General Probabilistic Surface Optimization and Log Density Estimation

Dmitry Kopitkov

General Probabilistic Surface Optimization and Log Density Estimation

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Dmitry Kopitkov

Submitted to the Senate
of the Technion — Israel Institute of Technology
Cheshvan 5781 Haifa October 2020

This research was carried out under the supervision of Associate Prof. Vadim Indelman, in the Faculty of Aerospace Engineering.

Acknowledgements

I would like to thank my advisor Prof. Vadim Indelman for all the guidance, help, support and research freedom I enjoyed through the years of my PhD's studies. My research skills and knowledge got elevated significantly all thanks to his high-standard demands and him driving me to reach my limits.

I would also like to thank the people from Technion Autonomous Systems Program for helping me getting over the bureaucracy of the university and for being supportive through sometimes very uneasy moments during these years.

Finally, I thank deeply all my friends for having my back and being there for me this entire time. You continued to believe in me even at my darkest days, and your help allowed me to keep standing on my legs and to accomplish this research. You have my immense gratitude and appreciation for all your support.

The generous financial help of the Technion is gratefully acknowledged.

Contents

List of Tables

Abstract

Probabilistic inference, such as density (ratio) estimation, is a fundamental and highly important problem that needs to be solved in many different domains. Recently, a lot of research was done to solve it by producing various objective functions optimized over neural network (NN) models. Such Deep Learning (DL) based approaches include *unnormalized* and *energy* models, as well as critics of Generative Adversarial Networks, where DL has shown top approximation performance. In this thesis we contribute a novel algorithm family, which generalizes all above, and allows us to infer different statistical modalities (e.g. data likelihood and ratio between densities) from data samples. The proposed *unsupervised* technique, named *Probabilistic Surface Optimization* (PSO), views a model as a flexible surface which can be pushed according to loss-specific virtual stochastic forces, where a dynamical equilibrium is achieved when the pointwise forces on the surface become equal. Concretely, the surface is pushed *up* and *down* at points sampled from two different distributions. The averaged *up* and *down* forces become functions of these two distribution densities and of force *magnitudes* defined by the loss of a particular PSO instance. Upon convergence, the force equilibrium imposes an optimized model to be equal to various statistical functions depending on the used *magnitude* functions. Furthermore, this dynamical-statistical equilibrium is extremely intuitive and useful, providing many implications and possible usages in probabilistic inference. We connect PSO to numerous existing statistical works which are also PSO instances, and derive new PSO-based inference methods as a demonstration of PSO exceptional usability. Likewise, based on the insights coming from the virtual-force perspective we analyze PSO stability and propose new ways to improve it. Finally, we present new instances of PSO, termed PSO-LDE, for data log-density estimation and also provide a new NN block-diagonal architecture for increased surface flexibility, which significantly improves estimation accuracy. Both PSO-LDE and the new architecture are combined together as a new density estimation technique. In our experiments we demonstrate this technique to be superior over state-of-the-art baselines in density estimation tasks for multi-modal 20D data.

Abbreviations and Notations

PSO	:	Probabilistic Surface Optimization
PSO-LDE	:	PSO *log density estimators*
PSO-CM	:	PSO *consistent magnitude* set
pdf	:	probability density function
NN	:	Neural Network
CNN	:	Convolutional NN
FC	:	Fully-connected NN
BD	:	Block-diagonal NN
RKHS	:	Reproducing Kernel Hilbert Space
DL	:	Deep Learning
GD	:	Gradient Descent optimization
NTK	:	Neural Tangent Kernel
FIM	:	Fisher Information Matrix
MLE	:	Maximum Likelihood Estimation
KDE	:	Kernel Density Estimation
NCE	:	Noise Contrastive Estimation
GAN	:	Generative Adversarial Network model
cGAN	:	Conditional GAN model
MC	:	Monte Carlo sampling
CD	:	Contrastive Divergence
LF	:	Legendre-Fenchel transform

Notation	**Description**
$\mathbb{R}_{>0}$, $\mathbb{R}_{\geq 0}$ and $\mathbb{R}_{<0}$	sets of real numbers; positive $\{x \in \mathbb{R} \| x > 0\}$, non-negative $\{x \in \mathbb{R} \| x \geq 0\}$ and negative $\{x \in \mathbb{R} \| x < 0\}$ respectively
$\mathcal{F}$	function space over which PSO is inferred
$f_\theta(X) : \mathbb{R}^n \rightarrow \mathbb{R}$	model $f_\theta \in \mathcal{F}$, parametrized by θ (e.g. a neural network), can be viewed as a surface with support in $\mathbb{R}^n$ whose height is the output of $f_\theta(X)$
$\theta \in \mathbb{R}^{\|\theta\|}$	model parameters (e.g. neural network weights vector)
$X \in \mathbb{R}^n$	input space of $f_\theta(X)$, can be viewed as support of model surface in space $\mathbb{R}^{n+1}$
$X^U \sim \mathbb{P}^U$	n-dimensional random variable with pdf $\mathbb{P}^U$, samples of which are the locations where we push the model surface *up*
$X^D \sim \mathbb{P}^D$	n-dimensional random variable with pdf $\mathbb{P}^D$, samples of which are the locations where we push the model surface *down*
$\mathbb{S}^U \subset \mathbb{R}^n$	support of $\mathbb{P}^U$
$\mathbb{S}^D \subset \mathbb{R}^n$	support of $\mathbb{P}^D$
$\mathbb{S}^{U \cup D} \subset \mathbb{R}^n$	support union of $\mathbb{P}^U$ and $\mathbb{P}^D$
$\mathbb{S}^{U \cap D} \subset \mathbb{R}^n$	support intersection of $\mathbb{P}^U$ and $\mathbb{P}^D$
$\mathbb{S}^{U \setminus D} \subset \mathbb{R}^n$	support of $\mathbb{P}^U$ where $\mathbb{P}^D(X) = 0$
$\mathbb{S}^{D \setminus U} \subset \mathbb{R}^n$	support of $\mathbb{P}^D$ where $\mathbb{P}^U(X) = 0$
$M^U\left[X, f_\theta(X)\right] : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$	*force-magnitude* function that amplifies an *up* push force which we apply at X^U
$M^D\left[X, f_\theta(X)\right] : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$	*force-magnitude* function that amplifies a *down* push force which we apply at X^D
$R\left[X, f_\theta(X)\right] : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$	ratio function $\frac{M^D}{M^U}$
$T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right] : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$	convergence function satisfying $T \equiv R^{-1}$, describes the modality that PSO optima $f^*(X)$ approximates
$\mathbb{K} = (s_{min}, s_{max}) \subseteq \mathbb{R}$	convergence interval defined as the range of $T\left[X, z\right]$ w.r.t. $z \in \mathbb{R}_{>0}$, represents a set of values f^* can have, $f^*(X) \in \mathbb{K}$
$\widetilde{M}^U$ and $\widetilde{M}^D$	antiderivatives of M^U and M^D (PSO *primitives*)
$F^U_\theta(X)$ and $F^D_\theta(X)$	point-wise *up* and *down* forces, that are applied (on average) at any point $X \in \mathbb{R}^n$
N^U and N^D	batch sizes of samples from $\mathbb{P}^U$ and from $\mathbb{P}^D$, that are used in a single optimization iteration
$L_{PSO} : \mathcal{F} \rightarrow \mathbb{R}$	population *PSO functional*
$\hat{L}^{N^U, N^D}_{PSO} : \mathcal{F} \rightarrow \mathbb{R}$	empirical *PSO functional*, approximates L_{PSO} via training points $\{X^U_i\}^{N^U}_{i=1}$ and $\{X^D_i\}^{N^D}_{i=1}$
$g_\theta(X, X')$	model kernel that is responsible for generalization and interpolation during the GD optimization
$r_\theta(X, X')$	relative model kernel, a scaled version $r_\theta(X, X') = g_\theta(X, X') / g_\theta(X, X)$ of $g_\theta(X, X')$ whose properties can be used to analyze the bias-variance trade-off of PSO

Table 1: Thesis Main Notations

CHAPTER 1

Introduction

Probabilistic inference is the wide domain of extremely important statistical problems including density (ratio) estimation, distribution transformation, density sampling and many more. Solutions to these problems are extensively used in domains of robotics, computer image, economics, and other scientific/industrial data mining cases. Particularly, in robotics we require to manually/automatically infer a measurement model between sensor measurements and the hidden state of the robot, which can further be used to estimate robot state during an on-line scenario. Considering the above, solutions to probabilistic inference and their applications to real-world problems are highly important for many scientific fields.

The universal approximation theory [48] states that an artificial neural network with fully-connected layers can approximate any continuous function on compact subsets of $\mathbb{R}^n$, making it an universal approximation tool. Moreover, in the last decade methods based on Deep Learning (DL) provided outstanding performance in areas of computer vision and reinforcement learning. Furthermore, recently strong frameworks (e.g. [19, 108, 135]) were developed that allow fast and sophisticated training of neural networks (NNs) using GPUs.

With the above motivation, in this thesis we contribute a novel unified paradigm, *Probabilistic Surface Optimization* (PSO), that allows to solve various probabilistic inference problems using DL, where we exploit the approximation power of NNs in full. PSO expresses the probabilistic inference as a virtual physical system where the surface, represented by a function from the optimized function space (e.g. NN), is pushed by forces that are outcomes of a Gradient Descent (GD) optimization. We show that this surface is pushed during the optimization to the target surface for which the averaged pointwise forces cancel each other. Further, by using different virtual forces we can enforce the surface to converge to different probabilistic functions of data, such as a data density, various density ratios, conditional densities and many other useful statistical modalities.

We show that many existing probabilistic inference approaches, like *unnormalized* models, GAN critics, *energy* models and cross-entropy based methods, already apply such PSO principles implicitly, even though their underlying dynamics were not explored before through the prism of virtual forces. Additionally, many novel and original methods can be forged in a simple way by following the same fundamental rules of the virtual surface and the force balance. Moreover, PSO framework permits the proposal and the practical usage of new objective functions that can not be expressed in closed-form, by instead defining their Euler-Lagrange equation. This allows introduction of new estimators that were not considered before. Furthermore, motivated by usefulness and intuitiveness of the proposed PSO paradigm, we derive sufficient conditions for its optimization stability and further analyze its convergence.

Importantly, we emphasize that PSO is not only a new interpretation that allows for simplified and intuitive understanding of statistical learning. Instead, in this thesis we show that optimization dynamics that the inferred model undergoes are indeed matching the picture of a physical surface with particular forces applied on it. Moreover, such match allowed us to understand the optimization stability of PSO instances in more detail and to suggest new ways to improve it.

Further, we apply PSO framework to solve the density estimation task - a fundamental statistical problem essential in many scientific fields and application domains. We analyze PSO sub-family with the corresponding equilibrium, proposing a novel PSO log-density estimators (PSO-LDE). These techniques, as also other PSO-based density estimation approaches presented in this thesis, do not impose any explicit constraint over a total integral of the learned model, allowing it to be entirely unnormalized. Yet, the implicit PSO force balance produces at the convergence density approximations that are highly accurate and *almost* normalized, with total integral being very close to 1.

Additionally, we examine several NN architectures for a better estimation performance, and propose new block-diagonal layers that led us to significantly improved accuracy. PSO-LDE approach combined with new NN architecture allowed us to learn multi-modal densities of 20D continuous data with superior precision compared to other state-of-the-art methods, including Noise Contrastive Estimation (NCE) [39, 126], which we demonstrate in our experiments.

Lastly, we relate the performance of PSO estimators to the model kernel also known in DL community as Neural Tangent Kernel (NTK) [54]. We analyze its impact on PSO accuracy and on bias-variance tradeoff, and further investigate its evolution during a typical optimization process. The revealed alignment of this kernel towards the target function provides insights on the learning dynamics of deep models which may lead in the future to a better understanding of DL theory.

1.1 Thesis Structure

The thesis is organized into 15 chapters as follows.

Chapter 2 In this chapter we cover the work related to PSO framework. Since the topic of this thesis is very wide and since PSO has tight relations with numerous existing works, here we describe only the most related scientific literature.

Chapter 3 In this chapter we formulate PSO algorithm family, the main contribution of this thesis. The proposed estimation procedure minimizes the novel *PSO functional*, whose Euler-Lagrange equation allows to learn many statistical objective functions. As described, such optimization resembles a physical system of forces applied over the virtual geometric body which we call the model surface. Such physical system perspective brings with itself a strong intuition and a conceptual simplicity, which makes PSO framework convenient and easy to use.

Chapter 4 In this chapter we use convex theory to derive sufficient optimality conditions over various PSO components, under which the Euler-Lagrange equation represents a minimum of *PSO functional*. Likewise, we derive PSO convergence in the neighborhoods where only one physical force is present, which can be used to understand PSO dynamics outside of the mutual support of involved densities.

Chapter 5 In this chapter we relate PSO framework to many other existing methods, showing them to be its instances. Additionally, we show how to derive convergence of any considered PSO instance and how to derive new PSO instances for inference of various statistical modalities. Likewise, here we introduce the terminology to modulate PSO into various subgroups.

Chapter 6 In this chapter we define *PSO divergence* and outline its relation towards Bregman divergence and f-divergence. Particularly, we show that estimators based on these divergences are subsets of PSO estimation family.

Chapter 7 In this chapter numerous estimation properties are proved, including consistency and asymptotic normality. Likewise, here we investigate the model kernel's impact on the optimization equilibrium.

Chapters 8-9 In these chapters we focus in more detail on application of PSO for (conditional) density estimation. We introduce novel PSO-LDE estimators with bounded *magnitude* functions, and likewise present their conditional form. Further, we relate conditional GAN (cGAN) methods with PSO framework.

Chapter 10 In this chapter we show various additional PSO applications and relations. Particularly, we describe in detail the connection between PSO, cross-entropy, MLE, and contrastive divergence [46]. Further, we describe the procedure for estimation of mutual information between two random variables, and PSO-based solution for occupancy mapping in robotics domain.

Chapter 11 In this chapter we investigate the NN design and its optimization influence. Specifically, here we propose a new block-diagonal (BD) NN architecture as an alternative to the fully-connected NN. We show that the new connectivity pattern, employed by BD NN, reduces the bandwidth of the corresponding model kernel which can be used to reduce the estimation bias of PSO.

Chapter 12 In this chapter we empirically illustrate the main practical issue of PSO algorithm - PSO *overfitting*. Particularly, we show that in case of a small amount of training points and a narrow bandwidth of the model kernel we may succeed in stretching the virtual surface to have spikes at each training point. Such convergence is very undesirable and may be interpreted as *overfitting* of the learning process. Further, here we describe several techniques that may prevent this over-flexibility phenomena.

Chapter 13 In this chapter we experiment with various PSO instances and solve numerous probabilistic inference tasks. Specifically, we apply PSO on the density estimation problem and investigate the actual performance of different PSO instances in large and small dataset size settings. Likewise, we compare the results with other state-of-the-art density estimation methods that are not part of PSO, and show the superiority of PSO-based techniques.

Chapter 14 In this chapter we empirically investigate dynamics of the model kernel of NNs during a typical optimization process, showing its extreme alignment towards the target function. Such surprising behavior of deep models may explain their approximation power supremacy compared to more shallow models. Further, in context of PSO this alignment improves the overall inference accuracy, and may lead in the future to additional theoretical insights of PSO dynamics.

Chapter 15 In this chapter we conclude the thesis and discuss its main contributions. We also describe possible directions for future research. These include many important PSO aspects that need to be solved in order to further enhance the estimation performance, such as an optimality of various PSO instances and a study of interplay between *magnitude* functions and the model kernel.

1.2 Thesis Contribution

The main contributions of this thesis are:

(a) We develop a *Probabilistic Surface Optimization* (PSO) that enforces any approximator function to converge to a target statistical function which nullifies a point-wise virtual force.

(b) We derive sufficient optimality conditions under which the functional implied by PSO is stable during the optimization.

(c) We show that many existing probabilistic and (un-)supervised learning techniques can be seen as instances of PSO.

(d) We show how new probabilistic techniques can be derived in a simple way by using PSO principles, and also propose several such new methods.

(e) We provide analysis of PSO convergence where we relate its performance towards properties of the model kernel implicitly defined by the optimized function space.

(f) We use PSO to approximate a logarithm of the target density, proposing for this purpose several hyper-parametric PSO subgroups and analyzing their properties.

(g) We present a new NN architecture with block-diagonal layers that allows for lower side-influence (a smaller bandwidth of the corresponding model kernel) between various regions of the input space and that leads to a higher NN flexibility and to more accurate density estimation.

(h) We experiment with different continuous 20D densities, and accurately infer all of them using the proposed PSO instances, thus demonstrating these instances' robustness and top performance. Further, we compare our methods with state-of-the-art baselines, showing the superiority of former over latter.

(i) We show that the model kernel of NN serves as a NN memory, with its *top* eigenfunctions changing to align with the learned target function. This improves the optimization performance since the convergence rate along kernel *top* eigenfunctions is typically higher.

The following works were published/in submission process as part of this thesis:

- D. Kopitkov, V. Indelman. Robot Localization through Information Recovered From CNN Classificators. *International Conference on Intelligent Robots and Systems (IROS)*, 2018.
- D. Kopitkov, V. Indelman. Neural Spectrum Alignment: Empirical Study. *International Conference on Artificial Neural Networks (ICANN)*, 2020 (accepted).
- D. Kopitkov, V. Indelman. General Probabilistic Surface Optimization and its Variational Equilibrium. Submitted to *Journal of Machine Learning Research (JMLR)*.

Preprints in arXiv (still to be submitted):

- D. Kopitkov, V. Indelman. General Probabilistic Surface Optimization and Log Density Estimation.
- D. Kopitkov, V. Indelman. Deep PDF: Probabilistic Surface Optimization and Density Estimation.

CHAPTER 2

Related work

In this section we consider very different problems all of which involve reasoning about statistical properties and probability density of a given data, which can be also solved by various instances of PSO as is demonstrated in later sections. We describe studies done to solve these problems, including both DL and not-DL based methods, and relate their key properties to attributes of PSO.

2.1 Unsupervised Probabilistic Inference

Statistical estimation consists of learning various probabilistic modalities from acquired sample realizations. For example, given a dataset we may want to infer the corresponding probability density function (pdf). Similarly, given two datasets we may want to approximate the density ratio between sample distributions. The usage of NNs for statistical estimation was studied for several decades [14, 16, 44, 127, 140]. Furthermore, there is a huge amount of work that treats statistical learning in a similar way to PSO, based on sample frequencies, the optimization *energies* and their forces. Arguably, the first methods were Boltzman machines (BMs) and Restricted Boltzman machines (RBMs) [1, 46, 98]. Similarly to PSO, RBMs can learn a distribution over data samples using a physical equilibrium, and were proved to be very useful for various ML tasks such as dimensionality reduction and feature learning. Yet, they were based on a very basic NN architecture, containing only hidden and visible units, arguably because of over-simplified formulation of the original BM. Moreover, the training procedure of these methods, the contrastive divergence (CD) [46], applies computationally expensive Monte Carlo (MC) sampling. In Section 10.3 we describe CD in detail and outline its exact relation to PSO procedure, showing that the latter replaces the expensive MC by sampling auxiliary distribution which is computationally cheap.

In [91] authors extended RBMs to Deep Energy Models (DEMs) that contained multiple fully-connected layers, where during training each layer was trained separately via CD. Further, in [151] Deep Structured Energy Based Models were proposed that used fully-connected, convolutional and recurrent NN architectures for an anomaly detection of vector data, image data and time-series data respectively. Moreover, in the latter work authors proposed to train *energy* based models via a score matching method [50], which does not require MC sampling. A similar training method was also recently applied in [119] for learning an *energy* function of data - an *unnormalized* model that is proportional to the real density function. However, the produced by score matching *energy* function is typically over-smoothed and entirely unnormalized, with its total integral being arbitrarily far from 1 (see Section 13.2.2). In contrast, PSO based density estimators (e.g. PSO-LDE) yield a model that is *almost* normalized, with its integral being very close to 1 (see Section 13.2.1).

In [69] authors examined many statistical loss functions under the perspective of *energy* model learning. Their overview of existing learning rules describes a typical optimization procedure as a physical system of model pushes at various data samples in *up* and *down* directions, producing the intuition very similar to the one promoted in this work. Although [69] and our works were done in an independent manner with the former preceding the latter, both acknowledged that many objective functions have two types of terms corresponding to two force directions, that are responsible to enforce model to output desired energy levels for various neighborhoods of the input space. Yet, unlike [69] we take one step further and derive the precise way to control the involved forces, producing a formal framework for the generation of infinitely many learning rules to infer an infinitely large number of target functions. The proposed PSO approach is conceptually very intuitive, and permits unification of many various methods under a single algorithm umbrella via a formal yet simple mathematical exposition. This in its turn allows to address the investigation of different statistical techniques and their properties as one mutual analysis study.

In context of pdf estimation, one of the most relevant works to presented in this thesis PSO-LDE approach is NCE [39, 126], which formulates the inference problem via a binary classification between original data samples and auxiliary noise samples. The derived loss allows for an efficient (conditional) pdf inference and is widely adapted nowadays in the language modeling domain [65,83,84]. Further, the proposed PSO-LDE can be viewed as a generalization of NCE, where the latter is a specific member of the former for a hyper-parameter $\alpha = 1$. Yet importantly, both algorithms were derived based on different mathematical principles, and their formulations do not exactly coincide.

Furthermore, the presented herein PSO family is not the first endeavor for unifying different statistical techniques under a general algorithm umbrella. In [105] authors proposed a family of *unnormalized* models to infer log-density, which is based on Maximum Likelihood Monte Carlo estimation [28]. Their method infers both the *energy* function of the data and the appropriate normalizing constant. Thus, the produced (log-)pdf estimation is *approximately* normalized. Further, this work was extended in [38] where it was related to the separable Bregman divergence and where various other statistical methods, including NCE, were shown to be instances of this inference framework. In Section 6 we prove Bregman-based estimators to be contained inside PSO estimation family, and thus both of the above frameworks are strict subsets of PSO.

Further, in [92] and [94] new techniques were proposed to infer various f-divergences between two densities, based on M-estimation procedure and Fenchel conjugate [47]. Likewise, the f-GAN framework in [94] was shown to include many of the already existing GAN methods. In Section 6 we prove that estimation methods from [92] and critic objective functions from [94] are also strict subsets of PSO.

The above listed methods, as also the PSO instances in Section 5, are all derived using various math fields, yet they also could be easily derived via PSO *balance state* as is described in this thesis. Further, the simplest way to show that PSO is a generalization and not just another perspective that is identical to previous works is as follows. In most of the above approaches optimization objective functions are required to have an analytically known closed form, whereas in our framework knowledge of these functions is not even required. Instead, we formulate the learning procedure via *magnitude* functions, the derivatives of various loss terms, knowing which is enough to solve the corresponding minimization problem. Furthermore, the *magnitudes* of PSO-LDE sub-family in Eq. (8.7)-(8.8) do not have a known antiderivative for the general case of any α, with the corresponding PSO-LDE loss being unknown. Thus, PSO-LDE (and therefore PSO) cannot be viewed as an instance of any previous statistical framework. Additionally, the intuition and simplicity in viewing the optimization as merely point-wise pushes over some virtual surface are very important for the investigation of PSO stability and for its applicability in numerous different areas.

2.2 Parametric vs Non-parametric Approaches

The most traditional probabilistic problem, which is also one of the main focuses of this thesis, is density approximation for an arbitrary data. Approaches for statistical density estimation may be divided into two different branches - parametric and non-parametric. Parametric methods assume data to come from a probability distribution of a specific family, and infer parameters of that family, for example via minimizing the negative log-probability of data samples. Non-parametric approaches are distribution-free in the sense that they do not take any assumption over the data population a priori. Instead they infer the distribution density totally from data.

The main advantage of the parametric approaches is their statistical efficiency. Given the assumption of a specific distribution family is correct, parametric methods will produce more accurate density estimation for the same number of samples compared to non-parametric techniques. However, in case the assumption is not entirely valid for a given population, the estimation accuracy will be poor, making parametric methods not statistically robust. For example, one of the most expressive distribution families is a Gaussian Mixture Model (GMM) [79]. One of its structure parameters is the number of mixtures. Using a high number of mixtures, it can represent multi-modal populations with a high accuracy. Yet, in case the real unknown distribution has even higher number of modes, or sometimes even an infinite number, the performance of a GMM will be low.

To handle the problem of unknown number of mixture components in parametric techniques, Bayesian statistics can be applied to model a prior over parameters of the chosen family. Models such as Dirichlet process mixture (DPM) and specifically Dirichlet process Gaussian mixture model (DPGMM) [4,33,122] can represent an uncertainty about the learned distribution parameters and as such can be viewed as infinite mixture models. Although these hierarchical models are more statistically robust (expressive), they still require to manually select a base distribution for DPM, limiting their robustness. Likewise, Bayesian inference applied in these techniques is more theoretically intricate and computationally expensive [75].

On the other hand, non-parametric approaches can infer distributions of an (almost) arbitrary form. Methods such as data histogram and kernel density estimation (KDE) [120,125] use frequencies of different points within data samples in order to conclude how a population pdf looks like. In general, these methods require more samples and prone to the *curse of dimensionality*, but also provide a more robust estimation by not taking any prior assumptions. Observe that "non-parametric" terminology **does not** imply lack of parametrization. Both histogram and KDE require selection of (hyper) parameters - bin width for histogram and kernel type/bandwidth for KDE.

In many cases a selection of optimal parameters requires the manual parameter search [125]. Although an automatic parameter deduction was proposed for KDE in several studies [24,43,95], it is typically computationally expensive and its performance is not always optimal. Furthermore, one of the major weaknesses of KDE is its time complexity during the query stage. Even the most efficient KDE methods (e.g. [95]) require an above linear complexity ($O(m \log m)$) in the number of query points m. In contrast, PSO yields robust non-parametric algorithms that optimize the NN model, which in its turn can be queried at any input point by a single forward pass. Since this pass is independent of m, the query runtime of PSO is linear in m. When the complexity of NN forward pass is lower than $\log m$, PSO methods become a much faster alternative. Moreover, existing KDE implementations do not scale well for data with a high dimension, unlike PSO methods.

2.3 Additional Density Estimation Techniques

A unique work combining DL and non-parametric inference was done by Baird et al. [10]. The authors represent a target pdf via Jacobian determinant of a bijective NN that has an implicit property of non-negativity with the total integral being 1. Additionally, their *pdf learning algorithm* has similarity to our *pdf loss* described in [61] and which is also shortly presented in Section 8.1. Although the authors did not connect their approach to virtual physical forces that are pushing a model surface, their algorithm can be seen as a simple instance of the more general DeepPDF method that we contributed in our previous work.

Furthermore, the usage of Jacobian determinant and bijective NNs in [10] is just one instance of DL algorithm family based on a nonlinear independent components analysis. Given the transformation $G_\phi(\cdot)$ typically implemented as a NN parametrized by ϕ, methods of this family [20–22,114] exploit the integration by substitution theorem that provides a mathematical connection between random X's pdf $\mathbb{P}[X]$ and random $G_\phi(X)$'s pdf $\mathbb{P}[G_\phi(X)]$ through Jacobian determinant of G_ϕ. In case we know $\mathbb{P}[X]$ of NN's input X, we can calculate in closed form the density $\mathbb{P}[G_\phi(X)]$ of NN's output and vice versa, which may be required in different applications. However, for the substitution theorem to work the transformation $G_\phi(\cdot)$ should be invertible, requiring to restrict NN architecture of $G_\phi(\cdot)$ which significantly limits NN expressiveness. In contrast, the presented PSO-LDE approach does not require any restriction over its NN architecture.

Further, another body of research in DL-based density estimation was explored in [27, 67, 140], where the autoregressive property of density functions was exploited. The described methods NADE, RNADE and MADE decompose the joint distribution of a multivariate data into a product of simple conditional densities where a specific variable ordering needs to be selected for better performance. Although these approaches provide high statistical robustness, their performance is still limited since every simple conditional density is approximated by a specific distribution family thus introducing a bias into the estimation. Moreover, the provided solutions are algorithmically complicated. In contrast, in this thesis we develop a novel statistically robust and yet conceptually very simple algorithm for density estimation, PSO-LDE.

2.4 Relation to GANs

Recently, Generative Adversarial Networks (GANs) [32, 70, 109] became popular methods to generate new data samples (e.g. photo-realistic images). GAN learns a generative model of data samples, thus implicitly learning also the data distribution. The main idea behind these methods is to have two NNs, a generator and a critic, competing with each other. The goal of the generator NN is to create new samples statistically similar as much as possible to the given dataset of examples; this is done by transformation of samples from a predefined prior distribution $z \sim \mathbb{P}^Z$ which is typically a multivariate Gaussian. The responsibility of the critic NN is then to decide which of the samples given to it is the real data example and which is the fake. This is typically done by estimating the ratio between real and fake densities. The latter is performed by minimizing a critic loss, where most popular critic losses [6, 36, 76, 85, 86, 153] can be shown to be instances of PSO (see Section 5). Further, both critic and generator NNs are trained in adversarial manner, forcing generator eventually to create very realistic data samples.

Another extension of GAN is Conditional GAN methods (cGANs). These methods use additional labels provided for each example in the training dataset (e.g. ground-truth digit of image from MNIST dataset), to generate new data samples conditioned on these labels. As an outcome, in cGAN methods we can control to some degree the distribution of the generated data, for example by conditioning the generation process on a specific data label (e.g. generate an image of digit ”5”). Similarly, we can use such a conditional generative procedure in robotics where we would like to generate future measurements conditioned on old observations/current state belief. Moreover, cGAN critics are also members of PSO framework as is demonstrated in Section 9.1.

Further, it is a known fact that optimizing GANs is very unstable and fragile, though during the years different studies analyzed various instability issues and proposed techniques to handle them [5]. In [109], the authors proposed the DCGAN approach that combines several stabilization techniques such as the batch normalization and Relu non-linearity usage for a better

GAN convergence. Further improvement was done in [118] by using a parameter historical average and statistical feature matching. Additionally, in [5] it was demonstrated that the main reason for instability in training GANs is the density support difference between the original and generated data. While this insight was supported by very intricate mathematical proofs, we came to the same conclusion in Section 7 by simply applying equilibrium concepts of PSO. As we show, if there are areas where only one of the densities is positive, the critic's surface is pushed by virtual forces to infinity, causing the optimization instability (see also Figure 7.2).

Moreover, in our analysis we detected another significant cause for estimation inaccuracy - the strong implicit bias produced by the model kernel. In our experiments in Section 13 the bandwidth of this kernel is shown to be one of the biggest factors for a high approximation error in PSO. Moreover, in this thesis we show that there is a strong analogy between the model kernel and the kernel applied in kernel density estimation (KDE) algorithms [120, 125]. Considering KDE, low/high values of the kernel *bandwidth* can lead to both *underfitting* and *overfitting*, depending on the number of training samples. We show the same to be correct also for PSO. See more details in Sections 12 and 13.

2.5 Classification Domain

Considering supervised learning and the image classification domain, convolutional neural networks (CNNs) produce discrete class conditional probabilities [64] for each image. The typical optimization loss used by classification tasks is a categorical cross-entropy of data and label pair, which can also be viewed as an instance of our discovered PSO family, as shown in Section 10.1. In particular, the classification cross-entropy loss can be seen as a variant of the PSO optimization, pushing in parallel multiple virtual surfaces connected by a softmax transformation, that concurrently estimates multiple Bernoulli distributions. These distributions, in their turn, represent one categorical distribution $\mathbb{P}(C|I)$ that models probability of each object class C given a specific image I.

Beyond cross-entropy loss, many possible objective functions for Bayes optimal decision rule inference exist [77, 112, 123]. These objectives have various forms and a different level of statistical robustness, yet all of them enforce the optimized model to approximate $\mathbb{P}(C|I)$. PSO framework promotes a similar relationship among its instances, by allowing the construction of infinitely many estimators with the same equilibrium yet with some being more robust to outliers than the others. Further, the recently proposed framework [17] of classification losses extensively relies on notions of Fenchel duality, which is also employed in this thesis to derive the sufficient conditions over PSO *magnitudes*.

CHAPTER 3

Probabilistic Surface Optimization

In this section we formulate the definition of *Probabilistic Surface Optimization* (PSO) algorithm framework. While in previous work [61] we already explored a particular instance of PSO specifically for the problem of density estimation, in Section 5 we will see that PSO is actually a very general family of probabilistic inference algorithms that can solve various statistical tasks and that include a great number of existing methods.

3.1 Formulation

Consider an input space $X \in \mathbb{R}^n$ and two densities $\mathbb{P}^U$ and $\mathbb{P}^D$ defined on it, with appropriate pdfs $\mathbb{P}^U(X)$ and $\mathbb{P}^D(X)$ and with supports $\mathbb{S}^U \subset \mathbb{R}^n$ and $\mathbb{S}^D \subset \mathbb{R}^n$; U and D denote the *up* and *down* directions of forces under a physical perspective of the optimization (see below). Denote by $\mathbb{S}^{U\cap D}$, $\mathbb{S}^{U\setminus D}$ and $\mathbb{S}^{D\setminus U}$ sets $\{X : X \in \mathbb{S}^U \vee X \in \mathbb{S}^D\}$, $\{X : X \in \mathbb{S}^U \vee X \notin \mathbb{S}^D\}$ and $\{X : X \notin \mathbb{S}^U \vee X \in \mathbb{S}^D\}$ respectively. Further, denote a model $f_\theta(X) : \mathbb{R}^n \to \mathbb{R}$ parametrized by the vector θ (e.g. NN or a function in Reproducing Kernel Hilbert Space, RKHS). Likewise, define two arbitrary functions $M^U(X, s) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ and $M^D(X, s) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$, which we name *magnitude* functions; both functions must be continuous *almost everywhere* w.r.t. argument s (see also Table 1 for list of main notations). We propose a novel PSO framework for probabilistic inference, that performs a gradient-based iterative optimization $\theta_{t+1} = \theta_t - \delta \cdot d\theta$ with a learning rate δ where:

$$d\theta = -\frac{1}{N^U}\sum_{i=1}^{N^U} M^U\left[X_i^U, f_\theta(X_i^U)\right] \cdot \nabla_\theta f_\theta(X_i^U) + \frac{1}{N^D}\sum_{i=1}^{N^D} M^D\left[X_i^D, f_\theta(X_i^D)\right] \cdot \nabla_\theta f_\theta(X_i^D). \tag{3.1}$$

$\{X_i^U\}_{i=1}^{N^U}$ and $\{X_i^D\}_{i=1}^{N^D}$ are sample batches from $\mathbb{P}^U$ and $\mathbb{P}^D$ respectively. Each PSO instance is defined by a particular choice of $\{\mathbb{P}^U, \mathbb{P}^D, M^U, M^D\}$ that produces a different convergence of $f_\theta(X)$ by approximately satisfying PSO *balance state* (within a mutual support $\mathbb{S}^{U\cap D}$):

$$\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)} = \frac{M^D\left[X, f_{\theta^*}(X)\right]}{M^U\left[X, f_{\theta^*}(X)\right]}. \tag{3.2}$$

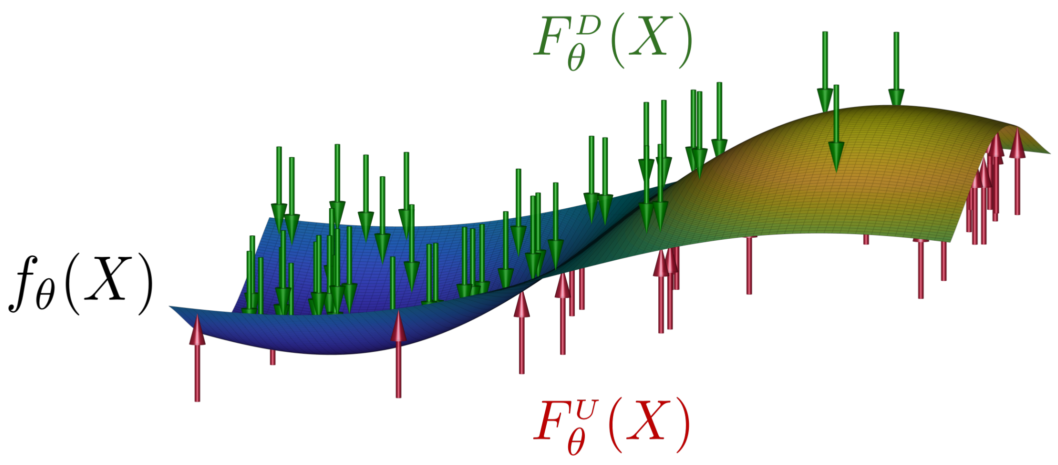


Figure 3.1: Illustration of PSO principles. Model $f_\theta(X) : \mathbb{R}^n \to \mathbb{R}$ (in this thesis NN parametrized by θ) represents a virtual surface that is pushed in opposite directions - *up* at points X^U sampled from $\mathbb{P}^U(X)$ and *down* at points X^D sampled from $\mathbb{P}^D(X)$. Magnitude of each push is amplified by analytical function - $M^U[X, f_\theta(X)]$ when pushing at X^U and $M^D[X, f_\theta(X)]$ when pushing at X^D, where both functions may have an almost arbitrary form, with only minor restrictions. During optimization via loss gradient in Eq. (3.1) the *up* and *down* forces $F_\theta^U(X) = \mathbb{P}^U(X) \cdot M^U[X, f_\theta(X)]$ and $F_\theta^D(X) = \mathbb{P}^D(X) \cdot M^D[X, f_\theta(X)]$, containing both frequency and analytical components, adapt to each other till point-wise *balance state* $F_\theta^U(X) = F_\theta^D(X)$ is achieved. Such convergence causes final $f_\theta(X)$ to be a particular function of $\mathbb{P}^U(X)$ and $\mathbb{P}^D(X)$, and can be used for inferring numerous statistical functions of arbitrary data (see Section 5).

Such optimization, outlined in Algorithm 3.1, will allow us to infer various statistical modalities from available data, making PSO a very useful and general algorithm family.

3.2 Derivation

Consider *PSO functional* over a function $f : \mathbb{R}^n \to \mathbb{R}$ as:

$$L_{PSO}(f) = - \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} \widetilde{M}^U[X, f(X)] + \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \widetilde{M}^D[X, f(X)] \tag{3.3}$$

where we define $\widetilde{M}^U[X, s] \triangleq \int_{s_0}^{s} M^U(X, t)dt$ and $\widetilde{M}^D[X, s] \triangleq \int_{s_0}^{s} M^D(X, t)dt$ to be antiderivatives of $M^U(\cdot)$ and $M^D(\cdot)$ respectively; these functions, referred below as *primitive* functions of PSO, are not necessarily known analytically. The above integral can be separated into several terms related to $\mathbb{S}^{U \cap D}$, $\mathbb{S}^{U \setminus D}$ and $\mathbb{S}^{D \setminus U}$. A minima f^* of $L_{PSO}(f)$ is described below, characterizing f^* within each of these areas.

Theorem 1 (Variational Characterization). *Consider densities $\mathbb{P}^U$ and $\mathbb{P}^D$ and magnitudes M^U and M^D. Define an arbitrary function $h : \mathbb{S}^{U \setminus D} \to \mathbb{R}$. Then for f^* to be minima of L_{PSO}, it must fulfill the following properties:*

1. *Mutual support: under "sufficient" conditions over $\{M^U, M^D\}$ the f^* must satisfy PSO balance state $\forall X \in \mathbb{S}^{U \cap D}$: $\mathbb{P}^U(X) \cdot M^U[X, f^*(X)] = \mathbb{P}^D(X) \cdot M^D[X, f^*(X)]$.*

2. *Disjoint support: depending on properties of a function M^U, it is necessary to satisfy $\forall X \in \mathbb{S}^{U \setminus D}$:*

   (a) If $\forall s \in \mathbb{R} : M^U(X, s) > 0$, then $f^*(X) = \infty$.

Algorithm 3.1 PSO estimation algorithm. Sample batches can be either identical or different for all iterations, which corresponds to GD and stochastic GD respectively.

```
Inputs:
$\mathbb{P}^U$ and $\mathbb{P}^D$ : up and down densities
$M^U$ and $M^D$ : magnitude functions
$\theta$ : initial parameters of model $f_\theta \in \mathcal{F}$
$\delta$ : learning rate
Outputs: $f_{\theta^*}$ : PSO solution that satisfies balance state in Eq. (3.2)
begin:
    while Not converged do
        Obtain samples $\{X_i^U\}_{i=1}^{N^U}$ from $\mathbb{P}^U$
        Obtain samples $\{X_i^D\}_{i=1}^{N^D}$ from $\mathbb{P}^D$
        Calculate $d\theta$ via Eq. (3.1)
        $\theta = \theta - \delta \cdot d\theta$
    end
    $\theta^* = \theta$
end
```

(b) If $\forall s \in \mathbb{R} : M^U(X, s) < 0$, *then* $f^*(X) = -\infty$.

(c) If $\forall s \in \mathbb{R} : M^U(X, s) \equiv 0$, *then* $f^*(X)$ *can be arbitrary.*

(d) If $\forall s \in \mathbb{R}$:

$$M^U(X, s) \rightarrow \begin{cases} = 0, & s = h(X) \\ > 0, & s < h(X) \\ < 0, & s > h(X) \end{cases} \tag{3.4}$$

then $f^*(X) = h(X)$.

(e) Otherwise, additional analysis is required.

The theorem's proof, showing PSO *balance state* to be Euler-Lagrange equation of $L_{PSO}(f)$, is presented in Section 4. Sufficient conditions over $\{M^U, M^D\}$ are likewise derived there. Part 2 helps to understand dynamics in areas outside of the mutual support, and its analogue for $\mathbb{S}^{D \setminus U}$ is stated in Section 4.2.2. Yet, below we will mostly rely on part 1, considering the optimization in area $\mathbb{S}^{U \cap D}$. Following from the above, finding a minima of L_{PSO} will produce a function f that satisfies Eq. (3.2).

To infer the above f^*, we consider a function space $\mathcal{F}$, whose each element f_θ can be parametrized by θ, and solve the problem $\min_{f_\theta \in \mathcal{F}} L_{PSO}(f_\theta)$. Assuming that $\mathcal{F}$ contains f^*, it will be obtained as a minima of the above minimization. Further, in practice L_{PSO} is optimized via gradient-based optimization where gradient w.r.t. θ is:

$$\nabla_\theta L_{PSO}(f_\theta) = - \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} M^U [X, f_\theta(X)] \cdot \nabla_\theta f_\theta(X) + \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} M^D [X, f_\theta(X)] \cdot \nabla_\theta f_\theta(X) \tag{3.5}$$

with Eq. (3.1) being its sampled approximation. Considering a hypothesis class represented by NN and the universal approximation theory [48], we assume that $\mathcal{F}$ is rich enough to learn the optimal f^* with high accuracy. Furthermore, in our experiments we show that in practice NN-based PSO estimators approximate the PSO *balance state* in a very accurate manner.

Remark 2. *PSO can be generalized into a functional gradient flow via the corresponding functional derivative of $L_{PSO}(f)$ in Eq. (3.3). Yet, in this thesis we will focus on GD formulation w.r.t. θ parametrization, outlined in Algorithm 3.1, leaving more theoretically sophisticated form for future work. Algorithm 3.1 is easy to implement in practice, if for example f_θ is represented as NN. Furthermore, in case $\mathcal{F}$ is RKHS, this algorithm can be performed by only evaluating RKHS's kernel at training points, by applying the kernel trick. The corresponding optimization algorithm is known as the kernel gradient descent. Further, in this thesis we consider loss functions without an additional regularization term such as RKHS norm or weight decay, since a typical GD optimization is known to implicitly produce a regularization effect [74]. Analysis of PSO combined with the explicit regularization term is likewise left for future work.*

3.3 PSO Balance State

Given that $\mathbb{P}^U$ and $\mathbb{P}^D$ have the same support, PSO will converge to PSO *balance state* in Eq. (3.2). By ignoring possible singularities (due to an assumed identical support) we can see that the converged surface f_{θ^*} will be such that the ratio of frequency components will be opposite-proportional to the ratio of *magnitude* components. To derive a value of the converged f_{θ^*} for a specific PSO instance, $\{M^U, M^D\}$ of that PSO instance, which typically involve f_θ inside them, must be substituted into Eq. (3.2) and then it needs to be solved for f_θ. This is equivalent to finding inverse $T(X, z)$ of the ratio $R(X, s) \triangleq \frac{M^D(X,s)}{M^U(X,s)}$ w.r.t. the second argument, $T \equiv R^{-1}$, with the convergence described as $f_{\theta^*}(X) = T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$. Such *balance state* can be used to mechanically recover many existing methods and to easily derive new ones for inference of numerous statistical functions of data; in Section 5 we provide full detail on this point. Furthermore, the above general formulation of PSO is surprisingly simple, considering that it provides a strong intuition about its optimization dynamics as a physical system, as explained below.

3.4 Virtual Surface Perspective

The main advantage of PSO is in its conceptual simplicity, revealed when viewed via a physical perspective. Specifically, $f_\theta(X)$ can be considered as a virtual surface in $\mathbb{R}^{n+1}$ space, with its support being $\mathbb{R}^n$, see Figure 3.1. Further, according to the update rule of a gradient-descent (GD) optimization and Eq. (3.1), any particular training point X updates θ during a single GD update by $\nabla_\theta f_\theta(X)$ magnified by the output of a *magnitude* function (M^U or M^D). Furthermore, considering the update of a simple form $\theta_{t+1} = \theta_t + \nabla_\theta f_{\theta_t}(X)$, it is easy to show that the height of the surface at any other point X' changes according to a first-order Taylor expansion as:

$$f_{\theta_{t+1}}(X') - f_{\theta_t}(X') \approx \nabla_\theta f_{\theta_t}(X')^T \cdot \nabla_\theta f_{\theta_t}(X). \tag{3.6}$$

Hence, by pushing (optimizing) a specific training point X, the surface at other points changes according to the elasticity properties of the model expressed via a *gradient similarity* kernel $g_\theta(X, X') \triangleq \nabla_\theta f_\theta(X)^T \cdot \nabla_\theta f_\theta(X')$. It helps also to think that during the above update we push at X with a virtual rod, appeared inside Figure 3.1 in form of green and red arrows, whose head shape is described by $g_\theta(X, X')$.

When optimizing over RKHS, the above expression turns to be identity and $g_\theta(X, X')$ collapses into the reproducing kernel[1]. For NNs, this model kernel is known as Neural Tangent Kernel (NTK) [54]. As was empirically observed, NTK has a strong local behavior with its outputs mostly being large only when points X and X' are close by. More insights about NTK can be found in [23, 54, 63]. Further assuming for simplicity that g_θ has zero-bandwidth $\forall X \neq X' : g_\theta(X, X') \equiv 0$ and that $\{M^U, M^D\}$ are non-negative functions, it follows then that each X_i^U in Eq. (3.1) pushes the surface at this point *up* by $g_\theta(X_i^U, X_i^U)$ magnified by $M^U\left[X_i^U, f_\theta(X_i^U)\right]$, whereas each X_i^D is pushing it *down* in a similar manner.

Considering a macro picture of such optimization, $f_\theta(X)$ is pushed *up* at samples from $\mathbb{P}^U$ and *down* at samples from $\mathbb{P}^D$, with the *up* and *down* averaged point-wise forces being $F_\theta^U(X) \triangleq \mathbb{P}^U(X) \cdot M^U\left[X, f_\theta(X)\right]$ and $F_\theta^D(X) \triangleq \mathbb{P}^D(X) \cdot M^D\left[X, f_\theta(X)\right]$ ($g_\theta(X, X)$ term is ignored since it is canceled out). Intuitively, such a physical process converges to a stable state when point-wise forces become equal, $F_\theta^U(X) = F_\theta^D(X)$. This is supported mathematically by the part 1 of Theorem 1, with such equilibrium being named as PSO *balance state*. Yet, it is important to note that this is only the variational equilibrium, which is obtained when training datasets are infinitely large and when the bandwidth of kernel g_θ is infinitely small. In practice the outcome of GD optimization **strongly** depends on the actual amount of available sample points and on various properties of g_θ, with the model kernel serving as a metric over the function space $\mathcal{F}$. Thus, the actual equilibrium somewhat deviates from PSO *balance state*, which we investigate in Section 7.

Additionally, the actual force direction at samples X_i^U and X_i^D depends on signs of $M^U\left[X_i^U, f_\theta(X_i^U)\right]$ and $M^D\left[X_i^U, f_\theta(X_i^D)\right]$, and hence may be different for each instance of PSO. Nonetheless, in most cases the considered *magnitude* functions are non-negative, and thus support the above exposition exactly. Moreover, for negative *magnitudes* the above picture of physical forces will still stay correct, after swapping between *up* and *down* terms.

The physical equilibrium can also explain dynamics outside of the mutual support $\mathbb{S}^{U \cap D}$. Considering the area $\mathbb{S}^{U \setminus D} \subset \mathbb{R}^n$ where only samples X_i^U are located, when M^U has positive outputs, the model surface is pushed indefinitely *up*, since there is no opposite force required for the equilibrium. Likewise, when M^U's outputs are negative - it is pushed indefinitely *down*. Further, when *magnitude* function is zero, there are no pushes at all and hence the surface can be anything. Finally, if the output of $M^U\left[X, f_\theta(X)\right]$ is changing signs depending on whether $f_\theta(X)$ is higher than $h(X)$, then $f_\theta(X)$ must be pushed towards $h(X)$ to balance the forces. Such intuition is supported mathematically by part 2 of Theorem 1. Observe that convergence at infinity actually implies that PSO will not converge to the steady state for any number of GD

[1]In RKHS defined via a feature map $\phi(X)$ and a reproducing kernel $k(X, X') = \phi(X)^T \cdot \phi(X')$, every function has a form $f_\theta(X) = \phi(X)^T \cdot \theta$. Since $\nabla_\theta f_\theta(X) = \phi(X)$, we obtain $g_\theta(X, X') \equiv k(X, X')$.

iterations. Yet, it can be easily handled by for example limiting range of functions within the considered $\mathcal{F}$ to some set $[a, b] \subset \mathbb{R}$.

Remark 3. *In this thesis we discuss PSO and its applications in context of only continuous multi-dimensional data, while in theory the same principles can work also for discrete data. The sampled points X_i^U and X_i^D will be located only at discrete locations of the surface $f_\theta(X)$ since the points are in $\mathbb{Z}^n \subset \mathbb{R}^n$. Yet, the balance at each such point will still be governed by the same up and down forces. Thus, we can apply similar PSO methods to also infer statistical properties of discrete data.*

CHAPTER 4

PSO Functional

Here we provide a detailed analysis of *PSO functional* $L_{PSO}(f)$, proving Theorem 1 and deriving sufficient optimality conditions to ensure its optimization stability for any considered pair of M^U and M^D. Examine L_{PSO}'s decomposition:

$$L_{PSO}(f) = L^{U\cap D}(f) + L^{U\setminus D}(f) + L^{D\setminus U}(f), \tag{4.1}$$

$$\begin{aligned}
L^{U\cap D}(f) &\triangleq \int_{\mathbb{S}^{U\cap D}} -\mathbb{P}^U(X)\cdot \widetilde{M}^U\left[X, f(X)\right] + \mathbb{P}^D(X)\cdot \widetilde{M}^D\left[X, f(X)\right] dX \\
L^{U\setminus D}(f) &\triangleq -\int_{\mathbb{S}^{U\setminus D}} \mathbb{P}^U(X)\cdot \widetilde{M}^U\left[X, f(X)\right] dX \\
L^{D\setminus U}(f) &\triangleq \int_{\mathbb{S}^{D\setminus U}} \mathbb{P}^D(X)\cdot \widetilde{M}^D\left[X, f(X)\right] dX.
\end{aligned} \tag{4.2}$$

In Section 4.1 we analyze $L^{U\cap D}$, specifically addressing non-differentiable functionals (Section 4.1.1), differentiable functionals (Section 4.1.2) and deriving extra conditions for unlimited range of $\mathcal{F}$ (Section 4.1.3). Further, in Section 4.2 we analyze $L^{U\setminus D}$ and $L^{D\setminus U}$, proving part 2 of Theorem 1.

4.1 Mutual Support Optima

Consider the loss term $L^{U\cap D}(f)$ corresponding to the area $\mathbb{S}^{U\cap D}$, where $\mathbb{P}^U(X) > 0$ and $\mathbb{P}^D(X) > 0$. The Euler-Lagrange equation of this loss is $-\mathbb{P}^U(X)\cdot M^U\left[X, f(X)\right] + \mathbb{P}^D(X)\cdot M^D\left[X, f(X)\right] = 0$, thus yielding the conclusion that $L^{U\cap D}$'s minimization must lead to the convergence in Eq. (3.2). Yet, calculus of variations does not easily produce the sufficient conditions that $\{M^U, M^D\}$ must satisfy for such steady state. Instead, below we will apply notions of Legendre-Fenchel (LF) transform from the convex optimization theory.

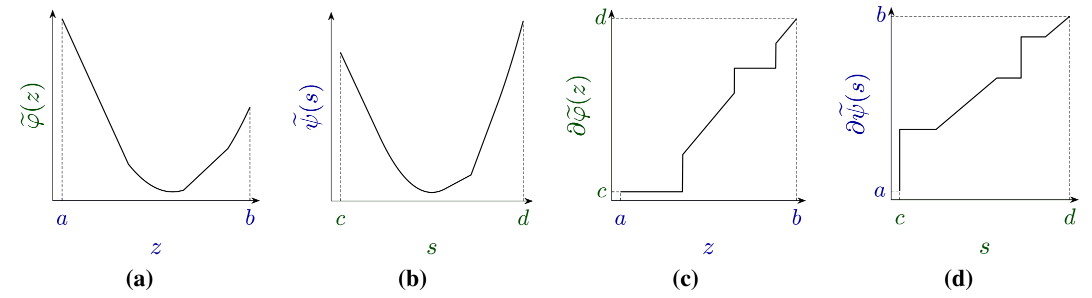


Figure 4.1: (a) Example of a convex function $\widetilde{\varphi}(z)$ defined on an interval (a, b), (b) its convex-conjugate $\widetilde{\psi}(s)$, (c) subderivative $\partial\widetilde{\varphi}(z)$ and (d) subderivative $\partial\widetilde{\psi}(s)$. The $\partial\widetilde{\varphi}(z)$ has jumps at points in (a, b) where $\widetilde{\varphi}(z)$ is non-differentiable, and is constant at points where $\widetilde{\varphi}(z)$ is not strictly convex. Further, $\partial\widetilde{\varphi}(z)$ and $\partial\widetilde{\psi}(s)$ are inverse mappings between (a, b) and (c, d), with constant regions in $\partial\widetilde{\varphi}(z)$ corresponding to jumps in $\partial\widetilde{\psi}(s)$, and vice versa. Additionally, any of the above four functions can be recovered from any other, thus each of them is a different representation of the same information. Observe that a convex function $\widetilde{\psi}(s)$ can be recovered from its subderivative $\partial\widetilde{\psi}(s)$ only up to an additive constant. Yet, this constant will not affect optima s^* of the considered below optimization $\min_{s\in(c,d)} \widetilde{\psi}(s) - z \cdot s$ and hence can be ignored. Furthermore, codomains of $\widetilde{\varphi}(z)$ and of $\widetilde{\psi}(s)$ do not play any role in our derivation.

4.1.1 PSO Non-Differentiable Case

The core concepts required for the below derivation are properties of convex functions and their derivatives, and an inversion relation between (sub-)derivatives of two convex functions that are also convex-conjugate of each other. Each convex function $\widetilde{\varphi}(z)$ on an interval (a, b) of real-line can be represented by its derivative $\varphi(z)$, with latter being increasing on (a, b) with finitely many discontinuities (jumps). Each non-differentiable point z_0 of $\widetilde{\varphi}$ is expressed within φ by a jump at z_0, and at each point where $\widetilde{\varphi}$ is not strictly convex the φ is locally constant. Left-hand and right-hand derivatives $\widetilde{\varphi}D_-(z)$ and $\widetilde{\varphi}D_+(z)$ of $\widetilde{\varphi}$

$$\widetilde{\varphi}D_-(z_0) = \lim_{z\to z_0^-} \frac{\widetilde{\varphi}(z) - \widetilde{\varphi}(z_0)}{z - z_0}, \quad \widetilde{\varphi}D_+(z_0) = \lim_{z\to z_0^+} \frac{\widetilde{\varphi}(z) - \widetilde{\varphi}(z_0)}{z - z_0} \tag{4.3}$$

can be constructed from φ by treating its finitely many discontinuities as left-continuities and right-continuities respectively. Further, $\widetilde{\varphi}$'s subderivative at z is defined as $\partial\widetilde{\varphi}(z) = [\widetilde{\varphi}D_-(z), \widetilde{\varphi}D_+(z)]$. Note that $\widetilde{\varphi}$ can be recovered from any one-sided derivative by integration [106, see Appendix C], up to an additive constant which will not matter for our goals. Hence, each one of $\widetilde{\varphi}$, φ, $\widetilde{\varphi}D_-$, $\widetilde{\varphi}D_+$ and $\partial\widetilde{\varphi}$ is just a different representation of the same information.

The LF transform $\widetilde{\psi}$ of $\widetilde{\varphi}$ (also known as convex-conjugate of $\widetilde{\varphi}$) is a convex function defined as $\widetilde{\psi}(s) \triangleq \sup_{z\in\mathbb{R}}\{sz - \widetilde{\varphi}(z)\}$. Subderivatives $\partial\widetilde{\varphi}$ and $\partial\widetilde{\psi}$ have the following useful inverse relation [115, see Proposition 11.3]: $\partial\widetilde{\psi}(s) = \{z : s \in \partial\widetilde{\varphi}(z)\}$. Moreover, in case $\widetilde{\varphi}$ and $\widetilde{\psi}$ are strictly convex and differentiable, their derivatives φ and ψ are strictly increasing and continuous, and actually are inverse functions between (a, b) and (c, d), with $c \triangleq \inf_{z\in(a,b)} \varphi(z)$ and $d \triangleq \sup_{z\in(a,b)} \varphi(z)$. Further, since $\partial\widetilde{\psi}$ can be recovered from $\partial\widetilde{\varphi}$, it contains the same information and is just one additional representation form. See illustration in Figure 4.1 and refer to [155] for a more intuitive exposition.

The above inverse relation can be used in optimization by applying the Fenchel-Young inequality: for any $z \in (a, b)$ and $s \in (c, d)$ we have $\widetilde{\varphi}(z) + \widetilde{\psi}(s) - z \cdot s \geq 0$, with equality obtained iff $s \in \partial\widetilde{\varphi}(z)$. Thus, for any given $z \in (a, b)$ the optima $s^* = \arg\min_{s \in (c,d)} \widetilde{\psi}(s) - z \cdot s$ must be within $\partial\widetilde{\varphi}(z)$. It is helpful to identify a role of each term within the above optimization problem. The $\widetilde{\psi}$ serves as part of the cost, $\partial\widetilde{\varphi}$ defines the optima s^*, $\partial\widetilde{\psi}$ is required to solve this optimization in practice (i.e. via subgradient descent), and $\widetilde{\psi}$ is not actually used. Thus, in order to define properties we want s^* to have and to perform the actual optimization, we only need to know $\partial\widetilde{\varphi}$ and $\partial\widetilde{\psi}$, with the latter being easily recovered from the former. For this reason, given an increasing function $\varphi(z)$ with domain (a, b) and codomain (c,d), in practice it is sufficient to know its inverse $\psi(s)$ (or the corresponding subderivative $\partial\widetilde{\psi}$) to solve the optimization and to obtain the optima $s^* \in (c, d)$ s.t. $s^* = \varphi(z)$ (or $s^* \in \partial\widetilde{\varphi}(z)$). Convex functions $\widetilde{\varphi}$ and $\widetilde{\psi}$ can be used symbolically for a math proof, yet their actual form is not required, which was also noted in [112]. This idea may seem pointless since to find s^* we can compute $\varphi(z)$ in the first place, yet it will help us in construction of a general optimization framework for probabilistic inference.

In following statements we define several functions with two arguments, where the first argument X can be considered as a "spectator" and where all declared functional properties are w.r.t. the second argument for any value of the first one. Define the required estimation convergence by a transformation $T(X, z) : \mathbb{R}^n \times \mathbb{R}_{>0} \to \mathbb{R}$. Given that T is increasing and right-continuous (w.r.t. $z \in \mathbb{R}_{>0}$ at any $X \in \mathbb{R}^n$), below we will derive a new objective functional whose minima is $f^*(X) = T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$ and which will have a form of $L^{U \cap D}$. Furthermore, this derivation will yield the sufficient conditions over PSO *magnitudes*.

Consider any fixed value of $X \in \mathbb{R}^n$. Denote by $\mathbb{K} = (s_{min}, s_{max})$ the effective convergence interval, with $s_{min} = \inf_{z \in \mathbb{R}_{>0}} T(X, z)$ and $s_{max} = \sup_{z \in \mathbb{R}_{>0}} T(X, z)$; at the convergence we will have $f^*(X) \in \mathbb{K}$. Further, below we will assume that the effective interval $\mathbb{K}$ is identical for any X, which is satisfied by all convergence transformations T considered in our work. It can be viewed as an assumption that knowing value of X without knowing value of $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$ does not yield any information about the convergence $T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$.

Due to its properties, $T(X, z) : \mathbb{R}^n \times \mathbb{R}_{>0} \to \mathbb{K}$ can be acknowledged as a right-hand derivative of some convex function $\widetilde{T}(X, z)$ (w.r.t. z). Denote by $\widetilde{T}D_-(X, z)$, $\widetilde{T}D_+(X, z) \equiv T(X, z)$ and $\partial\widetilde{T}(X, z)$ left-hand derivative, right-hand derivative and subderivative of $\widetilde{T}$.

Next, define a mapping $G(X, s)$ to be a strictly increasing and right-continuous function on $s \in \mathbb{K}$, with $\bar{\mathbb{K}} = \{G[X, s] : s \in \mathbb{K}\}$ being an image of $\mathbb{K}$ under G. Set $\bar{\mathbb{K}}$ may depend on value of X, although it will not affect below conclusions. We denote the left-inverse of $G(X, s) : \mathbb{R}^n \times \mathbb{K} \to \bar{\mathbb{K}}$ as $G^{-1}(X, t) : \mathbb{R}^n \times \bar{\mathbb{K}} \to \mathbb{K}$ s.t. $\forall s \in \mathbb{K} : G^{-1}(X, G(X, s)) = s$.

Define a mapping $\Phi(X, z) \triangleq G(X, T(X, z)) : \mathbb{R}^n \times \mathbb{R}_{>0} \to \bar{\mathbb{K}}$ and note that it is increasing (composition of two increasing functions is increasing) and right-continuous (right-continuity of G preserves all limits required for right-continuity of Φ). Similarly to T, define $\widetilde{\Phi}(X, z)$, $\widetilde{\Phi}D_-(X, z) \equiv G(X, \widetilde{T}D_-(X, z))$, $\widetilde{\Phi}D_+(X, z) \equiv G(X, \widetilde{T}D_+(X, z))$ and $\partial\widetilde{\Phi}(X, z)$ to be the corresponding convex function, its left-hand derivative, right-hand derivative and subderivative respectively.

Denote by $\widetilde{\Psi}(X, t)$ the LF transform of $\widetilde{\Phi}(X, z)$ w.r.t. z. Its subderivative at any $t \in \bar{\mathbb{K}}$ is $\partial\widetilde{\Psi}(X, t) = \{z : G^{-1}(t) \in \partial\widetilde{T}(X, z)\} \subset \mathbb{R}_{>0}$. According to the Fenchel-Young inequality, for any given $z \in \mathbb{R}_{>0}$ the optima t^* of $\min_{t \in \mathbb{K}} \widetilde{\Psi}(X, t) - t \cdot z$ must be within $\partial \widetilde{\Phi}(X, z)$. Further, this optimization can be rewritten as $\min_{G(X,s): s \in \mathbb{K}} \widetilde{\Psi}(X, G(X, s)) - G(X, s) \cdot z$ with its solution satisfying $s^* : G(X, s^*) \in \partial \widetilde{\Phi}(X, z)$, or:

$$s^* = \arg\min_{s \in \mathbb{K}} -z \cdot G(X, s) + \widetilde{\Psi}(X, G(X, s)), \tag{4.4}$$

$$G(X, s^*) \in [\widetilde{\Phi}D_-(X, z), \widetilde{\Phi}D_+(X, z)] \Rightarrow s^* \in [\widetilde{T}D_-(X, z), \widetilde{T}D_+(X, z)] \Rightarrow s^* \in \partial \widetilde{T}(X, z) \subset \mathbb{K}, \tag{4.5}$$

where we applied the left-inverse G^{-1}. The above statements are true for any considered $X \in \mathbb{R}^n$, with the convergence interval $\mathbb{K}$ being independent of X's value. Additionally, while the above problem is not necessarily convex in s (actually it is easily proved to be quasiconvex), it still has a well-defined minima s^*. Further, various methods can be applied if needed to solve this nonconvex nonsmooth optimization using $\partial \widetilde{\Psi}$ and various notions of G's subdifferential [9].

Substituting $z \equiv \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$, $G(X, s) \equiv \widetilde{M}^U(X, s)$ and $\widetilde{\Psi}(X, G(X, s)) \equiv \widetilde{M}^D(X, s)$ we get:

$$\begin{aligned} s^* = \arg\min_{s \in \mathbb{K}} -\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)} \cdot \widetilde{M}^U(X, s) + \widetilde{M}^D(X, s) = \\ = \arg\inf_{s \in \mathbb{K}} -\mathbb{P}^U(X) \cdot \widetilde{M}^U(X, s) + \mathbb{P}^D(X) \cdot \widetilde{M}^D(X, s), \end{aligned} \tag{4.6}$$

where we replaced minimum with infimum for the latter use. Optima s^* is equal to $T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$ if T is continuous at $z = \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$, or must be within $[\widetilde{T}D_-(X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}), \widetilde{T}D_+(X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)})]$ otherwise.

Next, denote $\mathcal{F}$ to be a function space with measurable functions $f : \mathbb{S}^{U \cap D} \to \mathbb{K}$ w.r.t. a base measure dX. Then, the optimization problem $\inf_{f \in \mathcal{F}} L^{U \cap D}(f)$ solves the problem in Eq. (4.6) for $f(X)$ at each $X \in \mathbb{S}^{U \cap D}$:

$$\inf_{f \in \mathcal{F}} L^{U \cap D}(f) = \int_{\mathbb{S}^{U \cap D}} \inf_{f(X)} \left[-\mathbb{P}^U(X) \cdot \widetilde{M}^U[X, f(X)] + \mathbb{P}^D(X) \cdot \widetilde{M}^D[X, f(X)] \right] dX, \tag{4.7}$$

where we can move infimum into the integral since f is measurable, the argument used also in works [92, 94]. Thus, the solution $f^* = \arg\inf_{f \in \mathcal{F}} L^{U \cap D}(f)$ must satisfy $\forall X \in \mathbb{S}^{U \cap D}$: $f^*(X) \in \partial \widetilde{T}\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$, given that $f^* \in \mathcal{F}$.

Finally, we summarize all conditions that are sufficient for the above conclusion: function $T(X, z)$ is increasing and right-continuous w.r.t. $z \in \mathbb{R}_{>0}$, the convergence interval $\mathbb{K}$ is X-invariant, $G(X, s)$ (also aliased as $\widetilde{M}^U(X, s)$) is right-continuous and strictly increasing w.r.t. $s \in \mathbb{K}$, and $\mathcal{F}$'s range is $\mathbb{K}$. Given T, $\mathbb{K}$, G and $\mathcal{F}$ have these properties, the entire above derivation follows.

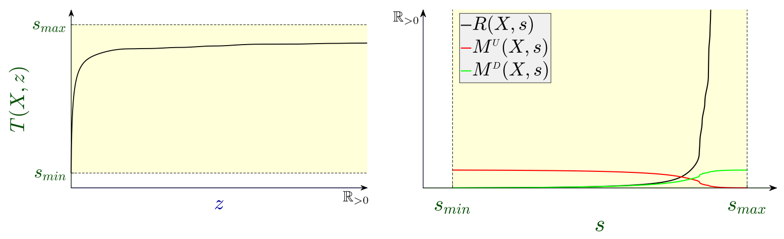


Figure 4.2: Summary of requirements over mappings $T(X,z)$, $R(X,s)$, $M^U(X,s)$ and $M^D(X,s)$. For any $X \in \mathbb{R}^n$, T must be strictly increasing and continuous on $z \in \mathbb{R}_{>0}$, with its image $\mathbb{K} = (s_{min}, s_{max})$ marked by yellow. The inverse R of T is likewise strictly increasing and continuous function from $\mathbb{K}$ to $\mathbb{R}_{>0}$. Further, M^U and M^D are any two functions that are positive and continuous on $\mathbb{K}$ with $\frac{M^D}{M^U} = R$.

4.1.2 PSO Differentiable Case

More "nice" results can be obtained if we assume additionally $T(X,z) : \mathbb{R}^n \times \mathbb{R}_{>0} \to \mathbb{K}$ to be strictly increasing and continuous w.r.t. $z \in \mathbb{R}_{>0}$, and $G(X,s) : \mathbb{R}^n \times \mathbb{K} \to \bar{\mathbb{K}}$ to be differentiable at $s \in \mathbb{K}$. This is the main setting on which our work is focused.

In such case T is invertible. Denote its inverse as $R(X,s) : \mathbb{R}^n \times \mathbb{K} \to \mathbb{R}_{>0}$. The R is strictly increasing and continuous w.r.t. $s \in \mathbb{K}$, and satisfies $\forall z \in \mathbb{R}_{>0} : R[X, T[X,z]] = z$ and $\forall s \in \mathbb{K} : T[X, R[X,s]] = s$.

Further, the derivative of $\widetilde{\Phi}(X,z)$ is $\Phi(X,z) = G(X, T(X,z))$. The Φ is strictly increasing and continuous w.r.t. $z \in \mathbb{R}_{>0}$, and thus $\widetilde{\Phi}$ is strictly convex and differentiable on $\mathbb{R}_{>0}$.

By LF transform's rules the derivative $\Psi(X,z)$ of $\widetilde{\Psi}$ is an inverse of $\widetilde{\Phi}$'s derivative $\Phi(X,z)$, and thus it can be expressed as $\Psi(X,z) = R(X, G^{-1}(X,z))$. This leads to $\frac{\partial \widetilde{\Psi}(X,G(X,s))}{\partial s} = R(X,s) \cdot G'(X,s)$ where $G'(X,s) \triangleq \frac{\partial G(X,s)}{\partial s}$ is the derivative of G.

From above we can conclude that $\widetilde{M}^U(X,s) \equiv G(X,s)$ and $\widetilde{M}^D(X,s) \equiv \widetilde{\Psi}(X, G(X,s))$ are both differentiable at $s \in \mathbb{K}$, with derivatives $M^U(X,s) = G'(X,s)$ and $M^D(X,s) = R(X,s) \cdot G'(X,s)$ satisfying $\frac{M^D(X,s)}{M^U(X,s)} = R(X,s)$. Functions M^U and M^D can be considered as *magnitudes* of physical forces, as explained in Section 3.4. Also, $M^U(X,s) > 0$ for any $s \in \mathbb{K}$ due to properties of G. Observe that we likewise have $R(X,s) > 0$ for any $s \in \mathbb{K}$ since $R(X,s) \in \mathbb{R}_{>0}$. This leads to $M^D(X,s) > 0$ at any $s \in \mathbb{K}$, which implies $\widetilde{M}^D(X,s)$ to be strictly increasing at $s \in \mathbb{K}$ (similarly to $\widetilde{M}^U$). Moreover, additionally taken assumptions will enforce the solution $f^* = \arg\inf_{f \in \mathcal{F}} L^{U \cap D}(f)$ to satisfy $\forall X \in \mathbb{S}^{U \cap D}$: $f^*(X) = T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$.

Above we derived a new objective function $L^{U \cap D}(f)$. Given that terms $\{T, \mathbb{K}, G, \mathcal{F}\}$ satisfy the declared above conditions, minima f^* of $L^{U \cap D}$ will be $T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$. Properties of $\{\widetilde{M}^U, \widetilde{M}^D, M^U, M^D\}$ follow from aforementioned conditions. This is summarized by below theorem, where instead of G we enforce the corresponding requirements over $\{M^U, M^D\}$. See also Figure 4.2 for an illustrative example.

Theorem 4 (Convergence-Focused). *Consider mappings $T(X,z)$, $M^U(X,s)$ and $M^D(X,s)$. Assume:*

1. *$T(X,z)$ is strictly increasing and continuous w.r.t. $z \in \mathbb{R}_{>0}$, with its inverse denoted by*

$R(X, s)$.

2. *The convergence interval $\mathbb{K} \triangleq \{T(X, z) | z \in \mathbb{R}_{>0}\}$ (image of $\mathbb{R}_{>0}$ under T) is X-invariant.*

3. *$M^U(X, s)$ and $M^D(X, s)$ are continuous and positive at $s \in \mathbb{K}$, satisfying $\frac{M^D(X,s)}{M^U(X,s)} = R(X, s)$.*

4. *Range of $\mathcal{F}$ is $\mathbb{K}$.*

Denote $\widetilde{M}^U(X, s)$ and $\widetilde{M}^D(X, s)$ to be antiderivatives of $M^U(X, s)$ and $M^D(X, s)$ at $s \in \mathbb{K}$, and construct the corresponding functional $L^{U \cap D}$. Then, the minima $f^* = \arg\inf_{f \in \mathcal{F}} L^{U \cap D}(f)$ will satisfy $\forall X \in \mathbb{S}^{U \cap D} : f^*(X) = T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$.

Continuity of $\{M^U, M^D\}$ in condition 3 is sufficient for existence of antiderivatives $\{\widetilde{M}^U, \widetilde{M}^D\}$. It is a little too strong criteria for integrability, yet it is more convenient to verify in practice. This leads to differentiability of $\widetilde{M}^U$, which in turn implies continuity and differentiability of G; positiveness of M^U in condition 3 implies G to be strictly increasing on $\mathbb{K}$. Conditions 2 and 4 restate assumptions over $\mathbb{K}$ and $\mathcal{F}$. Therefore, the sufficient conditions over $\{T, \mathbb{K}, G, \mathcal{F}\}$ follow from the above list, which leads to the required f^*.

Given any required convergence T, the above theorem can be applied to propose valid *magnitudes* $\{M^U, M^D\}$. This basically comes to requiring *magnitude* functions to be continuous and positive on $\mathbb{K}$, with their ratio being inverse of T. Once such pair of functions is obtained, the loss $L^{U \cap D}$ with corresponding optima, and more importantly its gradient, can be easily constructed. Observe that knowledge of $\{\widetilde{M}^U, \widetilde{M}^D\}$ is not necessary neither for condition verification nor for the optimization of $L^{U \cap D}$.

Further, given any $L^{U \cap D}$ with corresponding $\{M^U, M^D\}$, its convergence and sufficient conditions can be verified via Theorem 5.

Theorem 5 (Magnitudes-Focused). *Consider a functional $L^{U \cap D}$ with $\widetilde{M}^U(X, s)$ and $\widetilde{M}^D(X, s)$ whose derivatives are $M^U(X, s)$ and $M^D(X, s)$. Denote $R(X, s)$ to be the ratio $\frac{M^D(X,s)}{M^U(X,s)}$, and define a convergence interval as $\mathbb{K} \triangleq (s_{min}, s_{max})$. Assume:*

1. *$R(X, s) : \mathbb{R}^n \times \mathbb{K} \to \mathbb{R}_{>0}$ is continuous, strictly increasing and bijective w.r.t. domain $s \in \mathbb{K}$ and codomain $\mathbb{R}_{>0}$, for any $X \in \mathbb{R}^n$.*

2. *$M^U(X, s)$ and $M^D(X, s)$ are continuous and positive at $s \in \mathbb{K}$.*

3. *Range of $\mathcal{F}$ is $\mathbb{K}$.*

Then, the minima $f^* = \arg\inf_{f \in \mathcal{F}} L^{U \cap D}(f)$ will satisfy $\forall X \in \mathbb{S}^{U \cap D} : f^*(X) = T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$, where $T \triangleq R^{-1}$.

Condition sets in Theorem 4 and Theorem 5 are identical. Condition 1 of Theorem 5 is required for $T(X, z)$ to be strictly increasing, continuous and well-defined for each $z \in \mathbb{R}_{>0}$. $\mathbb{K}$ can be any interval as long as conditions of Theorem 5 are satisfied, yet typically it is a preimage

$\{s \in \mathbb{R} | R(X, s) \in \mathbb{R}_{>0}\}$ of $\mathbb{R}_{>0}$ under R. See examples in Section 5.1. Likewise, observe again that knowledge of PSO *primitives* $\{\widetilde{M}^U, \widetilde{M}^D\}$ is not required.

Below we will use Theorem 4 to derive valid $\{M^U, M^D\}$ for any considered T, and Theorem 5 to derive T for any considered $\{M^U, M^D\}$. Further, part 1 of Theorem 1 follows trivially from the above statements. Moreover, due to symmetry between *up* and *down* terms we can also have T and R to be strictly decreasing functions given $M^U(X, s)$ and $M^D(X, s)$ are negative at $s \in \mathbb{K}$. Furthermore, many objective functions satisfy the above theorems and thus can be recovered via PSO framework. Estimation methods for which the sufficient conditions do not hold include a hinge loss from the binary classification domain as also other threshold losses [93]. Yet, these losses can be shown to be included within PSO non-differentiable case in Section 4.1.1, whose analysis we leave for a future work.

4.1.3 Unlimited Range Conditions

Criteria 4 of Theorem 4 and 3 of Theorem 5 can be replaced by additional conditions over $\{M^U, M^D\}$. These derived below conditions are very often satisfied, which allows us to not restrict $\mathcal{F}$'s range in practice.

Recalling that $\mathbb{K}$ is an open interval (s_{min}, s_{max}), consider following sets $\mathbb{K}^- = \{s | s \leq s_{min}\}$ and $\mathbb{K}^+ = \{s | s \geq s_{max}\}$. Observe that if $s_{min} = -\infty$ then $\mathbb{K}^-$ is an empty set, and if $s_{max} = \infty$ - $\mathbb{K}^+$ is empty. Further, $\mathbb{K}$, $\mathbb{K}^-$ and $\mathbb{K}^+$ are disjoint sets.

To reduce limitation over $\mathcal{F}$'s range, it is enough to demand the inner optimization problem in Eq. (4.6) to be strictly decreasing on $\mathbb{K}^-$ and strictly increasing on $\mathbb{K}^+$. To this purpose, first we require $\{M^U, M^D\}$ to be well-defined on the entire real line $s \in \mathbb{R}$. This can be achieved by restricting $\widetilde{M}^U(X, s)$ and $\widetilde{M}^D(X, s)$ to be differentiable on $s \in \mathbb{R}$ - it is sufficient for $M^U(X, s)$ and $M^D(X, s)$ to be well-defined on $\mathbb{R}$. Alternatively, we may and will require $M^U(X, s)$ and $M^D(X, s)$ to be continuous at any $s \in \mathbb{R}$. This slightly stronger condition will ensure that $\{M^U, M^D\}$ are well-defined and that the antiderivatives $\{\widetilde{M}^U, \widetilde{M}^D\}$ exist on $\mathbb{R}$. Moreover, such condition is imposed over $\{M^U, M^D\}$, allowing to neglect properties of $\{\widetilde{M}^U, \widetilde{M}^D\}$.

Further, in case $\mathbb{K}^+$ is not empty, we require $-z \cdot \widetilde{M}^U(X, s) + \widetilde{M}^D(X, s)$ to be strictly increasing for any $s \in \mathbb{K}^+$ and any $z \in \mathbb{R}_{>0}$. Given that $\widetilde{M}^U(X, s)$ and $\widetilde{M}^D(X, s)$ are differentiable at $s \in \mathbb{K} \cup \mathbb{K}^+$ (which also implies their continuity at s_{max}), this requirement holds iff $\forall s \in \mathbb{K}^+, z \in \mathbb{R}_{>0} : M^D(X, s) > z \cdot M^U(X, s)$. Verifying all possible cases, the above criteria is satisfied iff: $\Big[\forall s \in \mathbb{K}^+ : [M^U(X, s) = 0 \vee M^D(X, s) > 0] \wedge [M^U(X, s) < 0 \vee M^D(X, s) \geq 0]\Big]$. This can be compactly written as $\forall s \in \mathbb{K}^+ : M^U(X, s) < M^D(X, s) \vee M^U(X, s) \cdot M^D(X, s) \leq 0$, where the second condition implies that *magnitudes* M^U and M^D can not have the same sign within $\mathbb{K}^+$.

Similar derivation for $\mathbb{K}^-$ will lead to demand the inner problem to be strictly decreasing for any $s \in \mathbb{K}^-$ and any $z \in \mathbb{R}_{>0}$. In turn, this leads to criteria $\forall s \in \mathbb{K}^- : M^U(X, s) > M^D(X, s) \vee M^U(X, s) \cdot M^D(X, s) \leq 0$. Below we summarize conditions under which no restrictions are required over $\mathcal{F}$'s range.

Theorem 6 (Unconstrained Function Range). *Consider the convergence interval $\mathbb{K} = (s_{min}, s_{max})$. Assume:*

1. *$M^U(X,s)$ and $M^D(X,s)$ are continuous on the entire real line $s \in \mathbb{R}$, and do not have the same sign outside of $\mathbb{K}$.*
2. *For any $s \leq s_{min}$: $M^U(X,s) > M^D(X,s)$.*
3. *For any $s \geq s_{max}$: $M^U(X,s) < M^D(X,s)$.*

Then, the condition 4 of Theorem 4 and the condition 3 of Theorem 5 can be removed.

Intuitively, conditions for $\mathbb{K}^+$ (and similarly for $\mathbb{K}^-$) can be interpreted as requiring *up* force $F_\theta^U(X)$ to be weaker than *down* force $F_\theta^D(X)$ for any ratio $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)} > 0$, once the surface height $f_\theta(X)$ got too high and exceeded the convergence interval $\mathbb{K}$. In Section 5 we will see that almost all PSO estimators satisfy Theorem 6.

4.2 Disjoint Support Optima

4.2.1 Area $\mathbb{S}^{U \setminus D}$

Consider the loss term $L^{U \setminus D}(f)$ corresponding to the area $\mathbb{S}^{U \setminus D}$, where $\mathbb{P}^U(X) > 0$ and $\mathbb{P}^D(X) = 0$. We are going to prove the below theorem (identical to part 2 of Theorem 1). The motivation behind this theorem is that in many PSO instances M^U satisfies one of its conditions. In such case the theorem can be applied to understand the PSO convergence behavior in the region $\mathbb{S}^{U \setminus D}$. Moreover, this theorem further supports the PSO framework's perspective, where virtual forces are pushing the model surface towards the physical equilibrium.

Theorem 7. *Define an arbitrary space $\mathcal{F}$ of functions from $\mathbb{S}^{U \setminus D}$ to $\mathbb{R}$, with $h \in \mathcal{F}$ being its element. Then, depending on properties of a function M^U, $f^* = \arg\inf_{f \in \mathcal{F}} L^{U \setminus D}(f)$ must satisfy:*

1. *If $\forall s \in \mathbb{R} : M^U(X,s) > 0$, then $f^*(X) = \infty$.*
2. *If $\forall s \in \mathbb{R} : M^U(X,s) < 0$, then $f^*(X) = -\infty$.*
3. *If $\forall s \in \mathbb{R} : M^U(X,s) \equiv 0$, then $f^*(X)$ can be arbitrary.*
4. *If $\forall s \in \mathbb{R}$:*
$$M^U(X,s) \to \begin{cases} = 0, & s = h(X) \\ > 0, & s < h(X) \\ < 0, & s > h(X) \end{cases} \tag{4.8}$$
   then $f^*(X) = h(X)$.
5. *Otherwise, additional analysis is required.*

Proof. The inner problem solved by $\inf_{f \in \mathcal{F}} L^{U\setminus D}(f)$ for each $X \in \mathbb{S}^{U\setminus D}$ is:

$$s^* = \arg\inf_{s \in \mathbb{R}} -z \cdot \widetilde{M}^U(X, s), \tag{4.9}$$

where $z \in \mathbb{R}_{>0}$. Given $\widetilde{M}^U(X, s)$ is differentiable, a derivative of the inner cost is $-z \cdot M^U(X, s)$. If the inner cost is a strictly decreasing function of s, $\forall s \in \mathbb{R} : M^U(X, s) > 0$, then the infimum is $\inf_{s \in \mathbb{R}} -z \cdot \widetilde{M}^U(X, s) = -\infty$ and $s^* = \infty$ - the inner cost will be lower for the bigger value of s. This leads to the entry 1 of the theorem. Similarly, if the cost is a strictly increasing function, $\forall s \in \mathbb{R} : M^U(X, s) < 0$, then the infimum is achieved at $s^* = -\infty$, yielding the entry 2.

If $\forall s \in \mathbb{R} : M^U(X, s) \equiv 0$, then the inner cost is constant. In such case the infimum is obtained at any $s \in \mathbb{R}$, hence the corresponding $f^*(X)$ can be arbitrary (the entry 3).

Further, denote $s' \equiv h(X)$. Conditions of the entry 4 imply that the inner cost in Eq. (4.9) is strictly decreasing at $s < s'$ and strictly increasing at $s > s'$. Since it is also continuous (consequence of being differentiable), its infimum must be at $s^* = s'$. Thus, we have the entry 4: $f^*(X) = s' = h(X)$.

Otherwise, if $M^U(X, s)$ does not satisfy any of the theorem's properties $(1) - (4)$, a further analysis of this particular *magnitude* function in the context of $L^{U\setminus D}$ needs to be done. ■

4.2.2 Area $\mathbb{S}^{D\setminus U}$

Consider the loss term $L^{D\setminus U}(f)$ corresponding to the area $\mathbb{S}^{D\setminus U}$, where $\mathbb{P}^U(X) = 0$ and $\mathbb{P}^D(X) > 0$. The below theorem explains PSO convergence in this area.

Theorem 8. *Define an arbitrary space $\mathcal{F}$ of functions from $\mathbb{S}^{D\setminus U}$ to $\mathbb{R}$, with $h \in \mathcal{F}$ being its element. Then, depending on properties of a function M^D, $f^* = \arg\inf_{f \in \mathcal{F}} L^{D\setminus U}(f)$ must satisfy:*

1. *If $\forall s \in \mathbb{R} : M^D(X, s) > 0$, then $f^*(X) = -\infty$.*
2. *If $\forall s \in \mathbb{R} : M^D(X, s) < 0$, then $f^*(X) = \infty$.*
3. *If $\forall s \in \mathbb{R} : M^D(X, s) \equiv 0$, then $f^*(X)$ can be arbitrary.*
4. *If $\forall s \in \mathbb{R}$:*

$$M^D(X, s) \to \begin{cases} = 0, & s = h(X) \\ > 0, & s > h(X) \\ < 0, & s < h(X) \end{cases} \tag{4.10}$$

   then $f^*(X) = h(X)$.

5. *Otherwise, additional analysis is required.*

The proof of Theorem 8 is symmetric to the proof of Theorem 7 and hence omitted.

CHAPTER 5

Instances of PSO

Many statistical methods exist whose loss and gradient have PSO forms depicted in Eq. (3.3) and Eq. (3.5) for some choice of densities $\mathbb{P}^U$ and $\mathbb{P}^D$, and of functions $\widetilde{M}^U$, $\widetilde{M}^D$, M^U and M^D, and therefore being instances of the PSO algorithm family. Typically, these methods defined via their loss which involves the pair of *primitives* $\{\widetilde{M}^U[X, s], \widetilde{M}^D[X, s]\}$. Yet, in practice it is enough to know their derivatives $\{M^U[X, s] = \frac{\partial \widetilde{M}^U(X,s)}{\partial s}, M^D[X, s] = \frac{\partial \widetilde{M}^D(X,s)}{\partial s}\}$ for the gradient-based optimization (see Algorithm 3.1). Therefore, PSO formulation focuses directly on $\{M^U, M^D\}$, with each PSO instance being defined by a particular choice of this pair.

Moreover, most of the existing PSO instances and subgroups actually require $\widetilde{M}^U$ and $\widetilde{M}^D$ to be analytically known, while PSO composition eliminates such demand. In fact, many pairs $\{M^U, M^D\}$ explored in this thesis do not have closed-form known antiderivatives $\{\widetilde{M}^U, \widetilde{M}^D\}$. Thus, PSO enriches the arsenal of available probabilistic methods.

In Tables 5.1-5.5 we show multiple PSO instances. We categorize all losses into two main categories - density estimation losses in Tables 5.1-5.2 and ratio density estimation losses in Tables 5.3-5.5. In the former class of losses we are interested to infer density $\mathbb{P}^U$ from its available data samples, while $\mathbb{P}^D$ represents some auxiliary distribution with analytically known pdf function $\mathbb{P}^D(X)$ whose samples are used to create the opposite force $F_\theta^D(X)$; this force will balance the force $F_\theta^U(X)$ from $\mathbb{P}^U$'s samples. Further, in the latter class we concerned to learn a density ratio, or some function of it, between two unknown densities $\mathbb{P}^U$ and $\mathbb{P}^D$ by using the available samples from both distributions.

In the tables we present the PSO loss of each method, if analytically known, and the corresponding pair $\{M^U, M^D\}$. We also indicate to what the surface $f(X)$ will converge assuming that PSO *balance state* in Eq. (3.2) was obtained. Derivation of this convergence appears below. Importantly, it describes the optimal PSO solution only within the area $\mathbb{S}^{U \cap D} \subset \mathbb{R}^n$. For X in $\mathbb{S}^{U \setminus D}$ or $\mathbb{S}^{D \setminus U}$, the convergence can be explained via theorems 7 and 8 respectively. Yet, in most of the thesis we will limit our discussion to the convergence within the mutual support, implicitly assuming $\mathbb{S}^U \equiv \mathbb{S}^D$.

Method	***Final*** $f_\theta(X)$ **and** $\mathbb{K}$ **/ *References* / *Loss* /** $M^U(\cdot)$ ***and*** $M^D(\cdot)$
DeepPDF	F: $\mathbb{P}^U(X), \mathbb{K} = \mathbb{R}_{>0}$ R: [10, 61] L: $-\mathbb{E}_{X\sim\mathbb{P}^U} f_\theta(X)\cdot\mathbb{P}^D(X) + \mathbb{E}_{X\sim\mathbb{P}^D} \frac{1}{2}\Big[f_\theta(X)\Big]^2$ M^U, M^D: $\mathbb{P}^D(X)$, $f_\theta(X)$
PSO-LDE (Log Density Estimators)	F: $\log\mathbb{P}^U(X), \mathbb{K} = \mathbb{R}$ R: Introduced and thoroughly analyzed in this thesis, see Section 8.2 L: unknown M^U, M^D: $\frac{\mathbb{P}^D(X)}{\big[[\exp f_\theta(X)]^\alpha + [\mathbb{P}^D(X)]^\alpha\big]^{\frac{1}{\alpha}}}$, $\frac{\exp f_\theta(X)}{\big[[\exp f_\theta(X)]^\alpha + [\mathbb{P}^D(X)]^\alpha\big]^{\frac{1}{\alpha}}}$ where α is a hyper-parameter
PSO-MAX	F: $\log\mathbb{P}^U(X), \mathbb{K} = \mathbb{R}$ R: This thesis, see Section 13.2.1 L: unknown M^U, M^D: $\exp\left[-\max\left[f_\theta(X) - \log\mathbb{P}^D(X), 0\right]\right]$, $\exp\left[\min\left[f_\theta(X) - \log\mathbb{P}^D(X), 0\right]\right]$
NCE (Noise Contrastive Estimation)	F: $\log\mathbb{P}^U(X), \mathbb{K} = \mathbb{R}$ R: [39, 126]; [84, 105]; [83] [40] L: $\mathbb{E}_{X\sim\mathbb{P}^U} \log\frac{\exp[f_\theta(X)]+\mathbb{P}^D(X)}{\exp[f_\theta(X)]} + \mathbb{E}_{X\sim\mathbb{P}^D} \log\frac{\exp[f_\theta(X)]+\mathbb{P}^D(X)}{\mathbb{P}^D(X)}$ M^U, M^D: $\frac{\mathbb{P}^D(X)}{\exp[f_\theta(X)]+\mathbb{P}^D(X)}$, $\frac{\exp[f_\theta(X)]}{\exp[f_\theta(X)]+\mathbb{P}^D(X)}$
IS (Importance Sampling)	F: $\log\mathbb{P}^U(X), \mathbb{K} = \mathbb{R}$ R: [105] L: $-\mathbb{E}_{X\sim\mathbb{P}^U} f_\theta(X) + \mathbb{E}_{X\sim\mathbb{P}^D} \frac{\exp[f_\theta(X)]}{\mathbb{P}^D(X)}$ M^U, M^D: 1, $\frac{\exp[f_\theta(X)]}{\mathbb{P}^D(X)}$

Table 5.1: PSO Instances For Density Estimation, Part 1

Method	***Final*** $f_\theta(X)$ **and** $\mathbb{K}$ **/** ***References*** **/** ***Loss*** **/** $M^U(\cdot)$ ***and*** $M^D(\cdot)$
Polynomial	F: $\log \mathbb{P}^U(X)$, $\mathbb{K} = \mathbb{R}$
	R: [105]
	L: $-\mathbb{E}_{X\sim\mathbb{P}^U} \frac{\exp[f_\theta(X)]}{\mathbb{P}^D(X)} + \mathbb{E}_{X\sim\mathbb{P}^D} \frac{1}{2} \frac{\exp[2\cdot f_\theta(X)]}{[\mathbb{P}^D(X)]^2}$
	M^U, M^D: $\frac{\exp[f_\theta(X)]}{\mathbb{P}^D(X)}$, $\frac{\exp[2\cdot f_\theta(X)]}{[\mathbb{P}^D(X)]^2}$
Inverse Polynomial	F: $\log \mathbb{P}^U(X)$, $\mathbb{K} = \mathbb{R}$
	R: [105]
	L: $\mathbb{E}_{X\sim\mathbb{P}^U} \frac{1}{2} \frac{[\mathbb{P}^D(X)]^2}{\exp[2\cdot f_\theta(X)]} - \mathbb{E}_{X\sim\mathbb{P}^D} \frac{\mathbb{P}^D(X)}{\exp[f_\theta(X)]}$
	M^U, M^D: $\frac{[\mathbb{P}^D(X)]^2}{\exp[2\cdot f_\theta(X)]}$, $\frac{\mathbb{P}^D(X)}{\exp[f_\theta(X)]}$
Inverse Importance Sampling	F: $\log \mathbb{P}^U(X)$, $\mathbb{K} = \mathbb{R}$
	R: [105]
	L: $\mathbb{E}_{X\sim\mathbb{P}^U} \frac{\mathbb{P}^D(X)}{\exp[f_\theta(X)]} + \mathbb{E}_{X\sim\mathbb{P}^D} f_\theta(X)$
	M^U, M^D: $\frac{\mathbb{P}^D(X^U)}{\exp[f_\theta(X^U)]}$, 1
Root Density Estimation	F: $\sqrt[d]{\mathbb{P}^U(X)}$, $\mathbb{K} = \mathbb{R}_{>0}$
	R: This thesis
	L: $-\mathbb{E}_{X\sim\mathbb{P}^U} f_\theta(X) \cdot \mathbb{P}^D(X) + \mathbb{E}_{X\sim\mathbb{P}^D} \frac{1}{d+1} \cdot \lvert f_\theta(X)\rvert^{d+1}$
	M^U, M^D: $\mathbb{P}^D(X)$, $\lvert f_\theta(X)\rvert^d \cdot sign[f_\theta(X)]$
PDF Convolution Estimation	F: $\log \big[[\mathbb{P}^U * \mathbb{P}^\upsilon](X)\big]$, $\mathbb{K} = \mathbb{R}$
	R: This thesis
	L: $-\mathbb{E}_{X\sim\bar{\mathbb{P}}^U} f_\theta(X) + \mathbb{E}_{X\sim\mathbb{P}^D} \frac{\exp[f_\theta(X)]}{\mathbb{P}^D(X)}$
	M^U, M^D: 1, $\frac{\exp[f_\theta(X)]}{\mathbb{P}^D(X)}$
	where $*$ is a convolution operator and
	$\bar{\mathbb{P}}^U(X) = \mathbb{P}^U(X) * \mathbb{P}^\upsilon(X)$ serves as *up* density,
	whose sample $X \sim \bar{\mathbb{P}}^U$ can be obtained via $X = X^U + \upsilon$
	with $X^U \sim \mathbb{P}^U(X)$ and $\upsilon \sim \mathbb{P}^\upsilon(X)$, see Section 12

Table 5.2: PSO Instances For Density Estimation, Part 2

Method	***F**inal* $f_\theta(X)$ **and** $\mathbb{K}$ **/ *R*eferences / *L*oss /** $M^U(\cdot)$ ***and*** $M^D(\cdot)$
"Unit" Loss	F: Kantorovich potential [142], only if the smoothness of $f_\theta(X)$ is heavily restricted R: see Section 10.3 L: $-\mathbb{E}_{X\sim\mathbb{P}^U} f_\theta(X) + \mathbb{E}_{X\sim\mathbb{P}^D} f_\theta(X)$ M^U, M^D: $1, 1$
EBGAN Critic	F: $f_\theta(X) = m$ at $\{X : \mathbb{P}^U(X) < \mathbb{P}^D(X)\}$, and $f_\theta(X) = 0$ otherwise R: [153], see Section 7.6 L: $\mathbb{E}_{X\sim\mathbb{P}^U} f_\theta(X) + \mathbb{E}_{X\sim\mathbb{P}^D} \max[m - f_\theta(X), 0]$ M^U, M^D: $-1, -cut\_at\,[X, f_\theta(X), m]$ where the considered model $f_\theta(X)$ is constrained to have non-negative outputs
uLSIF	F: $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}, \mathbb{K} = \mathbb{R}_{>0}$ R: [56, 129, 150]; [88, 138] L: $-\mathbb{E}_{X\sim\mathbb{P}^U} f_\theta(X) + \mathbb{E}_{X\sim\mathbb{P}^D} \frac{1}{2}\big[f_\theta(X)\big]^2$ M^U, M^D: 1, $f_\theta(X)$
KLIEP	F: $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}, \mathbb{K} = \mathbb{R}_{>0}$ R: [130, 131, 138] L: $-\mathbb{E}_{X\sim\mathbb{P}^U} \log f_\theta(X) + \mathbb{E}_{X\sim\mathbb{P}^D}[f_\theta(X) - 1]$ M^U, M^D: $\frac{1}{f_\theta(X)}, 1$
Classical GAN Critic	F: $\frac{\mathbb{P}^U(X)}{\mathbb{P}^U(X)+\mathbb{P}^D(X)}, \mathbb{K} = (0, 1)$ R: [32] L: $-\mathbb{E}_{X\sim\mathbb{P}^U} \log f_\theta(X) - \mathbb{E}_{X\sim\mathbb{P}^D} \log \Big[1 - f_\theta(X)\Big]$ M^U, M^D: $\frac{1}{f_\theta(X)}, \frac{1}{1-f_\theta(X)}$ * for $f_\theta(X) = sigmoid(h_\theta(X))$ this loss is identical to Logistic Loss in Table 5.4
NDMR (Noise-Data Mixture Ratio)	F: $\frac{\mathbb{P}^U(X)}{\mathbb{P}^U(X)+\mathbb{P}^D(X)}, \mathbb{K} = (0, 1)$ R: This thesis L: $-\mathbb{E}_{X\sim\mathbb{P}^U} f_\theta(X) + \mathbb{E}_{X\sim\mathbb{P}^M} f_\theta(X)^2$ M^U, M^D: $1, 2f_\theta(X)$ where $\mathbb{P}^M(X) = \frac{1}{2}\mathbb{P}^U(X) + \frac{1}{2}\mathbb{P}^D(X)$ serves as *down* density, instead of density $\mathbb{P}^D(X)$
NDMLR (Noise-Data Mixture Log-Ratio)	F: $\log \frac{\mathbb{P}^U(X)}{\mathbb{P}^U(X)+\mathbb{P}^D(X)}, \mathbb{K} = \mathbb{R}_{<0}$ R: This thesis L: $-\mathbb{E}_{X\sim\mathbb{P}^U} f_\theta(X) + \mathbb{E}_{X\sim\mathbb{P}^M} 2\exp[f_\theta(X)]$ M^U, M^D: $1, 2\exp[f_\theta(X)]$ where $\mathbb{P}^M(X) = \frac{1}{2}\mathbb{P}^U(X) + \frac{1}{2}\mathbb{P}^D(X)$ serves as *down* density, instead of density $\mathbb{P}^D(X)$

Table 5.3: PSO Instances For Density Ratio Estimation, Part 1

Method	***F**inal* $f_\theta(X)$ **and** $\mathbb{K}$ **/** ***R**eferences* **/** ***L**oss* **/** $M^U(\cdot)$ ***and*** $M^D(\cdot)$
Classical GAN Critic on log-scale	F: $\log \frac{\mathbb{P}^U(X)}{\mathbb{P}^U(X)+\mathbb{P}^D(X)}$, $\mathbb{K} = \mathbb{R}_{<0}$ R: This thesis L: $\mathbb{E}_{X\sim\mathbb{P}^U} \frac{1}{\exp[f_\theta(X)]} - \mathbb{E}_{X\sim\mathbb{P}^D} \log \frac{\exp[f_\theta(X)]}{1-\exp[f_\theta(X)]}$ M^U, M^D: $\frac{1}{\exp[f_\theta(X)]}$, $\frac{1}{1-\exp[f_\theta(X)]}$
Power Divergence Ratio Estimation	F: $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$, $\mathbb{K} = \mathbb{R}_{>0}$ R: $[80, 130]$ L: $-\mathbb{E}_{X\sim\mathbb{P}^U} \frac{f_\theta(X)^\alpha}{\alpha} + \mathbb{E}_{X\sim\mathbb{P}^D} \frac{f_\theta(X)^{\alpha+1}}{\alpha+1}$ M^U, M^D: $f_\theta(X)^{\alpha-1}$, $f_\theta(X)^\alpha$
Reversed KL	F: $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$, $\mathbb{K} = \mathbb{R}_{>0}$ R: [138] L: $\mathbb{E}_{X\sim\mathbb{P}^U} \frac{1}{f_\theta(X)} + \mathbb{E}_{X\sim\mathbb{P}^D} \log f_\theta(X)$ M^U, M^D: $\frac{1}{[f_\theta(X)]^2}$, $\frac{1}{f_\theta(X)}$
Balanced Density Ratio	F: $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$, $\mathbb{K} = \mathbb{R}_{>0}$ R: This thesis L: $-\mathbb{E}_{X\sim\mathbb{P}^U} \log[f_\theta(X)+1] + \mathbb{E}_{X\sim\mathbb{P}^D} f_\theta(X) - \log[f_\theta(X)+1]$ M^U, M^D: $\frac{1}{f_\theta(X)+1}$, $\frac{f_\theta(X)}{f_\theta(X)+1}$
Log-density Ratio	F: $\log \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$, $\mathbb{K} = \mathbb{R}$ R: This thesis L: $-\mathbb{E}_{X\sim\mathbb{P}^U} f_\theta(X) + \mathbb{E}_{X\sim\mathbb{P}^D} \exp[f_\theta(X)]$ M^U, M^D: 1, $\exp[f_\theta(X)]$
Square Loss	F: $\frac{\mathbb{P}^U(X)-\mathbb{P}^D(X)}{\mathbb{P}^U(X)+\mathbb{P}^D(X)}$, $\mathbb{K} = (-1, 1)$ R: [80] L: $\mathbb{E}_{X\sim\mathbb{P}^U} \frac{1}{2}[1-f_\theta(X)]^2 + \mathbb{E}_{X\sim\mathbb{P}^D} \frac{1}{2}[1+f_\theta(X)]^2$ M^U, M^D: $1-f_\theta(X)$, $1+f_\theta(X)$
Logistic Loss	F: $\log \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$, $\mathbb{K} = \mathbb{R}$ R: [80] L: $\mathbb{E}_{X\sim\mathbb{P}^U} \log\left[1+\exp[-f_\theta(X)]\right] + \mathbb{E}_{X\sim\mathbb{P}^D} \log\left[1+\exp[f_\theta(X)]\right]$ M^U, M^D: $\frac{1}{\exp[f_\theta(X)]+1}$, $\frac{1}{\exp[-f_\theta(X)]+1}$

Table 5.4: PSO Instances For Density Ratio Estimation, Part 2

Method	***F̲inal*** $f_\theta(X)$ **and** $\mathbb{K}$ **/ *R̲eferences* / *L̲oss* /** $M^U(\cdot)$ ***and*** $M^D(\cdot)$
Exponential Loss	F̲: $\frac{1}{2}\log\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$, $\mathbb{K}=\mathbb{R}$ R̲: [80] L̲: $\mathbb{E}_{X\sim\mathbb{P}^U}\exp[-f_\theta(X)]+\mathbb{E}_{X\sim\mathbb{P}^D}\exp[f_\theta(X)]$ M^U, M^D: $\exp[-f_\theta(X)]$, $\exp[f_\theta(X)]$
LSGAN Critic	F̲: $\frac{b\cdot\mathbb{P}^U(X)+a\cdot\mathbb{P}^D(X)}{\mathbb{P}^U(X)+\mathbb{P}^D(X)}$, $\mathbb{K}=(\min[a,b],\max[a,b])$ R̲: [76] L̲: $\mathbb{E}_{X\sim\mathbb{P}^U}\frac{1}{2}[f_\theta(X)-b]^2+\mathbb{E}_{X\sim\mathbb{P}^D}\frac{1}{2}[f_\theta(X)-a]^2$ M^U, M^D: $b-f_\theta(X)$, $f_\theta(X)-a$
Kullback-Leibler Divergence	F̲: $1+\log\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$, $\mathbb{K}=\mathbb{R}$ R̲: [94] L̲: $-\mathbb{E}_{X\sim\mathbb{P}^U}f_\theta(X)+\mathbb{E}_{X\sim\mathbb{P}^D}\exp[f_\theta(X)-1]$ M^U, M^D: 1, $\exp[f_\theta(X)-1]$
Reverse KL Divergence	F̲: $-\frac{\mathbb{P}^D(X)}{\mathbb{P}^U(X)}$, $\mathbb{K}=\mathbb{R}_{<0}$ R̲: [94] L̲: $-\mathbb{E}_{X\sim\mathbb{P}^U}f_\theta(X)+\mathbb{E}_{X\sim\mathbb{P}^D}[-1-\log[-f_\theta(X)]]$ M^U, M^D: 1, $\frac{1}{-f_\theta(X)}$
Lipschitz Continuity Objective	F̲: $\frac{1}{2}\cdot\frac{\mathbb{P}^U(X)-\mathbb{P}^D(X)}{\sqrt{\mathbb{P}^U(X)\cdot\mathbb{P}^D(X)}}$, $\mathbb{K}=\mathbb{R}$ R̲: [154] L̲: $-\mathbb{E}_{X\sim\mathbb{P}^U}\left[f_\theta(X)-\sqrt{f_\theta(X)^2+1}\right]+\mathbb{E}_{X\sim\mathbb{P}^D}\left[f_\theta(X)+\sqrt{f_\theta(X)^2+1}\right]$ M^U, M^D: $1-\frac{f_\theta(X)}{\sqrt{f_\theta(X)^2+1}}$, $1+\frac{f_\theta(X)}{\sqrt{f_\theta(X)^2+1}}$
LDAR (Log-density Atan-Ratio)	F̲: $\arctan\log\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$, $\mathbb{K}=(-\frac{\pi}{2},\frac{\pi}{2})$ R̲: This thesis L̲: unknown M^U, M^D: $\frac{1}{\exp[\tan f_\theta(X)]+1}$, $\frac{1}{\exp[-\tan f_\theta(X)]+1}$
LDTR (Log-density Tanh-Ratio)	F̲: $\tanh\log\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$, $\mathbb{K}=(-1,1)$ R̲: This thesis L̲: $\mathbb{E}_{X\sim\mathbb{P}^U}\frac{2}{3}\cdot[1-f_\theta(X)]^{\frac{3}{2}}+\mathbb{E}_{X\sim\mathbb{P}^D}\frac{2}{3}\cdot[1+f_\theta(X)]^{\frac{3}{2}}$ M^U, M^D: $\sqrt{1-f_\theta(X)}$, $\sqrt{1+f_\theta(X)}$

Table 5.5: PSO Instances For Density Ratio Estimation, Part 3

5.1 Deriving Convergence of PSO Instance

Given a PSO instance with a particular $\{\mathbb{P}^U, \mathbb{P}^D, M^U, M^D\}$, the convergence within $\mathbb{S}^{U \cap D}$ can be derived by solving PSO *balance state* $\mathbb{P}^U(X) \cdot M^U[X, f^*(X)] = \mathbb{P}^D(X) \cdot M^D[X, f^*(X)]$.

Example 1: From Table 5.1 we can see that IS method has $M^U[X, f(X)] = 1$ and $M^D[X, f(X)] = \frac{\exp[f(X)]}{\mathbb{P}^D(X)}$. Given that samples within the loss have densities $X^U \sim \mathbb{P}^U(X)$ and $X^D \sim \mathbb{P}^D(X)$, we can substitute the sample densities and the *magnitude* functions $\{M^U, M^D\}$ into Eq. (3.2) to get:

$$\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)} = \frac{\exp[f^*(X)]/\mathbb{P}^D(X)}{1} \quad \Rightarrow \quad f^*(X) = \log \mathbb{P}^U(X), \tag{5.1}$$

where we use an equality between density ratio of the samples and ratio of *magnitude* functions to derive the final $f^*(X)$. Thus, in case of IS approach, the surface will converge to the log-density $\log \mathbb{P}^U(X)$.

More formally, we can derive PSO convergence according to definitions of Theorem 5, using *magnitude* ratio R and its inverse T. The theorem allows us additionally to verify sufficient conditions required by PSO framework. Furthermore, we can decide whether the restriction of $\mathcal{F}$'s range is necessarily by testing criteria of Theorem 6.

Example 2: Consider the "Polynomial" method in Table 5.2, with $M^U[X, f(X)] = \frac{\exp[f(X)]}{\mathbb{P}^D(X)}$ and $M^D[X, f(X)] = \frac{\exp[2f(X)]}{[\mathbb{P}^D(X)]^2}$. Then, we have $\frac{M^D[X, f(X)]}{M^U[X, f(X)]} = \frac{\exp[f(X)]}{\mathbb{P}^D(X)}$ and hence $R(X, s) = \frac{\exp s}{\mathbb{P}^D(X)}$. Further, consider the convergence interval $\mathbb{K}$ to be entire $\mathbb{R}$. Both conditions 1 and 2 of Theorem 5 are satisfied - R is continuous, strictly increasing and bijective w.r.t. domain $\mathbb{R}$ and codomain $\mathbb{R}_{>0}$, and both *magnitudes* are continuous and positive on the entire $\mathbb{R}$. Additionally, $\mathcal{F}$'s range need not to be restricted since $\mathbb{K} \equiv \mathbb{R}$. Further, R has a simple form and its invert is merely $T(X, z) = \log \mathbb{P}^D(X) + \log z$. The above T and R are inverse of each other w.r.t. the second argument, which can be easily verified. Next, we can calculate PSO convergence as $f^*(X) = T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right] = \log \mathbb{P}^D(X) + \log \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)} = \log \mathbb{P}^U(X)$. Hence, "Polynomial" method converges to $\log \mathbb{P}^U(X)$.

Example 3: Consider the LDAR method in Table 5.5, with $M^U[X, f(X)] = \frac{1}{\exp[\tan f(X)]+1}$ and $M^D[X, f(X)] = \frac{1}{\exp[-\tan f(X)]+1}$. Then, $\frac{M^D(X, f(X))}{M^U(X, f(X))} = \frac{\exp[\tan f(X)]+1}{\exp[-\tan f(X)]+1}$ and hence $R(X, s) = \frac{\exp[\tan s]+1}{\exp[-\tan s]+1}$. Function $R(X, s)$ is not bijective w.r.t. $s \in \mathbb{R}$ - it has multiple positive increasing copies on each interval $(\pi k - \frac{\pi}{2}, \pi k + \frac{\pi}{2})$. Hence, it does not satisfy the necessary conditions. Yet, if we restrict it to a domain $(\pi k - \frac{\pi}{2}, \pi k + \frac{\pi}{2})$ for any $k \in \mathbb{Z}$, then Theorem 5 will hold. Particularly, if we choose $\mathbb{K} = (-\frac{\pi}{2}, \frac{\pi}{2})$ then all theorem's conditions are satisfied. Moreover, Theorem 6 is not applicable here since *magnitudes* are not defined at points $s \in \{\frac{\pi}{2} k\}$. Therefore, we are required to limit range of $\mathcal{F}$ to be $\mathbb{K}$. The inverse of R for the considered $\mathbb{K}$ is $T(X, z) = \arctan \log z$. Hence, LDAR converges to $\arctan \log \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$.

5.2 Deriving New PSO Instance

In order to apply PSO for learning any function of X and $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$, the appropriate PSO instance can be derived via Theorem 4. Denote the required PSO convergence by a transformation $T(X, z) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ s.t. $f^*(X) = T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$ is the function we want to learn. Then according to the theorem, any pair $\{M^U, M^D\}$ whose ratio $R \equiv \frac{M^D}{M^U}$ satisfies $R \equiv T^{-1}$ (i.e. inverses between $\mathbb{K}$ and $\mathbb{R}_{>0}$), will produce the required convergence, given that theorem's conditions hold. Further, if conditions of Theorem 6 likewise hold, then no range restriction over f is needed.

Therefore, to learn any function $T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$, first we obtain $R(X, s)$ by finding an inverse of $T(X, z)$ w.r.t. z. Any valid pair of *magnitudes* satisfying $\frac{M^D(X,s)}{M^U(X,s)} = R[X, s]$ will produce the desired convergence. For example, we can use a straightforward choice $M^D[X, s] = R[X, s]$ and $M^U[X, s] = 1$ in order to converge to the aimed target. Such choice corresponds to minimizing f-divergence [92, 94], see Section 6 for details. Yet, typically these *magnitude* functions will be sub-optimal if for example M^D is an unbounded function. When this is the case, we can derive a new pair of bounded *magnitude* functions by multiplying the old pair by the same factor $q[X, s]$ (see also Section 7.2).

Example 4: Consider a scenario where we would like to infer $f^*(X) = \frac{\mathbb{P}^U(X) - \mathbb{P}^D(X)}{\mathbb{P}^U(X) + \mathbb{P}^D(X)}$, similarly to "Square Loss" method in Table 5.4. Treating only points $X \in \mathbb{S}^D$, we can rewrite our objective as $f^*(X) = \frac{\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)} - 1}{\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)} + 1}$ and hence the required PSO convergence is given by $T(X, z) = \frac{z-1}{z+1}$. Further, its inverse function is given by $R(X, s) = \frac{1+s}{1-s}$. Therefore, *magnitude* functions must satisfy $\frac{M^D(X,s)}{M^U(X,s)} = \frac{1+s}{1-s}$. One choice for such *magnitudes* is $M^U[X, s] = 1 - s$ and $M^D[X, s] = 1 + s$, just like in the "Square Loss" method [80]. Note that the convergence interval of this PSO instance is $\mathbb{K} = (-1, 1)$ which is X-invariant. Further, T is strictly increasing and continuous at $z \in \mathbb{R}_{>0}$ and $\{M^U, M^D\}$ are continuous and positive at $s \in \mathbb{K}$, hence satisfying the conditions of Theorem 4. Moreover, $\{M^U, M^D\}$ are actually continuous on entire $s \in \mathbb{R}$, with $\forall s \leq -1$: $M^U(X, s) > M^D(X, s)$ and $\forall s \geq 1$: $M^U(X, s) < M^D(X, s)$. Since M^U and M^D do not have the same sign outside of $\mathbb{K}$, conditions of Theorem 6 are likewise satisfied and the $\mathcal{F}$'s range can be the entire $\mathbb{R}$. Furthermore, other variants with the same PSO *balance state* can be easily constructed. For instance, we can use $M^U[X, f(X)] = \frac{1-f(X)}{D(X,f(X))}$ and $M^D[X, f(X)] = \frac{1+f(X)}{D(X,f(X))}$ with $D(X, f(X)) \triangleq |1 - f(X)| + |1 + f(X)|$ instead. Such normalization by function $D(\cdot)$ does not change the convergence, yet it produces bounded *magnitude* functions that are typically more stable during the optimization. All the required conditions are satisfied also by these normalized *magnitudes*. Additionally, recall that we considered only points within support of $\mathbb{P}^D(X)$. Outside of this support, any $X \in \mathbb{S}^{U \setminus D}$ will push the model surface $f(X)$ according to the rules implied by M^U; note also that M^U changes signs at $f(X) = 1$, with force always directed toward the height $h(X) = 1$. Therefore, at points $\{X \in \mathbb{S}^{U \setminus D}\}$ the convergence will be $f^*(X) = 1$, which is also a corollary of the condition 4 in Theorem

Description	**Target Function**	$T(X,z)$	$R(X,s)$	$\mathbb{K}$
Density-Ratio Estimation	$\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$	z	s	$\mathbb{R}_{>0}$
Log-Density-Ratio Estimation	$\log\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$	$\log z$	$\exp s$	$\mathbb{R}$
Density Estimation	$\mathbb{P}^U(X)$	$\mathbb{P}^D(X)\cdot z$	$\frac{s}{\mathbb{P}^D(X)}$	$\mathbb{R}_{>0}$
Log-Density Estimation	$\log\mathbb{P}^U(X)$	$\log\mathbb{P}^D(X)+\log z$	$\frac{\exp s}{\mathbb{P}^D(X)}$	$\mathbb{R}$

Table 5.6: Common target functions, their corresponding T and R mappings, and the convergence interval $\mathbb{K}$. Note that for density estimation methods (2 last cases) the auxiliary pdf $\mathbb{P}^D(X)$ is known analytically.

7. Finally at points outside of both supports there is no optimization performed, and hence theoretically nothing moves there - no constraints are applied on the surface f in these areas. Yet, in practice $f(X)$ at $X \notin \mathbb{S}^{U\cup D}$ will be affected by pushes at the training points, according to the elasticity properties of the model expressed via kernel $g_\theta(X,X')$ (see Section 7.5 for details).

In Table 5.6 we present transformations T and R for inference of several common target functions. As shown, if $\mathbb{P}^D(X)$ is analytically known, Theorem 4 can be used to also infer any function of density $\mathbb{P}^U(X)$, by multiplying z argument by $\mathbb{P}^D(X)$ inside T. Thus, we can apply the theorem to derive a new PSO instances for pdf estimation, and to mechanically recover many already existing such techniques. In Section 8.2 we will investigate new methods provided by the theorem for the estimation of log-density $\log\mathbb{P}^U(X)$.

Remark 9. *The inverse relation $R \equiv T^{-1}$ and properties of R and T described in theorems 4 and 5 imply that antiderivatives $\widetilde{R}[X,s] \triangleq \int_{s_0}^{s} R(X,t)dt$ and $\widetilde{T}[X,z] \triangleq \int_{z_0}^{z} R(X,t)dt$ are Legendre-Fenchel transforms of each other. Such a connection reminds the relation between Langrangian and Hamiltonian mechanics, and opens a bridge between control and learning theories. A detailed exploration of this connection is an additional interesting direction for future research.*

Further, density of *up* and *down* sample points within PSO loss can be changed from the described above choice $\mathbb{P}^U$ and $\mathbb{P}^D$, to infer other target functions. For example, in NDMR method from Table 5.3 instead of $\mathbb{P}^D(X)$ we sample $\frac{1}{2}\mathbb{P}^U(X)+\frac{1}{2}\mathbb{P}^D(X)$ to construct training dataset of *down* points (denoted by $\{X_i^D\}$ in Eq. (3.1)). That is, the updated *down* density is mixture of two original densities with equal weights. Then, by substituting sample densities and appropriate *magnitude* functions $\{M^U[X,f(X)]=1, M^D[X,f(X)]=2f(X)\}$ into the *balance state* equilibrium in Eq. (3.2) we will get:

$$\frac{\mathbb{P}^U(X)}{\frac{1}{2}\mathbb{P}^U(X)+\frac{1}{2}\mathbb{P}^D(X)} = \frac{2f(X)}{1} \quad \Rightarrow \quad f(X) = \frac{\mathbb{P}^U(X)}{\mathbb{P}^U(X)+\mathbb{P}^D(X)}. \tag{5.2}$$

The NDMR infers the same target function as the Classical GAN Critic loss from Table 5.3, and can be used as its alternative. Therefore, an additional degree of freedom is acquired by considering different sampling strategies in PSO framework. Similar ideas will allow us to also infer conditional density functions, as shown in Section 9.

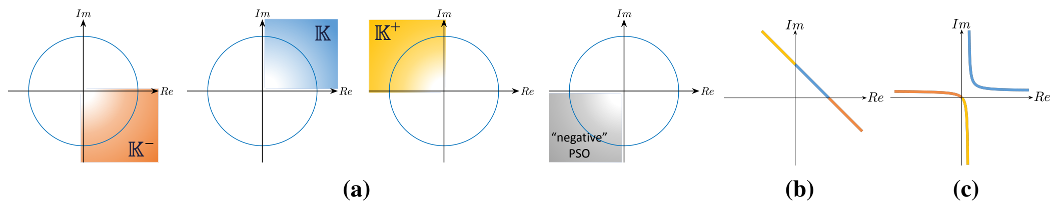


Figure 5.1: (a) Correspondence between different quadrants of a complex plane and different parts of $c[X, s]$. The quadrant I is where c must be for any $s \in \mathbb{K}$. Additionally, if c is continuous w.r.t. $s \in \mathbb{R}$ and if it is located in the quadrant IV at $s \in \mathbb{K}^-$ and in the quadrant II at $s \in \mathbb{K}^+$, then we are allowed to reduce restrictions over the function range. Further, the quadrant III is associated with "negative" PSO whose *magnitudes* have negative outputs and that can learn any decreasing target function $T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$. (b)-(c) $c[X, s]$ for two PSO instances is depicted, (b) "Square Loss" from Table 5.4 and (c) "Classical GAN Critic Loss" from Table 5.3. Colors red, blue and yellow represent parts of c at $s \in \mathbb{K}^-$, $s \in \mathbb{K}$ and $s \in \mathbb{K}^+$ respectively. As seen, in case of (b) the requirements are satisfied and hence there is no need to restrict the range of functions within $\mathcal{F}$. In contrast, in (c) the curve $c[X, s]$ parametrized by s is not even continuous, hence $\mathcal{F}$'s range must be $\mathbb{K}$.

5.3 PSO Feasibility Verification and Polar Parametrization

Sometimes it may be cumbersome to test if given $\{M^U, M^D\}$ satisfy all required conditions over sets $\mathbb{K}^-$, $\mathbb{K}$ and $\mathbb{K}^+$. Below we propose representing *magnitudes* within a complex plane, and use the corresponding polar parametrization. The produced representation yields a graphical visualization of PSO instance which permits for easier feasibility analysis.

For this purpose, define PSO complex-valued function as $c[X, s] \triangleq M^U[X, s] + M^D[X, s] \cdot i$ whose real part is *up magnitude*, and imaginary part - *down magnitude*. Further, denote by $c_\angle$ and c_r the angle and the radius of c defined as $c_\angle[X, s] = \text{atan2}(M^D[X, s], M^U[X, s])$ and $c_r[X, s] = \sqrt{M^U[X, s]^2 + M^D[X, s]^2}$. Conditions from theorems 5 and 6 can be translated into conditions over $c[X, s]$ as following.

Lemma 10 (Complex Plane Feasibility). *Consider PSO instance that is described by a complex-valued function $c[X, s] : \mathbb{R}^n \times \mathbb{R} \to \mathbb{C}$ and some convergence interval $\mathbb{K} \triangleq (s_{min}, s_{max})$. Assume:*

1. *$c[X, s]$ is continuous at any $s \in \mathbb{K}$, with $c_r[X, s] > 0$ and $0 < c_\angle[X, s] < \frac{\pi}{2}$.*

2. *$c_\angle[X, s]$ is strictly increasing and bijective w.r.t. domain $s \in \mathbb{K}$ and codomain $(0, \frac{\pi}{2})$.*

Then, given that the range of $\mathcal{F}$ is $\mathbb{K}$, the minima $f^* = \arg\inf_{f \in \mathcal{F}} L_{PSO}(f)$ will satisfy $\forall X \in \mathbb{S}^{U \cap D} : f^*(X) = T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$, where $T[X, z] \triangleq c_\angle^{-1}[X, \text{atan}(z)]$. Further, $\mathcal{F}$'s range can be entire $\mathbb{R}$ if the following conditions hold:

1. *$c[X, s]$ is continuous on $s \in \mathbb{R}$, with $c_r[X, s] > 0$.*

2. *$\forall s \in \mathbb{K}^- : \frac{3}{2}\pi \leq c_\angle[X, s] \leq 2\pi$.*

3. *$\forall s \in \mathbb{K}^+ : \frac{\pi}{2} \leq c_\angle[X, s] \leq \pi$.*

Proof. The necessary positivity of *magnitudes* over $s \in \mathbb{K}$ from Theorem 5 implies $\forall s \in \mathbb{K}$: $[0 < c_\angle[X, s] < \frac{\pi}{2}] \vee [c_r[X, s] > 0]$. Further, conditions of Theorem 6 are equivalent to require

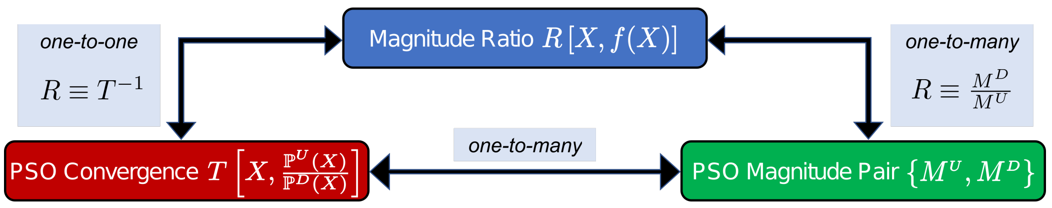

one-to-one
$R \equiv T^{-1}$
Magnitude Ratio $R[X, f(X)]$
one-to-many
$R \equiv \frac{M^D}{M^U}$
PSO Convergence $T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$
one-to-many
PSO Magnitude Pair $\{M^U, M^D\}$

Figure 5.2: Schematic relationship between PSO mappings of Theorem 4 and Theorem 5. T and R are inverse functions, with one-to-one relation between them. R is ratio of $\{M^U, M^D\}$, with infinitely many choices of the latter producing the same ratio. For this reason, many pairs of *magnitude* functions will yield the same PSO convergence T.

$\forall s \in \mathbb{K}^- : \left[\frac{3}{2}\pi \leq c_\angle[X, s] \leq 2\pi\right] \vee [c_r[X, s] > 0]$ and $\forall s \in \mathbb{K}^+ : [\frac{\pi}{2} \leq c_\angle[X, s] \leq \pi] \vee [c_r[X, s] > 0]$. Likewise, observe that the range of angles $(\pi, \frac{3}{2}\pi)$ is allocated by "negative" PSO family mentioned in Section 4.1.2, which can be formulated by switching between *up* and *down* terms of PSO family and which allows to learn any decreasing function $T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$. See also the schematic illustration in Figure 5.1a.

Further, continuity of $c[X, s]$ (over $\mathbb{K}$ or over entire $\mathbb{R}$) is equivalent to continuity of *magnitudes* enforced by theorems 5 and 6. Likewise, it leads to continuity of $c_\angle[X, s]$.

Moreover, given that $c[X, s]$ at $s \in \mathbb{K}$ is located within the quadrant I of a complex plane, its angle can be rewritten as $c_\angle[X, s] = \text{atan}\left(\frac{M^D[X,s]}{M^U[X,s]}\right) = \text{atan}(R[X, s])$, with $R[X, s] = \tan c_\angle[X, s]$ and $T[X, z] = R^{-1}[X, z] = c_\angle^{-1}[X, \text{atan}(z)]$ where $c_\angle^{-1}$ is an inverse of $c_\angle$ w.r.t. second argument. Hence, continuity of $c_\angle$ (implied by continuity of c) yields continuity of R at $s \in \mathbb{K}$, which is required by Theorem 5.

Finally, strictly increasing property of $c_\angle$ is equivalent to the same property of R since they are related via strictly increasing $\tan(\cdot)$ and $\text{atan}(\cdot)$. Similarly, bijectivity is also preserved, with codomain changed from $\mathbb{R}_{>0}$ to $(0, \frac{\pi}{2})$ due to limited range of $\text{atan}(\cdot)$.

■

The above lemma summarizes conditions required by PSO framework. As noted in Section 4.1.2, this condition set is overly restrictive and some of its parts may be relaxed. Particularly, we speculate that continuity may be replaced by continuity *almost everywhere*, and "increasing" (without "strictly") may be sufficient. We shall address such condition relaxation in future work.

To verify feasibility of any PSO instance, $c[X, s]$ can be drawn as a curve within a convex plane where conditions of the above lemma can be checked. In Figures 5.1b and 5.1c we show example of this curve for "Square Loss" from Table 5.4 and "Classical GAN Critic Loss" from Table 5.3 respectively. From the first diagram it is visible that the curve satisfies lemma's conditions, which allows us to not restrict $\mathcal{F}$'s range when optimizing via "Square Loss". In the second diagram conditions do not hold, leading to the conclusion that "Classical GAN Critic Loss" may be optimized only over $\mathcal{F}$ whose range is exactly $\mathbb{K}$.

5.4 PSO Subsets

Given any two densities $\mathbb{P}^U$ and $\mathbb{P}^D$, all PSO instances can be represented as a set of all feasible *magnitude* pairs $\mathbb{C} \triangleq \{[M^U, M^D] : feasible(M^U, M^D)\}$, where $feasible(\cdot)$ is a logical indicator that specifies whether the arguments satisfy conditions of Theorem 5 (for any set $\mathbb{K}$) or not. Below we systematically modulate the set $\mathbb{C}$ into subgroups, providing an useful terminology for the later analysis.

The relation of PSO mappings is presented in Figure 5.2. As observed, many different PSO instances have the same approximated target function. We will use this property to divide the set $\mathbb{C}$ of all feasible PSO instances into disjoint subsets, according to the estimation convergence. Considering any target function with the corresponding mapping T, we denote by $\mathbb{C}[T] \triangleq \{[M^U, M^D] : [M^U, M^D] \in \mathbb{C} \vee \frac{M^D}{M^U} = T^{-1}\}$ all feasible PSO instances that converge to T. The subset $\mathbb{C}[T]$ is referred below as T's PSO *consistent magnitude* set - PSO-CM set of T for shortness. Further, in this thesis we focus on two specific PSO subsets with $T(X, z) = \mathbb{P}^D(X) \cdot z$ and $T(X, z) = \log \mathbb{P}^D(X) + \log z$ that infer $\mathbb{P}^U(X)$ and $\log \mathbb{P}^U(X)$, respectively. For compactness, we denote the former as $\mathbb{A}$ and the latter as $\mathbb{B}$. Likewise, below we will term two pairs of *magnitude* functions as PSO *consistent* if they belong to the same subset $\mathbb{C}[T]$ (i.e. if their *magnitude* ratio is the same).

Given a specific PSO task at hand, represented by the required convergence T, it is necessary to choose the most optimal member of $\mathbb{C}[T]$ for the sequential optimization process. In Sections 7 and 8 we briefly discuss how to choose the most optimal PSO instance from PSO-CM set of any given T, based on the properties of *magnitude* functions.

5.5 PSO Methods Summary

The entire exposition of this section was based on a relation in Eq. (3.2) that sums up the main principle of PSO - *up* and *down* point-wise forces must be equal at the optimization equilibrium. In Tables 5.1-5.5 we refer to relevant works in case the specific PSO losses were already discovered in previous scientific studies. The previously discovered ones were all based on various sophisticated mathematical laws and theories, yet they all could be also derived in a simple unified way using PSO concept and Theorem 4. Additionally, besides the previously discovered methods, in Tables 5.1-5.5 we introduce several new losses for inference of different stochastic modalities of the data, as the demonstration of usage and usefulness of the general PSO formulation. In Section 6 we reveal that PSO framework has a tight relation with Bregman and "f" divergencies. Furthermore, in Section 10.1 we prove that the cross-entropy losses are also instances of PSO. Likewise, in Section 10.2 we derive Maximum Likelihood Estimation (MLE) from *PSO functional*.

CHAPTER 6

PSO, Bregman and "f" Divergencies

Below we define *PSO divergence* and show its connection to Bregman divergence [18, 30] and f-divergence [2], that are associated with many existing statistical methods.

6.1 PSO Divergence

Minimization of $L_{PSO}(f)$ corresponds also to minimization of:

$$D_{PSO}(f^*, f) \triangleq L_{PSO}(f) - L_{PSO}(f^*) = - \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} \int_{f^*(X)}^{f(X)} M^U(X,t)dt + \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \int_{f^*(X)}^{f(X)} M^D(X,t)dt \tag{6.1}$$

where $f^*(X) = T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$ is the optimal solution characterized by Theorem 5. Since f^* is unique minima (given that theorem's "sufficient" conditions hold) and since $L_{PSO}(f) \geq L_{PSO}(f^*)$, we have the following properties: $D_{PSO}(f^*, f) \geq 0$ and $D_{PSO}(f^*, f) = 0 \iff f^* = f$. Thus, D_{PSO} can be used as a "distance" between f and f^* and we name it *PSO divergence*. Yet, note that D_{PSO} does not measure a distance between any two functions; instead it evaluates distance between any f and the optimal solution of the specific PSO instance, derived from the corresponding $\{\mathbb{P}^U, \mathbb{P}^D, M^U, M^D\}$ via PSO *balance state*.

6.2 Bregman Divergence

Define $\mathcal{F}$ to be a convex set of non-negative functions from $\mathbb{R}^n$ to $\mathbb{R}_{\geq 0}$. Bregman divergence for every $q, p \in \mathcal{F}$ is defined as $D_\psi(q,p) = \psi(q) - \psi(p) - \int \nabla_q \psi(q(X)) \cdot [q(X) - p(X)]\, dX$, where $\psi(\mu)$ is a continuously-differentiable, strictly convex functional over $\mu \in \mathcal{F}$, and ∇_q is the differentiation w.r.t. q. When $\psi(\mu)$ has a form $\psi(\mu) = \int \varphi(\mu(X))dX$ with a strictly convex function $\varphi : \mathbb{R} \to \mathbb{R}$, the divergence is referred to as U-divergence [26] or sometimes as separable Bregman divergence [35]. In such case $D_\psi(q,p)$ has the form:

$$D_\varphi(q,p) = \int \varphi\left[q(X)\right] - \varphi\left[p(X)\right] - \varphi'\left[p(X)\right] \cdot \left[q(X) - p(X)\right] dX. \tag{6.2}$$

The above modality is non-negative for all $q, p \in \mathcal{F}$ and is zero if and only if $q = p$, hence it specifies a "distance" between q and p. For this reason, $D_\varphi(q, p)$ is widely used in the optimization $\min_p D_\varphi(q, p)$ where q usually represents an unknown density of available i.i.d. samples and p is a model which is optimized to approximate q. For example, when $\varphi(s) = s \cdot \log s$, we obtain the generalized KL divergence $D_\varphi(q, p) = \int q(X) \cdot [\log q(X) - 1] - q(X) \cdot \log p(X) + p(X) dX$ [137]. It can be further reduced to MLE objective $-\mathbb{E}_{X \sim q(X)} \log p(X)$ in case p is normalized, as is shown in Section 10.2.

Further, the term $\psi(\mu) \triangleq \int \varphi[\mu(X)] \, dX$ is called an entropy of μ, and $d_\varphi(q, p) \triangleq -\psi(p) - \int \varphi'[p(X)] \cdot [q(X) - p(X)] \, dX$ is referred to as a cross-entropy between q and p. Since $D_\varphi(q, p) = d_\varphi(q, p) - d_\varphi(q, q)$, usually the actual optimization being solved is $\min_p d_\varphi(q, p)$ where the cross-entropy objective is approximated via Monte-Carlo integration.

Claim 11. *Consider a strictly convex and twice differentiable function φ and two densities $\mathbb{P}^U$ and $\mathbb{P}^D$ that satisfy $\mathbb{S}^U \subseteq \mathbb{S}^D$. Likewise, define magnitudes $\{M^U[X, s] = \varphi''(s)$, $M^D[X, s] = \frac{s \cdot \varphi''(s)}{\mathbb{P}^D(X)}\}$ and a space of non-negative functions $\mathcal{F}$. Then PSO functional and PSO divergence are equal to U-cross-entropy (up to an additive constant) and U-divergence defined by φ, respectively:*

1. $\forall f \in \mathcal{F} : L_{PSO}(f) = d_\varphi(\mathbb{P}^U, f)$
2. $\forall f \in \mathcal{F} : D_{PSO}(\mathbb{P}^U, f) = D_\varphi(\mathbb{P}^U, f)$

Proof. First, *primitives* corresponding to the above *magnitude* functions are $\{\widetilde{M}^U[X, s] = \varphi'(s) + c_U, \widetilde{M}^D[X, s] = \frac{[s \cdot \varphi'(s) - \varphi(s)]}{\mathbb{P}^D(X)} + c_D\}$, where c_U and c_D are additive constants. Denote $c_T \triangleq c_D - c_U$. Introducing above expressions into L_{PSO} defined in Eq. (3.3) will lead to:

$$\forall f \in \mathcal{F} : \quad L_{PSO}(f) = \int -\varphi[f(X)] + \varphi'[f(X)] \cdot [f(X) - \mathbb{P}^U(X)] \, dX + c_T = d_\varphi(\mathbb{P}^U, f) + c_T. \tag{6.3}$$

Further, according to Table 5.6 the minimizer f^* of $L_{PSO}(f)$ for a given $\{M^U, M^D\}$ is $\mathbb{P}^U(X)$. Therefore, Eq. (6.1) leads to $D_{PSO}(f^*, f) \triangleq L_{PSO}(f) - L_{PSO}(f^*) = d_\varphi(\mathbb{P}^U, f) - d_\varphi(\mathbb{P}^U, f^*) = d_\varphi(\mathbb{P}^U, f) - d_\varphi(\mathbb{P}^U, \mathbb{P}^U) = D_\varphi(\mathbb{P}^U, f)$.

Function φ has to be twice differentiable in order to solve $\min_p d_\varphi(q, p)$ via gradient-based optimization - derivative of $d_\varphi(q, p)$ w.r.t. p involves φ''. Hence, this property is typically satisfied by all methods that minimize Bregman divergence. Furthermore, if $\varphi''(s)$ is a continuous function then the specified *magnitudes* are feasible w.r.t. Theorem 5. Otherwise, we can verify a more general set of conditions at the end of Section 4.1.1 which is satisfied for the claim's setting.

■

From the above we can conclude that definitions of Bregman and PSO divergencies coincide when the former is limited to U-divergence and the latter is limited to *magnitudes* that satisfy $\frac{M^D[X,s]}{M^U[X,s]} = \frac{s}{\mathbb{P}^D(X)}$. Such PSO subset has *balance state* at $f^*(X) = \mathbb{P}^U(X)$ and is denoted in Section 5.4 as $\mathbb{A}$. It contains all PSO instances for the density estimation, not including log-density estimation methods in subset $\mathbb{B}$. Further, since $L_{PSO}(f) = d_{\varphi}(\mathbb{P}^U, f)$, a family of algorithms that minimize U-divergence and PSO subset $\mathbb{A}$ of density estimators are equivalent. However, in general *PSO divergence* is different from Bregman as it can also be used to measure a "distance" between any PSO solution f^*, including even negative functions such as log-density $\log \mathbb{P}^U(X)$ and log-density-ratio $\log \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$ (see Section 5), and any $f \in \mathcal{F}$ without the non-negative constraint over $\mathcal{F}$.

6.3 f-Divergence

Define q and p to be probability densities so that q is absolutely continuous w.r.t. p. Then, f-divergence between q and p is defined as $D_\phi(q,p) = \int p(X) \cdot \phi(\frac{q(X)}{p(X)}) dX$, where $\phi : \mathbb{R} \to \mathbb{R}$ is a convex and lower semicontinuous function s.t. $\phi(1) = 0$. This divergence family contains many important special cases, such as KL divergence (for $\phi(z) = z \cdot \log z$) and Jensen-Shannon divergence (for $\phi(z) = -(z+1) \cdot \log \frac{1+z}{2} + z \cdot \log z$).

In [92] authors proved that it is lower-bounded as:

$$D_\phi(q,p) \geq \sup_{f \in \mathcal{F}} J_\phi(f,q,p), \quad J_\phi(f,q,p) \triangleq \mathop{\mathbb{E}}_{X \sim q(X)} f(X) - \mathop{\mathbb{E}}_{X \sim p(X)} \phi^c\left[f(X)\right], \tag{6.4}$$

where ϕ^c is the convex-conjugate function of ϕ: $\phi^c(s) \triangleq \sup_{z \in \mathbb{R}} \{z \cdot s - \phi(z)\}$. The above expression becomes an equality when $\phi'(\frac{q(X)}{p(X)})$ belongs to a considered function space $\mathcal{F}$. In such case, the supremum is obtained via $f(X) = \phi'(\frac{q(X)}{p(X)})$. Therefore, the above lower bound is widely used in the optimization $\max_f J_\phi(f,q,p)$ to approximate $D_\phi(q,p)$ [92], and to learn any function of $\frac{q(X)}{p(X)}$ [94].

Claim 12. *Consider a strictly convex and differentiable function ϕ s.t. $\phi(1) = 0$, its convex conjugate ϕ^c, and two densities $\mathbb{P}^U$ and $\mathbb{P}^D$ that satisfy $\mathbb{S}^U \subseteq \mathbb{S}^D$. Likewise, define PSO primitives $\{\widetilde{M}^U[X,s] = s, \widetilde{M}^D[X,s] = \phi^c(s)\}$ and assume that $\phi'(\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)})$ is contained in a function space $\mathcal{F}$. Then PSO functional and f-divergence have the following connection:*

1. *$\forall f \in \mathcal{F} : L_{PSO}(f) \equiv -J_\phi(f, \mathbb{P}^U, \mathbb{P}^D)$ and*
   $f^* = \arg\min_{f \in \mathcal{F}} L_{PSO}(f) = \arg\max_{f \in \mathcal{F}} J_\phi(f, \mathbb{P}^U, \mathbb{P}^D) = \phi'(\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)})$
2. *$\forall f \in \mathcal{F} : D_\phi(\mathbb{P}^U, \mathbb{P}^D) = -L_{PSO}(f^*) \geq -L_{PSO}(f)$*

Proof. First, introducing the given $\{\widetilde{M}^U, \widetilde{M}^D\}$ into Eq. (3.3) leads to $L_{PSO}(f) \equiv -J_\phi(f, \mathbb{P}^U, \mathbb{P}^D)$. Further, since $\{M^U[X,s] = 1, M^D[X,s] = \phi^{c\prime}(s)\}$ we have $\frac{M^D[X,s]}{M^U[X,s]} = \phi^{c\prime}(s)$. Denote $R[X,s] = \frac{M^D[X,s]}{M^U[X,s]} = \phi^{c\prime}(s)$ and observe that its inverse is $T[X,z] = \phi'(z)$ due properties of LF transform. Applying Theorem 5, we conclude $f^*(X) = \phi'(\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)})$. Finally, since we assumed $f^* \in \mathcal{F}$, we also have $D_\phi(\mathbb{P}^U, \mathbb{P}^D) = J_\phi(f^*, \mathbb{P}^U, \mathbb{P}^D) = -L_{PSO}(f^*)$. ∎

The required by claim strict convexity and differentiability of function ϕ are necessary in order to recover the estimation convergence, as was also noted in Section 5 of [92]. Likewise, these properties ensure that derivatives ϕ' and $\phi^{c\prime}$ are well-defined. Further, above *magnitudes*

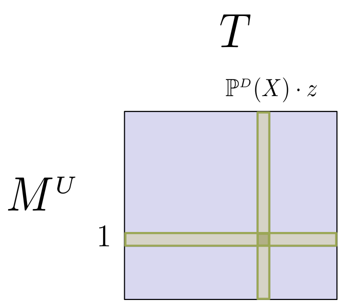


Figure 6.1: Schematic relation between PSO and its two subgroups related in Claim 11 and Claim 12. All PSO instances within $\mathbb{C}$ can be indexed by the convergence T, where each $\mathbb{C}[T]$ can be further indexed by function M^U (given T and M^U, M^D is derived as $M^D[X,s] = T^{-1}[X,s]\cdot M^U[X,s]$). Thus, all elements of $\mathbb{C}$ can be viewed as a table indexed by T and M^U. The column corresponding to $T[X,z] = \mathbb{P}^D(X)\cdot z$ encapsulates PSO methods referred by Claim 11 that minimize a separable Bregman divergence, and the row corresponding to $M^U[X,s] = 1$ - methods from Claim 12 that approximate f-divergence. The column-row intersection is obtained at $M^U[X,s] = 1$, $M^D[X,s] = \frac{s}{\mathbb{P}^D(X)}$ and $T[X,z] = \mathbb{P}^D(X)\cdot z$.

satisfy the requirements of PSO framework since the PSO derivation in Section 4.1.1 agrees with derivation in [92] when the mapping G considered in our proof is equal to $G(X,s) = s$.

Finally, observe that given the above pair of $\{M^U, M^D\}$, the primitives $\widetilde{M}^U$ and $\widetilde{M}^D$ can be recovered up to an additive constant. This constant does not affect the outcome of PSO inference since the optima f^* of *PSO functional* stays unchanged. Yet, it changes the output of $L_{PSO}(f)$ and thus for arbitrary antiderivatives $\{\widetilde{M}^U, \widetilde{M}^D\}$ the identity $D_\phi(\mathbb{P}^U, \mathbb{P}^D) = -L_{PSO}(f^*)$ will not be satisfied. To recover the correct additive constant, we can use information produced by $\phi(1) = 0$. Specifically, for the considered above setting the "proper" antiderivative $\widetilde{M}^D$ of M^D is the one that satisfies $\widetilde{M}^D[X,s'] = s'$ where $s' \triangleq \phi'(1)$.

■

Hence, in light of the above connection PSO framework can be seen as an estimation of a negative f-divergence between $\mathbb{P}^U$ and $\mathbb{P}^D$, $-D_\phi(\mathbb{P}^U, \mathbb{P}^D)$. Likewise, PSO is identical to an estimation framework of [94] when the former is limited to have *magnitudes* $\{M^U[X,s] = 1, M^D[X,s] = \phi^{c\prime}(s)\}$. Considering the terminology of Section 5.4, such set of *magnitude* functions can be obtained by retrieving from each PSO subset $\mathbb{C}[T]$ an particular *magnitude* pair with $M^U \equiv 1$ and $M^D \equiv T^{-1}$, with all collected pairs being equivalent to the estimation family of [94], as shown in Figure 6.1. Due to such limitation this algorithm family is a strict subgroup of PSO.

Furthermore, the formulation of this framework in Eq. (6.4) can be generalized as:

$$\min_{f\in\mathcal{F}} - \mathop{\mathbb{E}}_{X\sim\mathbb{P}^U} G[X, f(X)] + \mathop{\mathbb{E}}_{X\sim\mathbb{P}^D} \phi^c[X, G[X, f(X)]], \tag{6.5}$$

where f is replaced by a transformation $G(X,s) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ over the inner model f, and where ϕ^c is the convex-conjugate of $\phi(X,z) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ that was extended to accept two arguments. Consequently, G can be selected to obtain any required convergence $f^* = G^{-1} \odot \phi' \odot \frac{\mathbb{P}^U}{\mathbb{P}^D}$, where $\odot$ defines the composition via a second argument $g \odot h \iff g(X, h(\cdot))$. Such extension parametrized by a pair $\{\phi^c, G\}$ becomes identical to PSO parametrized by a pair $\{\widetilde{M}^U, \widetilde{M}^D\}$, as is evident from our derivation in Section 4.1.1. Particularly, the parametrization $\{\widetilde{M}^U, \widetilde{M}^D\}$ can be recovered from $\{\phi^c, G\}$ via $\widetilde{M}^U[X,s] = G[X,s]$ and $\widetilde{M}^D[X,s] = \phi^c[X, G[X,s]]$.

In the backward direction, $\{\phi^c, G\}$ can be recovered from $\{\widetilde{M}^U, \widetilde{M}^D\}$ as following. Denote the corresponding derivatives by $\{M^U, M^D\}$, and their ratio by $R[X,s]$. Further, denote $s^\bullet[X] \triangleq T[X,1]$ where T is an inverse of R w.r.t. second argument. Then, $\{\phi^c, G\}$ can be obtained via $G[X,s] = \widetilde{M}^U[X,s]$ and $\phi^c[X,s] = \widetilde{M}^D[X, G^{-1}[X,s]] + \alpha[X]$, with $\alpha[X] \triangleq \widetilde{M}^U[X, s^\bullet[X]] - \widetilde{M}^D[X, s^\bullet[X]]$. Note that the additive X-dependent component $\alpha[X]$ ensures $\phi^c[X,s'] = s'$ at $s' \triangleq \phi'[X,1]$, which produces the condition $\phi[X,1] = 0$ required by the definition of f-divergence. Thus, for the above "properly" normalized ϕ^c the expression in Eq. (6.5) is equal to $-D_\phi(\mathbb{P}^U, \mathbb{P}^D)$, where we extend f-divergence to have a form[1] $D_\phi(q,p) = \int p(X) \cdot \phi\left[X, \frac{q(X)}{p(X)}\right] dX$.

Moreover, the relation between parametrizations $\{\phi^c, G\}$ and $\{\widetilde{M}^U, \widetilde{M}^D\}$ allows us to compute $D_\phi(\mathbb{P}^U, \mathbb{P}^D)$ from PSO loss $L_{PSO}(f)$ even in case of arbitrary antiderivatives $\{\widetilde{M}^U, \widetilde{M}^D\}$ that were not necessarily "properly" normalized. Specifically, $D_\phi(\mathbb{P}^U, \mathbb{P}^D)$ can be recovered from $L_{PSO}(f^*) \equiv \min_{f \in \mathcal{F}} L_{PSO}(f)$ via:

$$D_\phi(\mathbb{P}^U, \mathbb{P}^D) = -L_{PSO}(f^*) - \beta, \quad \beta = \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \alpha[X]. \tag{6.6}$$

Furthermore, if $\{\widetilde{M}^U, \widetilde{M}^D\}$ depend only on their second argument, which is often the case, then $T[X,z]$ is also only a function of z and the additive constant $\beta = \alpha = \widetilde{M}^U[T(1)] - \widetilde{M}^D[T(1)]$ does not require the expected value computation.

The extension in Eq. (6.5) requires a remark about novelty and usefulness of PSO compared to works in [92, 94]. First, note that the generalization in Eq. (6.5) was not proposed by any study, to the best of our knowledge, although the original paper [94] was considering a transformation $G[f(X)]$ to control the domain of particular ϕ^c. Likewise, we note that the implied by f-divergence framework restriction $M^U[X,s] = 1$ over *up magnitude* inevitably causes $M^D[X,s]$ to be an unbounded function. Such unboundedness property causes instability during optimization as discussed and empirically shown in sections 7.2 and 13 respectively. In contrast, PSO formulation allows to easily construct bounded *magnitudes* for any desired target function by removing the aforementioned restriction (see Section 7.2 for details).

Further, there are three main points of difference between two estimation procedures. PSO is defined via $\{M^U, M^D\}$, permitting $\{\widetilde{M}^U, \widetilde{M}^D\}$ to not have an analytical form which in turn increases the number of feasible estimators. This allows us to propose novel techniques not considered before, such as PSO-LDE in Section 8.2 which is shown in Section 13 to be superior over other state-of-the-art baselines. Further, using $G[X, f(X)]$ instead of $G[f(X)]$ is important to allow more freedom in selecting the required convergence f^*. Particularly, employing PSO to learn the density $\mathbb{P}^U(X)$ is possible only under this two-argument setting. Finally, PSO allows for a better intuition which is not available for the f-divergence formulation

[1]This extended modality can be shown to satisfy the non-negativity $D_\phi(q,p) \geq 0$ and the identification $D_\phi(q,p) = 0 \Leftrightarrow q \equiv p$, required by the statistical divergence definition. We leave the proof of that for future work since it is not the main focus of this thesis.

in [94], and can further be used to analyze convergence outside of the mutual support $\mathbb{S}^{U \cap D}$.

6.4 Divergence Relation Summary

Due to above links PSO loss can be rewritten as $L_{PSO}(f) = D_{PSO}(f^*, f) - D_\phi(\mathbb{P}^U, \mathbb{P}^D) - \beta$, where *PSO divergence* $D_{PSO}(f^*, f)$ can be considered as an extension of separable Bregman divergence and where $D_\phi(\mathbb{P}^U, \mathbb{P}^D)$ is f-divergence defined via some ϕ. Further, minimization of $L_{PSO}(f)$ is same as the minimization of *PSO divergence* between f^* and f. Finally, at the convergence $f = f^*$, $D_{PSO}(f^*, f^*)$ becomes zero and $L_{PSO}(f^*)$ is equal to $-D_\phi(\mathbb{P}^U, \mathbb{P}^D) - \beta$.

CHAPTER 7

Properties of PSO Estimators

In above sections we saw that PSO principles are omnipresent within many statistical techniques. In this section we will investigate how these principles work in practice, by extending our analysis beyond the described above variational equilibrium. Particularly, we will study the consistency and the convergence of PSO, as also the actual equilibrium obtained via GD optimization. Furthermore, we will describe the model kernel impact on a learning task and emphasize the extreme similarity between PSO actual dynamics and the physical illustration in Figure 3.1. Likewise, we will propose several techniques to improve the practical stability of PSO algorithms.

7.1 Consistency and Asymptotic Normality

PSO solution f_{θ^*} is obtained by solving $\min_{f_\theta \in \mathcal{F}} L_{PSO}(f_\theta)$, and hence PSO belongs to the family of extremum estimators [3]. Members of this family are known to be consistent and asymptotically normal estimators under appropriate regularity conditions. Below we restate the corresponding theorems.

Herein, we will reduce the scope to PSO instances whose *up* and *down* densities have an identical support. This is required to avoid the special PSO cases for which f_{θ^*} has convergence at infinity outside of the mutual support (see theorems 7 and 8). Although it is possible to evade such convergence by considering $\mathcal{F}$ whose functions' output is lower and upper bounded (see Section 7.6), we sidestep this by assuming $\mathbb{S}^U \equiv \mathbb{S}^D$, for simplicity.

The empirical PSO loss over N^U samples from $\mathbb{P}^U$ and N^D samples from $\mathbb{P}^D$ has a form:

$$\hat{L}_{PSO}^{N^U,N^D}(f_\theta) = -\frac{1}{N^U}\sum_{i=1}^{N^U} \widetilde{M}^U\left[X_i^U, f_\theta(X_i^U)\right] + \frac{1}{N^D}\sum_{i=1}^{N^D} \widetilde{M}^D\left[X_i^D, f_\theta(X_i^D)\right], \tag{7.1}$$

with its gradient $\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)$ being defined in Eq. (3.1).

Theorem 13 (Consistency). *Assume:*

1. *Parameter space Θ is a compact set.*
2. *$L_{PSO}(f_\theta)$ (defined in Eq. (3.3)) is continuous on $\theta \in \Theta$.*
3. *$\exists! \theta^* \in \Theta, \forall X \in \mathbb{S}^{U \cap D} : \mathbb{P}^U(X) \cdot M^U[X, f_{\theta^*}(X)] = \mathbb{P}^D(X) \cdot M^D[X, f_{\theta^*}(X)]$.*
4. *$\sup_{\theta \in \Theta} \left| \hat{L}_{PSO}^{N^U, N^D}(f_\theta) - L_{PSO}(f_\theta) \right| \xrightarrow{p} 0$ along with $\min(N^U, N^D) \to \infty$.*

Define $\hat{\theta}_{N^U,N^D} = \arg\min_{\theta \in \Theta} \hat{L}_{PSO}^{N^U, N^D}(f_\theta)$. Then $\hat{\theta}_{N^U,N^D}$ converges in probability to θ^* along with $\min(N^U, N^D) \to \infty$, $\hat{\theta}_{N^U,N^D} \xrightarrow{p} \theta^*$.

The above assumptions are typically taken by many studies to claim the estimation consistency. Assumption 3 ensures that there is only single vector θ^* for which PSO *balance state* is satisfied. Hence it ensures that $L_{PSO}(f_\theta)$ is uniquely minimized at θ^*, according to Theorem 1. Assumption 4 is the uniform convergence of empirical loss towards its population form along with $\min(N^U, N^D) \to \infty$. The above consistency statement and its proof appear as theorem 2.1 in [90].

Theorem 13 ensures the estimation consistency of PSO under technical conditions over space Θ. However, some of these conditions (e.g. compactness of Θ) are not satisfied by NNs. Another way, taken by [57,92], is to use a complexity metric defined directly over the space $\mathcal{F}$, such as the bracketing entropy. We shall leave this more sophisticated consistency derivation for future work.

For asymptotic normality we will apply the following statements where we use notations $\mathcal{I}_\theta(X, X') \triangleq \nabla_\theta f_\theta(X) \cdot \nabla_\theta f_\theta(X')^T$, $N \triangleq N^U + N^D$, $\tau \triangleq \frac{N^U}{N^D}$, $M^{U\prime}(X, s) \triangleq \frac{\partial M^U(X,s)}{\partial s}$ and $M^{D\prime}(X, s) \triangleq \frac{\partial M^D(X,s)}{\partial s}$.

Lemma 14 (Hessian). *Assume $\exists \theta^* \in \Theta$ s.t. $\forall X \in \mathbb{S}^{U \cap D} : \mathbb{P}^U(X) \cdot M^U[X, f_{\theta^*}(X)] = \mathbb{P}^D(X) \cdot M^D[X, f_{\theta^*}(X)]$. Denote PSO convergence as $f_{\theta^*}(X) = T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right] \equiv f^*(X)$, according to Theorem 5. Then the Hessian $\mathcal{H}$ of population PSO loss at θ^* has a form:*

$$\mathcal{H} \equiv \nabla_{\theta\theta} L_{PSO}(f_{\theta^*}) = -\mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} M^{U\prime}[X, f^*(X)] \cdot \mathcal{I}_{\theta^*}(X, X) + \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} M^{D\prime}[X, f^*(X)] \cdot \mathcal{I}_{\theta^*}(X, X). \tag{7.2}$$

Lemma 15 (Gradient Variance). *Under the same setting, the variance of $\nabla_\theta \hat{L}_{PSO}^{N^U, N^D}(f_\theta)$ at θ^* has a form $Var\left[\nabla_\theta \hat{L}_{PSO}^{N^U, N^D}(f_{\theta^*})\right] = \frac{1}{N}\mathcal{J}$ where:*

$$\begin{aligned}\mathcal{J} = \frac{\tau + 1}{\tau} \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} \left[M^U[X, f^*(X)]^2 \cdot \mathcal{I}_{\theta^*}(X, X)\right] + \\ + (\tau + 1) \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \left[M^D[X, f^*(X)]^2 \cdot \mathcal{I}_{\theta^*}(X, X)\right] - \\ - \frac{(\tau + 1)^2}{\tau} \mathop{\mathbb{E}}_{\substack{X \sim \mathbb{P}^U \\ X' \sim \mathbb{P}^D}} M^U[X, f^*(X)] \cdot M^D[X', f^*(X')] \cdot \mathcal{I}_{\theta^*}(X, X'). \end{aligned} \tag{7.3}$$

$\mathcal{H}$ and $\mathcal{J}$ have several additional forms that appear in Appendix A along with the lemmas' proofs.

Theorem 16 (Asymptotic Normality). *Given assumptions and definitions of Theorem 13, assume additionally:*

1. *$\theta^* \in int(\Theta)$.*
2. *$\hat{L}_{PSO}^{N^U,N^D}(f_\theta)$ is twice continuously differentiable in a neighborhood of θ^*.*
3. *$\nabla_{\theta\theta} L_{PSO}(f_\theta)$ is continuous in θ.*
4. *Both $\mathcal{J}$ and $\mathcal{H}$ are non-singular matrices.*
5. *Each entry of $\nabla_{\theta\theta} L_{PSO}(f_\theta)$ and of $\mathcal{J}$ is uniformly bounded by an integrable function.*
6. *$\tau > 0$ is a finite and fixed scalar.*

Then $\sqrt{N} \cdot \left[\hat{\theta}_{N^U,N^D} - \theta^*\right]$ converges in distribution to $\mathcal{N}(0, \mathcal{H}^{-1}\mathcal{J}\mathcal{H}^{-1})$ along with $N \to \infty$.

Proof of the above theorem is in Appendix B. Thus, we can see that for large sample sizes the parametric estimation error $\sqrt{N^U + N^D} \cdot \left[\hat{\theta}_{N^U,N^D} - \theta^*\right]$ is normal with zero mean and covariance matrix $\Sigma = \mathcal{H}^{-1}\mathcal{J}\mathcal{H}^{-1}$. Further, it is possible to simplify an expression for Σ when considering $\{M^U, M^D\}$ of a specific PSO instance. Observe also that none of the above theorems and lemmas require the analytical knowledge of $\{\widetilde{M}^U, \widetilde{M}^D\}$.

7.2 Bounded vs Unbounded Magnitude Functions

Any considered functions $\{M^U, M^D\}$ will produce PSO *balance state* at the convergence, given that they satisfy conditions of Theorem 5. Further, as was shown in Section 5, there is infinite number of possible *magnitudes* within $\mathbb{C}[T]$ that will lead to the same convergence T. Thus, we need analytical tools to establish the superiority of one *magnitude* function pair over another. While this is an advanced estimation topic from robust statistics and is beyond this thesis's scope, herein we describe one desired property for these functions to have - boundedness.

In Tables 5.1-5.5 we can see many choices of $\{M^U, M^D\}$, with both bounded (e.g. NCE in Table 5.1) and unbounded (e.g. IS in Table 5.1) outputs. Further, note that IS method has a *magnitude* function $M^D[X, f_\theta(X)] = \frac{\exp(f_\theta(X))}{\mathbb{P}^D(X)}$ with outputs that can be extremely high or low, depending on the difference $[f_\theta(X) - \log \mathbb{P}^D(X)]$. From Eq. (3.1) we can likewise see that the gradient contribution of the *down* term in PSO loss is $M^D[X, f_\theta(X)] \cdot \nabla_\theta f_\theta(X)$. Thus, in case $f_\theta(X) \gg \log \mathbb{P}^D(X)$, high values from $M^D(\cdot)$ will produce gradients with large norm. Such large norm causes instability during the optimization, known as exploding gradients problem in DL community. Intuitively, when we make a large step inside θ-space, the consequences can be unpredictable, especially for highly non-linear models such as modern NNs.

Therefore, in practice the loss with bounded gradient is preferred. Such conclusion was also empirically supported in context of unnormalized density estimation [105]. Further, while it is possible to solve this issue by for example gradient clipping and by decreasing the learning rate [102], such solutions also slow down the entire learning process. PSO framework allows to achieve the desired gradient boundedness by bounding *magnitude* functions' outputs via the following lemma.

Lemma 17 (PSO Consistent Modification). *Consider any PSO instance $[M^U, M^D] \in \mathbb{C}$ with the corresponding convergence interval $\mathbb{K}$ on which criteria of Theorem 5 hold. Define $D(X, s) : \mathbb{R}^n \times \mathbb{K} \to \mathbb{R}$ to be a continuous and positive function on $s \in \mathbb{K}$ for any $X \in \mathbb{S}^{U \cap D}$. Then, $[M^U, M^D]$ and $[\frac{M^U}{D}, \frac{M^D}{D}]$ are PSO consistent (i.e. have identical convergence).*

The proof is trivial, by noting that the ratio of both pairs is preserved which places them into the same PSO subset $\mathbb{C}[T]$. The feasibility of the second pair is a result of the first pair's feasibility and of the fact that a devision by D does not change the sign and continuity of *magnitude* functions.

To produce bounded *magnitudes*, consider any pair of functions $\{M^U[X, s], M^D[X, s]\}$ with some desired PSO convergence T. In case these are unbounded functions, a new pair of bounded functions can be constructed as

$$M^U_{bounded}[X, s] = \frac{M^U[X, s]}{|M^U[X, s]| + |M^D[X, s]|}, \quad M^D_{bounded}[X, s] = \frac{M^D[X, s]}{|M^U[X, s]| + |M^D[X, s]|}. \tag{7.4}$$

It is clear from their structure that the new functions' outputs are in $[-1, 1]$. Furthermore, their PSO convergence will be identical to the one of the original pair, due to Lemma 17. Similarly, we can replace the absolute value also by other norms.

Magnitudes of many popular estimation methods (e.g. NCE, logistic loss, cross-entropy) are already bounded, making them more stable compared to unbounded variants. This may explain their wide adaptation in Machine Learning community. In Section 8.2 we will use the above transformation to develop a new family of robust log-pdf estimators.

7.3 Statistics of Surface Change

Herein we will analyze statistical properties of f_θ's evolution during the optimization. To this end, define the *differential* $df_{\theta_t}(X) \triangleq f_{\theta_{t+1}}(X) - f_{\theta_t}(X)$ as a change of $f_\theta(X)$ after t-th GD iteration. Its first-order Taylor approximation is:

$$\begin{aligned} &df_\theta(X) \approx -\delta \cdot \nabla_\theta f_\theta(X)^T \cdot \nabla_\theta \hat{L}^{N^U,N^D}_{PSO}(f_\theta) = \\ &= \delta \cdot \left[\frac{1}{N^U} \sum_{i=1}^{N^U} M^U[X^U_i, f_\theta(X^U_i)] \cdot g_\theta(X, X^U_i) - \frac{1}{N^D} \sum_{i=1}^{N^D} M^D[X^D_i, f_\theta(X^D_i)] \cdot g_\theta(X, X^D_i) \right], \end{aligned} \tag{7.5}$$

where δ is the learning rate, value of θ is the one before GD iteration, and $\nabla_\theta \hat{L}^{N^U,N^D}_{PSO}(f_\theta)$ is the loss gradient. When the considered model is NN, the above first-order dynamics are typically a very good approximation of the real $df_\theta(X)$ [54, 63]. Further, when f_θ belongs to RKHS, the above approximation becomes an identity. Therefore, below we will treat Eq. (7.5) as an equality, neglecting the fact that this is only the approximation.

Theorem 18 (Differential Statistics). *Denote $F_\theta(X) \triangleq \mathbb{P}^U(X) \cdot M^U[X, f_\theta(X)] - \mathbb{P}^D(X) \cdot M^D[X, f_\theta(X)]$ to be a difference of two point-wise forces F_θ^U and F_θ^D defined in Section 3, $F_\theta(X) = F_\theta^U(X) - F_\theta^D(X)$. Additionally, define G_θ to be the integral operator $[G_\theta u](\cdot) = \int g_\theta(\cdot, X)u(X)dX$. Then, considering training points as random i.i.d. realizations from the corresponding densities $\mathbb{P}^U$ and $\mathbb{P}^D$, the expected value and the covariance of $df_\theta(X)$ at any fixed θ are:*

$$\mathbb{E}[df_\theta(X)] = \delta \cdot \Big[\mathop{\mathbb{E}}_{X' \sim \mathbb{P}^U} \Big[M^U[X', f_\theta(X')] \cdot g_\theta(X, X') \Big] -$$
$$- \mathop{\mathbb{E}}_{X' \sim \mathbb{P}^D} \Big[M^D[X', f_\theta(X')] \cdot g_\theta(X, X') \Big] \Big] = \delta \cdot \int g_\theta(X', X) \cdot F_\theta(X')dX' = \delta \cdot [G_\theta F_\theta](X), \tag{7.6}$$

$$\mathit{Cov}[df_\theta(X), df_\theta(X')] = \delta^2 \cdot \nabla_\theta f_\theta(X)^T \cdot \mathit{Var}\Big[\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)\Big] \cdot \nabla_\theta f_\theta(X'), \tag{7.7}$$

with $\mathit{Var}\Big[\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)\Big]$ being proportional to $\frac{1}{N^U + N^D}$.

Proof of the above theorem together with the explicit form of Var $\Big[\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)\Big]$ appears in Appendix C. The theorem provides insights about stochastic dynamics of the surface f_θ caused by point-wise forces $F_\theta^U(X)$ and $F_\theta^D(X)$. Eq. (7.6) is an *integral transform* of the total force $F_\theta(X)$, which can also be seen as a point-wise error. That is, on the average $df_\theta(X)$ changes proportionally to the convolution of $[F_\theta^U(X') - F_\theta^D(X')]$ w.r.t. the kernel $g_\theta(X', X)$. When F_θ^U is larger than F_θ^D, after both being convolved via g_θ, then on the average $f_\theta(X)$ is pushed *up*, and vice versa. When convolutions of F_θ^U and F_θ^D are equal around the point X, $f_\theta(X)$ stays constant, again on the average. Further, the variance of $df_\theta(X)$ depends on the alignment between $\nabla_\theta f_\theta(X)$ and the eigenvectors of Var $\Big[\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)\Big]$ that correspond to the largest eigenvalues (the directions in θ-space to which PSO loss mostly propagates), and in addition can be reduced by increasing the size of training datasets.

Neglecting higher moments of random variable $df_\theta(X)$, this *differential* can be expressed as the sum $df_\theta(X) = \delta \cdot [G_\theta F_\theta](X) + \omega_\theta(X)$ where ω_θ is a zero-mean indexed-by-X stochastic process with the covariance function defined in Eq. (7.7). Such evolution implies that $f_\theta(X)$ will change on average towards the height where the *convoluted* PSO equilibrium $[G_\theta F_\theta](\cdot) = 0 \Leftrightarrow G_\theta F_\theta^U = G_\theta F_\theta^D$ is satisfied:

$$\int g_\theta(X, X') \cdot F_\theta^U(X')dX' = \int g_\theta(X, X') \cdot F_\theta^D(X')dX', \tag{7.8}$$

while ω_θ's variance will cause it to vibrate around such target height. Likewise, informally δ in Eq. (7.7) has a role of configuration parameter that controls the diapason around the target height where the current estimation $f_\theta(X)$ is vibrating. Further, sequential tuning/decaying of the learning rate δ will decrease this vibration amplitude (distance between the function that satisfies Eq. (7.8) and the current model).

Furthermore, considering a large training dataset regime where ω_θ's effect is insignificant, and replacing θ by the iteration time t, dynamics of the model can be written as $df_t(X) = f_{t+1}(X) - f_t(X) = \delta \cdot [G_t F_t](X)$, or as $f_{t+1} = f_t + \delta \cdot G_t F_t$. Here, F_t represents a negative functional derivative - the steepest descent direction of loss L_{PSO} in the function space. Further, G_t is GD operator that stretches and shrinks F_t according to the alignment of the latter with eigenfunctions of the model kernel. Hence, it serves as a metric over the function space, defining what directions are "fast" to move in and in which directions it is "slow".

7.4 Convoluted PSO Balance State

Above we observed that in fact GD optimization will lead to the *convoluted* PSO equilibrium. This can also be derived from the first-order-conditions argument as follows. Assume that at the convergence of PSO algorithm $\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta) = 0$ is satisfied, with θ containing final parameter values. Multiplying it by $\nabla_\theta f_\theta(X)^T$ will lead to:

$$\frac{1}{N^U}\sum_{i=1}^{N^U} M^U\left[X_i^U, f_\theta(X_i^U)\right] \cdot g_\theta(X, X_i^U) = \frac{1}{N^D}\sum_{i=1}^{N^D} M^D\left[X_i^D, f_\theta(X_i^D)\right] \cdot g_\theta(X, X_i^D). \quad (7.9)$$

According to the weak law of large numbers the above equality converges in probability (under appropriate regularity conditions) to Eq. (7.8) as batch sizes N^U and N^D increase.

Further, considering the asymptotic equilibrium $[G_\theta F_\theta](\cdot) = 0$, if the θ-dependent operator G_θ is injective then we also have $F_\theta(\cdot) = 0$ which leads to $F_\theta^U \equiv F_\theta^D$ and to the variational PSO *balance state* in Eq. (3.2). Yet, in general case during the optimization F_θ will project into the null-space of G_θ, with $F_\theta \not\equiv 0$. Moreover, even for injective G_θ it may take prohibitively many GD iterations in order to obtain the *convoluted* PSO equilibrium in Eq. (7.8), depending on the conditional number of G_θ. Hence, at the end of a typical optimization F_θ is expected to be outside of (orthogonal to) G_θ's high-spectrum space (i.e. a space spanned by G_θ's eigenfunctions related to its highest eigenvalues), and to be within G_θ's null-space or its low-spectrum space (associated with the lowest eigenvalues).

Since the low-spectrum is affiliated with high-frequency functions [116], $F_\theta \not\equiv 0$ will resemble some sort of a noise function. That is, during GD optimization the Fourier transform $\hat{F}_\theta(\xi)$ of $F_\theta(X)$ is losing its energy around the origin, yet it almost does not change at frequencies ξ whose norm is large. See [63] for the empirical evidence of the above conclusions and for additional analysis of G_θ's role in a least-squares optimization.

The above described behavior implicitly introduces a bias into PSO solution. Namely, when a function space $\mathcal{F}$ with particular g_θ and G_θ is chosen, this decision will affect our learning task exactly via the above relation to G_θ's null-space. The deeper analysis is required to answer the following questions: How closely are two equilibriums in Eq. (7.8) and Eq. (7.9)? What is the impact of batch sizes N^U and N^D, and what can we say about PSO solution when these batches are finite/small? How g_θ's properties, specifically its eigenvalues and eigenfunctions, will effect the PSO solution? What is the rate of converge towards G_θ's null-space? And how these aspects behave in a setting of stochastic mini-batch optimization? These advanced questions share their key concepts with the topics of RKHS estimation and Deep Learning theory, and deserve their own avenue. Therefore, we leave most of them out of this thesis's scope and shall address them as part of future research. Further, below we analyze a specific property of g_θ - its bandwidth.

7.5 Model Expressiveness and Smoothness vs Kernel Bandwidth

The model kernel $g_\theta(X, X')$ expresses the impact over $f_\theta(X)$ when we optimize at a data point X'. Intuitively, under the physical perspective (see Section 3) PSO algorithm can be viewed as pushing (*up* and *down*) at the training points with some wand whose end's shape is described by the above kernel. Here we will show that the flexibility of the surface f_θ strongly depends on $g_\theta(X, X')$'s bandwidth (i.e. on flatness of the pushing wand's end).

For this purpose we will define a notion of the model relative kernel:

$$r_\theta(X, X') \triangleq \frac{g_\theta(X, X')}{g_\theta(X, X)}, \tag{7.10}$$

which can be interpreted as a *relative* side-influence over $f_\theta(X)$ from X', scaled w.r.t. the self-influence $g_\theta(X, X)$ of X. Further, assume that the model relative kernel is bounded as:

$$0 < \exp\left[-\frac{d(X, X')}{h_{min}}\right] \leq r_\theta(X, X') \leq \exp\left[-\frac{d(X, X')}{h_{max}}\right] \leq 1, \tag{7.11}$$

where $d(X, X')$ is any function that satisfies the triangle inequality $d(X, X') \leq d(X, X'') + d(X', X'')$ (e.g. metric or pseudometric over $\mathbb{R}^n$), and where h_{min} and h_{max} can be considered as lower and upper bounds on $r_\theta(X, X')$'s bandwidth. Below, we will explore how h_{min} and h_{max} effect the smoothness of f_θ. Note, that in case $g_\theta(X, X)$ is identical for any X, the below analysis can be performed w.r.t. properties of g_θ instead of r_θ. However, in case f_θ is NN, the NTK g_θ can not be bounded as in Eq. (7.11), yet its scaled version r_θ clearly manifests such bounded-bandwidth properties. That is, $r_\theta(X, X')$ of NN typically decreases when the distance between X and X' increases (see Section 11), exhibiting some implicit bandwidth induced by a NN architecture. Moreover, while the magnitude of g_θ can be quiet different for various NN models and architectures, the normalized r_θ is on the same scale, allowing to compare the smoothness properties of particular models. Likewise, Eq. (7.11) is satisfied by many popular kernels used for RKHS construction, such as Gaussian and Laplacian kernels, making the below analysis relevant also for kernel models.

Theorem 19 (Spike Convergence). *Assume:*

1. *PSO algorithm converged, with* $\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta) = 0$.

2. $\{M^U, M^D\}$ *are non-negative functions.*

3. $M^U[X', s]$ *is continuous and strictly decreasing w.r.t.* s, *at* $\forall X' \in \mathbb{S}^U$.

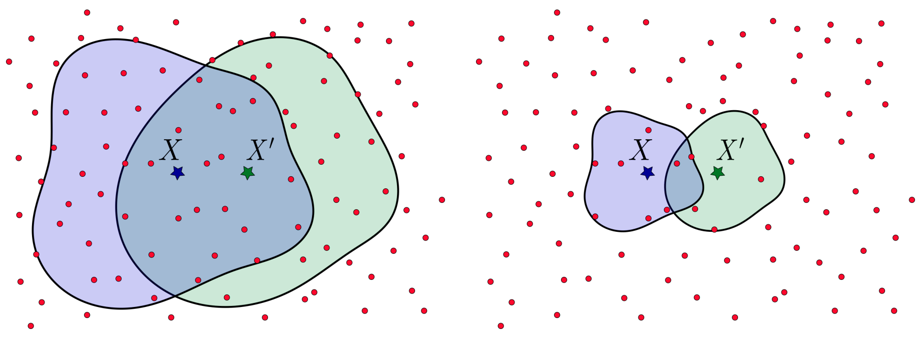


Figure 7.1: Model flexibility vs *influence* decay rate (bandwidth) of *gradient similarity* g_θ. Assume for simplicity $\forall X : g_\theta(X, X) \equiv \gamma$ for some constant γ. Considering Eq. (7.5), the *differential* at X is a weighted average of terms belonging to the training points around X, where $g_\theta(X, \cdot)$ serves as a weighting coefficient for each term. Further, consider *gradient similarity* to have a local-support. Its *influence* decay can be seen to express an *influence* area around each point X outside of which the *gradient similarity* $g_\theta(X, \cdot)$ becomes very small and negligible, on average. Above we illustrate two possible scenarios where the *influence* area is (a) large and (b) small. Red points represent training points, from both $\mathbb{P}^U$ and $\mathbb{P}^D$. Further, blue and green regions around points X and $X' = X + \Delta$ express the neighborhoods around the points where *gradient similarity* has large values. Note that in context of NNs these regions in general are not centered at X (or X') and are not symmetric, yet exhibit a particular *influence* decay rate (see Section 11 and Appendix H for the empirical evaluation of g_θ). The training points in each such region around some point X can be considered as *support* training points of X that will influence its surface height. As observed from plots, when the *influence* decay rate is low (i.e. large *influence* area, see plot (a)) the *differentials* of X and X' will be very similar since most of the *support* training points stay the same for both X and X'. In contrast, when the *influence* decay rate is high (see plot (b)), the *differentials* at X and X' will be very different since most of the *support* training points for both X and X' are not the same. Hence, the *differential* as a function of X changes significantly for a step Δ within the input space when the decay rate is high, and vice versa. Furthermore, when the *differential* $df_\theta(X)$ changes only slightly for different points, the overall update of the surface height at each point is similar/identical to other points. Such surface is pushed *up*/*down* as one physical rigid body, making $f_\theta(X)$ "inelastic." Moreover, when the *influence* decay rate is significantly high, the point X may have only a single *support* (*up* or *down*) training point and $f_\theta(X)$ will be pushed only in a single direction (*up* or *down*) yielding the spike near X. Finally, in case the *influence* area of X will not contain any training point, $f_\theta(X)$ will stay constant along the entire optimization.

Denote by $(M^U)^{-1}[X', z]$ *the inverse function of* M^U *w.r.t. second argument. Further, consider any training sample from* $\mathbb{P}^U$ *and denote it by* X*. Then the following is satisfied:*

1. $f_\theta(X) \geq (M^U)^{-1}[X, \alpha]$ *where* $\alpha = \frac{N^U}{N^D} \sum_{i=1}^{N^D} M^D[X_i^D, f_\theta(X_i^D)] \cdot \exp\left[-\frac{d(X, X_i^D)}{h_{max}}\right]$.

2. $(M^U)^{-1}[X, z]$ *is strictly decreasing w.r.t.* z*, with* $(M^U)^{-1}[X, \alpha] \to \infty$ *when* $\alpha \to 0$.

Proof of the above theorem is in Appendix D. From it we can see that for smaller α the surface at any *up* training point X converges to some very high height, where $f_\theta(X)$ can be arbitrary big. This can happen when X is faraway from all *down* samples $\{X_i^D\}_{i=1}^{N^D}$ (i.e. causing $d(X, X_i^D)$ to have a large output), or when h_{max} is very small (i.e. r_θ has a very narrow bandwidth). In these cases we will have *up* spikes at locations $\{X_i^U\}_{i=1}^{N^U}$. Similarly, when h_{max} is very small, there will be *down* spikes at locations $\{X_i^D\}_{i=1}^{N^D}$ - the corresponding theorem is symmetric to Theorem 19 and thus is omitted. Hence, the above theorem states that a very narrow bandwidth of r_θ will cause at the convergence *up* and *down* spikes within the surface f_θ, which can be interpreted as an overfitting behavior of PSO algorithm (see Section 12 for the empirical demonstration). Note that the theorem's assumptions are not very restrictive, with many PSO instances satisfying them such as PSO-LDE, NCE and logistic loss (see Section 5). Providing a more general theorem with less assumptions (specifically reducing the assumption 3) we shall leave for future work.

Theorem 20 (Change Difference). *Consider the differential df_θ defined in Eq. (7.5). For any two points X_1 and X_2, their change difference is bounded as:*

$$|df_\theta(X_1) - df_\theta(X_2)| \leq \delta \cdot g_\theta(X_1, X_1) \cdot \left[\frac{1}{N^U} \sum_{i=1}^{N^U} |M^U [X_i^U, f_\theta(X_i^U)]| \cdot \nu_\theta [X_1, X_2, X_i^U] + \right.$$
$$\left. + \frac{1}{N^D} \sum_{i=1}^{N^D} |M^D [X_i^D, f_\theta(X_i^D)]| \cdot \nu_\theta [X_1, X_2, X_i^D] \right], \quad (7.12)$$

where

$$\nu_\theta [X_1, X_2, X] \triangleq \epsilon [X_1, X_2, X] + \frac{|g_\theta(X_1, X_1) - g_\theta(X_2, X_2)|}{g_\theta(X_1, X_1)}, \quad (7.13)$$

$$\epsilon [X_1, X_2, X] \triangleq 1 - \exp\left[-\frac{1}{h_{min}} d(X_1, X_2)\right] \cdot \exp\left[-\frac{1}{h_{min}} \max [d(X_1, X), d(X_2, X)]\right]. \quad (7.14)$$

Proof of the above theorem is in Appendix E. In the above relation we can see that ϵ is smaller for a smaller distance $d(X_1, X_2)$. Likewise, $\epsilon \to 0$ along with $h_{min} \to \infty$. Neglecting the second term in ν_θ's definition ($g_\theta(X_1, X_1)$ and $g_\theta(X_2, X_2)$ are typically very similar for any two close-by points X_1 and X_2), ν_θ has analogous trends. Therefore, the upper bound in Eq. (7.12) is smaller when two points are nearby or when h_{min} is large. From this we can conclude that $f_\theta(X_1)$ and $f_\theta(X_2)$ are evolving in a similar manner for the above specified setting. Particularly, for $h_{min} \to \infty$ (and if the second term of ν_θ is relatively small) the entire surface f_θ will change almost identically at each point, intuitively resembling a rigid geometric body that moves *up* and *down* without changing its internal shape.

To conclude, when we decide which function space $\mathcal{F}$ to use for PSO optimization, this decision is equivalent to choosing the model kernel g_θ with the most desired properties. The effect of g_θ's bandwidth on the optimization is described by the above theorems, whose intuitive summary is given in Figure 7.1. In particular, when a bandwidth of the (relative) kernel is too narrow - there will be spikes at the training points, and when this bandwidth is too wide - the converged surface will be overly smoothed. These two extreme scenarios are also known as *overfitting* and *underfitting*, and the kernel bandwidth can be considered as a flexibility parameter of the surface f_θ. Furthermore, the above exposition agrees with existing works for RKHS models [107] where the kernel bandwidth is known to affect the estimation bias-variance trade-off.

7.6 Infinite Height Problem and its Solutions

In this section we study main stability issues encountered due to a mismatch between supports of $\mathbb{P}^U$ and $\mathbb{P}^D$. From part 2 of Theorem 1 we see that there are settings under which $f_\theta(X)$ will go to infinity at X outside of mutual support $\mathbb{S}^{U \cap D}$. In Figure 7.2 a simple experiment is shown that supports this conclusion empirically, where $f_\theta(X)$ at $X \in \mathbb{S}^{U \setminus D}$ is increasing during

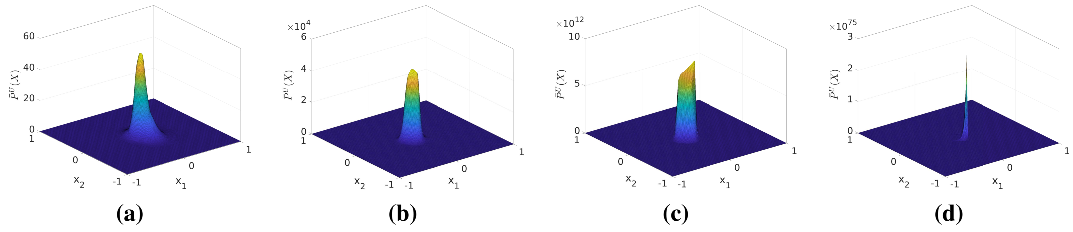


Figure 7.2: Illustration of PSO behavior in areas outside of the mutual support $\mathbb{S}^{U \cap D}$. We inferred 2D *Uniform* distribution $\mathbb{P}^U$ via $\bar{\mathbb{P}}^U(X) = \exp f_\theta(X)$ by using PSO-LDE with $\alpha = \frac{1}{4}$ (see Table 5.1 and Section 8.2). The applied NN architecture is block-diagonal with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). We plot $\bar{\mathbb{P}}^U(X)$ at different optimization times t: (a) $t = 100$, (b) $t = 200$, (c) $t = 10000$ and (d) $t = 100000$. The support $\mathbb{S}^U$ of $\mathbb{P}^U$ is $[-1, 1]$ for both dimensions. The chosen *down* density $\mathbb{P}^D$ is defined with $\mathbb{S}^D$ being identical to $\mathbb{S}^U$, minus the circle of radius 0.3 around the origin $(0, 0)$. In its entire support $\mathbb{P}^D$ is distributed uniformly. Thus, in this setup the zero-centered circle is outside of $\mathbb{S}^{U \cap D}$ - we have samples X_i^U there but no samples X_i^D. For this reason, there is only the *up* force F_θ^U that is present in the circle area, pushing the surface there indefinitely *up*. This can be observed from how the centered spire rises along the optimization time.

the entire learning process. Observe that while GD optimization obtains the *convoluted* PSO equilibrium in Eq. (7.8) instead of the variational PSO *balance state* in Eq. (3.2), the conclusions of Theorem 1 still remain valid in practice.

Similarly, the above described *infinite height* problem can happen at $X \in \mathbb{S}^{U \cap D}$ where one of the ratios $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$ and $\frac{\mathbb{P}^D(X)}{\mathbb{P}^U(X)}$ is too small, yet is not entirely zero (i.e. $|\log \mathbb{P}^U(X) - \log \mathbb{P}^D(X)|$ is large). Such *relative* support mismatch can cause instability as following. During the sampling process, at areas where $\mathbb{P}^U(X)$ and $\mathbb{P}^D(X)$ are very different, we can obtain many samples from one density yet almost no samples from the other. Taking Theorem 19 into the account, any training point X that is isolated from samples of the opposite force (i.e. when $d(X, \cdot)$ is large) will enforce a spike at $f_\theta(X)$. Moreover, when combined with the narrow kernel bandwidth, such spike behavior will be even more extreme with $f_\theta(X)$ being pushed to the *infinite height* (see Section 12 for the empirical illustration).

Hence, in both of the above cases f_θ will go to $\pm\infty$ at various locations. Meaning of this is lack of the optimization convergence. Furthermore, too large f_θ' outputs may cause arithmetic underflow and overflow instabilities when computing the loss gradient, and hence will lead to a divergence of the learning task.

In case $\mathbb{P}^U$ and $\mathbb{P}^D$ are relatively similar distributions and when g_θ's bandwidth is wide enough, the above problem of *infinite height* will not occur in practice. For other cases, there are two possible strategies to avoid the optimization divergence.

Indicator Magnitudes The above problem can be easily fixed by multiplying any given *magnitude* M^U (or M^D) with the following function:

$$reverse\_at\left[X, f_\theta(X), \varphi\right] = \begin{cases} -1, & \text{if } f_\theta(X) > \varphi \quad (\text{or } f_\theta(X) < \varphi) \\ 1, & \text{otherwise} \end{cases} \tag{7.15}$$

The $reverse\_at(\cdot)$ will change sign of near *magnitude* term when $f_\theta(X)$ at the training point X passes the threshold height φ. This in turn will change a direction of the force, making it to oscillate the surface $f_\theta(X)$ around φ (similarly to part 2-d of Theorem 1). Such behavior will

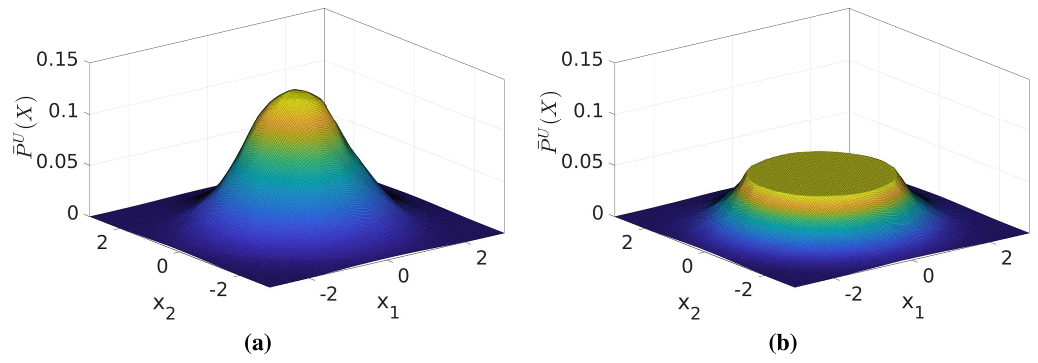


Figure 7.3: Impact illustration of the function $cut\_at\,[X, f_\theta(X), \varphi]$. (a) We inferred 2D *Normal* distribution via $\bar{\mathbb{P}}^U(X) = \exp f_\theta(X)$ by using PSO-LDE with $\alpha = \frac{1}{4}$ (see Table 5.1 and Section 8.2). The applied NN architecture is block-diagonal with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). (b) We performed the same learning algorithm as in (a), but with modified *up magnitude* function $\bar{M}^U\,[X, f_\theta(X)] = M^U\,[X, f_\theta(X)] \cdot cut\_at\,[X, f_\theta(X), -3]$, where cut_at is defined in Eq. (7.16). This function deactivates *up* pushes for points with $f_\theta(X) > -3$ (and $\exp f_\theta(X) > 0.0498$), thus the surface $f_\theta(X)$ at height above threshold -3 is only pushed by *down* force and hence is enforced to converge to the threshold. Yet, note that at points where the converged model satisfies $f_\theta(X) \leq -3$ the convergence is the same as in (a).

happen only at the "problematic" areas where $f_\theta(X)$ got too high/low. In other "safe" areas the function $reverse\_at(\cdot)$ will not have any impact. Hence, applying $reverse\_at(\cdot)$ next to both M^U and M^D will enforce the surface height at all X to be between some minimal and maximal thresholds, improving in this way the optimization stability.

The alternative to the above function is:

$$cut\_at\,[X, f_\theta(X), \varphi] = \begin{cases} 0, & \text{if } f_\theta(X) > \varphi \quad (\text{or } f_\theta(X) < \varphi) \\ 1, & \text{otherwise} \end{cases} \tag{7.16}$$

In contrast to $reverse\_at(\cdot)$, once the surface $f_\theta(X)$ at some training point X passed the threshold height φ, the corresponding gradient term of X (either *up* or *down*) is withdrawn from the total gradient $\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)$ in Eq. (3.1), entirely deactivating the influence of X on the learning task. Thus, the surface f_θ is not pushed anymore in areas where its height got too high/low; yet, it is still pushed at samples of the opposite force. Such force composition will constrain the surface height at unbalanced areas to converge to the threshold height φ, similarly to $reverse\_at(\cdot)$. Further, unlike $reverse\_at(\cdot)$, once any particular training point X got disabled by $cut\_at(\cdot)$, it also does not have a side-influence (via $g_\theta(X, X')$) on the surface at other points. Likewise, it stops affecting the value of θ, leading to a higher movement freedom within θ-space. Empirically we observed $cut\_at(\cdot)$ to result in overall higher approximation accuracy compared to $reverse\_at(\cdot)$. The optimization outcome of $cut\_at(\cdot)$ usage is illustrated in Figure 7.3.

Restriction over Functions' Range Alternatively, we can enforce all functions within the considered function space $\mathcal{F}$ to have any desired range $\mathbb{A} = [H_{min}, H_{max}]$. The constants H_{min} and H_{max} will represents the minimal and maximal surface heights respectively. For example, this can be accomplished by using the model:

$$f_\theta(X) = \frac{1}{2} \cdot [H_{max} - H_{min}] \cdot \tanh[h_\theta(X)] + \frac{1}{2} \cdot [H_{max} + H_{min}], \tag{7.17}$$

where h_θ represents an inner model that may have unbounded outputs. Since $\tanh(\cdot)$ is bounded to have values in $[-1, 1]$, it is easy to verify that the above model can return only values between H_{min} and H_{max}. Thus, using such model will eliminate the divergence of the surface to an infinite height. Likewise, other bounded functions can be used instead of $\tanh(\cdot)$ such as $\text{erf}(\cdot)$, sigmoid$(\cdot)$, $\arctan(\cdot)$ and many others.

While the impact of both above strategies is intuitive and simple - prevention of f_θ from being pushed beyond pre-defined thresholds, the rigorous math proof is more difficult to obtain. First strategy induces estimators within PSO non-differentiable family presented in Section 4.1.1 - the setting that is not analyzed by this work. Second strategy leads to PSO functionals minimized over a function space, whose range may not contain the entire convergence interval $\mathbb{K}$. A detailed analysis of the above special cases is left for future work since they are not the main focus of this thesis.

CHAPTER 8

Density Estimation via PSO

Till now we discussed a general formulation of PSO, where the presented analysis addressed properties of any PSO instance. In this section we focus in more detail on groups of PSO instances that can be applied for the density estimation problem, denoted above as $\mathbb{A}$ and $\mathbb{B}$. In particular, in Section 8.1 we shortly describe our previous work on density estimation, while in Section 8.2 we explore new PSO approaches to infer density on a logarithmic scale, with bounded *magnitude* functions that lead to the enhanced optimization performance.

8.1 DeepPDF

Here we briefly describe the density estimation approach, DeepPDF, introduced in [61], as a particular instance of the PSO paradigm. The density estimation problem involves learning a pdf function $\mathbb{P}^U(X)$ from a dataset of i.i.d. samples $\{X_i^U\}$. For this purpose, the proposed *pdf loss* was defined as:

$$L_{pdf}(\theta) = - \underset{X \sim \mathbb{P}^U}{\mathbb{E}} f_\theta(X) \cdot \mathbb{P}^D(X) + \underset{X \sim \mathbb{P}^D}{\mathbb{E}} \frac{1}{2}\Big[f_\theta(X)\Big]^2, \tag{8.1}$$

with corresponding *magnitude* functions $M^U[X, f_\theta(X)] = \mathbb{P}^D(X)$ and $M^D[X, f_\theta(X)] = f_\theta(X)$. $\mathbb{P}^D$ is an arbitrary density with a known pdf function which can be easily sampled.

The above loss is a specific instance of PSO with *balance state* achieved when the surface $f_\theta(X)$ converges to $\mathbb{P}^U(X)$ point-wise $\forall X \in \mathbb{S}^{U \cap D}$ (see Theorem 5), with the corresponding convergence interval being $\mathbb{K} = \mathbb{R}_{>0}$. Since conditions of Theorem 6 hold, no restriction over $\mathcal{F}$'s range is required. Moreover, according to the condition 3 in Theorem 7 upon the convergence $f_\theta(X)$ at $X \in \mathbb{S}^{U \setminus D}$ can be arbitrary - M^U is zero in the area $\mathbb{S}^{U \setminus D}$ and hence the force F_θ^U is disabled. Due to similar Theorem 8, at $X \in \mathbb{S}^{D \setminus U}$ the optimal solution must satisfy $f_\theta(X) = 0$. Therefore, any candidate for $\mathbb{P}^D$ with $\mathbb{S}^U \subseteq \mathbb{S}^D$ will lead to the convergence $\forall X \in \mathbb{S}^{U \cup D} : f_\theta(X) = \mathbb{P}^U(X)$. Outside of the support union $\mathbb{S}^{U \cup D}$ a convergence can be arbitrary, depending of the model kernel, which is true for any PSO method.

Concluding from the above, selected $\mathbb{P}^D$ must satisfy $\mathbb{S}^U \subseteq \mathbb{S}^D$. In our experiments we typically use a Uniform distribution for *down* density $\mathbb{P}^D$ (yet in practice any density can be applied). The minimum and maximum for each dimension of $\mathbb{P}^D$'s support are assigned to minimum and maximum of the same dimension from the available $\mathbb{P}^U$'s data points. Thus, the available samples $\{X_i^U\}$ define n-dimensional *hyperrectangle* in $\mathbb{R}^n$ as support of $\mathbb{P}^D$, with $\mathbb{P}^U$'s support being its subset. Inside this *hyperrectangle* the surface is pushed by F_θ^U and F_θ^D. Note that if borders of this support *hyperrectangle* can not be computed a priori (e.g. active learning), the $reverse\_at(\cdot)$ and $cut\_at(\cdot)$ functions can be used to prevent a possible optimization divergence as described in Section 7.6.

After training is finished, the converged $f_\theta(X)$ may have slightly negative values at points $\{X \in \mathbb{S}^{D \setminus U}\}$ being that during optimization the oscillation around height zero is stochastic in nature. Moreover, surface values outside of the *hyperrectangle* may be anything since the $f_\theta(X)$ was not optimized there. In order to deal with these possible inconsistencies, we can use the following *proxy* function as our estimation of target $\mathbb{P}^U(X)$:

$$\bar{f}_\theta(X) = \begin{cases} 0, & \text{if } f_\theta(X) < 0 \text{ or } \mathbb{P}^D(X) = 0 \\ f_\theta(X), & \text{otherwise} \end{cases} \tag{8.2}$$

which produces the desirable convergence $\forall X \in \mathbb{R}^n : \bar{f}_\theta(X) = \mathbb{P}^U(X)$.

In [61] we demonstrated that the above DeepPDF method with f_θ parametrized by NN outperforms the kernel density estimation (KDE) in an inference accuracy, and is significantly faster at the query stage when the number of training points is large.

8.2 PSO-LDE - Density Estimation on Logarithmic Scale

Typically, the output from a multidimensional density $\mathbb{P}^U(X)$ will tend to be extra small, where higher data dimension causes smaller pdf values. Representing very small numbers in a computer system may cause underflow and precision-loss problems. To overcome this, in general it is recommended to represent such small numbers at a logarithmic scale. Furthermore, the estimation of log-pdf is highly useful. For example, in context of robotics it can represent log-likelihood of sensor measurement and can be directly applied to infer an unobservable state of robot [62]. Likewise, once log-pdf $\log \mathbb{P}^U(X)$ is learned its average for data samples approximates the entropy of $\mathbb{P}^U$, which can further be used for robot planning [60].

Here we derive several estimator families from PSO subgroup $\mathbb{B}$ that infers logarithm of a pdf, $\log \mathbb{P}^U(X)$, as its target function. Although some members of these families were already reported before (e.g. NCE, [39]), the general formulation of these families was not considered previously. Further, presented below PSO instances with the convergence $\log \mathbb{P}^U(X)$ can be separated into two groups - instances with unbounded and bounded *magnitude* functions $\{M^U, M^D\}$. As was discussed in Section 7.2 and as will be shown in Section 13, the latter group yields a better optimization stability and also produces a higher accuracy.

According to Table 5.6, PSO convergence of the subgroup $\mathbb{B}$ is described by $T[X, z] = \log z + \log \mathbb{P}^D(X)$, with its inverse being $R[X, s] = \frac{\exp s}{\mathbb{P}^D(X)}$. Further, according to Theorem 4

Loss Version	***L**oss* / $M^U(\cdot)$ ***and*** $M^D(\cdot)$
1	$\underline{\text{L}}$: $-\mathbb{E}_{X\sim\mathbb{P}^U} f_\theta(X)\cdot\mathbb{P}^D(X)+\mathbb{E}_{X\sim\mathbb{P}^D}\exp(f_\theta(X))$ M^U, M^D: $\mathbb{P}^D(X)$, $\exp(f_\theta(X))$
2	$\underline{\text{L}}$: $-\mathbb{E}_{X\sim\mathbb{P}^U} f_\theta(X)+\mathbb{E}_{X\sim\mathbb{P}^D}\frac{\exp(f_\theta(X))}{\mathbb{P}^D(X)}$ M^U, M^D: 1, $\frac{\exp(f_\theta(X))}{\mathbb{P}^D(X)}$
3	$\underline{\text{L}}$: $\mathbb{E}_{X\sim\mathbb{P}^U}\frac{\mathbb{P}^D(X)}{\exp(f_\theta(X))}+\mathbb{E}_{X\sim\mathbb{P}^D} f_\theta(X)$ M^U, M^D: $\frac{\mathbb{P}^D(X)}{\exp(f_\theta(X))}$, 1
4	$\underline{\text{L}}$: $\mathbb{E}_{X\sim\mathbb{P}^U} 2\cdot\exp\left[\frac{1}{2}\cdot\left(\log\mathbb{P}^D(X)-f_\theta(X)\right)\right]+$ $+\mathbb{E}_{X\sim\mathbb{P}^D} 2\cdot\exp\left[\frac{1}{2}\cdot\left(f_\theta(X)-\log\mathbb{P}^D(X)\right)\right]$ M^U, M^D: $\exp\left[\frac{1}{2}\cdot\left(\log\mathbb{P}^D(X)-f_\theta(X)\right)\right]$, $\exp\left[\frac{1}{2}\cdot\left(f_\theta(X)-\log\mathbb{P}^D(X)\right)\right]$
5	$\underline{\text{L}}$: $-\mathbb{E}_{X\sim\mathbb{P}^U}\exp(f_\theta(X))\cdot\mathbb{P}^D(X)+\mathbb{E}_{X\sim\mathbb{P}^D}\frac{1}{2}\cdot\exp(2\cdot f_\theta(X))$ M^U, M^D: $\mathbb{P}^D(X)\cdot\exp(f_\theta(X))$, $\exp(2\cdot f_\theta(X))$

Table 8.1: Several PSO Instances that converge to $f_\theta(X)=\log\mathbb{P}^U(X)$

the convergence interval is $\mathbb{K}=\mathbb{R}$. Then, any pair of continuous positive functions $\{M^U, M^D\}$ that satisfies:

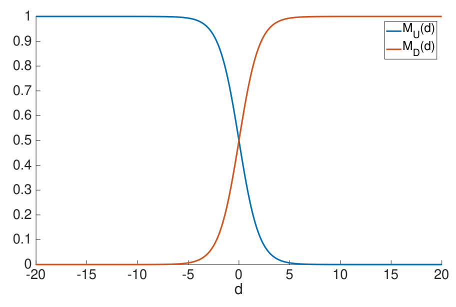
$$\frac{M^D(X, f_\theta(X))}{M^U(X, f_\theta(X))}=\frac{\exp f_\theta(X)}{\mathbb{P}^D(X)}, \tag{8.3}$$

will produce $f_\theta(X)=\log\mathbb{P}^U(X)$ at the convergence.

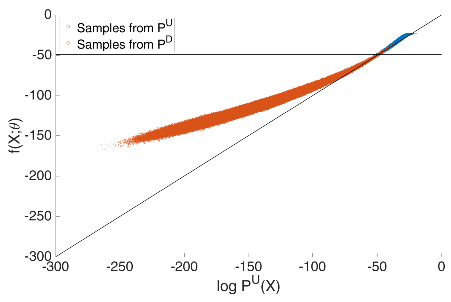
Unbounded *Magnitudes* To produce new PSO instances with the above equilibrium, infinitely many choices over $\{M^U, M^D\}$ can be taken. In Table 8.1 we show several such alternatives. Note that according to PSO we can merely move any term $q(X,s)$ from $M^U(X,s)$ into $M^D(X,s)$ as $\frac{1}{q(X,s)}$, and vice versa. Such modification will not change the PSO *balance state* and therefore allows for the exploration of various *magnitudes* that lead to the same convergence.

Remark 21. *Although we can see an obvious similarity and a relation between magnitudes in Table 8.1 (they all have the same ratio $M^D(\cdot)/M^U(\cdot)$), the corresponding losses have a much smaller resemblance. Without applying PSO rules, it would be hard to deduce that they all approximate the same target function.*

The Table 8.1 with acquired losses serves as a demonstration for simplicity of applying PSO concepts to forge new methods for the log-density estimation. However, produced losses have unbounded *magnitude* functions, and are not very stable during the real optimization, as will be shown in our experiments. The first loss in Table 8.1 can lead to precision problems since its *magnitudes* return (very) small outputs from $\mathbb{P}^D(X)$ and $\exp(f_\theta(X))$. Further, M^U of the third loss has devision by output from the current model $\exp(f_\theta(X))$ which is time-varying and can produce values arbitrarily close to zero. Likewise, methods 4 and 5 hold similar problems.



(a)



(b)

Figure 8.1: (a) NCE *magnitudes* as functions of a difference $\bar{d}[X, f_\theta(X)] \triangleq f_\theta(X) - \log \mathbb{P}^D(X)$. (b) Log density estimation via NCE for 20D data, where $\mathbb{P}^U$ is standard Normal 20D distribution and $\mathbb{P}^D$ is minimal Uniform 20D distribution that covers all samples from $\mathbb{P}^U$. Blue points are sampled from $\mathbb{P}^U$, while red points - from $\mathbb{P}^D$. The x axes represent $\log \mathbb{P}^U(X)$ for each sample, y axes - the surface height $f_\theta(X)$ after the optimization was finished. Diagonal line represents $f_\theta(X) = \log \mathbb{P}^U(X)$, where we would see all points in case of *perfect* model inference. The black horizontal line represents $\log \mathbb{P}^D(X) = -49$ which is constant for the Uniform density. As can be seen, these two densities have a *relative* support mismatch - the sampled points from both densities are obviously located mostly in different space neighborhoods; this can be concluded from values of $\log \mathbb{P}^U(X)$ that are very different for both point populations. Further, points with relatively small $|\bar{d}|$ (around the horizontal line) have a small estimation error since the ratio $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$ there is bounded and both points X^U and X^D are sampled from these areas. In contrast, we can see that in areas where $|\bar{d}| > \varepsilon$ for some positive constant ε we have samples only from one of the densities. Further, points with $\bar{d} > 0$ (*above* the horizontal line) are pushed *up* till a some threshold where the surface height is stuck due to *up magnitude* $M^U(\cdot)$ going to zero (see also Figure (a)). Additionally, points with $\bar{d} < 0$ (*below* the horizontal line) are pushed *down* till their *magnitude* $M^D(\cdot)$ also becomes zero. Note that *below* points are pushed further from the horizontal line than the *above* points. This is because the *above* points are near the origin (mean of Normal distribution) which is a less flexible area; it is side-influenced from all directions by surrounding *down* samples via g_θ, which prevents it from getting too high. On opposite, the *below* points are located far from the origin center on the edges of the considered point space. In these areas there are almost no samples from $\mathbb{P}^U$ and thus the surface is much more easily pushed *down*. Clearly, the PSO estimation task for the above choice of *up* and *down* densities can not yield a high accuracy, unless g_θ is a priori chosen in data-dependent manner (not considered in this thesis). Yet, we can see that NCE does not push the surface to $\pm\infty$ at the unbalanced areas.

Bounded *Magnitudes* Considering the above point, the PSO instances in Table 8.1 are sub-optimal. Instead, we want to find PSO losses with bounded $M^D(\cdot)$ and $M^U(\cdot)$. Further, the required relation in Eq. (8.3) between two *magnitudes* can be seen as:

$$\frac{M^D[X, f_\theta(X)]}{M^U[X, f_\theta(X)]} = \exp \bar{d}[X, f_\theta(X)], \tag{8.4}$$

$$\bar{d}[X, f_\theta(X)] \triangleq f_\theta(X) - \log \mathbb{P}^D(X). \tag{8.5}$$

$\bar{d}[X, f_\theta(X)]$ is a logarithm difference between the model surface and log-pdf of *down* density, which will play an essential role in *magnitude* functions below.

According to Section 7.2 and Lemma 17, from Eq. (8.4) we can produce the following family of PSO instances:

$$M^U[X, f_\theta(X)] = \frac{\mathbb{P}^D(X)}{D[X, f_\theta(X)]}, \quad M^D[X, f_\theta(X)] = \frac{\exp f_\theta(X)}{D[X, f_\theta(X)]}, \tag{8.6}$$

where the denominator function $D[X, f_\theta(X)] > 0$ takes the responsibility to normalize output of *magnitude* functions to be in some range $[0, \epsilon]$. Moreover, choice of $D[X, f_\theta(X)]$ does not affect the PSO *balance state*; it is reduced when the above *magnitudes* are introduced into Eq. (8.3).

To bound functions $M^D(\cdot)$ and $M^U(\cdot)$ in Eq. (8.6), $D[X, f_\theta(X)]$ can take infinitely many forms. One such form, that was implicitly applied by NCE technique [39, 126], is $D[X, f_\theta(X)] = \exp f_\theta(X) + \mathbb{P}^D(X)$ (see also Table 5.1). Such choice of normalization enforces outputs of both *magnitude* functions in Eq. (8.6) to be between 0 and 1. Moreover, NCE *magnitudes* can be seen as functions of a logarithm difference $\bar{d}[X, f_\theta(X)]$ in Eq. (8.5), $M^U(\bar{d}) = sigmoid(-\bar{d}[X, f_\theta(X)])$ and $M^D(\bar{d}) = sigmoid(\bar{d}[X, f_\theta(X)])$. Thus, an output of *magnitude* functions at point $X \in \mathbb{R}^n$ entirely depends on this logarithm difference at X. Furthermore, the *up magnitude* reduces to zero for a large positive $\bar{d}$ and the *down magnitude* reduces to zero for a large negative $\bar{d}$ (see also Figure 8.1a).

Such property, produced by bounding *magnitudes*, is highly helpful and intuitively can be viewed as an elastic springy constraint over the surface f_θ; it prevents *infinite height* problem described in Section 7.6, even when *up* and *down* densities are very different and when their support does not match. In neighborhoods where we sample many points from $\mathbb{P}^U$ but almost no points from $\mathbb{P}^D$ (ratio $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$ is large), the surface is pushed indefinitely *up* through *up* term within PSO loss, as proved by Theorem 19. Yet, as it pushed higher, $\bar{d}$ for these neighborhoods becomes larger and thus the *up magnitude* $M^U(\bar{d})$ goes quickly to zero. Therefore, when at a specific point X the surface $f_\theta(X)$ was pushed *up* too far from $\log \mathbb{P}^D(X)$, the *up magnitude* at this point becomes almost zero hence deactivating the *up* force $F_\theta^U(X)$ at this X. The same logic also applies to *down* force $F_\theta^D(X)$ - in NCE this force is deactivated at points where $f_\theta(X)$ was pushed *down* too far from $\log \mathbb{P}^D(X)$.

Critically, since $f_\theta(X)$ approximates $\log \mathbb{P}^U(X)$, $\bar{d}[X, f_\theta(X)]$ can also be viewed as an estimation of $\log \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$. Therefore, the above exposition of NCE dynamics can be also summarized as follows. At points where logarithm difference $\log \mathbb{P}^U(X) - \log \mathbb{P}^D(X)$ is in some dynamical active range $[-\varepsilon, \varepsilon]$ for positive ε, the *up* and *down* forces will be active and will reach the equilibrium with $f_\theta(X) = \log \mathbb{P}^U(X)$. At points where $[\log \mathbb{P}^U(X) - \log \mathbb{P}^D(X)] > \varepsilon \Leftrightarrow \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)} > \exp \varepsilon$, the surface will be pushed *up* to height ε. And at points where $[\log \mathbb{P}^U(X) - \log \mathbb{P}^D(X)] < -\varepsilon \Leftrightarrow \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)} < \frac{1}{\exp \varepsilon}$, the surface will be pushed *down* to height $-\varepsilon$. Once the surface at some point X passes above the height ε or below the height $-\varepsilon$, the NCE loss stops pushing it due to (near) zero *magnitude* component. Yet, the side-influence induced by model kernel $g_\theta(X, X')$ from non-zero *magnitude* areas can still affect the surface height at X. The above NCE behavior is illustrated in Figure 8.1b where 20D log-density estimation is performed via NCE for Gaussian distribution $\mathbb{P}^U$ and Uniform distribution $\mathbb{P}^D$.

Remark 22. *Note that the scalar ε represents a sensitivity threshold, where pushes at points with $\bar{d} > \varepsilon \Leftrightarrow |M^U(\cdot)| < sigmoid(-\varepsilon)$ or at points with $\bar{d} < -\varepsilon \Leftrightarrow |M^D(\cdot)| < sigmoid(-\varepsilon)$ have a neglectable effect on the surface due to their small magnitude component. Such sensitivity is different for various functional spaces $f_\theta \in \mathcal{F}$; for some spaces a small change of θ can only insignificantly affect the surface $f_\theta(X)$, while causing huge impact in others. Hence, the value of ε depends on specific choice of $\mathcal{F}$ and of magnitude functions $M^U(\cdot)$ and $M^D(\cdot)$.*

The above described relationship between NCE *magnitude* functions and ratio $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$ is very beneficial in the context of density estimation, since it produces high accuracy for points with bounded density ratio $|\log \mathbb{P}^U(X) - \log \mathbb{P}^D(X)| \leq \varepsilon$ and it is not sensitive to instabilities of areas where $|\log \mathbb{P}^U(X) - \log \mathbb{P}^D(X)| > \varepsilon$. Thus, even for very different densities $\mathbb{P}^U$ and $\mathbb{P}^D$ the optimization process is still very stable. Further, in our experiments we observed NCE to be much more accurate than unbounded losses in Table 8.1.

Moreover, such dynamics are not limited only to the loss of NCE, and can actually be enforced through other PSO variants. Herein, we introduce a novel general algorithm family for PSO *log density estimators* (PSO-LDE) that takes a *normalized* form in Eq. (8.6). The denominator function is defined as $D^{\alpha}_{PSO\text{-}LDE}\left[X, f_\theta(X)\right] \triangleq \left[[\exp f_\theta(X)]^\alpha + [\mathbb{P}^D(X)]^\alpha\right]^{\frac{1}{\alpha}}$ with α being family's hyper-parameter. Particularly, each member of PSO-LDE has bounded *magnitude* functions:

$$M^U_\alpha\left[X, f_\theta(X)\right] = \frac{\mathbb{P}^D(X)}{\left[[\exp f_\theta(X)]^\alpha + [\mathbb{P}^D(X)]^\alpha\right]^{\frac{1}{\alpha}}} = \left[\exp\left[\alpha \cdot \bar{d}\left[X, f_\theta(X)\right]\right] + 1\right]^{-\frac{1}{\alpha}}, \quad (8.7)$$

$$M^D_\alpha\left[X, f_\theta(X)\right] = \frac{\exp f_\theta(X)}{\left[[\exp f_\theta(X)]^\alpha + [\mathbb{P}^D(X)]^\alpha\right]^{\frac{1}{\alpha}}} = \left[\exp\left[-\alpha \cdot \bar{d}\left[X, f_\theta(X)\right]\right] + 1\right]^{-\frac{1}{\alpha}}. \quad (8.8)$$

In Figure 8.2 the above *magnitude* functions are plotted w.r.t. logarithm difference $\bar{d}$, for different values of α. As can be observed, α controls the smoothness and the rate of a *magnitude* decay to zero. Specifically, for smaller α *magnitudes* go faster to zero, which implies that the aforementioned active range $[-\varepsilon, \varepsilon]$ is narrower. Thus, small α introduce some elasticity constraints over f_θ that induce smoothness of the converged model. We argue that these smoother dynamics of smaller α values allow for a more stable optimization and a more accurate convergence, similarly to the robustness of redescending M-estimators [124]. Yet, we leave the theoretical analysis of this affect for future work. In Section 13 we will empirically investigate the impact of α on the performance of density estimation, where we will see that $\alpha = \frac{1}{4}$ typically has a better performance.

Additionally, the formulation of PSO-LDE in Eqs. (8.7)-(8.8) can be exploited to overcome possible underflow and overflow issues. In a typically used single-precision floating-point format the function $\exp(\cdot)$ can only be computed for values in the range $[-81, 81]$. Hence, there is an upper bound for values of $|\bar{d}\left[X, f_\theta(X)\right]|$ above which $M^U_\alpha(X, f_\theta(X))$ and $M^D_\alpha(X, f_\theta(X))$ can not be computed in practice. Yet, we can set the hyper-parameter α to be small enough to overcome this numerical limitation.

Remark 23. *Note that NCE is a member of the above PSO-LDE family for $\alpha = 1$. Further, the analytic loss for magnitudes in Eqs. (8.7)-(8.8) is unknown for general α. Yet, the gradient of this loss can be easily calculated.*

To summarize, by replacing *pdf loss* in Eq. (8.1) with PSO-LDE we succeeded to increase approximation accuracy of density estimation. We show these results in Section 13. Furthermore, unlike typical density estimators, for both DeepPDF and PSO-LDE cases the total integral of the density estimator is not explicitly constrained to 1, yet was empirically observed to be very

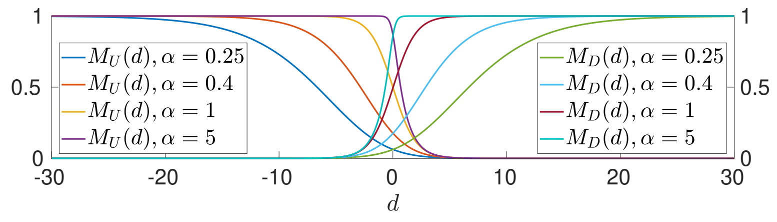


Figure 8.2: PSO-LDE *magnitudes* as functions of a difference $\bar{d} \triangleq f_\theta(X) - \log \mathbb{P}^D(X)$ for different values of a hyper-parameter α.

close to it. This implies that the proposed herein methods produce an *approximately* normalized density model. For many applications such approximate normalization is suitable. For example, in the estimation of a measurement likelihood model for Bayesian state inference in robotics [62] the model is required only to be proportional to the real measurement likelihood.

CHAPTER 9

Conditional Density Estimation

In this section we show how to utilize PSO *balance state* to infer conditional (ratio) density functions.

9.1 Conditional Density Estimation

Herein we will focus on problem of conditional density estimation, where i.i.d. samples of pairs $\{X_i^U, Y_i^U\}$ are given as:

$$\begin{array}{cc} & \begin{array}{cc} (\underline{\text{columns of } X^U}) & (\underline{\text{columns of } Y^U}) \end{array} \\ \mathcal{D} = & \left(\begin{array}{c|c} X_1^U & Y_1^U \\ X_2^U & Y_2^U \\ \vdots & \vdots \end{array}\right), \end{array} \quad \text{where } X_i^U \in \mathbb{R}^{n_x}, Y_i^U \in \mathbb{R}^{n_y}. \tag{9.1}$$

Again, we use U to refer to *up* force in PSO framework as will be described below. For any dataset $\mathcal{D}$, the generation process of its samples is governed by the following unknown data densities: $\mathbb{P}_{XY}^U(X, Y)$, $\mathbb{P}_X^U(X)$ and $\mathbb{P}_Y^U(Y)$. Specifically, in Eq. (9.1) rows under X^U columns will be distributed by marginal pdf $\mathbb{P}_X^U(X)$, rows under Y^U columns - by marginal pdf $\mathbb{P}_Y^U(Y)$, and entire rows of $\mathcal{D}$ will have the joint density $\mathbb{P}_{XY}^U(X, Y)$ (see also Table 9.1 for list of main notations). Likewise, these densities induce the conditional likelihoods $\mathbb{P}_{X|Y}^U(X|Y)$ and $\mathbb{P}_{Y|X}^U(Y|X)$, which can be formulated via Bayes theorem:

$$\mathbb{P}_{X|Y}^U(X|Y) = \frac{\mathbb{P}_{XY}^U(X, Y)}{\mathbb{P}_Y^U(Y)}, \quad \mathbb{P}_{Y|X}^U(Y|X) = \frac{\mathbb{P}_{XY}^U(X, Y)}{\mathbb{P}_X^U(X)}. \tag{9.2}$$

Depending on the task at hand, conditional pdf $\mathbb{P}_{X|Y}^U(X|Y)$ can produce valuable information about given data.

The simple way to infer $\mathbb{P}_{X|Y}^U(X|Y)$ is by first approximating separately the $\mathbb{P}_{XY}^U(X, Y)$

Notation	**Description**
$X^U \sim \mathbb{P}^U_X(X)$	n_x-dimensional random variable with marginal pdf $\mathbb{P}^U_X$
$Y^U \sim \mathbb{P}^U_Y(Y)$	n_y-dimensional random variable with marginal pdf $\mathbb{P}^U_Y$
$[X^U, Y^U]$	n-dimensional random variable with joint pdf $\mathbb{P}^U_{XY}(X, Y)$, at samples of which we push the model surface *up*
$n = n_x + n_y$	joint dimension of random variable $[X^U, Y^U]$
$\mathbb{P}^U_{X\|Y}(X\|Y)$	conditional probability density function of $X \equiv X^U$ given $Y \equiv Y^U$
$X^D \sim \mathbb{P}^D$	n_x-dimensional random variable with pdf $\mathbb{P}^D$
$[X^D, Y^D]$	n-dimensional random variable with joint pdf $\mathbb{P}^D(X) \cdot \mathbb{P}^U_Y(Y)$, at samples of which we push the model surface *down*

Table 9.1: Main Notations for Conditional Density Estimators

and $\mathbb{P}^U_Y(Y)$ from data samples (e.g. by using DeepPDF or PSO-LDE), and further applying Bayes theorem in Eq. (9.2). Yet, such method is not computationally efficient and typically is also not optimal, since approximation errors of both functions can produce even bigger error in the combined function.

A different technique, based on PSO principles, can be performed as follows. Consider a model (PSO surface) $f_\theta(X, Y) : \mathbb{R}^n \to \mathbb{R}$ with $n = n_x + n_y$, where the concatenated input $[X, Y]$ can be seen as the surface support. Define an arbitrary density $\mathbb{P}^D$ over $\mathbb{R}^{n_x}$ with a known pdf function which can be easily sampled (e.g. Uniform). Density $\mathbb{P}^D$ will serve as a *down* force to balance samples from $\mathbb{P}^U_X$, and thus is required to cover the support of $\mathbb{P}^U_X$. Further, $\mathbb{P}^U_{XY}(X, Y)$ will serve as *up* density in PSO framework, and its sample batch $\{X^U_i, Y^U_i\}_{i=1}^{N^U}$ will contain all rows from $\mathcal{D}$. As well, we will use $\mathbb{P}^D(X) \cdot \mathbb{P}^U_Y(Y)$ as *down* density. Corresponding samples $\{X^D_i, Y^D_i\}_{i=1}^{N^D}$ will be sampled in two steps. $\{Y^D_i\}_{i=1}^{N^D}$ are taken from $\mathcal{D}$ under Y^U columns; $\{X^D_i\}_{i=1}^{N^D}$ are sampled from $\mathbb{P}^D(X)$.

Considering the above setup, we can apply PSO to push $f_\theta(X, Y)$ via *up* and *down* forces. The optimization gradient will be identical to Eq. (3.1) where X^U_i and X^D_i are substituted by $\{X^U_i, Y^U_i\}$ and $\{X^D_i, Y^D_i\}$ respectively. For any particular $\{M^U, M^D\}$ the associated PSO *balance state* will be:

$$\frac{M^D\left[X, Y, f_\theta(X, Y)\right]}{M^U\left[X, Y, f_\theta(X, Y)\right]} = \frac{\mathbb{P}^U_{XY}(X, Y)}{\mathbb{P}^D(X) \cdot \mathbb{P}^U_Y(Y)} = \frac{\mathbb{P}^U_{X|Y}(X|Y)}{\mathbb{P}^D(X)}, \tag{9.3}$$

where we can observe $\mathbb{P}^U_{X|Y}(X|Y)$, which we aim to learn. Similarly to Section 5, below we formulate PSO subgroup for the conditional density estimation (or any function of it).

Theorem 24 (Conditional Density Estimation). *Denote the required PSO convergence by a transformation $T(X, Y, z) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ s.t. $f_\theta(X, Y) = T\left[X, Y, \mathbb{P}^U_{X|Y}(X|Y)\right]$ is the function we want to learn. Denote its inverse function w.r.t. z as $T^{-1}(X, Y, s)$. Then, any pair $\{M^U, M^D\}$ satisfying:*

$$\frac{M^D(X, Y, s)}{M^U(X, Y, s)} = \frac{T^{-1}(X, Y, s)}{\mathbb{P}^D(X)}, \tag{9.4}$$

will produce the required convergence.

The proof is trivial by noting that:

$$T^{-1}(X,Y,f_\theta(X,Y)) = \mathbb{P}^D(X)\cdot\frac{M^D(X,Y,f_\theta(X,Y))}{M^U(X,Y,f_\theta(X,Y))} = \mathbb{P}^D(X)\cdot\frac{\mathbb{P}^U_{X|Y}(X|Y)}{\mathbb{P}^D(X)} \quad \Rightarrow$$
$$\Rightarrow \quad T^{-1}(X,Y,f_\theta(X,Y)) = \mathbb{P}^U_{X|Y}(X|Y) \quad (9.5)$$

where we used both Eq. (9.3) and Eq. (9.4). From properties of inverse functions it follows:

$$T\left[X,Y,\mathbb{P}^U_{X|Y}(X|Y)\right] = T\left[X,Y,T^{-1}(X,Y,f_\theta(X,Y))\right] = f_\theta(X,Y). \quad (9.6)$$

Further, the sufficient conditions over mappings T, M^U and M^D are omitted since they already appear in Theorem 4.

Example 5: Consider a scenario where we would like to infer $f_\theta(X,Y) = \mathbb{P}^U_{X|Y}(X|Y)$. Thus, the PSO convergence is described by $T(X,Y,z) = z$. Its inverse is $T^{-1}(X,Y,s) = s$. Hence, *magnitude* functions must satisfy $\frac{M^D(X,Y,f_\theta(X,Y))}{M^U(X,Y,f_\theta(X,Y))} = \frac{f_\theta(X,Y)}{\mathbb{P}^D(X)}$. One choice for such *magnitudes* is $M^U[X,Y,f_\theta(X,Y)] = \mathbb{P}^D(X)$ and $M^D[X,Y,f_\theta(X,Y)] = f_\theta(X,Y)$, defined in Table 9.2 as "Conditional Density Estimation".

Example 6: Consider a scenario where we would like to infer $f_\theta(X,Y) = \log \mathbb{P}^U_{X|Y}(X|Y)$, which can be essential for high-dimensional data. The PSO convergence is described by $T(X,Y,z) = \log z$, and its inverse is $T^{-1}(X,Y,s) = \exp s$. Hence, *magnitude* functions must satisfy $\frac{M^D(X,Y,f_\theta(X,Y))}{M^U(X,Y,f_\theta(X,Y))} = \frac{\exp f_\theta(X,Y)}{\mathbb{P}^D(X)}$. One choice for such *magnitudes* is $M^U[X,Y,f_\theta(X,Y)] = \frac{\mathbb{P}^D(X)}{D(X,Y,f_\theta(X,Y))}$ and $M^D[X,Y,f_\theta(X,Y)] = \frac{\exp f_\theta(X,Y)}{D(X,Y,f_\theta(X,Y))}$, defined in Table 9.2 as "PSO-LDE Conditional Form". The denominator $D(X,Y,f_\theta(X,Y)) = [[\exp f_\theta(X,Y)]^\alpha + [\mathbb{P}^D(X)]^\alpha]^{\frac{1}{\alpha}}$ serves as a normalization to enforce *magnitudes* to be bounded functions, similarly to PSO-LDE method in Section 8.2.

Thus, we can estimate the conditional density, or any function of it, in a one-step algorithm by applying PSO procedure with *up* and *down* densities defined above. This again emphasizes the simplicity and usability of PSO formulation. Further, note that it is also possible to reuse sample Y_i^U as Y_i^D, since within the *down* term this sample will still be independent of X_i^D and its density is still the marginal $\mathbb{P}^U_Y(Y)$. Such reuse is popular for example in NCE methods [83, 84] in context of language modeling.

The above examples and several other options are listed in Table 9.2. Similarly to a case of the ordinary density estimation, also in the conditional case there are numerous PSO instances with the same target function $\mathbb{P}^U_{X|Y}(X|Y)$ (or $\log \mathbb{P}^U_{X|Y}(X|Y)$). Analyses of these techniques and search for the most "optimal" can be an interesting direction for future research.

Method	***F**inal* $f_\theta(X)$ / ***R**eferences* / ***L**oss* / $M^U(\cdot)$ *and* $M^D(\cdot)$
Conditional Density Estimation	F: $\mathbb{P}^U(X\mid Y)$ R: This thesis L: $-\mathbb{E}_{[X,Y]\sim\mathbb{P}^U_{XY}(X,Y)}\, f_\theta(X,Y)\cdot\mathbb{P}^D(X)+$ $+\mathbb{E}_{[X,Y]\sim\mathbb{P}^D(X)\cdot\mathbb{P}^U_Y(Y)}\,\frac{1}{2}\Big[f_\theta(X,Y)\Big]^2$ M^U, M^D: $\mathbb{P}^D(X)$, $f_\theta(X,Y)$
Conditional Log-density Estimation	F: $\log\mathbb{P}^U(X\mid Y)$ R: This thesis L: $-\mathbb{E}_{[X,Y]\sim\mathbb{P}^U_{XY}(X,Y)}\, f_\theta(X,Y)+\mathbb{E}_{[X,Y]\sim\mathbb{P}^D(X)\cdot\mathbb{P}^U_Y(Y)}\,\frac{\exp[f_\theta(X,Y)]}{\mathbb{P}^D(X)}$ M^U, M^D: 1, $\frac{\exp[f_\theta(X,Y)]}{\mathbb{P}^D(X)}$
NCE Conditional Form	F: $\log\mathbb{P}^U(X\mid Y)$ R: [83,84] L: $\mathbb{E}_{[X,Y]\sim\mathbb{P}^U_{XY}(X,Y)}\log\frac{\exp[f_\theta(X,Y)]+\mathbb{P}^D(X)}{\exp[f_\theta(X,Y)]}+$ $+\mathbb{E}_{[X,Y]\sim\mathbb{P}^D(X)\cdot\mathbb{P}^U_Y(Y)}\log\frac{\exp[f_\theta(X,Y)]+\mathbb{P}^D(X)}{\mathbb{P}^D(X)}$ M^U, M^D: $\frac{\mathbb{P}^D(X)}{\exp[f_\theta(X,Y)]+\mathbb{P}^D(X)}$, $\frac{\exp[f_\theta(X,Y)]}{\exp[f_\theta(X,Y)]+\mathbb{P}^D(X)}$
PSO-LDE Conditional Form	F: $\log\mathbb{P}^U(X\mid Y)$ R: This thesis L: unknown M^U, M^D: $\frac{\mathbb{P}^D(X)}{\left[[\exp f_\theta(X,Y)]^\alpha+[\mathbb{P}^D(X)]^\alpha\right]^{\frac{1}{\alpha}}}$, $\frac{\exp f_\theta(X,Y)}{\left[[\exp f_\theta(X,Y)]^\alpha+[\mathbb{P}^D(X)]^\alpha\right]^{\frac{1}{\alpha}}}$
Conditional GAN Critic	F: $\frac{\mathbb{P}^U_{X\mid Y}(X\mid Y)}{\mathbb{P}^U_{X\mid Y}(X\mid Y)+\mathbb{P}^D_\phi(X\mid Y)}$, where $\mathbb{P}^D_\phi(X\mid Y)$ is density of generator h_ϕ parametrized by ϕ R: [81] L: $-\mathbb{E}_{[X,Y]\sim\mathbb{P}^U_{XY}(X,Y)}\log f_\theta(X,Y)-$ $-\mathbb{E}_{[X,Y]\sim\mathbb{P}^D_\phi(X\mid Y)\cdot\mathbb{P}^U_Y(Y)}\log\Big[1-f_\theta(X,Y)\Big]$ M^U, M^D: $\frac{1}{f_\theta(X,Y)}$, $\frac{1}{1-f_\theta(X,Y)}$
Likelihood-Ratio with Logistic Loss	F: $\log\frac{\mathbb{P}^U_{X\mid Y}(X\mid Y)}{\mathbb{P}^D_\phi(X\mid Y)}$, where $\mathbb{P}^D_\phi(X\mid Y)$ is density of generator h_ϕ parametrized by ϕ R: This thesis L: $\mathbb{E}_{[X,Y]\sim\mathbb{P}^U_{XY}(X,Y)}\log\left[1+\exp[-f_\theta(X,Y)]\right]+$ $+\mathbb{E}_{[X,Y]\sim\mathbb{P}^D_\phi(X\mid Y)\cdot\mathbb{P}^U_Y(Y)}\log\left[1+\exp[f_\theta(X,Y)]\right]$ M^U, M^D: $\frac{1}{\exp[f_\theta(X,Y)]+1}$, $\frac{1}{\exp[-f_\theta(X,Y)]+1}$

Table 9.2: PSO Instances For Conditional Density (Ratio) Estimation, see Sections 9.1 and 9.2 for a detailed exposition of conditional PSO

9.2 Relation to Conditional GANs

Furthermore, a similar idea was also presented in the context of GANs, where a conditional generation of data (e.g. images given labels) was explored. Below we show its connection to PSO framework.

Denote the dataset $\mathcal{D}$ as in Eq. (9.1), where *real* sample pairs $\{X_i^U, Y_i^U\}$ are distributed according to unknown $\mathbb{P}_{XY}^U(X,Y)$. In Conditional GAN (cGAN) [76] the generator produces *fake* samples from the generator's conditional density $\mathbb{P}_\phi^D(X|Y)$, where we again use notations U and D to refer to PSO forces, as is described below. Density $\mathbb{P}_\phi^D(X|Y)$ is an implicit distribution of *fake* samples that are returned by the generator $h_\phi(\upsilon, Y)$ from the latent space $\upsilon \in \mathbb{R}^{n_\upsilon}$, where the label Y was a priori sampled from $\mathbb{P}_Y^U(Y)$; ϕ is a generator's parametrization. Further, the critic sees pairs $[X, Y]$ coming from $\mathcal{D}$ and from the generator, and tries to decide where the pair is originated from. This is done by estimating a statistical divergence between $\mathbb{P}_{X|Y}^U(X|Y)$ implicitly defined by $\mathcal{D}$, and between $\mathbb{P}_\phi^D(X|Y)$ implicitly defined by h_ϕ. Moreover, the divergence estimation is typically done by first inferring the ratio $\frac{\mathbb{P}_{X|Y}^U(X|Y)}{\mathbb{P}_\phi^D(X|Y)}$ (or some function of this ratio).

The proposed by [76] algorithm is identical to PSO procedure, when $\mathbb{P}_{XY}^U(X,Y)$ serves as *up* density, and $\mathbb{P}_\phi^D(X|Y) \cdot \mathbb{P}_Y^U(Y)$ - as *down* density. The *up* sample batch $\{X_i^U, Y_i^U\}_{i=1}^{N^U}$ will contain all rows from $\mathcal{D}$. Further, samples $\{X_i^D, Y_i^D\}_{i=1}^{N^D}$ from *down* density will be sampled in three steps. $\{Y_i^D\}_{i=1}^{N^D}$ are taken from $\mathcal{D}$ under Y^U columns; $\{\upsilon^i\}_{i=1}^{N^D}$ are sampled from generator's base distribution; $\{X_i^D\}_{i=1}^{N^D}$ are generator's outputs for inputs $\{Y_i^D, \upsilon_i\}_{i=1}^{N^D}$. Extending the setup of Section 9.1 to the above sampling procedure, any particular $\{M^U, M^D\}$ will produce PSO *balance state*:

$$\frac{M^D\left[X, Y, f_\theta(X,Y)\right]}{M^U\left[X, Y, f_\theta(X,Y)\right]} = \frac{\mathbb{P}_{XY}^U(X,Y)}{\mathbb{P}_\phi^D(X|Y)\cdot\mathbb{P}_Y^U(Y)} = \frac{\mathbb{P}_{X|Y}^U(X|Y)}{\mathbb{P}_\phi^D(X|Y)}, \tag{9.7}$$

where the conditional ratio shows up. Similarly to the conditional density estimation, below we formulate PSO subgroup for inference of this ratio (or any function of it).

Theorem 25 (Conditional Ratio Estimation). *Denote the required PSO convergence by a transformation $T(X,Y,z) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ s.t. $f_\theta(X,Y) = T\left[X, Y, \frac{\mathbb{P}_{X|Y}^U(X|Y)}{\mathbb{P}_\phi^D(X|Y)}\right]$ is the function we want to learn. Denote its inverse function w.r.t. z as $T^{-1}(X,Y,s)$. Then, any pair $\{M^U, M^D\}$ satisfying:*

$$\frac{M^D(X,Y,s)}{M^U(X,Y,s)} = T^{-1}(X,Y,s), \tag{9.8}$$

will produce the required convergence.

The proof is trivial by noting that:

$$T^{-1}(X,Y,f_\theta(X,Y)) = \frac{M^D(X,Y,f_\theta(X,Y))}{M^U(X,Y,f_\theta(X,Y))} = \frac{\mathbb{P}^U_{X|Y}(X|Y)}{\mathbb{P}^D_\phi(X|Y)} \quad \Rightarrow$$
$$\Rightarrow \quad T^{-1}(X,Y,f_\theta(X,Y)) = \frac{\mathbb{P}^U_{X|Y}(X|Y)}{\mathbb{P}^D_\phi(X|Y)}. \tag{9.9}$$

From properties of inverse functions the Theorem follows.

The cGAN method aimed to infer $\frac{\mathbb{P}^U_{X|Y}(X|Y)}{\mathbb{P}^U_{X|Y}(X|Y)+\mathbb{P}^D_\phi(X|Y)}$ to measure Jensen-Shannon divergence between *real* and *fake* distributions. This is associated with PSO convergence $T(X,Y,z) = \frac{z}{z+1}$ and the corresponding inverse $T^{-1}(X,Y,s) = \frac{s}{1-s}$. According to Theorem 25 the *magnitudes* must satisfy:

$$\frac{M^D(X,Y,f_\theta(X,Y))}{M^U(X,Y,f_\theta(X,Y))} = \frac{f_\theta(X,Y)}{1-f_\theta(X,Y)}, \tag{9.10}$$

with the specific choice $\{M^U(X,Y,f_\theta(X,Y)) = \frac{1}{f_\theta(X,Y)}, M^D(X,Y,f_\theta(X,Y)) = \frac{1}{1-f_\theta(X,Y)}\}$ selected by the critic loss in [76].

Hence, we can see that cGAN critic loss is a particular instance of PSO, when the sampling procedure of *up* and *down* samples is as described above. Further, two main problems of the classical cGAN critic are the not-logarithmic scale of the target function and unboundedness of *magnitude* functions (for a general model $f_\theta(X,Y)$). In Table 9.2 we propose a "Likelihood-Ratio with Logistic Loss" to learn $\log \frac{\mathbb{P}^U_{X|Y}(X|Y)}{\mathbb{P}^D_\phi(X|Y)}$ whose *magnitudes* are bounded. We argue such choice to be more stable during the optimization which will lead to a better accuracy. Moreover, for specific case when cGAN critic $f_\theta(X,Y)$ is parameterized as $sigmoid(h_\theta(X,Y))$ with $h_\theta(X,Y)$ being the inner model, cGAN critic loss can be shown to be reduced to the above logistic loss. Thus, with such parametrization the inner NN $h_\theta(X,Y)$ within cGAN critic will converge to $\log \frac{\mathbb{P}^U_{X|Y}(X|Y)}{\mathbb{P}^D_\phi(X|Y)}$.

CHAPTER 10

Additional Applications and Relations of PSO Framework

In this section we demonstrate how PSO principles can be exploited beyond the (conditional) pdf inference problem. Particularly, we relate PSO and cross-entropy loss, showing the latter to be a specific instance of the former. We also outline relation between PSO and MLE by deriving the latter from *PSO functional*. Further, we analyze PSO instance with unit *magnitude* functions and describe its connection to CD method [46]. Additionally, we show how to use PSO for learning mutual information from available data samples, and how to employ it in the solution of occupancy mapping.

10.1 Cross-Entropy as Instance of PSO

In this section we will show that the binary cross-entropy loss combined with a $sigmoid$ non-linearity, typical in binary classification problems, is instance of PSO. Further, in Appendix F we extend this setup also to a more general case of $softmax$ cross-entropy. Similarly to a binary $sigmoid$ cross-entropy, a multi-class $softmax$ cross-entropy is shown to be a PSO instance, extended to models with multi-dimensional outputs. Thus, the optimization of multi-class $softmax$ cross-entropy can be seen as pushes of dynamical forces over C different surfaces $\{f_\theta(X)_i\}_{i=1}^C$ - the outputs of the model $f_\theta(X) \in \mathbb{R}^C$ per each class.

To prove the above point, we derive the binary cross-entropy loss using PSO principles. Define training dataset of pairs $\{X_i, Y_i\}_{i=1}^N$ where $X_i \in \mathbb{R}^n$ is data point of an arbitrary dimension n (e.g. image) and Y_i is its label - the discrete number that takes values from $\{0, 1\}$. Denote by N_1 and N_0 the number of samples with labels 1 and 0 respectively. Further, assume each sample pair to be i.i.d. sampled from an unknown density $\mathbb{P}(X, Y) = \mathbb{P}(X) \cdot \mathbb{P}(Y|X)$. Our task is to enforce the output of $\sigma(f_\theta(X))$, the $sigmoid$ non-linearity over inner model

$f_\theta(X)$, to converge to unknown conditional $\mathbb{P}(Y=1|X)$. Such convergence is equivalent to

$$f_\theta(X) = -\log\left[\frac{1}{\mathbb{P}(Y=1|X)} - 1\right] = \log\frac{\mathbb{P}(X,Y=1)}{\mathbb{P}(X,Y=0)} = \log\frac{\mathbb{P}(X|Y=1)\cdot\mathbb{P}(Y=1)}{\mathbb{P}(X|Y=0)\cdot\mathbb{P}(Y=0)}. \tag{10.1}$$

To apply PSO, we consider $\mathbb{P}(X|Y=1)$ as *up* density $\mathbb{P}^U$ and $\mathbb{P}(X|Y=0)$ as *down* density $\mathbb{P}^D$. Sample batches $\{X_i^U\}_{i=1}^{N_1}$ and $\{X_i^D\}_{i=1}^{N_0}$ from both can be obtained by fetching X_i with appropriate label Y_i. Then the required convergence is described by $T(X,z) = \log\frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=0)}\cdot z$, with $f^*(X) = T(X, \frac{\mathbb{P}(X|Y=1)}{\mathbb{P}(X|Y=0)})$. Further, the T's inverse is $R(X,s) = \frac{\mathbb{P}(Y=0)}{\mathbb{P}(Y=1)}\cdot\exp s$, and according to Theorem 4 *magnitudes* must satisfy $\frac{M^D[X,f_\theta(X)]}{M^U[X,f_\theta(X)]} = \frac{\mathbb{P}(Y=0)\cdot\exp f_\theta(X)}{\mathbb{P}(Y=1)}$. One possible choice is:

$$M^U[X,f_\theta(X)] = \frac{\mathbb{P}(Y=1)}{1+\exp f_\theta(X)}, \quad M^D[X,f_\theta(X)] = \frac{\mathbb{P}(Y=0)\cdot\exp f_\theta(X)}{1+\exp f_\theta(X)} \tag{10.2}$$

where the denominator $1+\exp f_\theta(X)$ serves as a normalization factor that enforces $\{M^U, M^D\}$ to be between 0 and 1. Further, the above *magnitude* functions have known antiderivatives:

$$\widetilde{M}^U[X,f_\theta(X)] = \mathbb{P}(Y=1)\cdot\log\left[\sigma(f_\theta(X))\right], \ \widetilde{M}^D[X,f_\theta(X)] = -\mathbb{P}(Y=0)\cdot\log\left[1-\sigma(f_\theta(X))\right] \tag{10.3}$$

that produce the following *PSO functional*:

$$L_{PSO}(f) = -\mathop{\mathbb{E}}_{X\sim\mathbb{P}(X|Y=1)}\mathbb{P}(Y=1)\cdot\log\left[\sigma(f_\theta(X))\right] - \mathop{\mathbb{E}}_{X\sim\mathbb{P}(X|Y=0)}\mathbb{P}(Y=0)\cdot\log\left[1-\sigma(f_\theta(X))\right]. \tag{10.4}$$

Finally, considering $\frac{N_1}{N}$ and $\frac{N_0}{N}$ as estimators of $\mathbb{P}(Y=1)$ and $\mathbb{P}(Y=0)$ respectively, the empirical version of the above loss is:

$$\begin{aligned} L_{PSO}(f) &\approx -\frac{1}{N_1}\sum_{i=1}^{N_1}\frac{N_1}{N}\cdot\log\left[\sigma(f_\theta(X_i^U))\right] - \frac{1}{N_0}\sum_{i=1}^{N_0}\frac{N_0}{N}\cdot\log\left[1-\sigma(f_\theta(X_i^D))\right] = \\ &= -\frac{1}{N}\sum_{i=1}^{N}\Big[Y_i\cdot\log\left[\sigma(f_\theta(X_i))\right] + [1-Y_i]\cdot\log\left[1-\sigma(f_\theta(X_i))\right]\Big], \end{aligned} \tag{10.5}$$

where we combine two sums of the first row into a single sum after introducing indicators Y_i and $1-Y_i$.

The second row is known in Machine Learning community as the binary cross-entropy loss. Therefore, we can conclude that PSO instance with *magnitudes* in Eq. (10.2) corresponds to cross-entropy when $\mathbb{P}(X|Y=1)$ and $\mathbb{P}(X|Y=0)$ serve as *up* and *down* densities respectively. See a similar derivation for multi-class cross-entropy in Appendix F. Therefore, convergence and stability properties of PSO are also shared by the *supervised* classification domain which further motivates PSO analysis.

10.2 Relation to Maximum Likelihood Estimation

Below we establish the relation between MLE and PSO procedures, by deriving MLE approach from principles of PSO. Consider a batch of i.i.d. samples $\{X_i^U\}_{i=1}^{N^U}$ sampled from density $\mathbb{P}^U$, whose pdf we aim to estimate. Define an auxiliary distribution $\mathbb{P}^D$ with analytically known pdf $\mathbb{P}^D(X)$ that satisfies $\mathbb{S}^U \subseteq \mathbb{S}^D$, and define *PSO functional* as:

$$L_{PSO}(f) = - \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} [1 + \log f(X)] + \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \frac{f(X)}{\mathbb{P}^D(X)} \tag{10.6}$$

that is induced by the following *magnitude* functions:

$$M^U [X, f(X)] = \frac{1}{f(X)}, \quad M^D [X, f(X)] = \frac{1}{\mathbb{P}^D(X)}. \tag{10.7}$$

Define the hypothesis class $\mathcal{F}$ with functions that are positive on $\mathbb{S}^U$ so that $\log f(X)$ is properly defined for all $f \in \mathcal{F}$ and $X \in \mathbb{S}^U$. Then, the optimal $f^* = \arg\min_{f \in \mathcal{F}} L_{PSO}(f)$ will satisfy PSO *balance state* in Eq. (3.2) which yields $f^*(X) = \mathbb{P}^U(X)$.

Furthermore, in case $\mathcal{F}$ is a space of positive functions whose total integral is equal to 1 (i.e. probability measure space), the above loss can be reduced to:

$$L_{PSO}(f) = \int -\mathbb{P}^U(X) \cdot [1 + \log f(X)] + f(X) dX = - \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} \log f(X), \tag{10.8}$$

where we apply an equality $\int \mathbb{P}^U(X) dX = \int f(X) dX$ since f is normalized. Note that limiting $\mathcal{F}$ to be a probability measure space does not affect the *balance state* of PSO since the optimal solution f^* is also a probability measure. Further, considering the physical perspective of PSO such choice of $\mathcal{F}$ has an implicit regularization affect, removing a need for the *down* force F_θ^D in order to achieve the force equilibrium over the surface $f(X)$.

The loss in Eq. (10.8) and its empirical variant $L_{PSO}(f) \approx -\frac{1}{N^U} \sum_{i=1}^{N^U} \log f(X_i^U)$ define the standard MLE procedure. Therefore, we can conclude that PSO with *magnitudes* defined in Eq. (10.7) corresponds to MLE when the data distribution $\mathbb{P}^U$ is absolutely continuous w.r.t. $\mathbb{P}^D$ and when each $f \in \mathcal{F}$ is a normalized function. Likewise, such relation can also be explained by the connection between PSO and Kullback-Leibler (KL) divergencies described in Section 6.

10.3 PSO with Unit *Magnitudes* and Contrastive Divergence

The PSO instance with unit *magnitudes* $M^U [X, f_\theta(X)] = M^D [X, f_\theta(X)] = 1$ can be frequently met in Machine Learning (ML) literature. For example, Integral Probability Metrics (IPMs) [87], contrastive divergence (CD) [46], Maximum Mean Discrepancy (MMD) [34] and critic of the Wasserstein GAN [6] all rely on this loss to measure some distance between densities $\mathbb{P}^U$ and $\mathbb{P}^D$. In this section we will explore this *unit* loss

$$L_{unit}(f_\theta) = - \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} f_\theta(X) + \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} f_\theta(X) \tag{10.9}$$

in a context of the proposed PSO framework.

By following the derivation from Section 4.1, the inner minimization problem solved by $\inf_{f \in \mathcal{F}} L_{unit}(f)$ for each X is:

$$s^* = \arg\inf_{s \in \mathbb{R}} [-\mathbb{P}^U(X) + \mathbb{P}^D(X)] \cdot s. \tag{10.10}$$

Since it is linear in s, the optima s^* will be either $+\infty$ (if $\mathbb{P}^U(X) > \mathbb{P}^D(X)$) or $-\infty$ (if $\mathbb{P}^U(X) < \mathbb{P}^D(X)$). Using physical system perspective, we can say that given a flexible enough surface $f_\theta(X)$ (e.g. typical NN) the straight forward optimization via *unit* loss in Eq. (10.9) will diverge since forces $F^U_\theta(X) = \mathbb{P}^U(X)$ and $F^D_\theta(X) = \mathbb{P}^D(X)$ are actually independent of θ and cannot adapt to each other. That is, *balance state* $F^U_\theta(X) = F^D_\theta(X)$ can not be achieved by the *unit* loss. Thus, the model is pushed to $\pm\infty$ at various input points, up to the surface flexibility. During such optimization, training will eventually fail due to numerical instability that involves too large/small numbers. Furthermore, this point can be easily verified in practice by training NN with loss in Eq. (10.9).

CD One way to enforce the convergence of *unit* loss is by adapting/changing density $\mathbb{P}^D$ towards $\mathbb{P}^U$ along the optimization. Indeed, this is the main idea behind the CD method presented in [46] and further improved in [91] and [73]. In CD, the *down* density $\mathbb{P}^D(X)$ in Eq. (10.9) represents the current model distribution $\hat{\mathbb{P}}_\theta(X) \triangleq \exp[f_\theta(X)] / \int \exp[f_\theta(X')] dX'$, $\mathbb{P}^D \equiv \hat{\mathbb{P}}_\theta$. At each iteration, $\{X^D_i\}_{i=1}^{N^D}$ are sampled from $\hat{\mathbb{P}}_\theta(X)$ by Gibbs sampling [46], Monte Carlo with Langevin dynamics [51], Hybrid Monte Carlo sampling [91], or Stein Variational Gradient Descent (SVGD) [72, 73]. Thus, in CD algorithm forces $F^U_\theta(X) = \mathbb{P}^U(X)$ and $F^D_\theta(X) = \hat{\mathbb{P}}_\theta(X)$ are adapted to each other via their frequency components $\mathbb{P}^U$ and $\mathbb{P}^D$ instead of their *magnitude* components $M^U[X, f_\theta(X)]$ and $M^D[X, f_\theta(X)]$. The dynamics of such optimization will converge to the equilibrium only when $\mathbb{P}^U(X) = \hat{\mathbb{P}}_\theta(X)$ which will also lead to $\exp[f_\theta(X)] \propto \mathbb{P}^U(X)$.

WGAN We additionally consider the relation between PSO concepts and Wasserstein GAN [6] (WGAN) which has been recently proposed and is considered nowadays to be state-of-the-art. Apparently, the critic's loss in WGAN is exactly Eq. (10.9). It pushes the surface $f_\theta(X)$ *up* at points sampled from the real data distribution $\mathbb{P}^U$, and pushes *down* at points sampled from the generator density $\mathbb{P}^D_\phi$, which is an implicit distribution of fake samples returned by a generator from the latent space, with ϕ being a generator parametrization.

The critic's loss of WGAN was chosen as proxy to force critic's output to approximate Earth Mover (Wasserstein) distance between $\mathbb{P}^U$ and $\mathbb{P}^D_\phi$. Specifically, the *unit* loss is a dual form of Wasserstein distance under the constraint that $f_\theta(X)$ is 1-Lipschitz continuous. Intuitively, the critic network will return high values for samples coming from $\mathbb{P}^U$, and low values for samples coming from $\mathbb{P}^D_\phi$, thus it learns to deduce if its input is sampled from $\mathbb{P}^U$ or from $\mathbb{P}^D_\phi$. Once critic's optimization stage ends, the generator of WGAN optimizes its weights ϕ in order to increase $f_\theta(X)$'s output for samples coming from $\mathbb{P}^D_\phi$ via the loss

$$L^G_{WGAN}(\phi) = - \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D_\phi} f_\theta(X). \tag{10.11}$$

The described above "infinity" divergence of *unit* loss and 1-Lipschitz constraint may explain why the authors needed to clip NN weights to stabilize the approach's learning. In [6] after each iteration the NN weights are constrained to be between $[-c, c]$ for some constant c. Likely, such handling reduces the flexibility of a surface $f_\theta(X)$, thus preventing it from getting too high/low output values. Such conclusion about the reduced flexibility is also supported by [36].

Further, in [36] authors prove that 1-Lipschitz constraint of WGAN implies that the surface has gradient (w.r.t. X) with norm at most 1 everywhere, $\left\| \frac{\partial f_\theta(X)}{\partial X} |_{\theta=\theta^*} \right\| = 1$. Instead of weight clipping, they proposed to combine the *unit* loss with a X-gradient penalty term that forces this gradient norm to be close to 1. The effect of such regularization can be explained as follows. Considering the PSO principles, the optimal surface for the *unit* loss has areas of $+\infty$ and $-\infty$, thus requiring sharp slopes between these areas. The gradient penalty term constrains these slopes to be around 1, hence it prevents the surface from getting too high/low, solving in this way the "infinity" oscillations. Overall, by using weight clipping and other regularization techniques like gradient penalty [36], WGAN is in general highly successful in data generation task. Thus, we can see that basically unstable PSO instance with unit *magnitudes* can be stabilized by a surface flexibility restriction via appropriate regularization terms within the loss.

MMD Finally, MMD algorithm [34] exploits the *unit* loss in Eq. (10.9) to test if two separate datasets are generated from the same distribution. Authors express this loss over RKHS function with a bounded RKHS norm, thus implicitly constraining the model smoothness and eliminating the *infinite height* problem.

Remark 26. *As was observed empirically on 20D data, even the prolonged GD optimization via the above unit loss in Eq. (10.9) leaves the randomly initialized NN surface $f_\theta(X)$ almost unchanged for the case when $\mathbb{P}^U \equiv \mathbb{P}^D$. This is due to the implicit force balance produced by the identical densities. In contrast, when densities are different the optimization diverges very fast, after only a few thousands of iterations. Also, the optimization gradient during these iterations is typically smaller for the same density scenario than for the different densities. Similarly to MMD method, such behavior can be exploited for example to test if samples from two datasets have the same density or not, by performing the optimization and seeing if it diverges.*

Overall, all of the above PSO instances with unit *magnitudes*, except for CD, handle the instabilities of *unit* loss by restricting the flexibility of the model $f_\theta(X)$. Thus, a typical strategy is to enforce K-Lipschitz constraint. Yet, in context of DL it is still unclear if and how it is possible to enforce a model to be exact K-Lipschitz, even though there are several techniques recently proposed for this goal [36, 82, 104, 154].

10.4 Mutual Information Estimation

Mutual information (MI) between two random multi-variable distributions represents correlation between their samples, and is highly useful in the Machine Learning domain [13]. Here we shortly describe possible techniques to learn MI from data, based on PSO principles.

Consider two random variables $X \in \mathbb{R}^{n_x}$ and $Y \in \mathbb{R}^{n_y}$ with marginal densities $\mathbb{P}_X$ and $\mathbb{P}_Y$. Additionally, denote by $\mathbb{P}_{XY}$ their joint distribution. The MI between X and Y is defined as:

$$I(X,Y) = \int\int \mathbb{P}_{XY}(X,Y) \cdot V(X,Y) dX dY, \quad V(X,Y) \triangleq \log \frac{\mathbb{P}_{XY}(X,Y)}{\mathbb{P}_X(X) \cdot \mathbb{P}_Y(Y)}. \tag{10.12}$$

If log-ratio $V(X,Y)$ is known/learned in some way, and if we have samples $\{X^i, Y^i\}_{i=1}^N$ from joint density $\mathbb{P}_{XY}$, we can approximate MI via a sample approximation:

$$I(X,Y) \approx \frac{1}{N} \sum_{i=1}^{N} V(X_i, Y_i). \tag{10.13}$$

Further, $V(X,Y)$ can be easily learned by one of PSO instances in Tables 5.1-5.5 for logarithm density-ratio estimation as follows. Consider a model $f_\theta(X,Y) : \mathbb{R}^n \to \mathbb{R}$, with $n = n_x + n_y$. Additionally, we will use $\mathbb{P}_{XY}(X,Y)$ as *up* density in PSO framework, and $\mathbb{P}_X(X) \cdot \mathbb{P}_Y(Y)$ - as *down* density. To obtain sample from *up* density, we can pick random pair from available dataset $\{X^i, Y^i\}_{i=1}^N$, similarly to conditional density estimation in Section 9.1. Further, samples from *down* density can be acquired by picking X^i and Y^i from dataset independently.

Considering the above sampling procedure, we can apply PSO to push $f_\theta(X,Y)$ via *up* and *down* forces, using the corresponding *magnitude* functions. For any particular $\{M^U, M^D\}$ the associated PSO *balance state* will be:

$$\frac{M^D[X, Y, f_\theta(X,Y)]}{M^U[X, Y, f_\theta(X,Y)]} = \frac{\mathbb{P}_{XY}(X,Y)}{\mathbb{P}_X(X) \cdot \mathbb{P}_Y(Y)}. \tag{10.14}$$

Hence, to produce the convergence $f_\theta(X,Y) = V(X,Y)$, the appropriate choice of *magnitudes* must satisfy $\frac{M^D[X,Y,s]}{M^U[X,Y,s]} = \exp s$ - to infer log-ratio between *up* and *down* densities the *magnitude* ratio R must be equal to $\exp s$, according to Table 5.6.

For example, $M^U[X, Y, f_\theta(X,Y)] = \frac{1}{\exp[f_\theta(X,Y)]+1}$ and $M^D[X, Y, f_\theta(X,Y)] = \frac{1}{\exp[-f_\theta(X,Y)]+1}$ can be used (e.g. a variant of the logistic loss in Table 5.4), yet many other alternatives can also be considered. Recently, similar ideas were also presented in [13].

10.5 Learning Probabilistic Occupancy Mapping

Problem Definition Statistical representation of space occupancy around the robot is mandatory for autonomous navigation. Borrowing the formulation from [121], training data for this learning task can be acquired from lidar scans; the generated dataset is $\mathcal{D} = \{X_i, Y_i\}_{i=1}^N$ where X_i is the observed space location (2D or 3D) and $Y_i \in \{\mathbf{f}, \mathbf{o}\} \triangleq \{free, occupied\}$ is its occupancy label. Laser hit points can be considered as samples X with $Y = \mathbf{o}$, while samples with $Y = \mathbf{f}$ can be sampled (i.e. uniformly) along the laser beam. The dataset $\mathcal{D}$ implies existence of a joint distribution $\mathbb{P}(X, Y)$ from which it was sampled, and a typical objective is to estimate $\mathbb{P}(Y|X) = \mathbb{P}(X, Y)/\mathbb{P}(X)$. Applying PSO for this task can be done similarly to learning a (conditional) pdf in sections 8-9.

Yet, observe that the marginal $\mathbb{P}(X)$ represents a probability of location X being observed, and the above objective is well-defined only in areas of obtained scans. Instead, we propose to estimate a different objective $J(X) \triangleq \mathbb{P}(X) \cdot [\mathbb{P}(Y = \mathbf{o}|X) - \mathbb{P}(Y = \mathbf{f}|X)]$ since it is defined in any area of the map and produces valuable information about the environment. First, note that its second term, likelihood difference, defines the sign of $J(X)$ which can be interpreted as which label of X is more likely. Further, when $\mathbb{P}(X) = 0$, meaning that location X was not observed during scan gathering, $J(X)$ is also zero, allowing to decide when there is enough information about X. Moreover, locations that were observed by multiple scans will have higher $\mathbb{P}(X)$, and hence also $|J(X)|$, which can be used as a measure of confidence about X's occupancy state. Furthermore, $J(X)$ is continuous function of location X, allowing to avoid a discretization of the map typically done by occupancy grid methods. Thus, below we show how $J(X)$ can be inferred via PSO. Yet, importantly, we emphasize $J(X)$ is only one possible candidate target, and PSO is definitely not limited to only this case.

PSO-based Solution To learn $J(X)$, we extend PSO framework to setting of 3 different forces applied over the model $f_\theta(X)$. For this purpose, we denote by $S \subset \mathbb{R}^n$ the subset of n-dimensional space that we want to map. Here n can have value of 2 or 3, and S can represent the entire space of the considered indoor/outdoor environment (i.e. a navigated-through building). Further, denote by $\Omega \subseteq S$ the space that was observed by any of the acquired lidar scans.

Each training sample $X_i \in \mathcal{D}$ also belongs to Ω, since it is the observed location. Further, we can consider $\mathcal{D}$ to be implicitly sampled from the joint $\mathbb{P}(X, Y)$, with $\mathcal{X} = \{X_i\}_{i=1}^N$ and $\mathcal{Y} = \{Y_i\}_{i=1}^N$ being distributed according to marginal probabilities $\mathbb{P}(X)$ and $\mathbb{P}(Y)$. Also, as mentioned above $\mathbb{P}(X)$ can be interpreted as a likelihood of X being observed during gathering of lidar scans, with $\forall X \notin \Omega : \mathbb{P}(X) = 0$. Further, $\mathbb{P}(Y)$ can be inferred from $\mathcal{Y}$ via $\mathbb{P}(Y = \mathbf{f}) = \frac{N^{\mathbf{f}}}{N}$ and $\mathbb{P}(Y = \mathbf{o}) = \frac{N^{\mathbf{o}}}{N}$, with $N^{\mathbf{f}}$ being number of samples $Y_i = \mathbf{f}$ and $N^{\mathbf{o}}$ - number of samples $Y_i = \mathbf{o}$.

Next, we construct two datasets $\mathcal{X}^{\mathbf{o}} = \{X_i^{\mathbf{o}}\}_{i=1}^{N^{\mathbf{o}}}$ and $\mathcal{X}^{\mathbf{f}} = \{X_i^{\mathbf{f}}\}_{i=1}^{N^{\mathbf{f}}}$, with first dataset containing each location $X_i \in \mathcal{D}$ with $Y_i = \mathbf{o}$, and the second - the rest of the locations in $\mathcal{D}$. Note that according to the above considered implicit distributions $X_i^{\mathbf{o}}$ is distributed along $\mathbb{P}(X|Y = \mathbf{o}) = \frac{\mathbb{P}(X, Y=\mathbf{o})}{\mathbb{P}(Y=\mathbf{o})}$, and $X_i^{\mathbf{f}}$ - along $\mathbb{P}(X|Y = \mathbf{f}) = \frac{\mathbb{P}(X, Y=\mathbf{f})}{\mathbb{P}(Y=\mathbf{f})}$, with supports of both distributions being contained within Ω. Finally, we construct one more dataset $\mathcal{X}^{\mathcal{U}} = \{X_i^{\mathcal{U}}\}_{i=1}^{N^{\mathcal{U}}}$ with $N^{\mathcal{U}}$ points sampled from $\mathbb{P}^{\mathcal{U}}(X)$ - a uniform distribution over the entire S.

Further, we propose to extend PSO in Eq. (3.1) to a 3-term form:

$$d\theta = -\frac{1}{N^{\mathbf{o}}} \sum_{i=1}^{N^{\mathbf{o}}} M^{\mathbf{o}} \left[X_i^{\mathbf{o}}, f_\theta(X_i^{\mathbf{o}})\right] \cdot \nabla_\theta f_\theta(X_i^{\mathbf{o}}) + \frac{1}{N^{\mathbf{f}}} \sum_{i=1}^{N^{\mathbf{f}}} M^{\mathbf{f}} \left[X_i^{\mathbf{f}}, f_\theta(X_i^{\mathbf{f}})\right] \cdot \nabla_\theta f_\theta(X_i^{\mathbf{f}}) + \\ + \frac{1}{N^{\mathcal{U}}} \sum_{i=1}^{N^{\mathcal{U}}} M^{\mathcal{U}} \left[X_i^{\mathcal{U}}, f_\theta(X_i^{\mathcal{U}})\right] \cdot \nabla_\theta f_\theta(X_i^{\mathcal{U}}), \quad (10.15)$$

with $M^{\mathbf{o}}(\cdot) = \frac{N^{\mathbf{o}}}{N^{\mathbf{o}}+N^{\mathbf{f}}}\mathbb{P}^{\mathcal{U}}(X)$, $M^{\mathbf{f}}(\cdot) = \frac{N^{\mathbf{f}}}{N^{\mathbf{o}}+N^{\mathbf{f}}}\mathbb{P}^{\mathcal{U}}(X)$, and $M^{\mathcal{U}}(\cdot) = f_\theta(X)$. Note that $M^{\mathbf{o}}(\cdot)$ and $M^{\mathbf{f}}(\cdot)$ always return positive values. Therefore, Eq. (10.15) pushes $f_\theta(X)$ at occupied locations $X_i^{\mathbf{o}}$ *up* and at unoccupied locations $X_i^{\mathbf{f}}$ - *down*, similarly to the original PSO equation. Further, $M^{\mathcal{U}}(\cdot)$ is positive when $f_\theta(X) > 0$ and is negative when $f_\theta(X) < 0$. Hence, the applied force at $X_i^{\mathcal{U}}$ is always pushing the surface towards $f_\theta(X) = 0$ (see also the forth property of Theorem 7).

The convergence $f_\theta(X) = J(X)$ at the optimization equilibrium can be verified via PSO *balance state* below. The equality of forces at X (i.e. Euler-Lagrange equation at X) is satisfied when:

$$M^{\mathbf{o}}[X, f_\theta(X)] \cdot \mathbb{P}(X|Y=\mathbf{o}) - M^{\mathbf{f}}[X, f_\theta(X)] \cdot \mathbb{P}(X|Y=\mathbf{f}) - M^{\mathcal{U}}[X, f_\theta(X)] \cdot \mathbb{P}^{\mathcal{U}}(X) = 0, \quad (10.16)$$

where each term represents a physical force - a product of corresponding *magnitude* function and data density. After replacing each term with its definition, this equation turns to be:

$$\frac{N^{\mathbf{o}}}{N^{\mathbf{o}}+N^{\mathbf{f}}}\mathbb{P}^{\mathcal{U}}(X) \cdot \frac{\mathbb{P}(X, Y=\mathbf{o})}{\mathbb{P}(Y=\mathbf{o})} - \frac{N^{\mathbf{f}}}{N^{\mathbf{o}}+N^{\mathbf{f}}}\mathbb{P}^{\mathcal{U}}(X) \cdot \frac{\mathbb{P}(X, Y=\mathbf{f})}{\mathbb{P}(Y=\mathbf{f})} - f_\theta(X) \cdot \mathbb{P}^{\mathcal{U}}(X) = 0, \quad (10.17)$$

$$\frac{N^{\mathbf{o}}}{N^{\mathbf{o}}+N^{\mathbf{f}}} \cdot \frac{\mathbb{P}(X, Y=\mathbf{o})}{\mathbb{P}(Y=\mathbf{o})} - \frac{N^{\mathbf{f}}}{N^{\mathbf{o}}+N^{\mathbf{f}}} \cdot \frac{\mathbb{P}(X, Y=\mathbf{f})}{\mathbb{P}(Y=\mathbf{f})} - f_\theta(X) = 0, \quad (10.18)$$

$$\mathbb{P}(X, Y=\mathbf{o}) - \mathbb{P}(X, Y=\mathbf{f}) - f_\theta(X) = 0, \quad (10.19)$$

$$f_\theta(X) = \mathbb{P}(X, Y=\mathbf{o}) - \mathbb{P}(X, Y=\mathbf{f}) = \mathbb{P}(X) \cdot [\mathbb{P}(Y=\mathbf{o}|X) - \mathbb{P}(Y=\mathbf{f}|X)] = J(X). \quad (10.20)$$

Therefore, the described by Eq. (10.15) approach will converge to $J(X)$.

Furthermore, we can find a ”loss” form of the proposed PSO method in Eq. (10.15) since its *magnitude* functions have analytical anti-derivatives. Specifically, Eq. (10.15) can be considered as a gradient w.r.t. θ of the following loss:

$$L(\theta) = -\frac{1}{N^{\mathbf{o}}} \sum_{i=1}^{N^{\mathbf{o}}} \frac{N^{\mathbf{o}}}{N^{\mathbf{o}}+N^{\mathbf{f}}}\mathbb{P}^{\mathcal{U}}(X_i^{\mathbf{o}}) \cdot f_\theta(X_i^{\mathbf{o}}) + \frac{1}{N^{\mathbf{f}}} \sum_{i=1}^{N^{\mathbf{f}}} \frac{N^{\mathbf{f}}}{N^{\mathbf{o}}+N^{\mathbf{f}}}\mathbb{P}^{\mathcal{U}}(X_i^{\mathbf{f}}) \cdot f_\theta(X_i^{\mathbf{f}}) + \\ + \frac{1}{N^{\mathcal{U}}} \sum_{i=1}^{N^{\mathcal{U}}} \frac{1}{2}\left[f_\theta(X_i^{\mathcal{U}})\right]^2, \quad (10.21)$$

which in its turn is a sample approximation of:

$$\begin{aligned}
L(\theta) = \int -\mathbb{P}(X|Y=\mathbf{o}) \cdot \frac{N_{\mathbf{o}}}{N_{\mathbf{o}}+N_{\mathbf{f}}} \mathbb{P}^{\mathcal{U}}(X) \cdot f_\theta(X) + \\
+ \mathbb{P}(X|Y=\mathbf{f}) \cdot \frac{N_{\mathbf{f}}}{N_{\mathbf{o}}+N_{\mathbf{f}}} \mathbb{P}^{\mathcal{U}}(X) \cdot f_\theta(X) + \frac{1}{2}\mathbb{P}^{\mathcal{U}}(X) \cdot [f_\theta(X)]^2 \, dX = \\
= \int \mathbb{P}^{\mathcal{U}}(X) \cdot \Big[- [\mathbb{P}(X, Y=\mathbf{o}) - \mathbb{P}(X, Y=\mathbf{f})] \cdot f_\theta(X) + \frac{1}{2} [f_\theta(X)]^2 \Big] dX = \\
= \frac{1}{2} \int \mathbb{P}^{\mathcal{U}}(X) \cdot \Big[f_\theta(X) - [\mathbb{P}(X, Y=\mathbf{o}) - \mathbb{P}(X, Y=\mathbf{f})] \Big]^2 dX - \\
- \frac{1}{2} \int \mathbb{P}^{\mathcal{U}}(X) \cdot [\mathbb{P}(X, Y=\mathbf{o}) - \mathbb{P}(X, Y=\mathbf{f})]^2 dX. \quad (10.22)
\end{aligned}$$

Taking into account that the last term is independent of f_θ, this loss can be replaced by:

$$L(\theta) = \int \mathbb{P}^{\mathcal{U}}(X) \cdot [f_\theta(X) - J(X)]^2 dX. \quad (10.23)$$

whose minimizer is obviously $J(X)$.

Above we can observe that PSO allowed us to create a new probabilistic loss for statistical occupancy mapping by only considering the various force terms in Eq. (10.15) and verifying their convergence according to PSO *balance state*. Apparently, it is also possible to derive the very same method by considering the loss $L(\theta)$ in Eq. (10.23) in the first place. However, from $L(\theta)$'s definition it is not obvious that it can be even computed/approximated, due to unknown $\mathbb{P}(X, Y = \mathbf{o})$ and $\mathbb{P}(X, Y = \mathbf{f})$. In contrast, the physical paradigm of PSO leads to a very simple and systematic solution. The empirical evaluation of the above method appears in Section 13.6.

CHAPTER 11

NN Architecture

In this section we describe various design choices when constructing NN $f_\theta(X)$, and their impact on density estimation task. The discussed below are various connectivity architectures, activation functions and pre-conditioning techniques that helped us to improve overall learning accuracy. Likewise, where possible we relate the design choice to corresponding properties acquired by a model kernel $g_\theta(X, X')$.

Algorithms DeepPDF (see Section 8.1) and log-density estimators in Section 8.2 typically produce highly accurate density approximations in low dimensional cases. For example, in [61] we showed that DeepPDF produces a better accuracy than KDE methods in 2D and 3D scenarios. This likely can be accounted to the flexibility of NN - its universal ability to approximate any function. As empirically observed in [63], the implicit model kernel $g_\theta(X', X)$ adapts to better represent any learned target function. Further, its bandwidth, discussed in Section 7.5, is typically different in various areas of the considered input space. This allows to prevent *overfitting* in areas with small amount of training points, and to reduce *underfitting* in areas where the amount of training data is huge. In contrast, KDE methods are typically limited to a specific choice of a kernel and a bandwidth (yet variable-bandwidth KDE methods exist, see [136]) that is applied to estimate the entire pdf surface with its many various details.

Yet, we also observed a considerable *underfitting* problem of the above PSO instances that grows with larger data dimension and with higher frequency/variability contained within the target function. Particularly, even in case where a high-dimensional training dataset is huge, for a typical fully-connected (FC) NN architecture the produced estimation is far away from the real data density, and often contains mode-collapses and other inference inconsistencies. We argue that it is caused by too wide bandwidth of $g_\theta(X', X)$ in FC architecture, which leads to a growing estimation bias. Such conclusion is supported by Theorem 20 which stated that the bandwidth of $g_\theta(X, X')$ defines the flexibility of $f_\theta(X)$.

As was observed, in FC architecture g_θ's bandwidth is growing considerably with the higher data dimension, thus producing more side interference between the different training points and decreasing the overall elasticity of the network. In its turn, this limits the accuracy produced

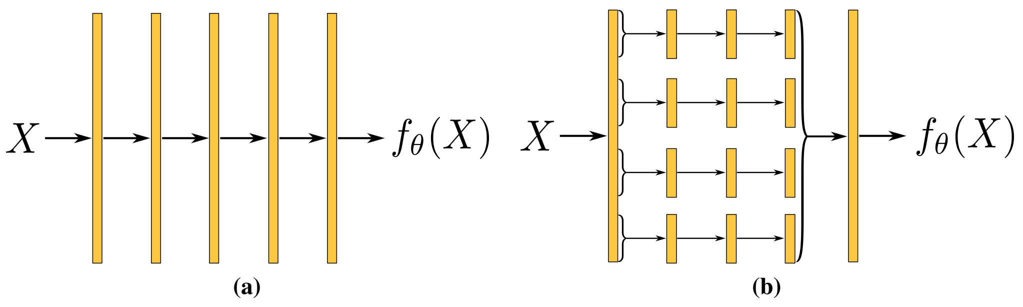


Figure 11.1: (a) Typical NN architecture used in [61]. Yellow blocks are FC layers with non-linearity, except for last layer which is FC layer without activation function. Entire network can be seen as single transformation channel. (b) Proposed NN architecture used in this thesis. Block-diagonal layers can be seen as set of independent transformation channels. This independence improves network's flexibility. Output vector of a first FC layer is sliced into N_B separate vectors. Each small vector is used as input to separate channel - FC sub-network. At the end outputs of all channels are concatenated and sent to the final FC layer. All FC layers in this architecture (the yellow blocks) use non-linearity (typically Relu or Leaky-Relu), except for the final FC layer.

by PSO. Below we propose a new NN architecture that mitigates the bandwidth problem and increases a flexibility of the surface. Further, in Section 13.2.3 we show that such architecture extremely improves the estimation accuracy.

Remark 27. *Note that in context of generative adversarial networks (GANs), whose critics are also instances of PSO (see Tables 5.1-5.5), the convergence problems (e.g. mode-collapse and non-convergence) were also reported. Typically, these problems are blamed on Nash equilibrium between critic and generator networks which is hard to optimize. Yet, in our work we see that such problems exist even without a two-player optimization. That is, even when only a specific PSO instance (the critic in GAN's context) is trained separately, it is typically underfitting and has mode-collapses within the converged surface $f_\theta(X)$ where several separate "hills" from target $T\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]$ are represented as one hill inside $f_\theta(X)$. Below we present several techniques that allowed us to reduce these convergence problems.*

11.1 Block-Diagonal Layers

A typical FC network (Figure 11.1a) can be seen as one channel transformation from point X to its surface height $f_\theta(X)$. During the optimization, due to such NN structure almost every parameter inside θ is updated as a consequence of pushing/optimizing at any training point $X \in \mathbb{R}^n$ within PSO loss. Such high sharing of the weights between various regions of $\mathbb{R}^n$ creates a huge side-influence between them - a dot product between $\nabla_\theta f_\theta(X)$ and $\nabla_\theta f_\theta(X')$ for faraway points X and X' is large. This in turn increases bandwidth of g_θ and decreases the flexibility of f_θ.

The above line of thought guided us to propose an alternative NN architecture where several separate transformation channels are built into one network (see Figure 11.1b). As we will see below, such architecture is identical to a simple FC network where each layer's weight matrix W_i is block-diagonal.

Specifically, we propose to pass input X through a typical FC layer with output dimension S, and split this output into a set of N_B smaller vectors of size $S_B = S/N_B$. Further, for each

S_B-sized vector we construct a channel: a subnetwork with $[N_L - 2]$ FC layers of the same size S_B. Finally, the outputs of all channels are concatenated into vector of size S and this vector is sent to the final FC layer (see illustration in Figure 11.1b). All FC layers within this architecture use non-linearity (typically Relu or Leaky-Relu), except for the last layer.

Exactly the same computational flow will be produced if we use the usual FC network from Figure 11.1a with inner layers having block-diagonal weight matrices. Namely, we can build the same simple network as in Figure 11.1a, with N_L layers overall, where $[N_L - 2]$ inner FC layers have block-diagonal weight matrices W_i of size $S \times S$. Each W_i in its turn will contain N_B blocks at its diagonal, of $S_B \times S_B$ size each, with rest of the entries being constant zeros.

A straightforward implementation of block-diagonal (BD) layers by setting off-diagonal entries to be constant zeros can be wasteful w.r.t. memory and computation resources. Instead, we can use multi-dimensional tensors for a more efficient implementation as follows. Consider output of the first FC layer as a tensor $\bar{v}$ with dimensions $[B, S]$, where B is a batch dimension and S is an output dimension of the layer. We can reshape $\bar{v}$ to have dimensions $[B, N_B, S_B]$, where the last dimension of $\bar{v}$ will contain small vectors $\bar{u}_j$ of size S_B each, i.e. inputs for independent channels. Further, each inner BD layer can be parametrized by a weight matrix W with dimensions $[N_B, S_B, S_B]$ and bias vector b with dimensions $[N_B, S_B]$. The multiplication between $\bar{v}$ and W, denoted as V, has to be done for each $\bar{u}_j$ with an appropriate slice of weight matrix, $W[j, :, :]$. Moreover, it should be done for every instance of the batch. This can be done via the following Einstein summation convention:

$$V[i, j, k] = \sum_{m=1}^{S_B} W[j, k, m] \cdot \bar{v}[i, j, m], \tag{11.1}$$

which produces tensor V with size $[B, N_B, S_B]$. Further, bias can be added as:

$$U[i, :, :] = V[i, :, :] + b, \tag{11.2}$$

where afterwards the tensor U is transformed by point-wise activation function $\sigma(\cdot)$, finally producing the output of BD layer $\hat{U} = \sigma(U)$ of size $[B, N_B, S_B]$.

We construct $[N_L - 2]$ such BD layers that represent N_B independent channels. Further, the output of the last BD layer is reshaped back to have dimensions $[B, S]$, and is sent to the final ordinary FC layer that returns a scalar.

Remark 28. *The Einstein summation operation is typically offered by modern DL frameworks, thus implementing the above BD layers is convenient and easy. Yet, their runtime is slower relative to FC layers. We hope that in future versions of DL frameworks such BD layers would be implemented efficiently on GPU level. Also, our code for this layer can be found in open source library `https://bit.ly/2AMwyJT`.*

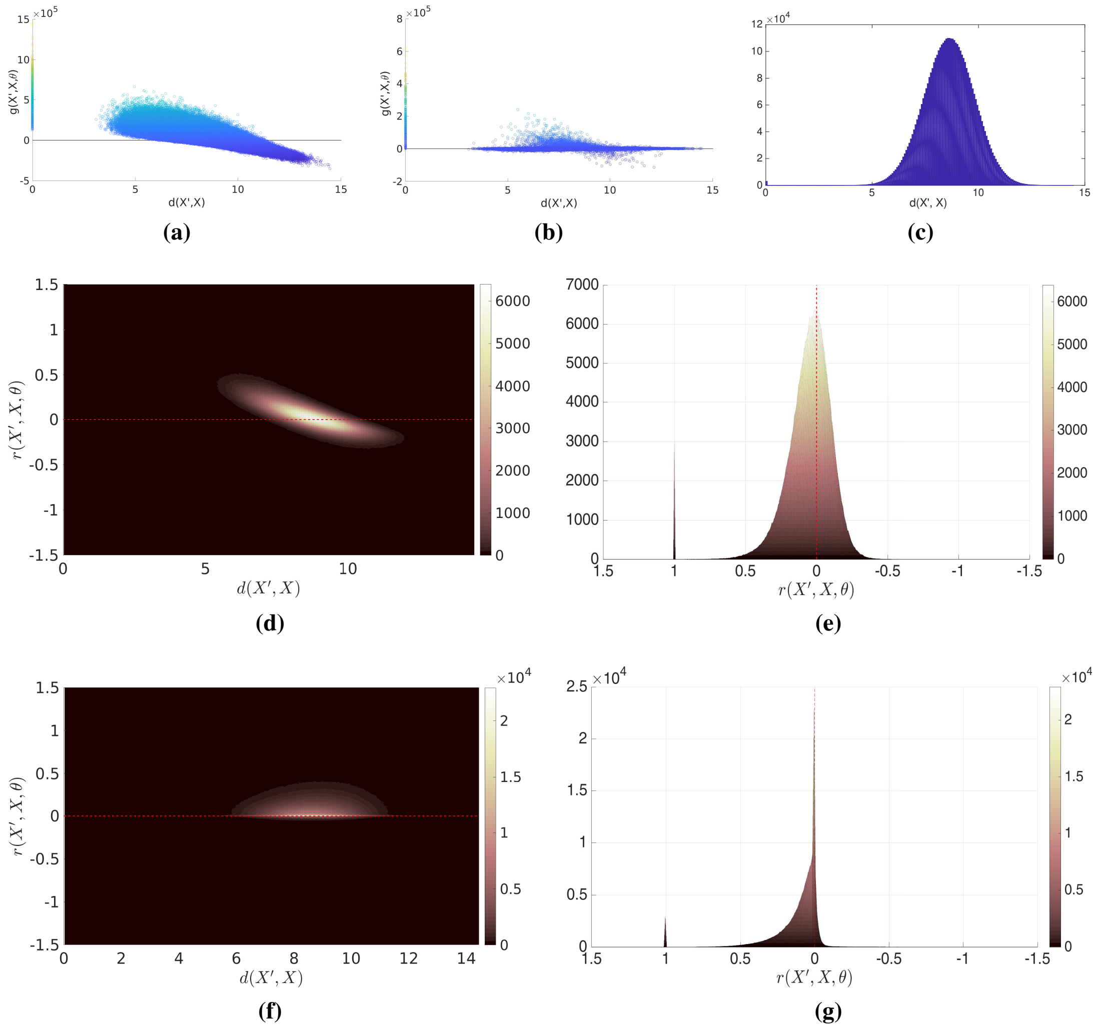


Figure 11.2: Bandwidth of $g_\theta(X', X)$ and the surface flexibility of FC and BD models. We infer 20D *Columns* distribution (see Section 13.2) by using PSO-LDE with $\alpha = \frac{1}{4}$ (see Table 5.1 and Section 8.2). Two networks were trained, FC and BD. The applied FC architecture contains 4 FC layers of size 1024. The applied BD architecture has 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$. Values of *gradient similarity* $g_\theta(X', X)$ and values of Euclidean distance $d(X', X)$ are plotted for (a) FC network and (b) BD network. (c) Histogram of $d(X', X)$ calculated between all sample pairs from dataset D. Further, a histogram of obtained $\{r_\theta(X_i, X_j)\}$ and $\{d(X_i, X_j)\}$ is plotted for (d) FC model and (f) BD model. Side views of these histograms are depicted in (e) and (g). See more details in the main text.

11.1.1 Flexibility of BD vs FC

The above BD architecture allowed us to tremendously improve accuracy of density estimation for high-dimensional data. This was achieved due to the enhanced flexibility of BD architecture vs FC, as we empirically demonstrate below.

To this end, we analyze the bandwidth of $g_\theta(X, X')$ for each of the architectures as follows. We perform a typical density estimation task via PSO-LDE method proposed in Section 8.2. After training a model we sample $D = \{X_i\}_{i=1}^{3000}$ testing points from the target density $\mathbb{P}^U$ and calculate their gradients $\nabla_\theta f_\theta(X_i)$. Further, we calculate Euclidean distance $d(X_i, X_j) = \|X_i - X_j\|$ and *gradient similarity* $g_\theta(X_i, X_j)$ between every two points within D, producing $\frac{3000 \cdot 3001}{2}$ pairs of distance and similarity values (we consider only unique pairs here). These values are plotted in Figures 11.2a-11.2b. As can be seen, $g_\theta(X', X)$ values (y axis) of FC network are much higher than these values in BD network. Likewise, there is strong correlation between values of *gradient similarity* and Euclidean distance. In FC case, for $d(X', X) > 0$ values of $g_\theta(X', X)$ are far away from being zeros, thus implying strong side-influence of optimization pushes on surface $f_\theta(X)$ even between far away points. In contrast, for BD case we can see that $g_\theta(X', X)$ is centered around zero for $d(X', X) > 0$, hence side-influence here is less significant. Furthermore, we stress that similar trends were achieved in all our experiments, for various densities $\mathbb{P}^U$ and $\mathbb{P}^D$.

Remark 29. *The gap in Figures 11.2a-11.2b between points with $d(X', X) = 0$ and rest of the samples is explained as follows. At $d(X', X) = 0$ all point pairs are of a form (X_i, X_i), with overall 3000 such pairs. Rest of the samples are $\{(X_i, X_j)|i \neq j\}$. Furthermore, the histogram of $d(X', X)$ between points in D is illustrated in Figure 11.2c. As can be observed, $d(X', X)$ is distributed with Gaussian-like density centered around 8.6. Hence, the gap between the points in Figures 11.2a-11.2b can be explained by a very low probability of two sampled points to be close to each other when the considered space volume (here the subset of $\mathbb{R}^{20}$) is huge.*

Further, for each sample pair in D we also calculate the *relative* model kernel $r_\theta(X_i, X_j)$ defined in Eq. (7.10). When r_θ is greater than 1, it implies that point X_j has stronger impact over $f_\theta(X_i)$ than the point X_i itself, and vice versa. Hence, for each point X_i the $r_\theta(X_i, \cdot)$ can be interpreted as a *relative* side-influence from other areas over $f_\theta(X_i)$, scaled w.r.t. the self-influence of X_i. Such normalization allows us to see the actual side-influence impact between two different points, since the value of $g_\theta(X, X')$ by itself is meaningless and only achieves significance when compared to the self-similarity $g_\theta(X, X)$. Moreover, unlike $g_\theta(X, X)$, $r_\theta(X, X')$ of different models and NN architectures is on the same scale, allowing to compare the side-influence level between different models.

For $9 \cdot 10^6$ calculated pairs of a *relative* side-influence $r_\theta(X_i, X_j)$ and a Euclidean distance $d(X_i, X_j)$ we constructed a histogram in Figures 11.2d and 11.2f for FC and BD networks respectively. Here, we can see the real difference between side-similarities of two models. Within FC network we have a strong *relative* side-influence even between far away regions. This side-influence interferes with the proper PSO optimization by introducing a bias, as was explained in Section 7.5. In contrast, within BD model the *relative* side-influence between far away regions stays very close to zero, implying that the surface height $f_\theta(X)$ at point X is only pushed by training points that are relatively close to X. Furthermore, this increases the flexibility of the surface, since with a less side-influence the surface is less constrained and can be pushed at each specific neighborhood more freely.

Hence, we see empirically that the kernel bandwidth of BD NN is smaller than the bandwidth of FC NN, which implies that BD surface is much more flexible than FC. Such flexibility also improves the overall accuracy performance achieved by BD networks (see Section 13.2.3). The exact mechanism responsible for such difference in the *gradient similarity* is currently unknown and we shall investigate it in future work.

11.1.2 Relation between BD and FC - Additional Aspects

The multi-dimensional tensor implementation of BD layers in Eq. (11.1-11.2) allows to significantly reduce size of θ. For example, BD network applied in Figure 11.2 with 6 layers has less than 10^6 weights ($|\theta| = 902401$), while the straightforward implementation would require above 10^7 weights - 10 times more; the same size that appropriate FC network with 6 layers of size 3200 would take. Further, the size of FC network used in Figure 11.2 is $|\theta| = 2121729$. Yet, surprisingly the more compact BD network produces a narrower model kernel (and higher approximation accuracy as will be shown in Section 13.2.3) than the more memory consuming FC network.

Interestingly, BD layers are contained in the hypothesis class of FC layers, thus being instance of the latter. Yet, a typical optimization of FC architecture will not impose weight matrices to be block-diagonal, since the local minima of FC network typically has dense weight matrices. However, as already stated, an optimized network with BD structure has a significantly lower error compared to FC structure. This suggests that local minima of FC networks has a much bigger error compared to the error of the global minima for such architecture, since the global minima should be even smaller than the one achieved by BD network. Hence, this implies that common statement about local and global errors of NN being close is not always correct.

11.1.3 Similar Proposed Architectures

The BD model can be expressed as a sum of sub-models, each representing separate network channel. Such design has a high resemblance to the products of experts (PoE) [45] where model is constructed as sum (or product) of smaller models. Yet, in typical PoE each expert is trained separately, while herein we represent our block-diagonal model as single computational graph that is trained as whole by the classical backpropagation method.

In addition, we argue that also other DL domains can benefit from a BD architecture, and such investigation can be an interesting future work. In fact, separating network into several independent channels is not new. The family of convolutional Inception models [132–134] also applied the *split-transform-merge* paradigm, where each network block was separated into a set of independent transformations (channels). These models succeeded to achieve high accuracy at the image classification problem. Further, ResNeXt convolutional model in [149] generalized this idea to produce NN computational blocks that contain C independent identical transformations, where C is a *cardinality* parameter of NN. Authors showed that increasing *cardinality* instead of width/number of layers can significantly improve the accuracy produced by NN. In context of BD architecture, we have seen a similar trend where increasing number of channels N_B (which is parallel to C) allows to provide a better approximation of the target function. We demonstrate this in our experiments in Section 13.

Further, a similar architecture was proposed also in [89] in the context of the classification, although it was implemented in a different way. The main motivation of that work was to condense a network size to improve the computational complexity of a NN. Authors showed that by forcing weight matrices of FC layers to be block-diagonal a significant speedup in time can be achieved with small loss in accuracy. In contrast, in our work we see that such NN structure not only improves runtime and reduces number of weights, but also produces a higher approximation performance.

11.2 NN Pre-Conditioning

It is a common practice in Machine Learning to pre-condition a learning algorithm by, for example, whitening and uncorrelating data or performing any other transformation that improves a condition number of the optimization. We also found that such techniques can be valuable for the density estimation task. Specifically, the main considered by this thesis application of PSO framework is to learn $\log \mathbb{P}^U(X)$. In our experiments we combine two pre-conditioning methods within our NN $f_\theta(X)$: data normalization and NN height bias.

First, we normalize data to have zero mean and unit standard deviation for each dimension i independently, via:

$$\hat{X}_i = \frac{X_i - \mu_i}{\sigma_i}, \tag{11.3}$$

where μ and σ are mean and standard deviation vectors calculated for all available data $\{X_i^U\}_{i=1}^{N^U}$ from the target density $\mathbb{P}^U$.

Second, we bias an initial surface $f_\theta(X)$ to coincide with logarithm of the chosen auxiliary *down* density $\mathbb{P}^D(X)$. We assume that the target $\log \mathbb{P}^U(X)$ and $\log \mathbb{P}^D(X)$ reside on a similar height on average. Thus, to accelerate the convergence we force the initial height of surface $f_\theta(X)$ to be identical to $\log \mathbb{P}^D(X)$ as follows. First, as observed a typical initialization of NN produces the initial surface $f_\theta(X) \approx 0$ for all points X. Hence, in order to bias it to the initial height $\log \mathbb{P}^D(X)$, we only need to add this log-pdf function to the output of the last layer, $f_\theta^L(X)$:

$$f_\theta(X) = f_\theta^L(X) + \log \mathbb{P}^D(X). \tag{11.4}$$

Moreover, such NN initialization enforces the logarithm difference $\bar{d}\,[X, f_\theta(X)] \triangleq f_\theta(X) - \log \mathbb{P}^D(X)$ from Eq. (8.5) to be approximately zero for all points $X \in \mathbb{R}^n$ at beginning of the optimization. Further, considering *magnitudes* of PSO-LDE in Eqs. (8.7)-(8.8), for the above initialization each of $\{M_\alpha^U, M_\alpha^D\}$ will return $2^{-\frac{1}{\alpha}}$ for any point X. Since both $M_\alpha^U(X, f_\theta(X))$ and $M_\alpha^D(X, f_\theta(X))$ have the same value at every point $X \in \mathbb{R}^n$, such NN bias produces a more balanced PSO gradient (see Eq. (3.1)) at start of the training, which improves the optimization numerical stability. Furthermore, as mentioned above in case the chosen $\mathbb{P}^D$ is indeed close to the target $\mathbb{P}^U$, the initial value of the surface $f_\theta(X)$ is also close to its final converged form; this in turn increases the convergence rate of PSO-LDE. Further, recently a similar idea was suggested in [65] specifically for NCE method (PSO-LDE with $\alpha = 1$) in the context of discrete density estimation for language modeling, where NN initialization according to outputs of the noise density (parallel to $\mathbb{P}^D(X)$ in our work) helped to improve the learned model accuracy.

We perform both techniques inside the computational graph of NN $f_\theta(X)$, by adding at the beginning of graph the operation in Eq. (11.3), and at the end of graph - the operation in Eq. (11.4).

11.3 Other NN Architecture Aspects

In our experiments we also explored two choices of non-linear activation function to use within NN $f_\theta(X)$, Relu and Leaky Relu. We found that both have their advantages and disadvantages. Relu reduces training time by 30% w.r.t. Leaky Relu, allegedly due to its implicit gradient sparsity, and the converged surface looks more *smooth*. Yet, it often contains mode collapse areas where several modes of $\mathbb{P}^U$ are represented within $f_\theta(X)$ by a single "hill". On the other hand, Leaky-Relu sometimes produces artifacts near sharp edges within f_θ, that resembles Gibbs phenomenon. Yet, it yields significantly less mode collapses.

We argue that these mode collapses are in general caused by the reduced model flexibility, which in case of Relu is induced by more sparse gradients $\nabla_\theta f_\theta(\cdot)$ as follows. The implicit gradient sparsity of Relu, i.e. zero-gradient $\frac{\partial f_\theta(X)}{\partial \theta_i} = 0$ for the most part of the weights $\theta_i \in \theta$ at all input points $X \in \mathbb{R}^n$, also reduces the effective dimension of subspace spanned by $\nabla_\theta f_\theta(\cdot)$ evaluated at training points. Thus, the number of possible independent gradient vectors at different points (i.e. the rank of the aforementioned subspace) is also reduced, which increases (on average) the correlation $g_\theta(X, X') \triangleq \nabla_\theta f_\theta(X)^T \cdot \nabla_\theta f_\theta(X')$ between various (even faraway) points X and X'. This increase can also be interpreted as an increase of the kernel bandwidth, which will lead to expressiveness reduction of the model, according to Section 7.5.

Further, residual (skip) connections between different NN layers became very popular in recent NN architectures [42, 132, 149]. Such connections allowed for using deeper neural networks and for the acceleration of learning convergence. Yet, in our work we did not observe any performance improvement from introducing skip connections into NN $f_\theta(X)$ with 8 or less layers. Thus, in most part of our experiments we did not employ these shortcuts. The only part where they were used is the Section 13.4 where networks with 14 layers were trained.

Additionally, the Batch Normalization (BN) [53] technique was shown in many DL works to stabilize the training process and improve the overall approximation accuracy. However, in our experiments on the density estimation we saw the opposite trend. That is, when BN is combined with PSO density estimators, the outcome is usually worsen than without it.

Finally, dropout [128] is known to be an useful regularization method to fight the *overfitting*. In our experiment we indeed observed the $g_\theta(X, X')$'s bandwidth increase along with an increase in the dropout probability. Hence, dropout can be considered as a tool to increase side-influence and bias of the estimation, and to reduce its variance. Further, the detailed investigation of dropout impact over $g_\theta(X, X')$ is outside of this thesis scope.

CHAPTER 12

Overfitting of PSO

In this section we will illustrate one of the major challenges involved in training PSO in a small dataset setting - the over-flexibility of model f_θ that induces *overfitting*. Likewise, herein we will also discuss possible solutions to overcome this issue.

12.1 Problem Illustration

As was described in [61], the accuracy of the estimated density can be very low for a small dataset setting, since the converged surface can be flat with several spikes at locations of available training data points. This is due the fact that apparently we estimate the empirical density of data which in case of sparse datasets can be represented as a flat surface with several peaks. If the used model $f_\theta(X)$ is overly flexible and is not properly regularized, it can be indeed pushed to such spiky form, as was proved by Theorem 19 and as we empirically demonstrate below.

According to Section 7.5, the flexibility of the surface f_θ can be expressed via properties of the model kernel g_θ, such as its bandwidth. The kernel acts as a connector of various input space areas, creating the side influence/force between these areas; it is balancing the overall physical force at each input point, with its equilibrium described by the convoluted PSO balance state in Eq. (7.8). Further, when the bandwidth of the model kernel is too narrow w.r.t. distance between training points, the *influence* area of any training point X will be some small neighborhood around X. Any input point X' outside of all *influence* areas is basically not optimized during the learning - $f_\theta(X')$ will mostly not change during such optimization. Furthermore, the size of available dataset also has its impact, since with more data the distance between training samples decreases (on average) and the volume of overall *influence* area increases.

In Figure 12.1 the *overfitting* nature of NN surface is illustrated in an experiment where 2D Gaussian distribution is inferred. When the same network is used and the number of samples is decreasing, the outcome is the spiky surface at the end of the optimization.

Onwards, in Figure 12.2 we can see the experiment where a small dataset of a size 10000 is used for the same pdf inference, and where the number of used layers is decreased. As

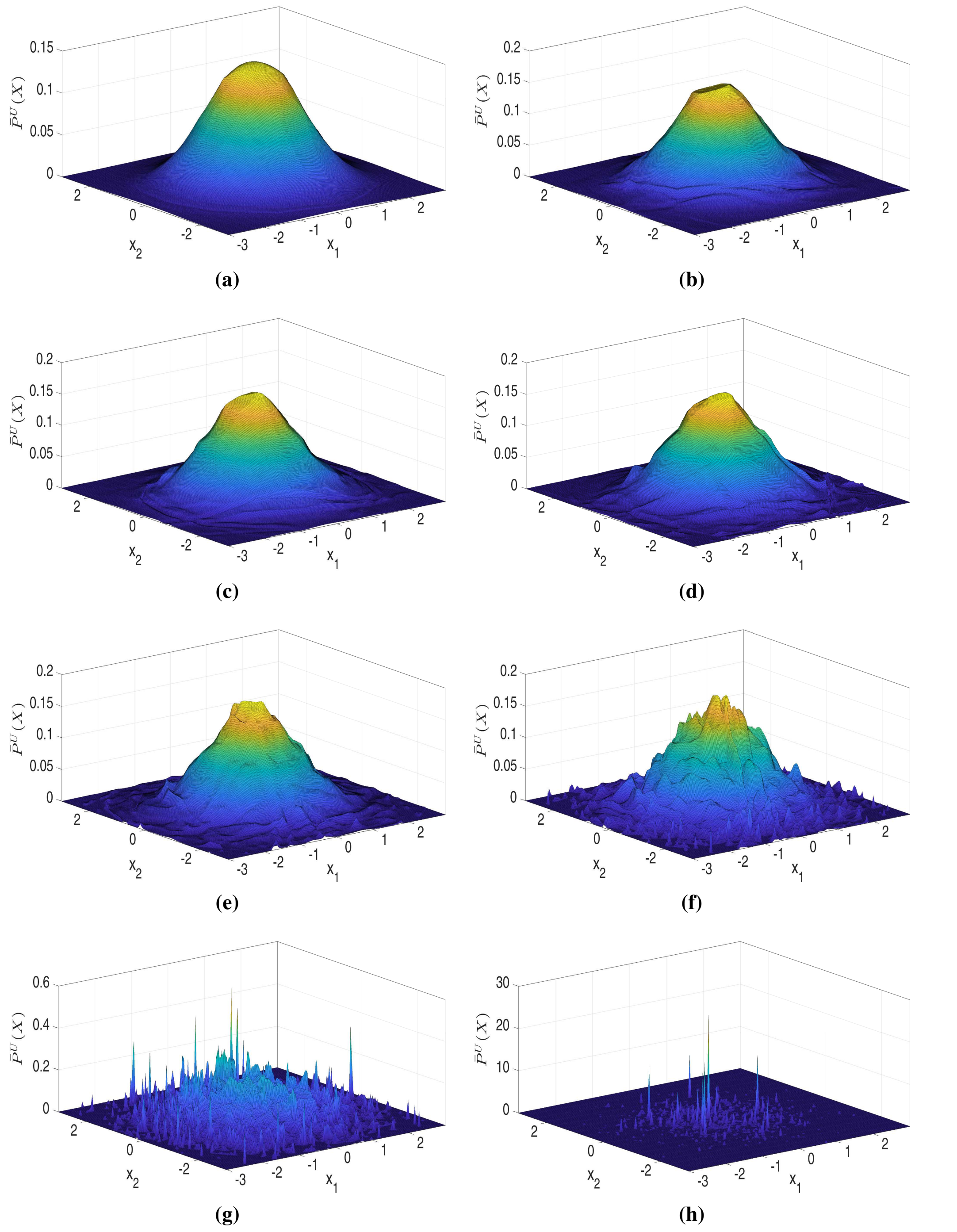


Figure 12.1: Illustration of PSO *overfitting* when the training dataset is small. We infer 2D *Normal* distribution via $\bar{\mathbb{P}}^U(X) = \exp f_\theta(X)$ by using PSO-LDE with $\alpha = \frac{1}{4}$ (see Table 5.1 and Section 8.2). The applied NN architecture is block-diagonal with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). Number of *up* training points $\{X_i^U\}$ is (a) 10^6, (b) 10^5, (c) 80000, (d) 60000, (e) 40000, (f) 20000, (g) 10000 and (h) 1000. As observed, when using the same NN architecture, that is when we do not reduce the flexibility level of the model, the smaller number of training points leads to the spiky approximation. In other words, the converged model will contain a peak around each training sample point.

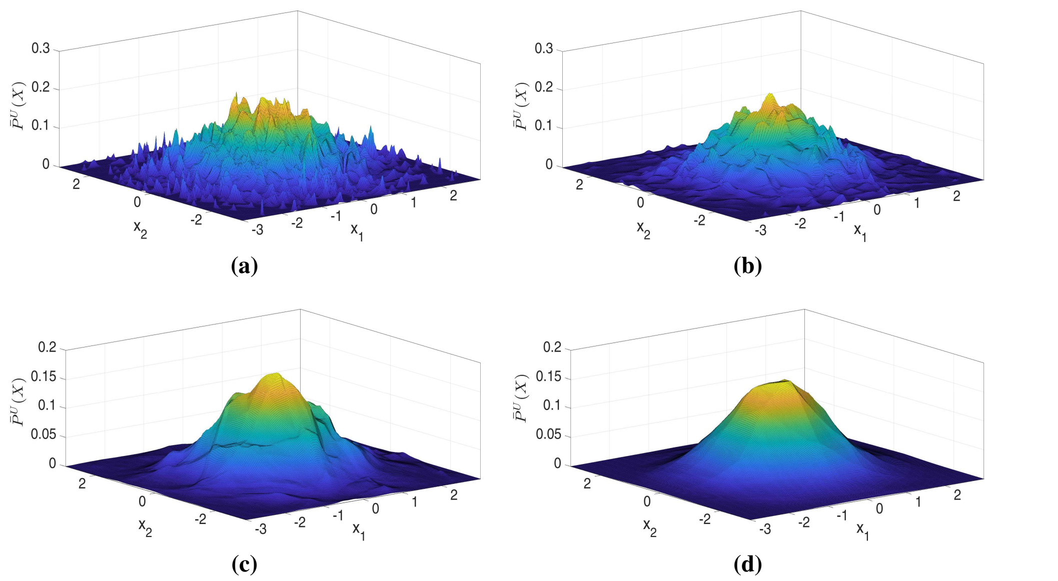


Figure 12.2: Illustration of decrease in PSO *overfitting* when the NN flexibility is reduced. We infer 2D *Normal* distribution via $\bar{\mathbb{P}}^U(X) = \exp f_\theta(X)$, using only 10000 training samples $\{X_i^U\}$. The applied loss is PSO-LDE with $\alpha = \frac{1}{4}$ (see Table 5.1 and Section 8.2). The applied NN architecture is block-diagonal with number of blocks $N_B = 20$ and block size $S_B = 64$ (see Section 11.1). Number of layers within NN is (a) 5, (b) 4, (c) 3 and (d) 2. As observed, when the number of layers is decreasing, the converged model is more smooth, with less peaks around the training points.

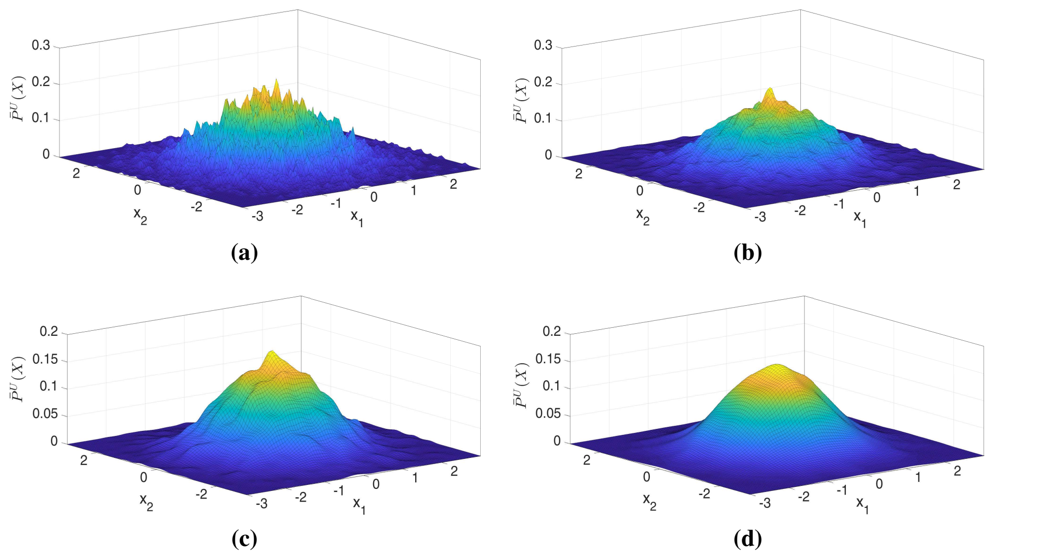
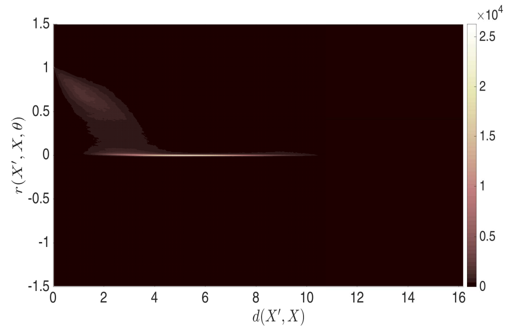
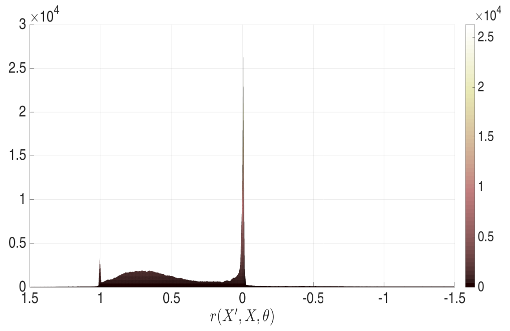


Figure 12.3: Illustration of decrease in KDE (kernel density estimation) *overfitting* when the *bandwidth* h of applied Gaussian kernel is increased. We infer 2D *Normal* distribution via KDE, using only 10000 training samples $\{X_i^U\}$. Used kernel has h equal to (a) 0.04, (b) 0.08, (c) 0.12 and (d) 0.2. As observed, when the *bandwidth* h is increasing, the converged model is more smooth, with less peaks around the training points. Similar trend is observed for PSO in Figure 12.2.

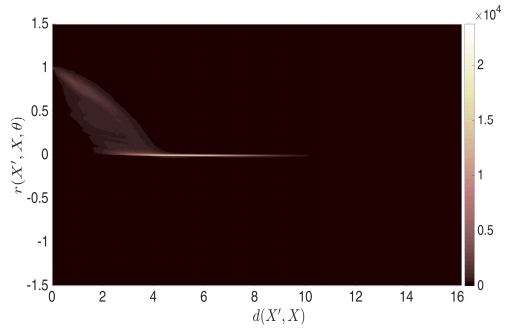
observed, with less layers the spiky nature of the surface is decreasing due to the reduced NN flexibility/capacity. Similar behavior is also observed in KDE method in Figure 12.3 where the bandwidth of Gaussian kernel is increasing; we can see that the surface estimated via KDE becomes more and more flexible for a smaller kernel *bandwidth*, similarly to what we observed in Figure 12.2. Thus, both KDE and PSO exhibit a similar flexibility behavior when the bandwidth of former is increased and when the layers depth of latter is reduced. Moreover,



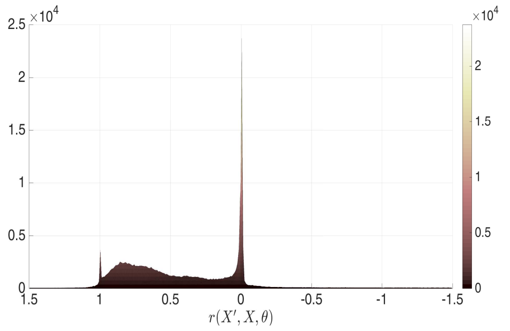
(a)



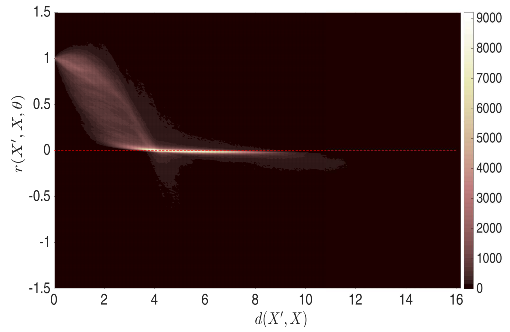
(b)



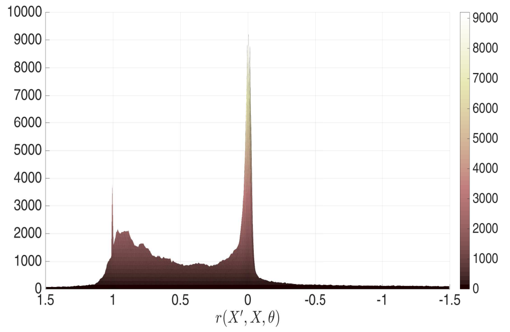
(c)



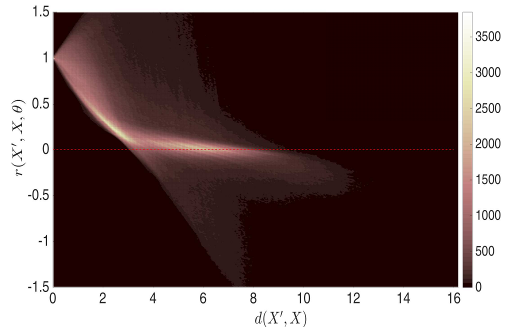
(d)



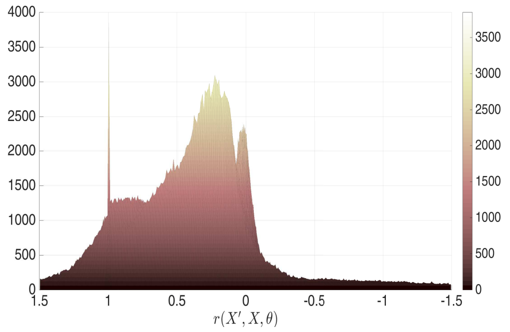
(e)



(f)



(g)



(h)

Figure 12.4: Illustration of NN flexibility and the corresponding bandwidth of $g_\theta(X, X')$, for each model in Figure 12.2. We calculate a *relative* model kernel $r_\theta(X_i, X_j)$ and a Euclidean distance $d(X_i, X_j)$ for $9 \cdot 10^6$ point pairs and depict a histogram of obtained $\{r_\theta(X_i, X_j)\}$ and $\{d(X_i, X_j)\}$ in left column. Likewise, a side view of this histogram is depicted in right column. Number of layers within NN is (a)-(b) 5, (c)-(d) 4, (e)-(f) 3 and (g)-(h) 2. See more details in the main text.

the reduction of layers in case of PSO produces a similar increase of $g_\theta(X, X')$'s bandwidth as we further show.

Particularly, in Figure 12.4 we present the bandwidth histogram of $g_\theta(X, X')$ for each trained model in Figure 12.2. We sample $\{X_i\}_{i=1}^{3000}$ testing points from 2D Gaussian and calculate *relative* side-influence $r_\theta(X_i, X_j)$ defined in Eq. (7.10) for each pair of points. Further, for each pair we also compute the Euclidean distance $d(X_i, X_j)$.

For $9 \cdot 10^6$ pairs of a *relative* side-influence $r_\theta(X_i, X_j)$ and a Euclidean distance $d(X_i, X_j)$ we construct a histogram in Figure 12.4. As observed, the *relative* side-influence is reduced with $d(X_i, X_j)$ - faraway points affect each other on much lower level. Further, we can see in left column of Figure 12.4 a sleeve right from a vertical line $d(X', X) = 0$ that implies an existence of overall local-support structure of $r_\theta(X_i, X_j)$, and a presence of some implicit kernel *bandwidth*. Likewise, we can also see a clear trend between $r_\theta(X_i, X_j)$ and the number of NN layers. For shallow networks (see Figures 12.4g-12.4h) the *relative* side-influence is strong even for faraway regions. In contrast, in deeper networks (see Figures 12.4a-12.4b) $r_\theta(X_i, X_j)$ is centered around zero for a pair of faraway points, with some close by points having non-zero side-influence. Hence, we can see the obvious relation between the network depth, the bandwidth of model kernel and the model flexibility, which supports conclusions made in Section 7.5.

Furthermore, since the impact of a kernel bandwidth is similar for PSO and KDE, we can compare our conclusions with well-known properties of KDE methods. For instance, optimality of the KDE bandwidth was already investigated in many works [24, 43, 95, 125] and is known to strongly depend on the number of training data samples. Hence, this implies that the optimal/"desired" bandwidth of $g_\theta(X, X')$ also depends on the size of training dataset, which agrees with statements of Section 7.5.

Remark 30. *In Figure 12.4a we can see that the side-influence between most points is zero, implying that gradients $\nabla_\theta f_\theta(X_i)$ at different points tend to be orthogonal for a highly flexible model. This gradient orthogonality does not present at NN initialization, and there is some mechanism that enforces it during NN training as shown in Appendix H. The nature of this mechanism is currently unknown.*

12.2 Possible Solutions

How big the training dataset should be and how to control NN flexibility to achieve the best performance are still open research questions. Some insights can be taken from KDE domain, yet we shall leave such analysis for future investigation. Further, the over-flexibility issue yields a significant challenge for application of PSO density estimators on small datasets, as well as also for other PSO instances. However, there are relatively simple regularization methods to reduce such *overfitting* and to eliminate peaks from the converged surface f_θ.

The first method is to introduce a weight regularization term into the loss, such as L2 norm of θ. This will enforce the weight vector to be inside a ball in the parameter space, thus limiting the flexibility of the NN. Yet, it is unclear what is the exact impact of any specific weight regularization method on the final surface and on properties of $g_\theta(X, X')$, due to highly non-linear nature of modern deep models. Typically, this regularization technique is used in try-and-fail regime, where different norms of θ and regularization coefficients are applied till a good performance is achieved.

Another arguably more consistent method is data augmentation, which is highly popular in Machine Learning. In context of PSO and its gradient in Eq. (3.1), we can consider to introduce an additive noise into each sample X_i^U as:

$$\bar{X}_i^U = X_i^U + \upsilon \tag{12.1}$$

where X_i^U is the original sample from the data density $\mathbb{P}^U(X)$ and υ is a random noise sampled from some density $\mathbb{P}^\upsilon(X)$ (e.g. Gaussian distribution). When using $\bar{X}_i^U$ instead of X_i^U, we will actually estimate the density of the random variable $\bar{X}_i^U$ which is the convolution between two densities. Thus, for PSO-LDE the converged surface $f_\theta(X)$ will be:

$$f_{\theta^*}(X) = \log\left([\mathbb{P}^U * \mathbb{P}^\upsilon](X)\right), \tag{12.2}$$

where $*$ defines the convolution operator.

Considering $\mathbb{P}^\upsilon(X)$ to be Gaussian and recalling that it is a solution of the *heat equation* [12], the above expression elucidates the effect of such data augmentation as a simple *diffusion* of the surface that would be estimated for the original X_i^U. That is, assuming that $f_\theta(X)$ would get a spiky form when approximating $\log \mathbb{P}^U(X)$, the updated target density function (and thus also its approximation f_θ) undergoes *diffusion* in order to yield a *smoother* final surface. In case of Gaussian noise, the smoothness depends on its covariance matrix. Yet, the other distributions can be used to perform appropriate convolution and to achieve different *diffusion* effects. We employ the above technique to improve an inference accuracy under a small training dataset setting in Section 13.2.5.

Remark 31. *Additionally, in context of image processing, a typical data augmentation involves image flipping, resizing and introducing various photographic effects [103, 147]. Such methods produce new samples $\bar{X}_i^U$ that are still assumed to have the original density $\mathbb{P}^U(X)$, which can be justified by our prior knowledge about the space of all possible images. Given this knowledge is correct, the final estimation is still of $\mathbb{P}^U(X)$ and not of its convolution (or any other operator) with the noise.*

CHAPTER 13

Experimental Evaluation of PSO Framework

Below we report several experimental scenarios that demonstrate the efficiency of the proposed PSO algorithm family. Concretely, in Section 13.2 we apply PSO to infer a pdf of 20D *Columns* distribution, where in sub-section 13.2.1 we compare between various PSO instances; in 13.2.2 we experiment with state-of-the-art baselines and compare their accuracy with PSO-LDE; in 13.2.3 we evaluate the pdf inference performance for different NN architectures; in 13.2.4 we investigate the impact of a batch size on PSO performance; and in 13.2.5 we show how different sizes of training dataset affect inference accuracy and explore different techniques to overcome difficulties of a small dataset setting. Furthermore, in Section 13.3 we perform pdf inference over a more challenging distribution *Transformed Columns*, in Section 13.4 we apply our pdf estimation approach over 3D densities generated from pixel landscape of RGB images, and in Section 13.5 we use PSO in robotics domain to infer joint density over robot poses and acquired measurements. Further, in Section 13.6 we solve the occupancy mapping problem via PSO-based technique. Additionally, in Appendix G we show that the first-order Taylor approximation of the surface *differential* in Eq. (7.5) is actually very accurate in practice, and in Appendix H we empirically explore dynamics of $g_\theta(X, X')$ and of its bandwidth during a learning process.

Importantly, our main focus in this paper is to introduce a novel paradigm for inferring various statistics of an arbitrary data in a highly accurate and consistent manner. To this end and concretely in context of density estimation, we are required to demonstrate **quantitatively** that the converged approximation $\bar{\mathbb{P}}_\theta(X)$ of the pdf function $\mathbb{P}(X)$ is indeed very close to its target. Therefore, except for Section 13.5 we mostly avoid experiments on real datasets (e.g. MNIST, [68]), since they lack information about the true pdf values of the samples. Instead, we generate datasets for our experiments from analytically known pdf functions, which allows us to evaluate the *ground truth* error between $\bar{\mathbb{P}}_\theta(X)$ and $\mathbb{P}(X)$. However, all selected pdf functions are highly multi-modal and therefore are very challenging to infer.

Likewise, in this work we purposely consider vector datasets instead of image data, to decouple our main approach from complexities coming with images and CNN models. Our main goal is to solve general unsupervised learning, and we do not want it to be biased towards spatial data. Moreover, vector data is mostly neglected in modern research and our method together with the new BD architecture addresses this gap.

13.1 Learning Setup

All the pdf inference experiments were done using Adam optimizer [59], since Adam showed better convergence rate compared to stochastic GD. Note that replacing GD with Adam does not change the target function approximated by PSO estimation, although it changes the implicit model kernel. That is, the variational PSO *balance state* in Eq. (3.2) stays the same while the convoluted equilibrium described in Section 7.4 will change according to the model kernel associated with Adam update rule.

The used Adam hyper-parameters are $\beta_1 = 0.75$, $\beta_2 = 0.999$ and $\epsilon = 10^{-10}$. Each experiment optimization is performed for 300000 iterations, which typically takes about one hour to run on a GeForce GTX 1080 Ti GPU card. The batch size is $N^U = N^D = 1000$. During each iteration next batch of *up* points $\{X_i^U\}_{i=1}^{1000}$ is retrieved from the training dataset of size N_{DT}, and next batch of *down* points $\{X_i^D\}_{i=1}^{1000}$ is sampled from *down* density $\mathbb{P}^D$. For *Columns* distribution in Section 13.2 we use a Uniform distribution as $\mathbb{P}^D$. Next, the optimizer updates the weights vector θ according to the loss gradient in Eq. (3.1), where the *magnitude* functions are specified by a particular PSO instance. The applied learning rate is 0.0035. We keep it constant for first 40000 iterations and then exponentially decay it down to a minimum learning rate of $3 \cdot 10^{-9}$. Further, in all our models we use Leaky-Relu as a non-linearity activation function. Additionally, weights are initialized via popular Xavier initialization [29]. Each model is learned 5 times; we report its mean accuracy and the standard deviation. Further, PSO implementation based on Tensorflow framework [135] can be accessed via open source library `https://bit.ly/2AMwyJT`.

To evaluate performance and consistency of each learned model, we calculate three different errors over testing dataset $\{X_i^U\}_{i=1}^N$, where each point was sampled from $\mathbb{P}^U$ and N is 10^5. First one is pdf squared error $PSQR = \frac{1}{N}\sum_{i=1}^N \left[\mathbb{P}^U(X_i^U) - \bar{\mathbb{P}}_\theta(X_i^U)\right]^2$, with $\bar{\mathbb{P}}_\theta(\cdot)$ being the pdf estimator produced by a specific model after an optimization convergence. Further, since we deal with high-dimensional data, $PSQR$ involves operations with very small numbers. To prevent inaccuracies caused by the computer precision limit, the second used error is log-pdf squared error $LSQR = \frac{1}{N}\sum_{i=1}^N \left[\log \mathbb{P}^U(X_i^U) - \log \bar{\mathbb{P}}_\theta(X_i^U)\right]^2$. Since in this thesis we target $\log \mathbb{P}(\cdot)$ in the first place, the $LSQR$ error expresses a distance between the data log-pdf and the learned NN surface $f_\theta(\cdot)$. Moreover, $LSQR$ is related to statistical divergence between $\mathbb{P}^U$ and $\bar{\mathbb{P}}_\theta$ (see more details in Appendix I).

Further, the above two errors require to know ground truth $\mathbb{P}^U$ for their evaluation. Yet, in real applications such ground truth is not available. As an alternative, we can approximate *PSO functional* in Eq. (3.3) over testing dataset. As explained in Section 6.1, this loss is equal to *PSO divergence* up to an additive constant, and thus can be used to measure a discrepancy between the PSO-optimized model and the target function. However, for most of the below applied PSO

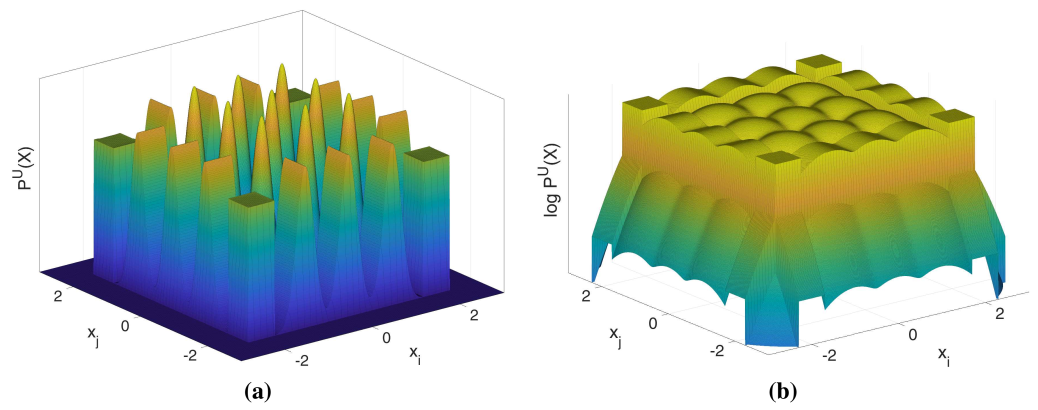


Figure 13.1: (a) Illustration of *Columns* distribution. Every slice of its pdf function $\mathbb{P}(x_i, x_j) = \mathbb{P}^U(0, \dots, 0, x_i, 0, \dots, 0, x_j, 0, \dots, 0)$ in Eq. (13.1) contains 25 modes of different shape. Overall, this distribution has 5^{20} modes. (b) Logarithm of pdf slice in (a) that will be learned via NN surface.

instances $L_{PSO}(f)$ is not analytically known and hence can not be computed. To overcome this problem and to measure model performance in real applications, we propose to use the loss of IS method from Table 5.1, $IS = -\frac{1}{N}\sum_{i=1}^{N} f_\theta(X_i^U) + \frac{1}{N}\sum_{i=1}^{N} \frac{\exp[f_\theta(X_i^D)]}{\mathbb{P}^D(X_i^D)}$, where $\{X_i^D\}_{i=1}^N$ are i.i.d. samples from $\mathbb{P}^D$. As we will see, while IS is less accurate than the ground truth errors, it still is a reliable indicator for choosing the best member from a set of learned models. Additionally, during the optimization IS is correlated with the real error and if required can be used to monitor current convergence and to allow an early stop evaluation.

Remark 32. *Note that unlike typical density estimator evaluation, herein we do not use performance metrics such as perplexity [55] and various kinds of f-divergences, or negative-likelihood scores. This is because the PSO-learned models are only approximately normalized, while the aforementioned metrics typically require strictly normalized models for their metric consistency. Still, both $PSQR$ and $LSQR$ are mean squared errors between target and approximation functions, and are similar to other performance metrics that are widely applied in regression problems of Machine Learning domain.*

13.2 PDF Estimation via PSO - *Columns* Distribution

In this Section we will infer a 20D *Columns* distribution from its sampled points, using various PSO instances and network architectures. The target pdf here is $\mathbb{P}^U(X) = \mathbb{P}^{Clmns}(X)$ and is defined as:

$$\mathbb{P}^{Clmns}(x_1, \dots, x_{20}) = \prod_{i=1}^{20} p(x_i), \tag{13.1}$$

where $p(\cdot)$ is a 1D mixture distribution with 5 components $\{Uniform(-2.3, -1.7), \mathcal{N}(-1.0, std = 0.2), \mathcal{N}(0.0, std = 0.2), \mathcal{N}(1.0, std = 0.2), Uniform(1.7, 2.3)\}$; each component has weight 0.2. This distribution has overall $5^{20} \approx 9.5 \cdot 10^{13}$ modes, making the structure of its entire pdf surface very challenging to learn. For the illustration see Figure 13.1a.

First, we evaluate the proposed density estimation methods under the setting of infinite

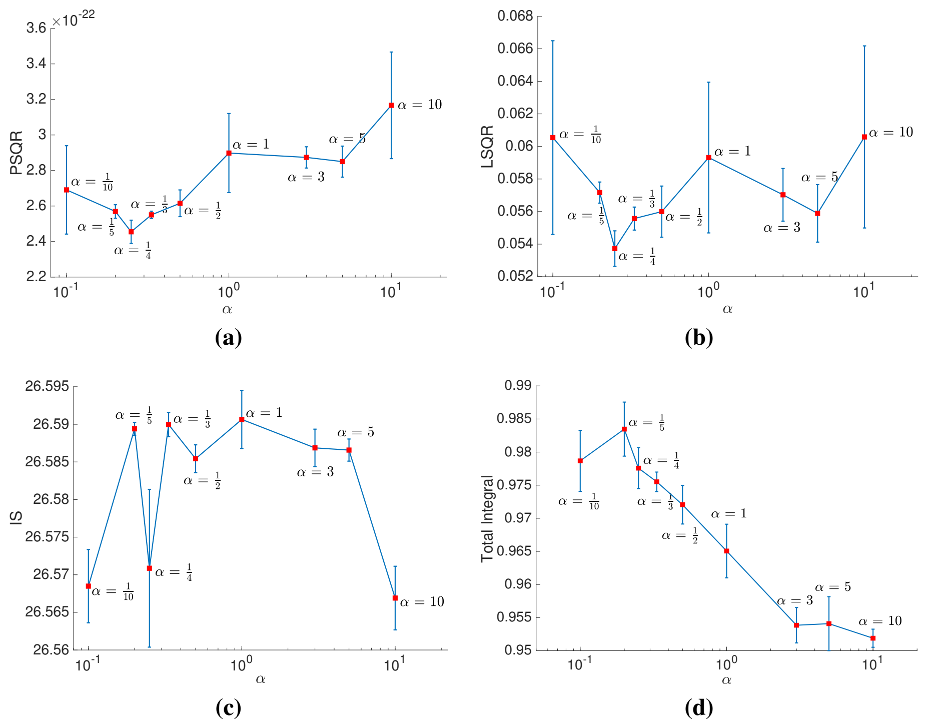


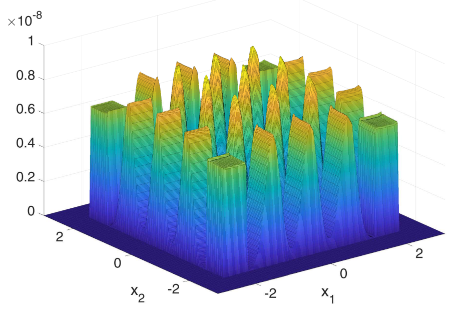
Figure 13.2: Evaluation of PSO-LDE for estimation of *Columns* distribution, where NN architecture is block-diagonal with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). For different values of a hyper-parameter α, (a) $PSQR$, (b) $LSQR$ and (c) IS are reported, along with their empirical standard deviation. (d) Estimators of the total integral $TI = \int \bar{\mathbb{P}}_\theta(X)dX$ of learned models for each value of α. For a specific learned model $\bar{\mathbb{P}}_\theta(X)$ this integral is estimated through importance sampling as $\overline{TI} = \sum_{i=1}^{N} \frac{\bar{\mathbb{P}}_\theta(X_i^D)}{\mathbb{P}^D(X_i^D)}$ over $N = 10^8$ samples from *down* density $\mathbb{P}^D$. Note that such estimator is consistent, with $TI = \overline{TI}$ for $N \to \infty$.

training dataset, with number of overall training points being $N_{DT} = 10^8$. Later, in Section 13.2.5 we will investigate how a smaller dataset size affects the estimation accuracy, and propose various techniques to overcome issues of sparse data scenario.

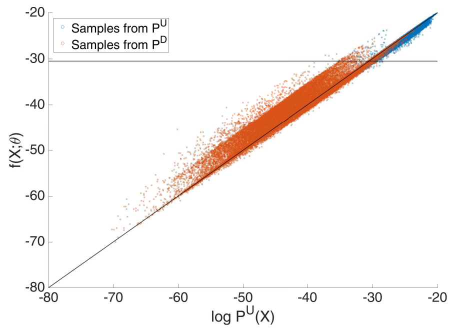
13.2.1 PSO Instances Evaluation

Here we perform pdf learning using different PSO instances, and compare their performance. The applied NN architecture is block-diagonal from Section 11.1, with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$.

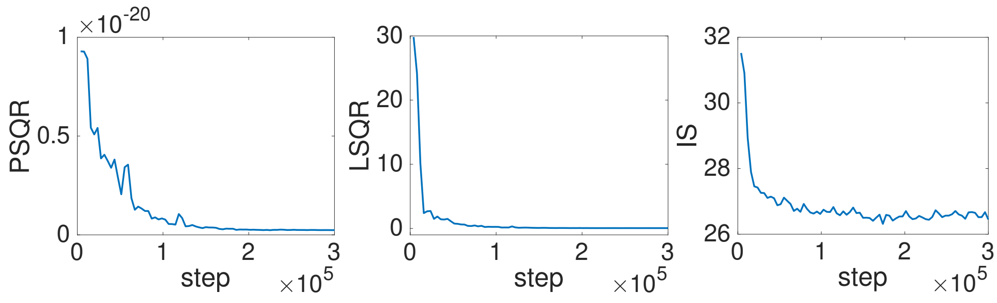
PSO-LDE and α First, we apply the PSO-LDE instances from Section 8.2, where we try various values for the hyper-parameter α. In Figure 13.2 we can see all three errors for different α. All models produce highly accurate pdf estimation, with average $LSQR$ being around 0.057. That is, the learned NN surface $f_\theta(X)$ is highly close to the target $\log \mathbb{P}^U(X)$. Further, we can see that some α values (e.g. $\alpha = \frac{1}{4}$) produce slightly better accuracy than others. This can be explained by smoother *magnitude* dynamics with respect to logarithm difference $\bar{d}$ from Eq. (8.5), that small values of α yield (see also Section 8.2). Note that here IS error is not very correlative with ground truth errors $PSQR$ and $LSQR$ since the accuracy of all models is very similar and IS is not sensitive enough to capture the difference.



(a)



(b)



(c)

Figure 13.3: Learned pdf function of *Columns* distribution by PSO-LDE with $\alpha = \frac{1}{4}$, where NN architecture is block-diagonal with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). (a) Illustration of learned pdf function. The depicted slice is $\mathbb{P}(x_1, x_2) = \bar{\mathbb{P}}^U(x_1, x_2, 0, \ldots, 0)$, with x_1 and x_2 forming a grid of points in first two dimensions of the density's support. As can be seen, all modes (within first two dimensions) and their appropriate shapes are recovered. (b) Illustration of the learned surface $f_\theta(X)$. Blue points are sampled from $\mathbb{P}^U$, while red points - from $\mathbb{P}^D$, minimal 20D Uniform distribution that covers all samples from $\mathbb{P}^U$. The x axis represents $\log \mathbb{P}^U(X)$ for each sample, y axis represents the surface height $f_\theta(X)$ after the optimization was finished. The diagonal line represents $f_\theta(X) = \log \mathbb{P}^U(X)$, where we would see all points in case of *perfect* model inference. The black horizontal line represents $\log \mathbb{P}^D(X) = -30.5$ which is a constant for the Uniform density. As can be seen, these two densities have a *relative* support mismatch - although their pdf values are not zero within the considered point space, the sampled points from both densities are obviously located mostly in different space neighborhoods. This can be concluded from values of $\log \mathbb{P}^U(X)$ that are very different for both point populations. Further, we can see that there are errors at both X^U and X^D locations, possibly due to high bias of the surface estimator $f_\theta(X)$ imposed by the model kernel g_θ (see also Figure 13.4). (c) Testing errors as functions of the optimization iteration. All three errors can be used to monitor the learning convergence. Further, IS error can be calculated without knowing the ground truth.

Further, we also estimate the total integral $TI = \int \bar{\mathbb{P}}_\theta(X)dX$ for each learned model via importance sampling. In Figure 13.2d we can see that the learned models are indeed very close to be *normalized*, with the estimated total integral being on average 0.97 - very close to the proper value 1. Note that in our experiments the model normalization was not enforced in any explicit way, and PSO-LDE achieved it via an implicit force equilibrium.

Furthermore, in Figure 13.2d it is also shown that smaller values of α are more properly *normalized*, which also correlates with the approximation error. Namely, in Figure 13.2a same models with smaller α are shown to have a lower error. We argue that further approximation improvement (e.g. via better NN architecture) will also increase the *normalization* quality of produced models.

Moreover, in Figure 13.3a we can see a slice of the learned $\exp f_\theta(X)$ for the first two dimensions, where the applied PSO instance was PSO-LDE with $\alpha = \frac{1}{4}$. As observed, it is highly close to the real pdf slice from Figure 13.1a. In particular, all modes and their shapes (within this slice) were recovered during learning. Further, in Figure 13.3b we can observe the learned surface height $f_\theta(X)$ and the ground truth height $\log \mathbb{P}^U(X)$ for test sample points from $\mathbb{P}^U$ and $\mathbb{P}^D$. As shown, there are approximation errors at both X^U and X^D, with *down* points having bigger error than *up* points. As we will see below, these errors are correlated with the norm of θ gradient at each point.

Additionally, in Figure 13.3b we can see an asymmetry of error w.r.t. horizontal line $\log \mathbb{P}^D(X) = -30.5$, where points above this line (mostly blue points) have a NN height $f_\theta(X)$ slightly lower than a target $\log \mathbb{P}^U(X)$, and points below this line (mostly red points) have a NN height $f_\theta(X)$ slightly higher than target $\log \mathbb{P}^U(X)$. This trend was observed in all our experiments. Importantly, this error must be accounted to an estimation bias (in contrast to an estimation variance), since the considered herein setting is of infinite dataset setting where theoretically the variance is insignificant.

Further, we speculate that the reason for such bias can be explained as follows. The points above this horizontal line have a positive logarithm difference $\bar{d} = f_\theta(X) + 30.5$ defined in Eq. (8.5), $\bar{d} \geq 0$, whereas points below this line have a negative $\bar{d} \leq 0$. From the relation between $\bar{d}$ and *magnitude* functions discussed in Section 8.2, we know that for "above" points the *up magnitude* $M^U(\cdot)$ is on average smaller than *down magnitude* $M^D(\cdot)$ (see Figure 8.2). The opposite trend can be observed within "below" points. There $M^D(\cdot)$ has smaller values relatively to $M^U(\cdot)$. Thus, the surface parts above this horizontal line have large *down magnitudes*, while parts below the line have large *up magnitudes*, which in its turn creates a global side-influence imposed via the model kernel. Finally, these global side influences generate this asymmetric error with $\log \mathbb{P}^D(X) = -30.5$ being the center of the pressure. Likewise, we argue that this asymmetric tendency can be reduced by selecting $\mathbb{P}^U$ and $\mathbb{P}^D$ densities that are closer to each other, as also enhancing NN architecture to be more flexible, with the side-influence between far away regions being reduced to zero. In fact, we empirically observed that NN architectures with bigger side-influence (e.g. FC networks) have a greater error asymmetry; the angle identified in Figure 13.3b between a point cloud and line $f_\theta(X) = \log \mathbb{P}^U(X)$ is bigger for a bigger overall side-influence within the applied model (see also Figure 13.9b below). We leave a more thorough investigation of this asymmetry nature for future research.

Further, the above asymmetry also clears out why all learned models in Figure 13.2d had the total integral less than 1. Since this integral is calculated by taking exponential over the learned $f_\theta(X)$, red points in Figure 13.3b almost do not have any impact on it, compared with the blue points (red points' exponential is much lower than exponential of blue points). Yet, blue points have smaller $\exp f_\theta(X)$ than their real pdf values $\mathbb{P}^U(X)$. Therefore, the total integral comes out to be slightly smaller than 1.

Also, in Figure 13.3c we can see all three errors along the optimization time; the IS is shown to monotonically decrease, similarly to ground truth errors. Hence, in theory it can be used in real applications where no ground truth is available, to monitor the optimization convergence.

Point-wise Error Furthermore, we empirically observe a direct connection between point-wise ground truth error and self *gradient similarity* $g_\theta(X, X)$ (squared norm of gradient $\nabla_\theta f_\theta(X)$

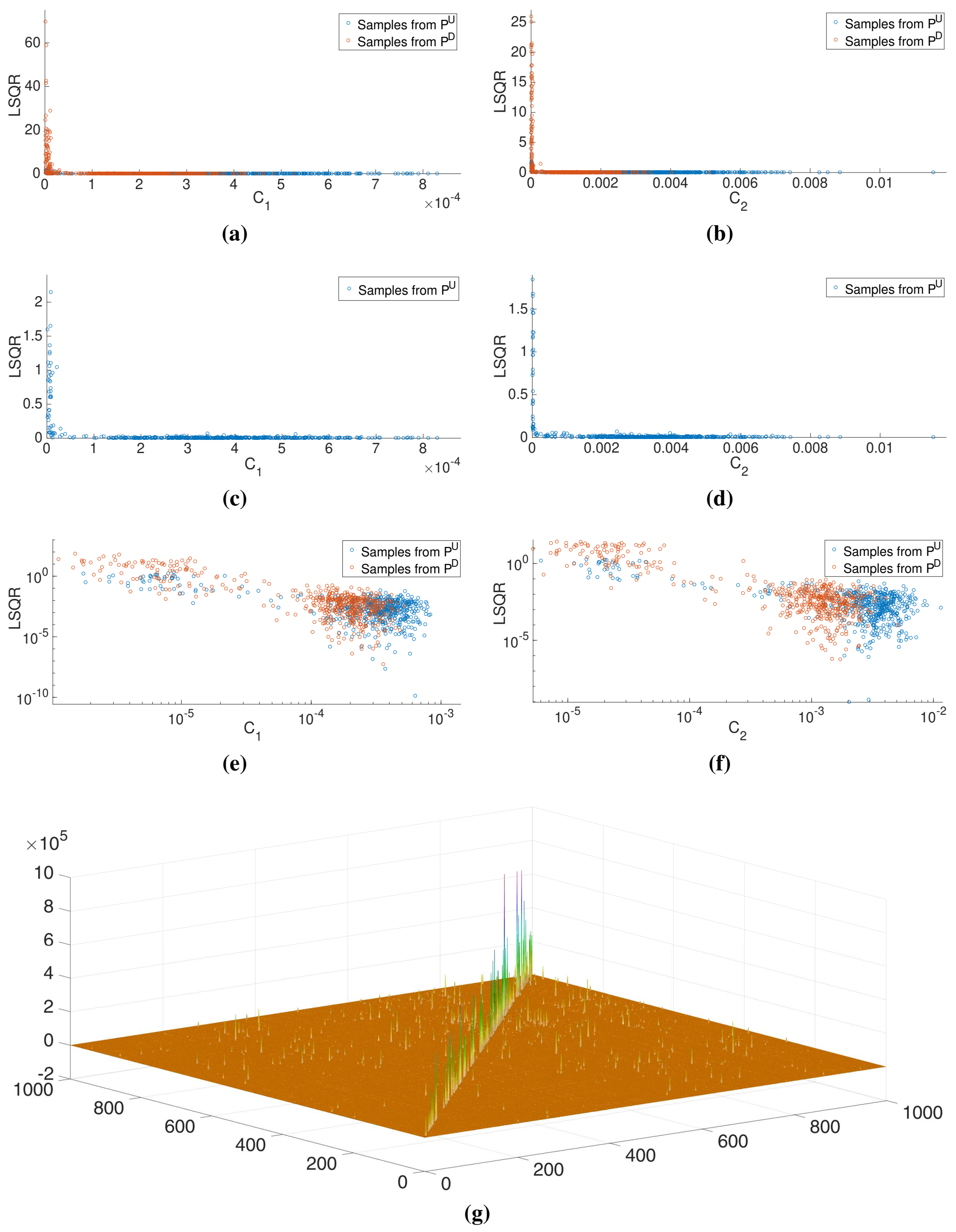


Figure 13.4: Relation between a point-wise error and a gradient norm. The pdf function of *Columns* distribution is learned by PSO-LDE with $\alpha = \frac{1}{4}$, where NN architecture is block-diagonal with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). (a)-(b) Relation between inverse-gradient-norm metrics C_1 and C_2 and a point-wise error $LSQR$. As can be seen, points with the smaller inverse-gradient-norm (that is, with a bigger norm of θ gradient) have a greater approximation error. See details in the main text. (c)-(d) Plots of (a)-(b) with only samples from $\mathbb{P}^U$ density. (e)-(f) Plots of (a)-(b) with both x and y axes scaled logarithmically. (g) Matrix G, with $G_{ij} = g_\theta(X_i, X_j)$.

at the point). To demonstrate this, we define two inverse-gradient-norm empirical metrics as follows. First, after training was finished we sample 1000 points $D = \{X_i\}$, where 500 are sampled from $\mathbb{P}^U$ and 500 - from $\mathbb{P}^D$, and calculate their gradients $\nabla_\theta f_\theta(X_i)$. Next, we compute the *Gramian* matrix G that contains all *gradient similarities* among the samples, with

Method	$PSQR$	$LSQR$	IS
PSO-LDE, averaged over all α	$2.7 \cdot 10^{-22} \pm 2.58 \cdot 10^{-23}$	0.057 ± 0.004	26.58 ± 0.01
IS	$1.79 \cdot 10^{-21} \pm 5 \cdot 10^{-22}$	0.46 ± 0.14	26.84 ± 0.07
PSO-MAX	$3.04 \cdot 10^{-22} \pm 1.55 \cdot 10^{-23}$	0.058 ± 0.002	26.57 ± 0.001

Table 13.1: Performance comparison between various PSO instances

$G_{ij} = g_\theta(X_i, X_j)$. Then, the first empirical metric C_1 for sample X_i is calculated as

$$C_1(X_i) = \frac{1}{G_{ii}} = \frac{1}{g_\theta(X_i, X_i)}. \tag{13.2}$$

The above $C_1(X_i)$ is bigger if $g_\theta(X, X)$ is smaller, and vice versa. The second metric C_2 is defined as

$$C_2(X_i) = \left[G^{-1}\right]_{ii}. \tag{13.3}$$

Since matrix G is almost diagonal (see Figure 13.4g), both C_1 and C_2 usually have a similar trend.

In Figure 13.4 we can see that the above metrics $C_1(X_i)$ and $C_2(X_i)$ are highly correlated with point-wise $LSQR(X_i) = \left[\log \mathbb{P}^U(X_i) - \log \bar{\mathbb{P}}_\theta(X_i)\right]^2$. That is, points with a bigger norm of the gradient $\nabla_\theta f_\theta(X)$ (bigger $g_\theta(X, X)$) have a bigger approximation error. One possible explanation for this trend is that there exists an estimation bias, which is amplified by a bigger gradient norm at the point. Further investigation is required to clarify this aspect. Concluding, we empirically demonstrate that in the infinite data setting we can measure model uncertainty (error) at query point X via a norm of its gradient. For a smaller dataset size the connection between the gradient norm and the approximation error is less obvious, probably because there we have another/additional factors that increase the approximation error (e.g. an estimation variance). Also, note that herein we use metrics C_1 and C_2 that are opposite-proportional to the gradient norm instead of using the gradient norm directly since the *inverse* relation is visually much more substantial.

Additionally, in Figure 13.4 it is visible that on average samples from $\mathbb{P}^D$ have a bigger gradient norm than samples from $\mathbb{P}^U$. This can explain why in Figure 13.3b we have higher error at samples from *down* density.

Other PSO Instances Further, several other PSO instances were executed to compare with PSO-LDE. First is the IS method from Table 5.1. As was discussed in Section 8.2, its *magnitude* functions are unbounded which may cause instability during the optimization. In Table 13.1 we can see that indeed its performance is much inferior to PSO-LDE with bounded *magnitudes*.

Additionally, we used an instance of a normalized family defined in Eq. (8.6), which we

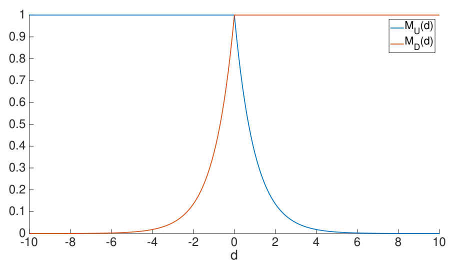


Figure 13.5: PSO-MAX *magnitudes* as functions of a difference $\bar{d}\left[X, f_\theta(X)\right] = f_\theta(X) - \log \mathbb{P}^D(X)$.

name PSO-MAX, with the following *magnitude* functions:

$$M^U\left[X, f_\theta(X)\right] = \frac{\mathbb{P}^D(X)}{\max\left[\mathbb{P}^D(X), \exp f_\theta(X)\right]} = \exp\left[-\max\left[\bar{d}\left[X, f_\theta(X)\right], 0\right]\right], \qquad (13.4)$$

$$M^D\left[X, f_\theta(X)\right] = \frac{\exp f_\theta(X)}{\max\left[\mathbb{P}^D(X), \exp f_\theta(X)\right]} = \exp\left[\min\left[\bar{d}\left[X, f_\theta(X)\right], 0\right]\right]. \qquad (13.5)$$

In Figure 13.5 the above *magnitudes* are depicted as functions of a logarithm difference $\bar{d}$ where we can see them to be also bounded. In fact, PSO-MAX is also an instance of PSO-LDE for a limit $\alpha \to \infty$. Similarly to other instances of PSO-LDE, the bounded *magnitudes* of PSO-MAX allow to achieve a high approximation accuracy, which gets very close to the performance of PSO-LDE for finite values of α (see Table 13.1). Yet, PSO-MAX is slightly worse, suggesting that very high values of α are sub-optimal for the task of pdf inference.

In overall, our experiments show that PSO instances with bounded *magnitudes* have superior performance at pdf inference task. Further, PSO-LDE with $\alpha = \frac{1}{4}$ has better accuracy w.r.t. other values of α. Note that this implies PSO-LDE with $\alpha = \frac{1}{4}$ is being superior to NCE [39, 126], which is PSO-LDE with $\alpha = 1$. Finally, in an infinite dataset setting and when using BD network architecture, we can measure model uncertainty of a specific query point X via the self *gradient similarity* $g_\theta(X, X)$.

13.2.2 Baselines

In the above section we showed that particular instances of PSO-LDE perform better than the NCE method (i.e. PSO-LDE with $\alpha = 1$). Likewise, in our previous work [61] we showed on 2D and 3D data that PSO-based methods are much more accurate than kernel density estimation (KDE) approach. Unfortunately, the KDE method does not scale well with higher dimensions, with very few implementations handling data of arbitrary dimension. Instead, below we evaluate score matching [50, 52, 119, 151], Masked Auto-encoder for Distribution Estimation (MADE) [27] and Masked Autoregressive Flow (MAF) [100] as state-of-the-art baselines in the context of density estimation.

Score Matching The originally introduced score matching approach [50] employed the following loss over samples $\{X_i^U\}_{i=1}^{N^U}$ from the target density $\mathbb{P}^U(X)$:

$$L_{SM}(\theta, \{X_i^U\}_{i=1}^{N^U}) = \frac{1}{N^U} \sum_{i=1}^{N^U} \sum_{j=1}^{n} \left[- \left(\frac{\partial^2 f_\theta(X_i^U)}{\partial (X_{ij}^U)^2} \right) + \frac{1}{2} \left(\frac{\partial f_\theta(X_i^U)}{\partial X_{ij}^U} \right)^2 \right], \tag{13.6}$$

where $\frac{\partial f_\theta(X_i^U)}{\partial X_{ij}^U}$ and $\frac{\partial^2 f_\theta(X_i^U)}{\partial (X_{ij}^U)^2}$ are first and second derivatives of $f_\theta(X_i^U)$ w.r.t. j-th entry of the n-dimensional sample X_i^U. Intuitively, we can see that this loss tries to construct a surface $f_\theta(X)$ where each sample point will be a local minima - its first derivative is "softly" enforced to be zero via the minimization of a term $\left(\frac{\partial f_\theta(X_i^U)}{\partial X_{ij}^U} \right)^2$, whereas the second one is "softly" optimized to be positive via maximization of $\left(\frac{\partial^2 f_\theta(X_i^U)}{\partial (X_{ij}^U)^2} \right)$. The inferred $f_\theta(X)$ of such optimization converges to the data *energy* function, which is proportional to the real negative log-pdf with some unknown partition constant, $\exp\left[-f_\theta(X)\right] \sim \mathbb{P}^U(X)$. Further, note that to optimize a NN model via $L_{SM}(\cdot)$, the typical GD-based back-propagation process will require to compute a third derivative of $f_\theta(X)$, which is typically computationally unfeasible for large NN models.

Due to the last point, in [119, 151] it was proposed to use the following loss as a proxy:

$$L_{SM}(\theta, \{X_i^U\}_{i=1}^{N^U}) = \frac{1}{N^U} \sum_{i=1}^{N^U} \left\| -v_i + \sigma^2 \cdot \frac{\partial f_\theta(X)}{\partial X} |_{X = X_i^U + v_i} \right\|_2^2, \tag{13.7}$$

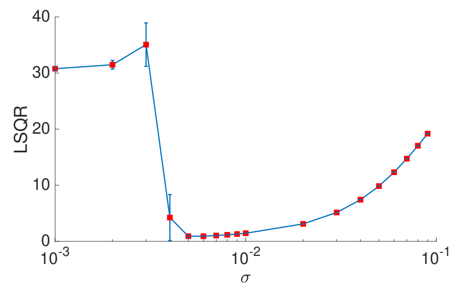
where v_i is zero-centered i.i.d. noise that is typically sampled from $\sim \mathcal{N}(0, \sigma^2 \cdot I)$. This "denoising" loss was shown in [143] to converge to the same target of the score matching loss in Eq. (13.6).

Furthermore, the above loss enforces $f_\theta(X)$ to converge to the data *energy* function. However, in this thesis we are interested to estimate the data log-pdf, which is proportional to the negative data *energy* function. To infer the latter via score matching, we employ the following sign change of the noise term:

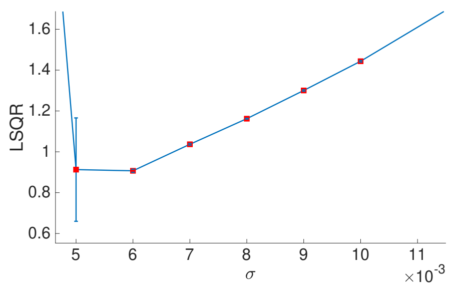
$$L_{SM}(\theta, \{X_i^U\}_{i=1}^{N^U}) = \frac{1}{N^U} \sum_{i=1}^{N^U} \left\| v_i + \sigma^2 \cdot \frac{\partial f_\theta(X)}{\partial X} |_{X = X_i^U + v_i} \right\|_2^2, \tag{13.8}$$

which has the same equilibrium as the loss in Eq. (13.7), yet with the negative sign. Namely, at a convergence $f_\theta(X)$ will satisfy now $\exp\left[f_\theta(X)\right] \sim \mathbb{P}^U(X)$. In our experiments we used this version of score matching loss for the density estimation of 20D *Columns* distribution.

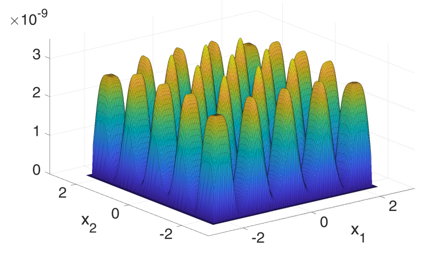
The employed learning setup of score matching is identical to PSO-LDE, with the loss in Eq. (13.8) being applied in a mini-batch mode, where at each optimization iteration a batch of samples $\{X_i^U\}_{i=1}^{N^U}$ was fetched from the training dataset and the new noise batch $\{v^i\}_{i=1}^{N}$ was generated. The learning rate of Adam optimizer was 0.003. Note that this method infers $\exp\left[f_\theta(X)\right]$ which is only proportional to the real pdf with some unknown partition constant. Therefore, in order to compute $LSQR$ of such model we also calculated its partition via importance sampling. Specifically, for each learned model $\exp\left[f_\theta(X)\right]$ its integral was



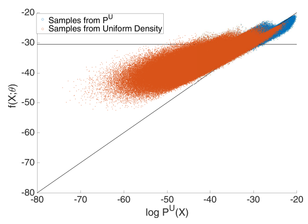
(a)



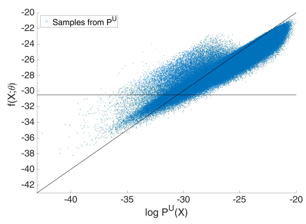
(b)



(c)



(d)



(e)

Figure 13.6: Learned pdf function of *Columns* distribution by score matching, where NN architecture is block-diagonal with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). The employed activation function is $tanh()$. (a) $LSQR$ error (mean and standard deviation) for various values of a scaling hyper-parameter σ; (b) Zoom of (a); (c) Illustration of learned pdf function for best model with $\sigma = 0.006$. The depicted slice is $\mathbb{P}(x_1, x_2) = \bar{\mathbb{P}}^U(x_1, x_2, 0, \ldots, 0)$, with x_1 and x_2 forming a grid of points in first two dimensions of the density's support. As can be seen, the estimated pdf is over-smoothed w.r.t. real pdf in Figure 13.1a. In contrast, PSO-LDE estimation in Figure 13.3a does not have this extra-smoothing nature. (d) Illustration of the learned surface $f_\theta(X)$. Blue points are sampled from $\mathbb{P}^U$, while red points - from $\mathbb{P}^D$, minimal 20D Uniform distribution that covers all samples from $\mathbb{P}^U$. The x axis represents $\log \mathbb{P}^U(X)$ for each sample, y axis represents the surface "normalized" height $\bar{f}_\theta(X) = f_\theta(X) - \log\left(\overline{TI}\right) = \log \bar{\mathbb{P}}_\theta(X)$ after optimization was finished. The diagonal line represents $\bar{f}_\theta(X) = \log \mathbb{P}^U(X)$, where we would see all points in case of *perfect* model inference. (e) Plot from (d) with only samples from $\mathbb{P}^U$. We can see that the produced surface is significantly less accurate than the one produced by PSO-LDE in Figures 13.3a-13.3b.

estimated through $\overline{TI} = \sum_{i=1}^{N^D} \frac{\exp\left[f_\theta(X_i^D)\right]}{\mathbb{P}^D(X_i^D)}$ over $N^D = 10^8$ samples from density $\mathbb{P}^D$, which is the minimal 20D Uniform distribution that covers all samples from $\mathbb{P}^U$. Further, we used $\bar{\mathbb{P}}_\theta(X) = \exp\left[f_\theta(X) - \log\left(\overline{TI}\right)\right]$ as the final estimation of data pdf.

Furthermore, we trained the score matching model for a range of σ values. After the explicit normalization of each trained model, in Figures 13.6a-13.6b we can see the $LSQR$ error for each value of a hyper-parameter σ. Particularly, for $\sigma = 0.006$ we got the smaller error $LSQR = 0.907 \pm 0.0075$, which is still much inferior to the accuracy obtained by PSO-LDE. Moreover, in Figure 13.6c we can see that the estimated surface is over-smoothed and does not accurately approximate sharp edges of the target pdf. In contrast, PSO-LDE produces a very close pdf estimation of an arbitrary shape, as was shown in Section 13.2.1. Likewise, comparing Figures 13.6d and 13.3b we can see again that PSO-LDE yields a much better accuracy.

Masked Auto-encoder for Distribution Estimation This technique is based on the autoregressive property of density functions, $\mathbb{P}(x_1, \ldots, x_n) = \prod_{i=1}^{n} \mathbb{P}(x_i | x_1, \ldots, x_{i-1})$, where each conditional $\mathbb{P}(x_i | x_1, \ldots, x_{i-1})$ is parameterized by NN. MADE constructs a network with sequential FC layers, where the autoregressive property is preserved via masks applied on activations of each layer [27]. Likewise, each conditional can be modeled as 1D density of

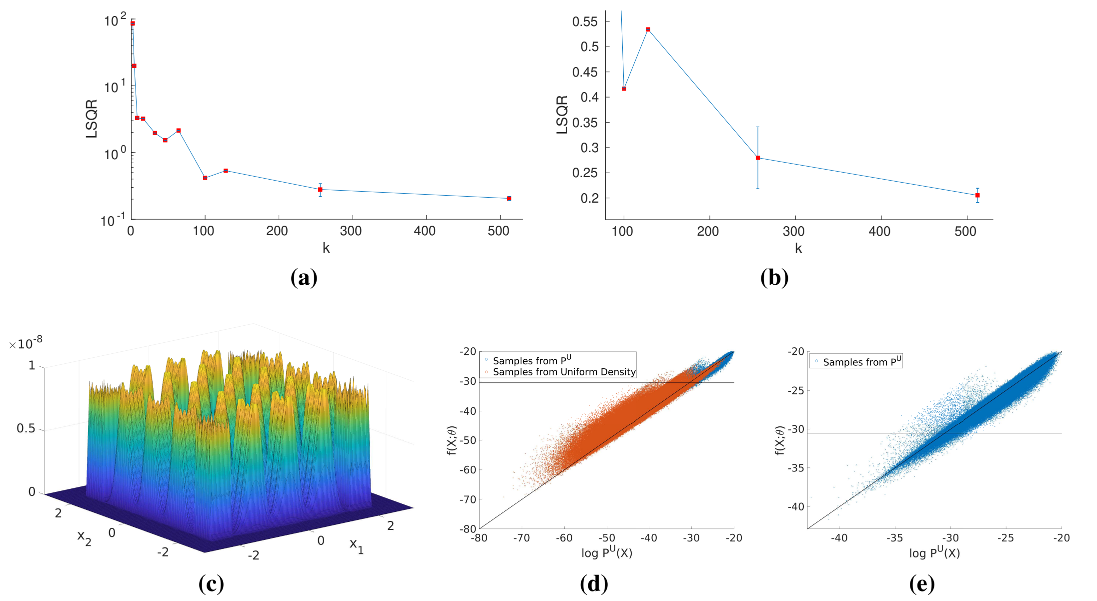
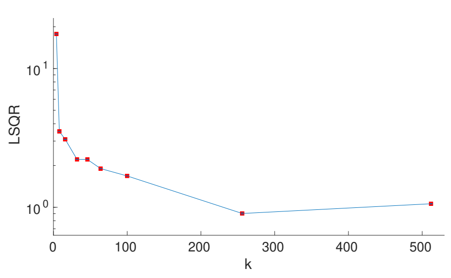


Figure 13.7: Learned pdf function of *Columns* distribution by MADE, where NN architecture is fully-connected with 4 layers of size 1024. The employed activation function is Relu. (a) $LSQR$ error (mean and standard deviation) for various values of a k - a number of mixture components; (b) Zoom of (a); (c) Illustration of learned pdf function for the best model with $k = 512$. The depicted slice is $\mathbb{P}(x_1, x_2) = \bar{\mathbb{P}}^U(x_1, x_2, 0, \dots, 0)$, with x_1 and x_2 forming a grid of points in first two dimensions of the density's support. As can be seen, the estimated pdf is over-spiky in areas where the real pdf in Figure 13.1a is flat. This is due to an inability of MoG model to represent flat non-zero surfaces. In contrast, PSO-LDE estimation in Figure 13.3a does not have this issue. (d) Illustration of the estimated pdf $\bar{\mathbb{P}}_\theta(X)$. Blue points are sampled from $\mathbb{P}^U$, while red points - from $\mathbb{P}^D$, minimal 20D Uniform distribution that covers all samples from $\mathbb{P}^U$. The x axis represents $\log \mathbb{P}^U(X)$ for each sample, y axis represents $\bar{f}_\theta(X) \triangleq \log \bar{\mathbb{P}}_\theta(X)$ after optimization was finished. The diagonal line represents $\bar{f}_\theta(X) = \log \mathbb{P}^U(X)$, where we would see all points in case of *perfect* model inference. (e) Plot from (d) with only samples from $\mathbb{P}^U$.

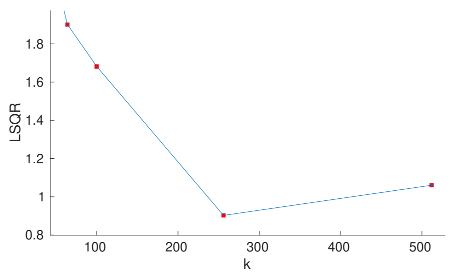
any known distribution family, with a typical choice being Gaussian or Mixture of Gaussians (MoG).

In our experiments we used MoG with k components to model each conditional, due to the highly multi-modal nature of *Columns* distribution. Moreover, we evaluated MADE for a range of various k, to see how the components number affects the technique's performance. Furthermore, the learning setup was similar to other experiments, with the only difference that the applied NN architecture was FC, with 4 layers of size 1024 each, and the exploited non-linearity was Relu.

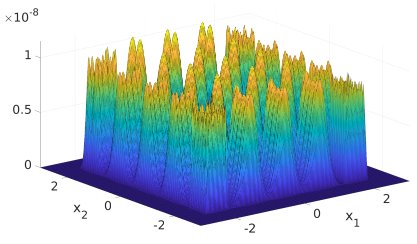
In Figures 13.7a-13.7b the $LSQR$ error is shown for each value of k. We can clearly see that with higher number of components the accuracy improves, where the best performance was achieved by $k = 512$ with $LSQR = 0.2 \pm 0.0141$. Furthermore, in Figure 13.7c we can see an estimated surface for the best learned model. As observed, most of the MoG components are spent to represent flat peaks of the target density. Such outcome is natural since for MoG to approximate flat areas the value of k has to go to infinity. Moreover, this demonstrates the difference between parametric and non-parametric techniques. Due to an explicit parametrization of each conditional, MADE can be considered as a member of the former family, while PSO-LDE is definitely a member of the latter. Further, non-parametric approaches are known to be more robust/flexible in general. In overall, we can see that PSO-LDE outperforms MADE even for a large number of mixture components.



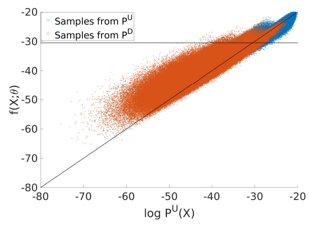
(a)



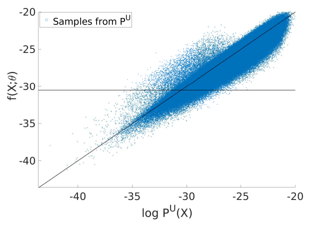
(b)



(c)



(d)



(e)

Figure 13.8: Learned pdf function of *Columns* distribution by MAF, with 5 inner MADE bijections and MADE MoG as a base density. (a) $LSQR$ error (mean and standard deviation) for various values of a k - a number of mixture components; (b) Zoom of (a); (c) Illustration of learned pdf function for the best model with $k = 256$. The depicted slice is $\mathbb{P}(x_1, x_2) = \bar{\mathbb{P}}^U(x_1, x_2, 0, \ldots, 0)$, with x_1 and x_2 forming a grid of points in first two dimensions of the density's support. The same over-spiky behavior can be observed as in Figure 13.7c. (d) Illustration of the estimated pdf $\bar{\mathbb{P}}_\theta(X)$ for the best model, constructed similarly to Figure 13.7d. (e) Plot from (d) with only samples from $\mathbb{P}^U$.

Masked Autoregressive Flow Shortly MAF, this technique combines an NN architecture of the previous MADE method with the idea of a normalizing flow [113] where a bijective transformation $h(\cdot)$ is applied to transform a priori chosen base density into the target density. Such bijective transformation allows to re-express the density of target data via an inverse of $h(\cdot)$ and via the known pdf of a base density, and further to infer the target pdf via a standard MLE loss. Moreover, the architecture of MADE can be seen as such bijective transformation, which is specifically exploited by MAF method [100]. Particularly, several MADE transformations are stuck together into one large bijective transformation, which allows for richer representation of the inferred pdf. In our experiments we evaluated MAF method with 5 inner MADE bijections.

Furthermore, the original paper proposed two MAF types. First one, referred as MAF($\cdot$) in the paper, uses multivariate normal distribution as a base density. During the evaluation this type did not succeed to infer 20D *Columns* distribution at all, probably because of its inability to handle distribution with 5^{20} modes.

The second MAF type, referred as MAF MoG($\cdot$) in the paper, uses MADE MoG as a base density in addition to the MADE-based bijective transformation. This type showed better performance w.r.t. first type, and we used it as an another baseline. Likewise, note that also in the original paper [100] this type was shown on average to be superior between the two.

Like in MADE experiments, also here we tested MAF MoG for different values of k - mixture components number of the base density, parametrized by a separate MADE MoG model. In Figures 13.8a-13.8b the $LSQR$ error is shown for each value of k. As in MADE case, also here accuracy improves with a higher number of components. The top accuracy was achieved by $k = 256$ with $LSQR = 0.9 \pm 0.009$. On average, MAF MoG showed the same trends as MADE method, yet with some higher $LSQR$ error. Moreover, during the experiments it was observed as a highly unstable technique, with thorough hyper-parameter tuning needed to overcome numerical issues of this approach.

In overall, we observed that non-parametric PSO-LDE is superior to other state-of-the-art baselines when dealing with highly multi-modal *Columns* distribution.

13.2.3 NN Architectures Evaluation

Here we compare performance of various NN architectures for the pdf estimation task.

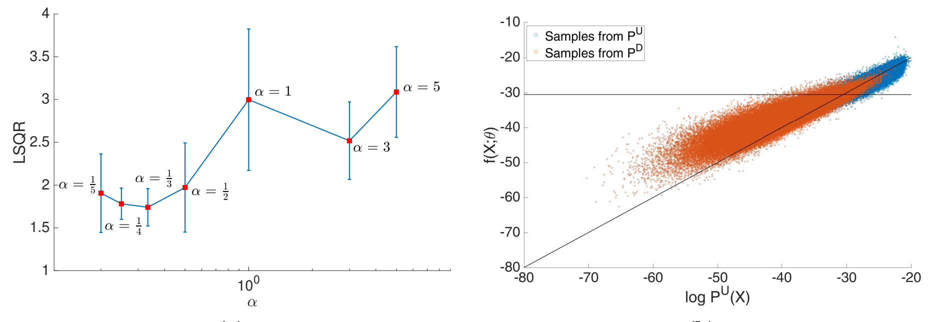
FC Architecture We start with applying PSO-LDE with different values of α where the used NN architecture is now fully-connected (FC), with 4 layers of size 1024. In Figure 13.9a we show $LSQR$ for different α, where again we can see that $\alpha = \frac{1}{4}$ (and now also $\alpha = \frac{1}{3}$) performs better than other values of α. On average, $LSQR$ error is around 2.5 which is significantly higher than 0.057 for BD architecture. Note also that BD network, used in Section 13.2.1, is twice smaller than FC network, containing only 902401 weights in BD vs 2121729 in FC, yet it produced a significantly better performance.

Further, in Figure 13.9b we illustrated the learned surface $f_\theta(X)$ for a single FC model with $\alpha = \frac{1}{4}$. Compared with Figure 13.3b, we can see that FC architecture produces a much less accurate NN surface. We address it to the fact that in BD network the *gradient similarity* $g_\theta(X, X')$ has much smaller overall side-influence (bandwidth) and the induced bias compared to the FC network, as was demonstrated in Section 11.1. Hence, BD models are more flexible than FC and can be pushed closer to the target function $\log \mathbb{P}^U(X)$, producing more accurate estimations.

Additionally, note the error asymmetry in Figure 13.9b which was already observed in Figure 13.3b. Also here we can see that the entire cloud of points is rotated from zero error line $f_\theta(X) = \log \mathbb{P}^U(X)$ by some angle where the rotation axis is also around horizontal line $\log \mathbb{P}^D(X) = -30.5$. As explained in Section 13.2.1, according to our current hypotheses there are global *up* and *down* side-influence forces that are responsible for this angle.

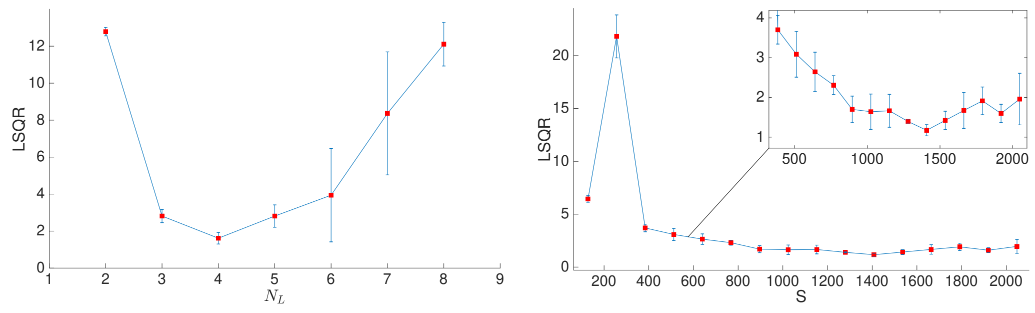
Further, to ensure that FC architecture can not produce any better results for the given inference task, we also evaluate it for different values of N_L and S - number of layers and size of each layer respectively. In Figure 13.10 we see that $N_L = 4$ and $S = 1408$ achieve best results for FC NN. Yet, the achieved performance is only $LSQR = 1.17$, which is still nowhere near the accuracy of BD architecture.

BD Architecture Further, we performed learning with a BD architecture, but with increasing number of blocks N_B. For N_B taking values between 20 and 200, in Figure 13.11 we can see that with bigger N_B there is improvement in approximation accuracy. This can be explained by the fact that bigger N_B produces bigger number of independent transformation channels inside NN; with more such channels there is less parameter sharing and side-influence between far



(a) **(b)**

Figure 13.9: Evaluation of PSO-LDE for estimation of *Columns* distribution, where NN architecture is fully-connected with 4 layers of size 1024 (see Section 11.1). (a) For different values of a hyper-parameter α, $LSQR$ error is reported along with its empirical standard deviation. (b) Illustration of the learned surface $f_\theta(X)$. Blue points are sampled from $\mathbb{P}^U$, while red points from $\mathbb{P}^D$. The x axes represent $\log \mathbb{P}^U(X)$ for each sample, y axes - the surface height $f_\theta(X)$ after optimization was finished. Diagonal line represents $f_\theta(X) = \log \mathbb{P}^U(X)$, where we would see all points in case of *perfect* model inference. The black horizontal line represents $\log \mathbb{P}^D(X) = -30.5$ which is constant for the Uniform density. As can be seen, there are high approximation errors at both X^U and X^D locations. Compared with BD architecture in Figure 13.3b, on average the error is much higher for FC network.



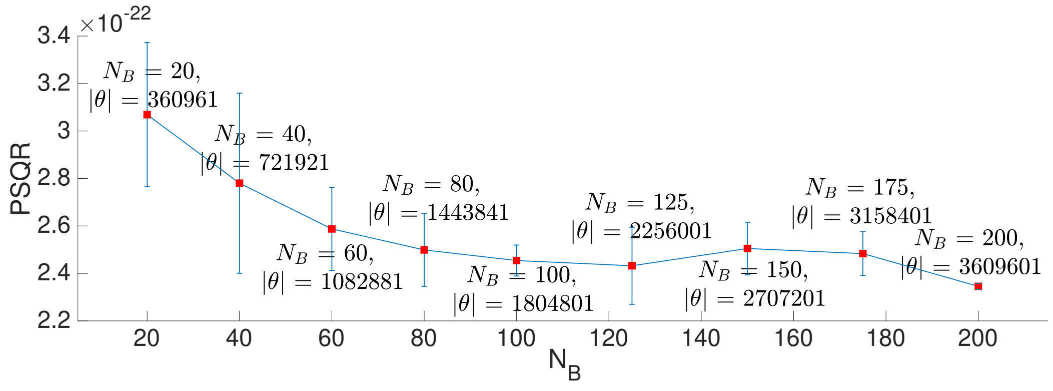
(a) **(b)**

Figure 13.10: Evaluation of PSO-LDE for estimation of *Columns* distribution, where NN architecture is fully-connected (see Section 11.1). The applied loss is PSO-LDE with $\alpha = \frac{1}{4}$. (a) For different number of layers N_L, $LSQR$ error is reported, where a size of each layer is $S = 1024$. (b) For different values of layer size S, $LSQR$ error is reported, where a number of layers is $N_L = 4$.

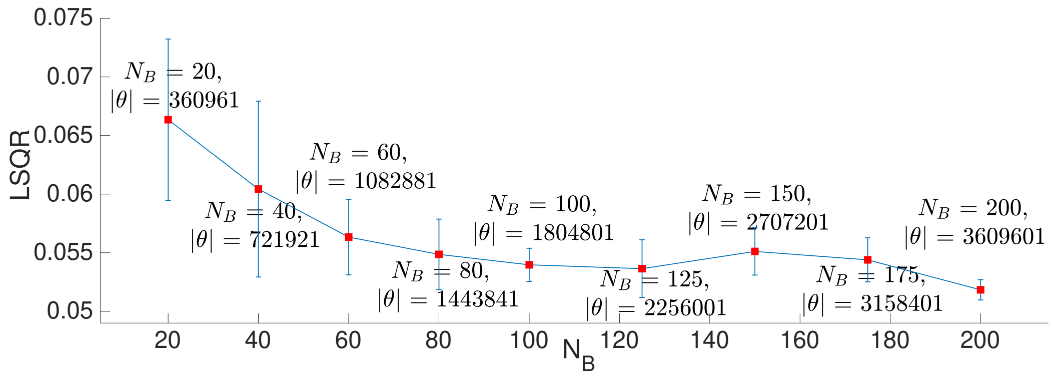
away input regions - different regions on average rely on different transformation channels. As a result, the NN becomes highly flexible. Further, in the setting of infinite dataset such high NN flexibility is desirable, and leads to a higher approximation accuracy. In contrast, in Section 13.2.5 below we will see that for a smaller dataset size the relation between NN flexibility and the accuracy is very different.

Likewise, we experiment with number of layers N_L to see how the network depth of a BD architecture affects the accuracy of pdf inference. In Figure 13.12 we see that deeper networks allow us to further decrease $LSQR$ error to around 0.03. Also, we can see that at some point increasing N_L causes only a slight error improvement. Thus, increasing N_L beyond that point is not beneficial, since for a very small error reduction we will pay with higher computational cost due to the increased size of θ.

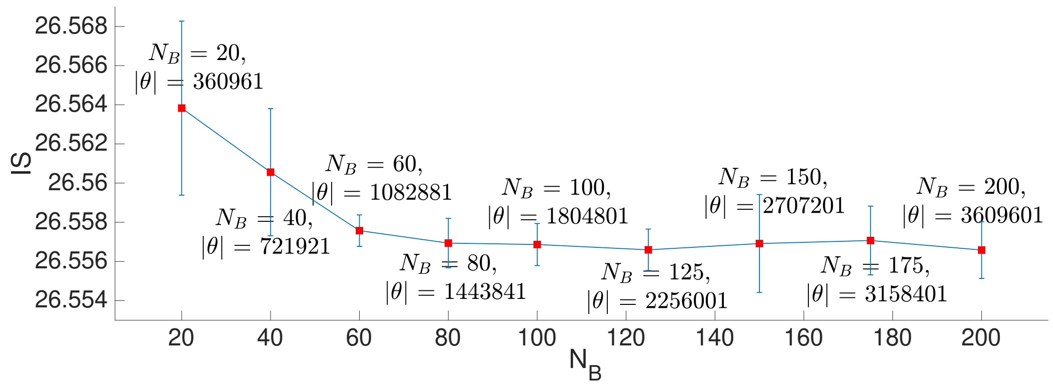
Furthermore, in Figure 13.13 we evaluate BD performance for different sizes of blocks S_B. Here we don't see anymore a monotonic error decrease that we observed above for N_B and N_L. The error is big for S_B below 32 or above 160. This probably can be explained as follows. For a small block size S_B each independent channel has too narrow width that is not



(a)

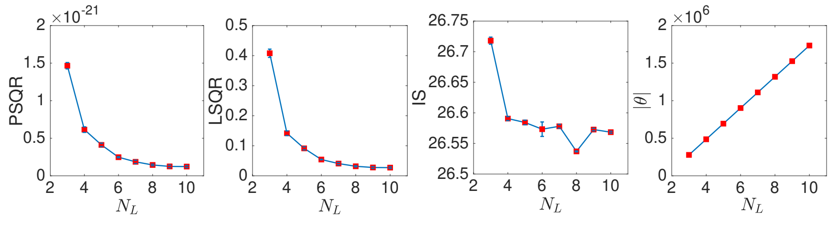


(b)

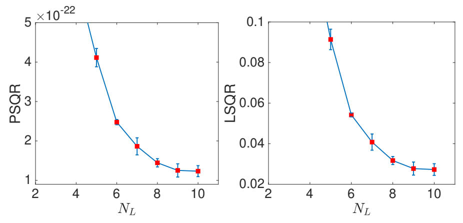


(c)

Figure 13.11: Evaluation of PSO-LDE for estimation of *Columns* distribution, where NN architecture is block-diagonal with 6 layers and block size $S_B = 64$ (see Section 11.1). The number of blocks N_B is changing. The applied loss is PSO-LDE with $\alpha = \frac{1}{4}$. For different values of N_B, (a) $PSQR$ (b) $LSQR$ and (c) IS are reported. As observed, the bigger number of blocks (e.g. independent channels) N_B improves the pdf inference.



(a)



(b)

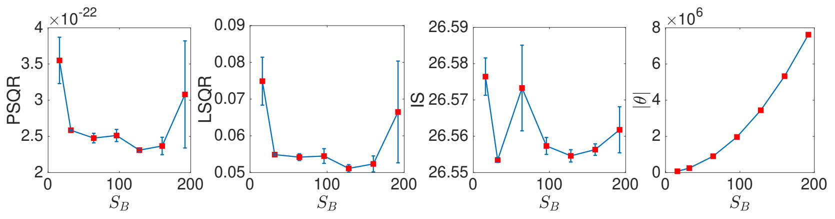
Figure 13.12: Evaluation of PSO-LDE for estimation of *Columns* distribution, where NN architecture is block-diagonal with number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). The number of layers N_L is changing from 3 to 10. The applied loss is PSO-LDE with $\alpha = \frac{1}{4}$. (a) For different values of N_L we report $PSQR$, $LSQR$ and IS, and their empirical standard deviation. Additionally, in the last column we depict the size of θ for each value of N_L. (b) Zoom of (a).

Method	$PSQR$	$LSQR$	IS
BD, ✓ DT, ✓ HB	$2.48 \cdot 10^{-22} \pm 6.63 \cdot 10^{-24}$	0.054 ± 0.0009	26.57 ± 0.01
BD, ✗ DT, ✓ HB	$2.49 \cdot 10^{-22} \pm 9.8 \cdot 10^{-24}$	0.055 ± 0.0021	26.57 ± 0.002
BD, ✓ DT, ✗ HB	$2.51 \cdot 10^{-22} \pm 1.1 \cdot 10^{-23}$	0.056 ± 0.0026	26.57 ± 0.002
BD, ✗ DT, ✗ HB	$3 \cdot 10^{-22} \pm 4.79 \cdot 10^{-23}$	0.066 ± 0.011	26.57 ± 0.005
FC, ✓ DT, ✓ HB	$5.56 \cdot 10^{-18} \pm 1.02 \cdot 10^{-17}$	1.78 ± 0.18	27.29 ± 0.05
FC, ✗ DT, ✓ HB	$1.2 \cdot 10^{-16} \pm 2.67 \cdot 10^{-16}$	1.35 ± 0.058	27.157 ± 0.029
FC, ✓ DT, ✗ HB	$7.9 \cdot 10^{-21} \pm 2.37 \cdot 10^{-21}$	2.38 ± 0.3	27.52 ± 0.08
FC, ✗ DT, ✗ HB	$1.36 \cdot 10^{-12} \pm 3 \cdot 10^{-12}$	2.5 ± 0.23	27.54 ± 0.08

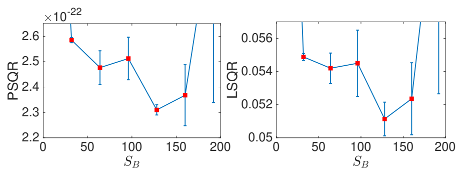
Table 13.2: Performance comparison between various NN pre-conditioning ways. The pdf function of *Columns* distribution is learned by PSO-LDE with $\alpha = \frac{1}{4}$. The applied models are fully-connected (FC) with 4 layers of size 1024, and block-diagonal (BD) with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). Evaluated pre-conditioning techniques are the data normalization in Eq. (11.3) (DT), and the height bias in Eq. (11.4) (HB).

enough to properly transfer the required signal from NN input to output. Yet, surprisingly it still can achieve a very good approximation, yielding $LSQR$ error of 0.075 for $S_B = 16$ with only 72001 weights, which is very impressive for such small network. Further, for a large block size S_B each independent channel becomes too wide, with information from too many various regions in $\mathbb{R}^n$ passing through it. This in turn causes interference (side-influence) between different regions and reduces overall NN flexibility, similarly to what is going on inside a regular FC network.

NN Pre-conditioning Finally, we verified efficiency of pre-conditioning techniques proposed in Section 11.2, namely the data normalization in Eq. (11.3) and the height bias in Eq. (11.4). In Table 13.2 we see that both methods improve the estimation accuracy. Further, in case the used model is FC, the $LSQR$ error improvement produced by the height bias is much more



(a)



(b)

Figure 13.13: Evaluation of PSO-LDE for estimation of *Columns* distribution, where NN architecture is block-diagonal with 6 layers and number of blocks $N_B = 50$ (see Section 11.1). The block size S_B is taking values $\{16, 32, 64, 96, 128, 160, 192\}$. The applied loss is PSO-LDE with $\alpha = \frac{1}{4}$. (a) For different values of S_B we report $PSQR$, $LSQR$ and IS, and their empirical standard deviation. Additionally, in the last column we depict the size of θ for each value of S_B. (b) Zoom of (a).

significant. Yet, for FC architecture it is unclear if the data normalization indeed helps.

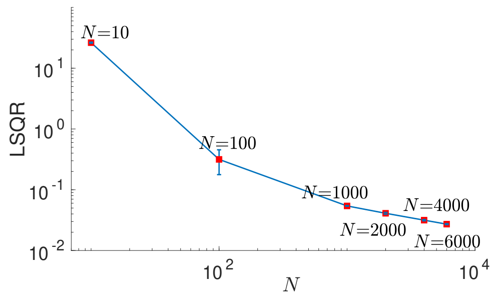
Overall, our experiments **combined** with empirical observations from Section 11.1.1 show that BD architecture has a smaller side-influence (small values of $g_\theta(X, X')$ for $X \neq X'$) and a higher flexibility than FC architecture. This in turn yields superior accuracy for BD vs FC networks. Moreover, in an infinite dataset setting we can see that further increase of NN flexibility by increasing N_B or N_L yields even a better approximation accuracy. The block size S_B around 64 produces a better performance in general. Yet, its small values are very attractive since they yield small networks with appropriate computational benefits, with relatively only a little error increase.

13.2.4 Batch Size Impact

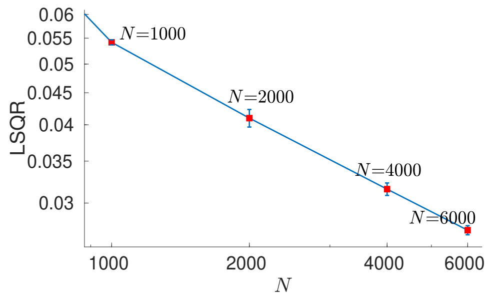
Herein we investigate the relation between PSO approximation error and a batch size of training points. In particular, in PSO loss we have two terms, the *up* term which is a sum over batch points $\{X_i^U\}_{i=1}^{N^U}$ and the *down* term which is a sum over batch points $\{X_i^D\}_{i=1}^{N^D}$. We will see how values of N^U and N^D affect PSO performance.

Increasing both N^U and N^D First, we run a scenario where both batch sizes are the same, $N^U = N^D = N$. We infer *Columns* distribution with different values of N, ranging from 10 to 6000. In Figures 13.14a-13.14b we can observe that $LSQR$ error is decreasing for bigger N, which is expected since then the stochastic forces at each point $X \in \mathbb{R}^n$ are getting closer to the averaged forces $F_\theta^U(X) = \mathbb{P}^U(X) \cdot M^U[X, f_\theta(X)]$ and $F_\theta^D(X) = \mathbb{P}^D(X) \cdot M^D[X, f_\theta(X)]$. In other words, the sampled approximation of PSO gradient in Eq. (3.5) becomes more accurate.

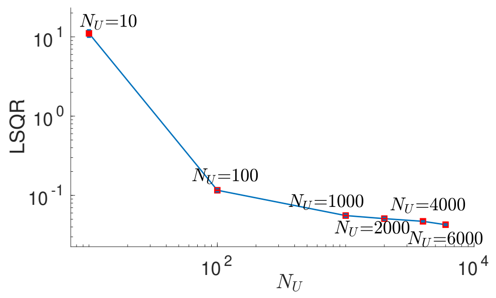
Moreover, we can observe that for the smaller batch size the actual PSO-LDE performance is very poor, with $LSQR$ being around 26.3 for $N = 10$ and decreasing to 0.31 for $N = 100$. The high accuracy, in range 0.03-0.05, is only achieved when we increase the number of batch



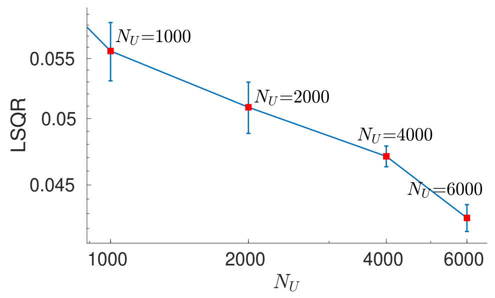
(a)



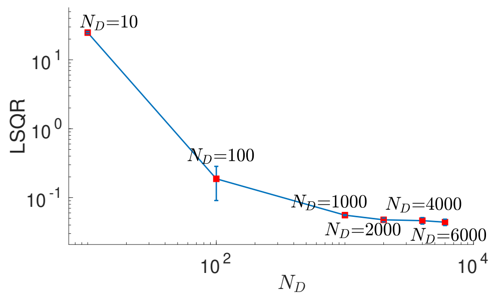
(b)



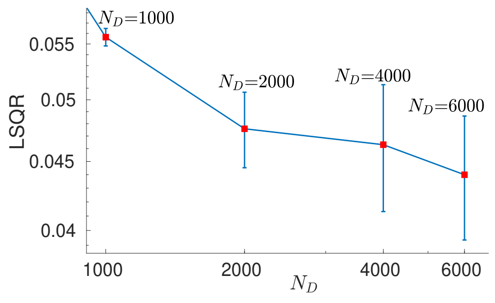
(c)



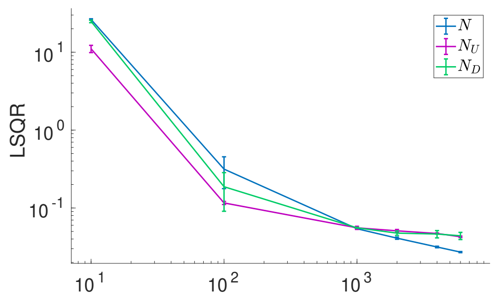
(d)



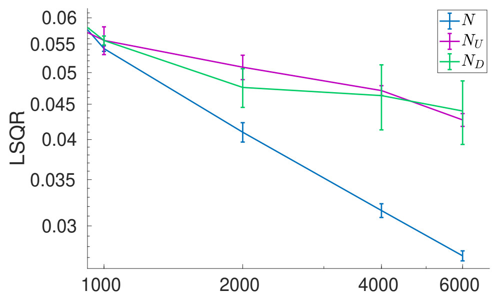
(e)



(f)



(g)



(h)

Figure 13.14: Batch size evaluation. *Columns* distribution is estimated via PSO-LDE with $\alpha = \frac{1}{4}$, where NN architecture is block-diagonal with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). (a)-(b) For different values of a batch size $N = N^U = N^D$ we report $LSQR$ and its empirical standard deviation. Both the *up* batch size N^U and the *down* batch size N^D are kept the same. (c)-(d) The N^U receives different values while the N^D is 1000. (e)-(f) The N^D receives different values while the N^U is 1000. (g)-(h) All scenarios are plotted together in the same graph. Note that both x and y axes are log-scaled. Right column is zoom-in of left column.

points to be above 1000. This implies that to reach a higher accuracy, PSO will require a higher demand over the memory/computation resources. Therefore, the higher available resources, expected from future GPU cards, will lead to a higher PSO accuracy.

Increasing only N^U Further, we experiment with increasing/decreasing only one of the batch sizes while the other stays constant. In Figures 13.14c-13.14d a scenario is depicted where N^U is changing while N^D is 1000. Its error for small values of N^U is smaller than in the previous scenario, with $LSQR$ being around 11 for $N^U = 10$ and decreasing to 0.11 for $N^U = 100$. Comparing with the previous experiment, we can see that even if N^U is small, a high value of N^D (1000) improves the optimization performance.

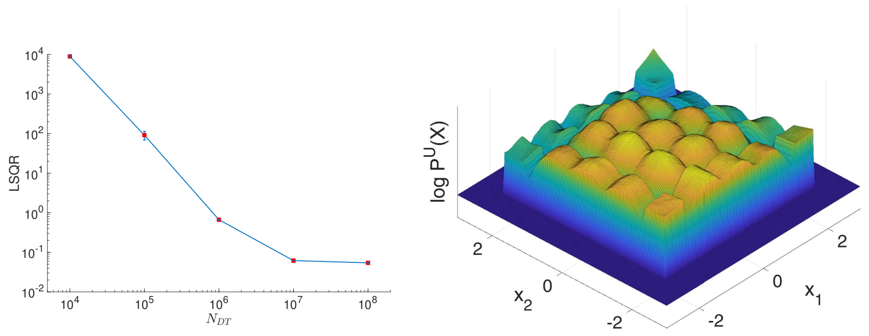
Increasing only N^D Furthermore, in Figures 13.14e-13.14f we depict the opposite scenario where N^D is changing while N^U is 1000. Unlike the experiment in Figures 13.14c-13.14d, here the improvement of error for small values of N^D w.r.t. the first experiment is not that significant. For $N^D = 10$ the error is 24.8 and for $N^D = 100$ it is 0.19. Hence, the bigger number of *up* points ($N^U = 1000$) does not lead to a much higher accuracy if a number of *down* points N^D is stll too small.

Finally, in Figures 13.14g-13.14h we plot all three experiments together. Note that all lines cross at the same point, where all experiments were configured to have $N^U = N^D = 1000$. We can see that in case our resource budget is low (smaller values of x in Figure 13.14g), it is more efficient to spend them to increase N^D. Yet, for a high overall resource budget (higher values of x in Figure 13.14g) both N^U and N^D affect the error similarly, and it is better to keep them equal and increase them as much as possible.

13.2.5 Small Training Dataset

In this section we will learn 20D *Columns* distribution using only 100000 training sample points. As we will see below, the density inference task via non-parametric PSO becomes much more challenging when the size of the training dataset is limited.

Various Sizes of Dataset To infer the data pdf, here we applied PSO-LDE with $\alpha = \frac{1}{4}$. Further, we use BD NN architecture since it has superior approximation performance over FC architecture. First, we perform the inference task using the same BD network as in Section 13.2.1, with 6 layers, $N_B = 50$ and $S_B = 64$, where the size of the entire weights vector is $|\theta| = 902401$. The *Columns* pdf is inferred using a various number N_{DT} of **overall** training points $\{X_i^U\}_{i=1}^{N_{DT}}$. As can be observed from Figure 13.15a, $LSQR$ error increases for a smaller size N_{DT} of the training dataset. From error 0.05 for $N_{DT} = 10^8$ it gets to 0.062 for $N_{DT} = 10^7$, 0.67 for $N_{DT} = 10^6$, 91.2 for $N_{DT} = 10^5$ and 8907 for $N_{DT} = 10^4$. As we will see below and as was already discussed in Section 12, one of the main reasons for such large errors is a too flexible NN model, which in a small dataset setting can significantly damage the performance of PSO-LDE, and PSO in general.



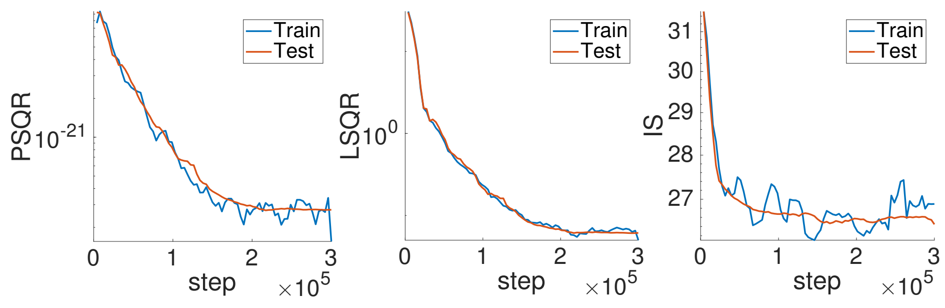
(a) (b)

Figure 13.15: Evaluation of PSO-LDE for estimation of *Columns* distribution with different sizes N_{DT} of the training dataset. Used NN architecture is block-diagonal with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). Number of weights is $|\theta| = 902401$. Applied PSO instance is PSO-LDE with $\alpha = \frac{1}{4}$. (a) $LSQR$ error as a function of the training dataset size N_{DT}, where both x and y axes are log-scaled. (b) Illustration of the learned log-pdf surface $f_\theta(X)$ for the dataset size $N_{DT} = 10^5$. The depicted slice is $\log \mathbb{P}(x_1, x_2) \equiv f_\theta([x_1, x_2, 0, \dots, 0])$, with x_1 and x_2 forming a grid of points in first two dimensions of the density's support. Note the resemblance similarity between this plot and the target surface in Figure 13.1b. Even though $LSQR$ error of this model in (a) is very high (≈ 91.2), the local structure within the converged surface is close to the local structure of the target function. Unfortunately, its global structure is inconsistent and far away from the target.

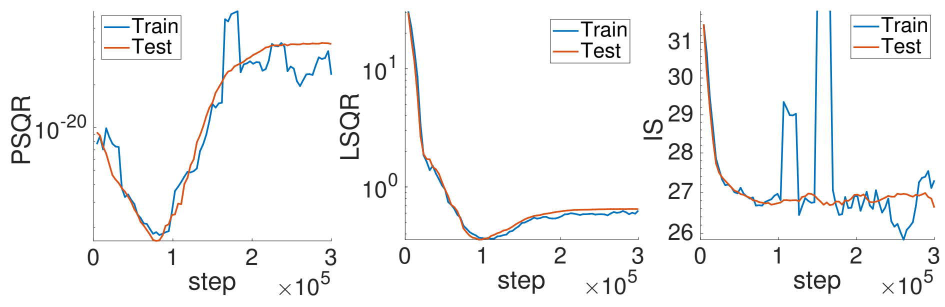
Interestingly, although $LSQR$ error is high for models with $N_{DT} = 10^5$, in Figure 13.15b we can see that the converged surface $f_\theta(X)$ visually highly resembles the real target log-pdf in Figure 13.1b. We can see that main lines and forms of the target surface were learned by $f_\theta(X)$, yet the overall global shape of NN surface is far away from the target. Furthermore, if we calculate $\exp f_\theta(X)$, we will get a surface that is very far from its target $\mathbb{P}^U(X)$ since the exponential will amplify small errors into large.

Further, in Figure 13.16 we can observe error curves for models learned in Figure 13.15a. Both train and test errors are reported for all three error types, per a different dataset size N_{DT}. Train and test errors are very similar for big N_{DT} in Figures 13.16a-13.16b. Moreover, in Figure 13.16b we can see according to the $LSQR$ error (the middle column) that at the beginning error decreases but after 10^5 steps it starts increasing, which suggests a possible *overfitting* to the training dataset at $N_{DT} = 10^6$. Further, in Figures 13.16c-13.16d we can see that train and test errors are more distinct from each other. Likewise, here $PSQR$ and $LSQR$, both train and test, are increasing almost from the start of the optimization. In contrast, in Figures 13.16c-13.16d in case of IS the train and test errors have different trends compared with each other. While the test IS is increasing, the train IS is decreasing. This is a typical behavior of the optimization error that indicates strong *overfitting* of the model, here for $N_{DT} = 10^5$ and $N_{DT} = 10^4$. In turn, this means that we apply a too rich model family - over-flexible NN which can be pushed to form peaks around the training points, as was demonstrated in Section 12. Further, we can detect such *overfitting* by comparing train and test IS errors, which do not depend on ground truth.

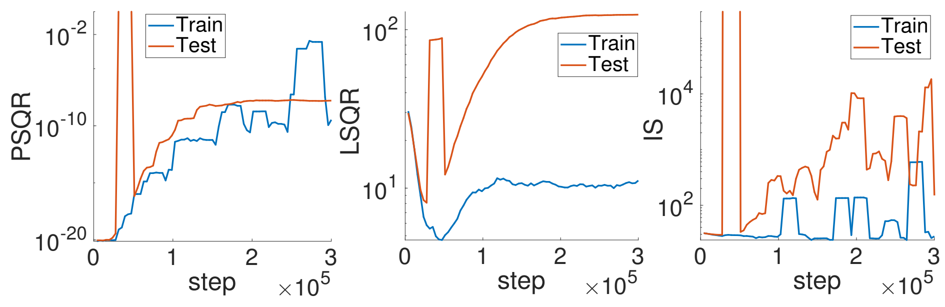
Note that train and test errors of $PSQR$/$LSQR$ have the same trend herein, unlike IS. The reason for this is that $PSQR$ and $LSQR$ express a real ground truth distance between NN surface and the target function, whereas IS error is only a some rough estimation of it.



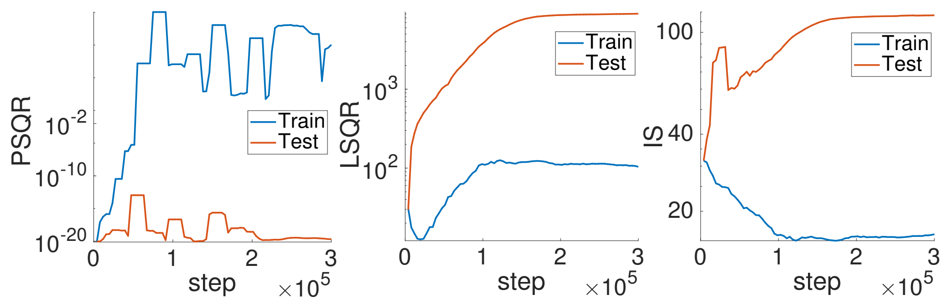
(a)



(b)

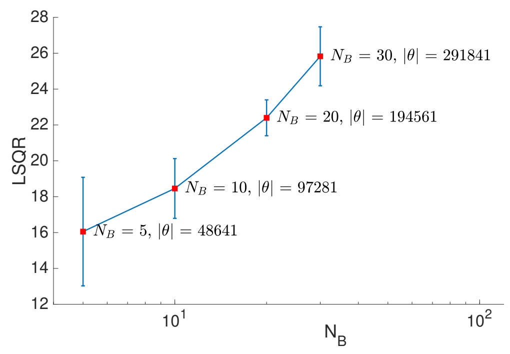


(c)

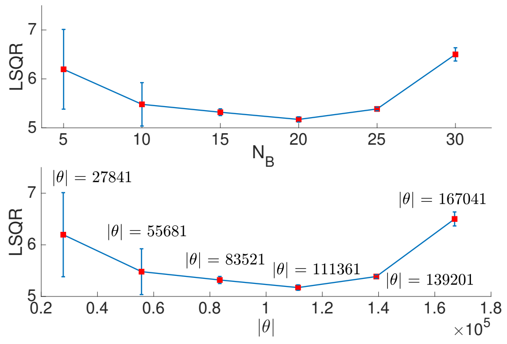


(d)

Figure 13.16: Error curves during the optimization for models learned in Figure 13.15a, where various dataset sizes N_{DT} were evaluated. The error is reported for (a) $N_{DT} = 10^7$, (b) $N_{DT} = 10^6$, (c) $N_{DT} = 10^5$ and (d) $N_{DT} = 10^4$. In each row we report: $PSQR$ - first column; $LSQR$ - middle column; IS - last column. Each plot contains both train and test errors; the former is evaluated over 10^3 *up* points from training dataset chosen as a batch for a specific optimization iteration, whereas the latter - over 10^5 *up* points from the testing dataset.



(a)



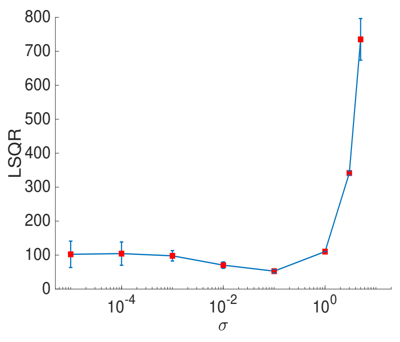
(b)

Figure 13.17: Evaluation of PSO-LDE for estimation of *Columns* distribution with only $N_{DT} = 10^5$ training samples. Used NN architecture is block-diagonal with block size $S_B = 64$ (see Section 11.1). Applied PSO instance is PSO-LDE with $\alpha = \frac{1}{4}$. (a) $LSQR$ error for models with 4 layers and various values of the blocks number N_B. Size of weights vector θ is depicted for each model. (b) $LSQR$ error for models with 3 layers and various values of blocks number N_B. Top - $LSQR$ is shown as a function of N_B; bottom - $LSQR$ is shown as a function of the size $|\theta|$. As observed, too big and too small number of parameters, $|\theta|$, produces a less accurate pdf approximation in the small dataset setting.

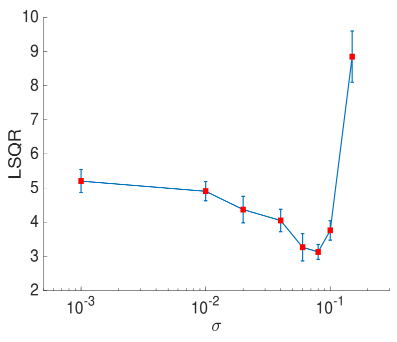
Reduction of NN Flexibility to Tackle Overfitting Next, we perform the inference task using BD network with only 4 layers, on the training dataset of size $N_{DT} = 10^5$. We learn models for various numbers of blocks N_B inside our network, being between 5 and 30. In Figure 13.17a we can see that $LSQR$ error is still very high compared to the results of an infinite dataset setting in Section 13.2.1: ≈ 20 vs ≈ 0.05. Yet, it is smaller than in Figure 13.15a, where we used 6 layers instead of 4 and $N_B = 50$. Moreover, we can see in Figure 13.17a that the error is reducing with a smaller number of blocks N_B and a smaller number of NN parameters $|\theta|$.

Further, we perform the same experiment where BD architecture has only 3 layers (first and last are FC layers and in the middle there is BD layer). In Figure 13.17b we can observe that for a too small/large value of N_B the $LSQR$ is higher. That is, for a too big/small number of weights in θ we have a worse pdf approximation. This can be explained as follows. Small size $|\theta|$ implies NN with low flexibility which is not enough to closely approximate a target surface $\log \mathbb{P}^U(X)$, thus producing *underfitting*. We observed similar results also in an infinite dataset setting in Section 13.2.3, where bigger size of θ yielded an even smaller error. Furthermore, when $|\theta|$ is too big, the NN surface becomes too flexible and causes *overfitting*. Such over-flexibility is not appropriate for small dataset setting, since it allows to closely approximate a peak around each training sample X^U (or X^D), where the produced spiky surface $f_\theta(X)$ will obviously have a high approximation error, as was demonstrated in Section 12. In other words, in contrast to common regression learning, in case of *unsupervised* PSO approaches the size (and the flexibility) of NN should be adjusted according to the number of available training points, otherwise the produced approximation error will be enormous. In contrast, in common DL-based regression methods such over-flexible NN may cause *overfitting* to training samples and increase the testing error, yet it will not affect the overall approximation performance as destructively as in PSO.

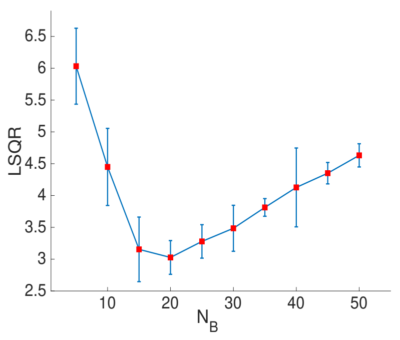
Interestingly, the optimal size $|\theta|$ in Figure 13.17b-bottom is around 100000 - the number of available training points. It would be an important investigatory direction to find the exact



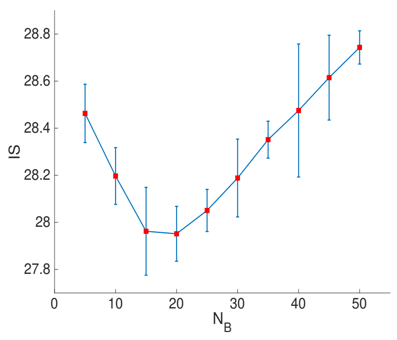
(a)



(b)



(c)



(d)

Figure 13.18: Evaluation of PSO-LDE for estimation of *Columns* distribution with only $N_{DT} = 10^5$ training samples, using a data augmentation noise. The used NN architecture is block-diagonal with block size $S_B = 64$ (see Section 11.1). Applied PSO instance is PSO-LDE with $\alpha = \frac{1}{4}$. Each *up* training point X^U is sampled from the data density $\mathbb{P}^U$. Further, an additive 20D noise $\upsilon \sim \mathcal{N}(0, \sigma^2 \cdot I)$ is sampled and added to X^U. The PSO-LDE loss is applied on $\bar{X}^U = X^U + \upsilon$ instead of the original X^U. Number of overall $\mathbb{P}^U$'s samples is constant 10^5; new samples of noise υ are sampled at each optimization iteration. We learn models using various values of σ. (a) $LSQR$ error as a function of σ, for models with 6 layers and the blocks number $N_B = 50$. (b) $LSQR$ error as a function of σ, for models with 3 layers and the blocks number $N_B = 20$. (c) $LSQR$ error and (d) IS error as a function of the blocks number N_B, for models with 3 layers and $\sigma = 0.08$. As can be seen, smaller size (up to some threshold) of NN produces much higher accuracy in a limited training dataset setting. Also, the additive noise can yield an accuracy improvement.

mathematical relation between the dataset size and properties of NN (e.g. its size, architecture and *gradient similarity*) for the optimal inference. We shall leave it for future research.

Data Augmentation to Tackle Overfitting Additionally, we also apply the augmentation data technique to smooth the converged surface $f_\theta(X)$, as was described in Section 12. Concretely, we use samples of r.v. $\bar{X}^U = X^U + \upsilon$ as our *up* points, where we push the NN surface *up*. The X^U is sampled from the data density $\mathbb{P}^U$, while the additive 20D noise υ is sampled from $\sim \mathcal{N}(0, \sigma^2 \cdot I)$. At each optimization iteration the next batch of $\{X_i^U\}_{i=1}^{N^U}$ is fetched from a priori prepared training dataset of size 10^5, and new noise instances $\{\upsilon^i\}_{i=1}^{N^U}$ are generated. Further, $\{\bar{X}_i^U\}_{i=1}^{N^U}$ is used as the batch of *up* points within PSO-LDE loss, where $\bar{X}_i^U = X_i^U + \upsilon^i$. Such a method allows us to push the $f_\theta(X)$ *up* not only at the limited number of training points X_i^U, but also at other points in some ball neighborhood around each X_i^U, thus implicitly changing the approximated function to be smoother and less spiky. Another perspective to look over it is that we apply Gaussian diffusion over our NN surface, since adding Gaussian noise is identical to replacing the target pdf $\mathbb{P}^U(X)$ with the convolution $\mathbb{P}^U(X) * \mathcal{N}(0, \sigma^2 \cdot I)$.

In Figure 13.18a we can see results of such data augmentation for BD network with 6 layers and 50 blocks, with $S_B = 64$. In such case the used model is over-flexible with too many degrees of freedom, and the data augmentation is not helpful, with an overall error being similar to the one obtained in Figure 13.15a. Yet, when the model size (and its flexibility) is reduced to only 3 layers and 20 blocks, the performance trend becomes different. In Figure 13.18b we can see that for particular values of the noise s.t.d. σ (e.g. 0.08) the data augmentation technique reduces $LSQR$ error from 5.17 (see Figure 13.17b) to only 3.13.

Further, in Figures 13.18c-13.18d we can see again that there is an optimal NN size/flexibility that produces the best performance, where a smaller NN suffers from *underfitting* and a larger NN suffers from *overfitting*. Moreover, in Figure 13.18d we can see again that the empirical error IS is correlated with the ground truth error $LSQR$, although the former is less accurate than the latter. Hence, IS can be used in practice to select the best learned model.

Overall, in our experiments we observed both *underfitting* and *overfitting* cases of PSO optimization. The first typically happens for large dataset size when NN is not flexible enough to represent all the information contained within training samples. In contrast, the second typically happens for small datasets when NN is over-flexible so that it can be pushed to have spikes around the training points. Further, *overfitting* can be detected via IS test error. Finally, the data augmentation reduces effect of *overfitting*.

13.3 PDF Estimation via PSO - *Transformed Columns* Distribution

In this section we will show that PSO is not limited only to isotropic densities (e.g. *Columns* distribution from Section 13.2) where there is no correlation among different data dimensions, and can be actually applied also to data with a complicated correlation structure between various dimensions. Specifically, herein we infer a 20D *Transformed Columns* distribution, $\mathbb{P}^{\mathcal{U}}(X) = \mathbb{P}^{TrClmns}(X)$, which is produced from isotropic *Columns* by multiplying a random variable $X \sim \mathbb{P}^{Clmns}$ (defined in Eq. (13.1)) by a dense invertible matrix A that enforces correlation between different dimensions. Its pdf can be written as:

$$\mathbb{P}^{TrClmns}(X) = \frac{1}{abs\,[\det A]}\mathbb{P}^{Clmns}(A^{-1} \cdot X), \tag{13.9}$$

where A appears in Appendix J. As we will see below, the obtained results for this more sophisticated distribution have similar trends to results of *Columns*. Additionally, we will also show how important is the choice of $\mathbb{P}^{\mathcal{D}}$. Unconcerned reader may skip it to the next Section 13.4.

Uniform $\mathbb{P}^{\mathcal{D}}$ First, we evaluate PSO-LDE for different values of α on the density inference task. The applied model is BD with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$. The dataset size is $N_{DT} = 10^8$ and $\mathbb{P}^{\mathcal{D}}$ is Uniform. In Figures 13.19a-13.19b-13.19c we can see the corresponding errors for learned models. The errors are huge, implying that the inference task failed. The reason for this is the *relative support mismatch* between Uniform and

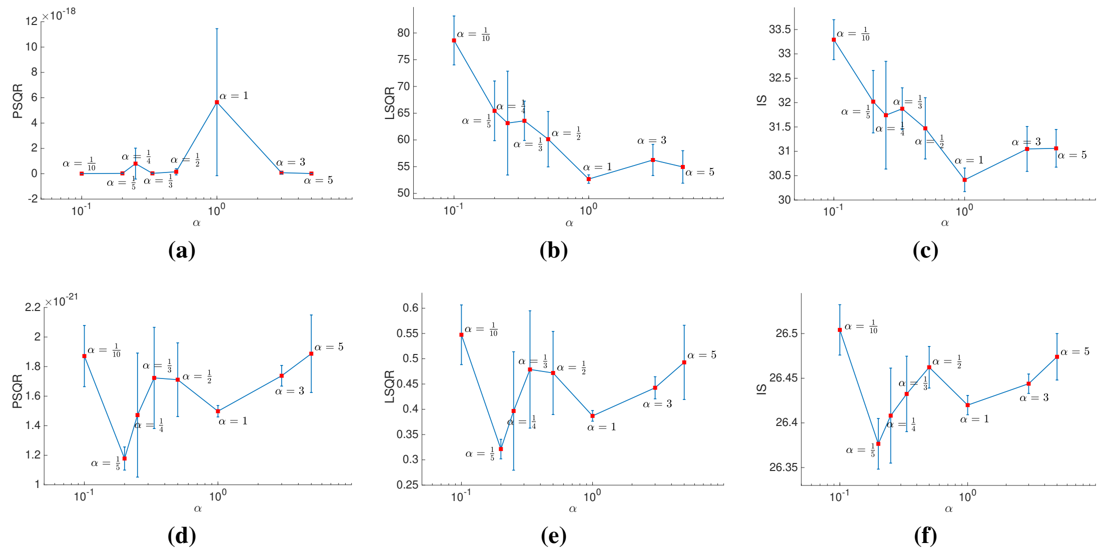


Figure 13.19: Evaluation of PSO-LDE for estimation of *Transformed Columns* distribution, where NN architecture is BD with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). The *down* density $\mathbb{P}^D$ is Uniform in the top row (a)-(b)-(c), and Gaussian in the bottom row (d)-(e)-(f). For different values of a hyper-parameter α, (a)-(d) $PSQR$, (b)-(e) $LSQR$ and (c)-(f) IS are reported, along with their empirical standard deviation. As observed, when $\mathbb{P}^D$ is Uniform, PSO-LDE fails to learn the data density due to large *relative support mismatch*. On the other hand, when $\mathbb{P}^D$ is Gaussian, the target surface is accurately approximated.

Transformed Columns densities. After transformation by matrix A the samples from $\mathbb{P}^{TrClmns}$ are more widely spread out within the space $\mathbb{R}^{20}$. The range of samples along each dimension is now around $[-10, 10]$ instead of the corresponding range $[-2.3, 2.3]$ in *Columns* distribution. Yet, the samples are mostly located in a small subspace of *hyperrectangle* $\mathcal{R} = [-10, 10]^{20}$. When we choose $\mathbb{P}^D$ to be Uniform with $\mathcal{R}$ as its support, most of the samples X^U and X^D are located in different areas of this huge *hyperrectangle* and cannot balance each other to reach PSO equilibrium. Such *relative support mismatch* prevents proper learning of the data density function.

Gaussian $\mathbb{P}^D$ Next, instead of Uniform we used Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ as our *down* density $\mathbb{P}^D$. The mean vector μ is equal to the mean of samples from $\mathbb{P}^U$; the Σ is a diagonal matrix whose non-zero values are empirical variances for each dimension of available *up* samples. In Figures 13.19d-13.19e-13.19f we can observe that overall achieved accuracy is high, yet it is worse than the results for *Columns* distribution. Such difference can be again explained by a mismatch between $\mathbb{P}^U$ and $\mathbb{P}^D$ densities. While *Columns* and Uniform densities in Section 13.2 are relatively aligned to each other, the *Transformed Columns* and Gaussian distributions have a bounded ratio $\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$ only around their mean point μ. In far away regions such ratio becomes too big/small, causing PSO inaccuracies. Hence, we argue that a better choice of $\mathbb{P}^D$ would further improve the accuracy.

Additionally, we can see that in case of *Transformed Columns* PSO-LDE with $\alpha = \frac{1}{5}$ achieves the smallest error, with its $LSQR$ being 0.32 ± 0.02. Additionally, we evaluated the Importance Sampling (IS) method from Table 5.1 and PSO-MAX defined in Eq. (13.4)-(13.5).

Method	$PSQR$	$LSQR$	IS
PSO-LDE, averaged over all α	$1.6 \cdot 10^{-21} \pm 3.17 \cdot 10^{-22}$	0.442 ± 0.095	26.44 ± 0.05
IS	$9.3 \cdot 10^{-21} \pm 3.4 \cdot 10^{-24}$	26.63 ± 0.86	31.3 ± 0.04
PSO-MAX	$2.1 \cdot 10^{-21} \pm 2.96 \cdot 10^{-22}$	0.54 ± 0.088	26.52 ± 0.03

Table 13.3: Performance comparison between various PSO instances for *Transformed Columns* density

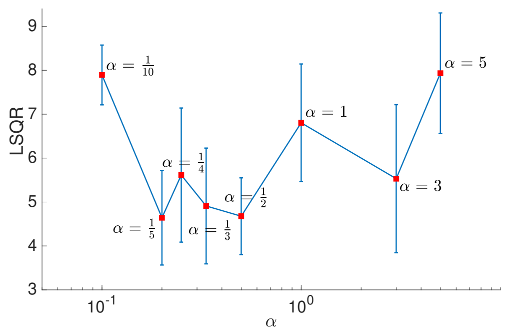


Figure 13.20: Evaluation of PSO-LDE for estimation of *Transformed Columns* distribution, where NN architecture is FC with 4 layers of size 1024 (see Section 11.1). For different values of a hyper-parameter α, $LSQR$ error is reported, along with its empirical standard deviation. Again, the small values of α (around $\frac{1}{5}$) have a lower error.

In Table 13.3 we see that the performance of PSO-MAX is slightly worse than of PSO-LDE, similarly to what was observed for *Columns*. Moreover, the IS fails entirely, producing a very large error. Furthermore, in order to stabilize its learning process we were required to reduce the learning rate from 0.0035 to 0.0001. Hence, here we can see again the superiority of bounded *magnitude* functions over not bounded.

Various NN Architectures Additionally, we evaluated several different NN architectures for *Transformed Columns* distribution, with $\mathbb{P}^D$ being Gaussian. In Figure 13.20 we report the performance for FC networks. As observed, the FC architecture has a higher error w.r.t. BD architecture in Figure 13.19e. Moreover, PSO-LDE with α around $\frac{1}{5}$ performs better.

Further, in Figure 13.21 we experiment with BD architecture for different values of the

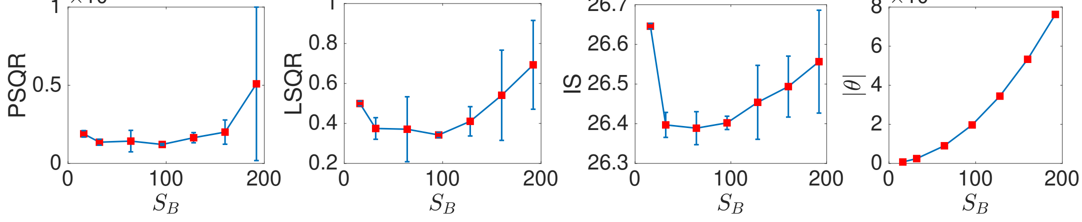


Figure 13.21: Evaluation of PSO-LDE for estimation of *Transformed Columns* distribution, where NN architecture is BD with 6 layers and number of blocks $N_B = 50$ (see Section 11.1). The block size S_B is taking values $\{16, 32, 64, 96, 128, 160, 192\}$. The applied loss is PSO-LDE with $\alpha = \frac{1}{5}$. For different values of S_B we report $PSQR$, $LSQR$ and IS, and their empirical standard deviation. Additionally, in the last column we depict the size of θ for each value of S_B.

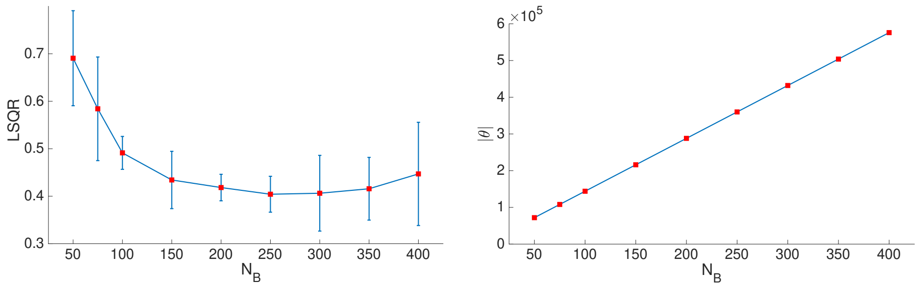


Figure 13.22: Evaluation of PSO-LDE for estimation of *Transformed Columns* distribution, where NN architecture is BD with 6 layers and block size $S_B = 16$ (see Section 11.1). The number of blocks N_B is changing. The applied loss is PSO-LDE with $\alpha = \frac{1}{5}$. For different values of N_B, (a) $LSQR$ error and (b) the number of weights $|\theta|$ are reported.

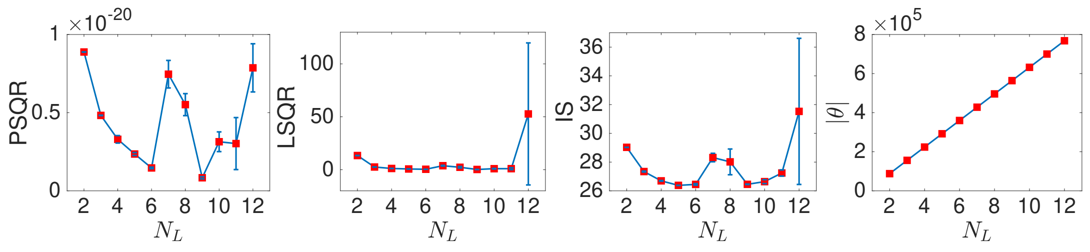
block size S_B. Similarly to what was observed in Section 13.2.3 for *Columns* distribution, also here a too small/large block size has worsen accuracy.

We also perform additional experiments for BD architecture where this time we use small blocks, with $S_B = 16$. In Figure 13.22 we see that for such networks the bigger number of blocks per layer N_B, that is the bigger number of transformation channels, yields better performance. However, we also observe the drop in an accuracy when the number of these channels grows considerably (above 250). Furthermore, no matter how big is N_B, the models with small blocks ($S_B = 16$) in Figure 13.22 produced inferior results w.r.t. model with big blocks ($S_B = 64$, see Figure 13.19e for $\alpha = \frac{1}{5}$).

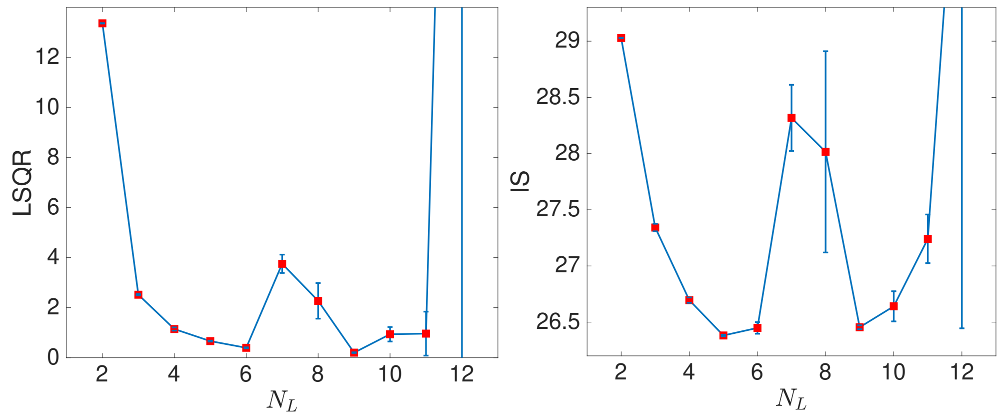
Further, we evaluated the same small-block network for a different number of layers N_L, with $S_B = 16$ and $N_B = 250$. In Figure 13.23 we can see that when N_L grows from 2 to 6, the overall performance is getting better. Yet, for a larger number of layers the performance trend is inconsistent. When N_L is between 7 and 11, some values of N_L are better than other, with no evident improvement pattern for large N_L. Moreover, for $N_L = 12$ the error grows significantly. More so, we empirically observed that 2 out of 5 runs of this setting totally failed due to the zero loss gradient. Thus, the most likely conclusion for this setting is that for a too deep networks the signal from input fails to reach its end, which is a known issue in DL domain. Also, from our experiments it follows that learning still can succeed, depending on the initialization of network weights.

Furthermore, along with the above inconsistency that can occur for too deep networks, for $N_L = 9$ in Figure 13.23 we still received our best results on *Transformed Columns* dataset, with the mean $LSQR$ being 0.204. This leads to the conclusion that even when BD network uses blocks of a small size ($S_B = 16$), it still can produce a superior performance if it is deep enough. Besides, the zero-gradient issue may be tackled by adding shortcut connections between various layers [42].

Overall, in our experiments we saw that when $\mathbb{P}^U$ and $\mathbb{P}^D$ are not properly aligned (i.e. far from each other), PSO fails entirely. Further, PSO-LDE with values of α around $\frac{1}{5}$ was observed to perform better, which is an another superiority evidence for small values of α. Moreover, IS with unbounded *magnitude* functions could not be applied at all for faraway densities. BD



(a)



(b)

Figure 13.23: Evaluation of PSO-LDE for estimation of *Transformed Columns* distribution, where NN architecture is BD with number of blocks $N_B = 250$ and block size $S_B = 16$ (see Section 11.1). The number of layers N_L is changing from 2 to 12. The applied loss is PSO-LDE with $\alpha = \frac{1}{5}$. (a) For different values of N_L we report $PSQR$, $LSQR$ and IS, and their empirical standard deviation. Additionally, in the last column we depict the size of θ for each value of N_L. (b) Zoom of (a).

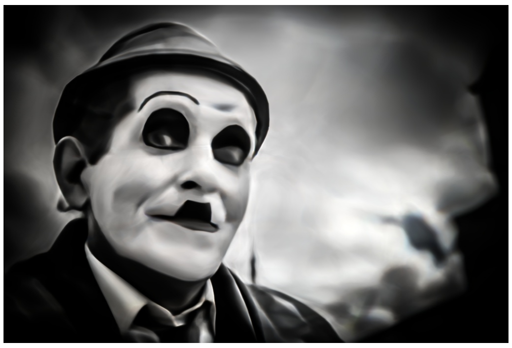
architecture showed again a significantly higher accuracy over FC networks. Block size $S_B = 96$ produced a better inference. Finally, for BD networks with small blocks ($S_B = 16$) the bigger number of blocks N_B and the bigger number of layers N_L improve an accuracy. However, at some point the increase in both can cause the performance drop.

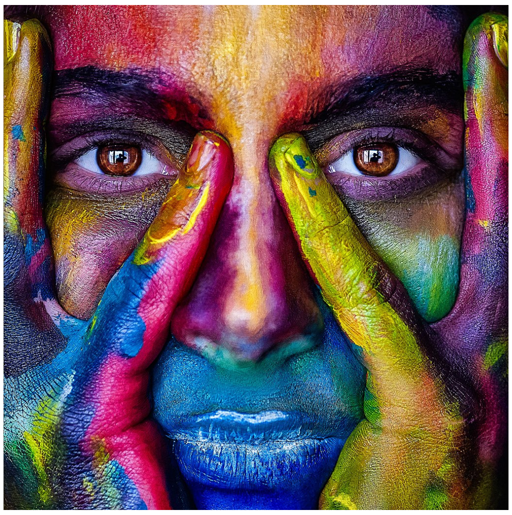
13.4 PDF Estimation via PSO - 3D *Image-based* Densities

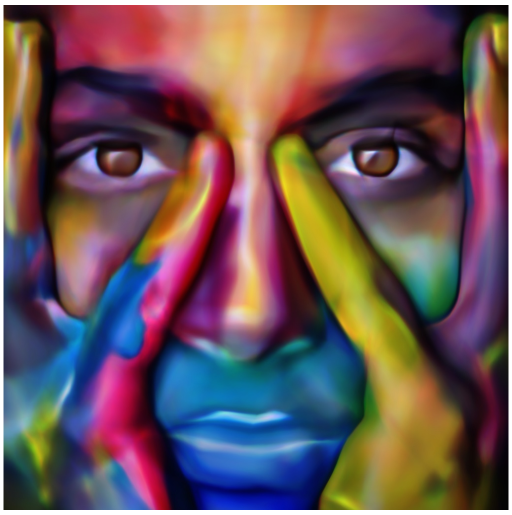
In order to further evaluate the presented herein PSO-LDE density estimation approach, we use intricate 3D densities that are based on image surfaces. More specifically, we consider a given RGB image I as a function $F(x, y, c)$ from $\mathbb{R}^3$ to $\mathbb{R}$ where x, y and c represent width, height and color channel of I respectively. For simplicity we define the range for each input scalar variable from $[x, y, c]$ to be $[0, 1]$, with $F : [0, 1]^3 \rightarrow [0, 1]$. We use grid of values from an image I to appropriately interpolate outputs of the function $F(X)$ at any input point $X \in [0, 1]^3 \subseteq \mathbb{R}^3$. In our experiments we used a linear interpolation.

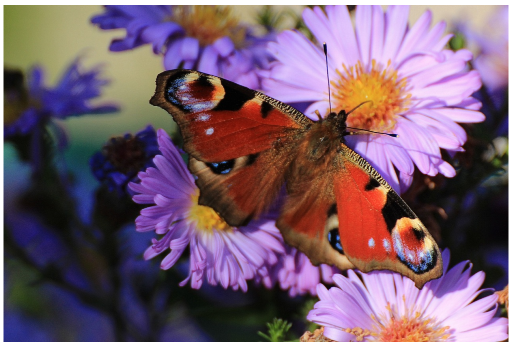
Further, we use F as pdf function which we sample to create a dataset for our PSO-LDE experiments. Yet, the function F, interpolated from image I, is not a valid pdf function since its integral can be any positive number. Thus, we normalize it by its total integral to get a normalized function $\bar{F}$ which we use as an intricate 3D pdf function for further density estimation evaluation. Next, we sample the density $\bar{F}$ via rejection sampling method and gather a dataset of size 10^7

(a)


(b)


(c)


(d)


(e)

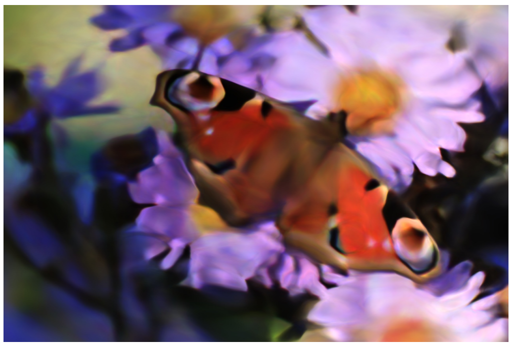

(f)

Figure 13.24: (a),(c),(e) *Image-based* densities and (b),(d),(f) their approximations - Part 1.

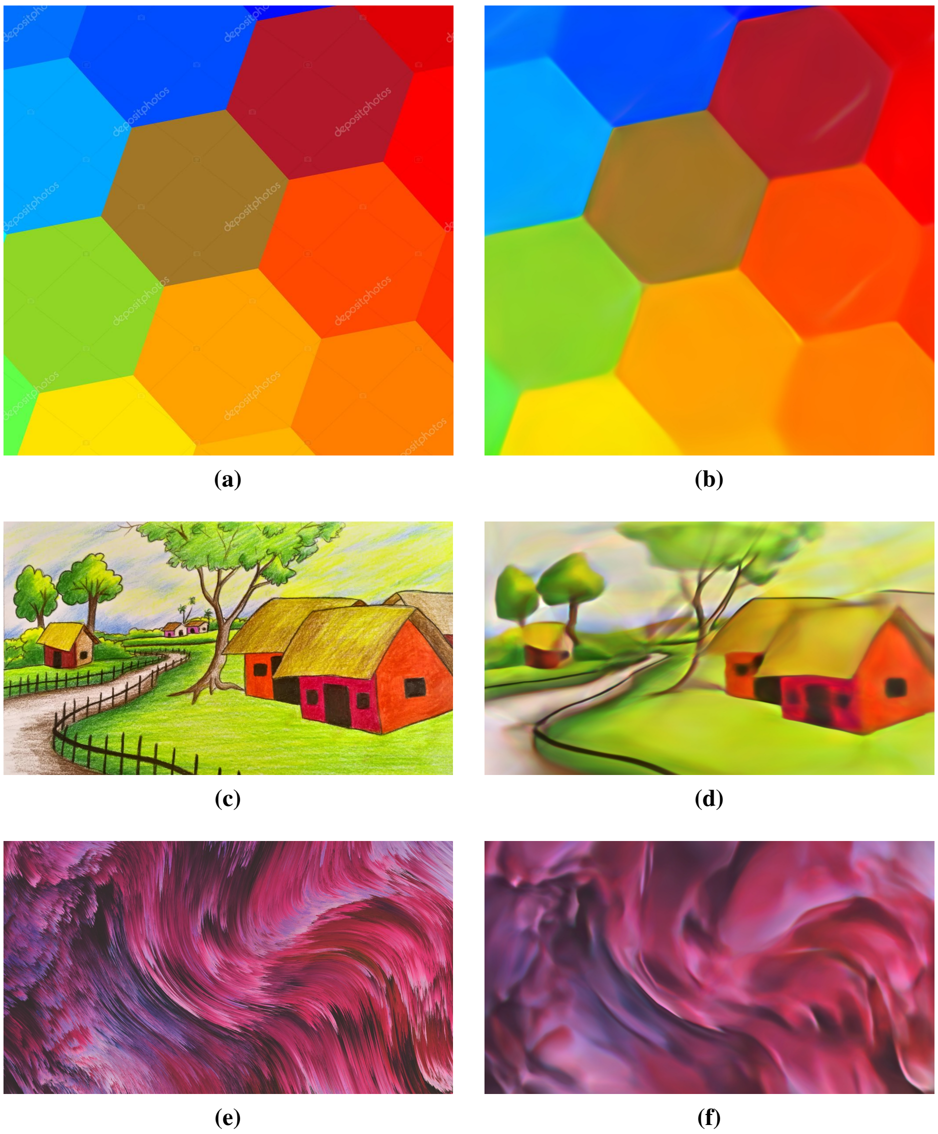


Figure 13.25: (a),(c),(e) *Image-based* densities and (b),(d),(f) their approximations - Part 2.

per each *image-based* density. Furthermore, we approximate the target surface $\log \bar{F}(X)$ via PSO-LDE only from these sampled points.

Note that $\log \bar{F}$ has a very sophisticated structure since used images have a very high contrast and nearby pixels typically have significantly different values. This makes $\log \bar{F}$ a highly non-linear function, which cannot be easily approximated by typical parametric density estimation techniques. Yet, as we will see below, due to the approximation power of DL, which is exploited in full by PSO, and due to a high flexibility of the proposed BD architecture, the non-parametric PSO-LDE allows us to accurately estimate even such complex distributions. Importantly, we emphasize that the evaluated distributions in this section have their support in $\mathbb{R}^3$, and are **not** some very high-dimensional densities over image data that can be often encountered in DL domain.

We use PSO-LDE with $\alpha = \frac{1}{4}$ in order to learn the target $\log \bar{F}$. The applied NN architecture is block-diagonal with a number of blocks $N_B = 15$, a block size $S_B = 64$ and a number of layers $N_L = 14$ (see Section 11.1). In order to tackle the problem of vanishing gradients in such a deep model, we introduced shortcut connections into our network, where each layer has a form $u_i = h_i(u_{i-1}; \theta_i) + u_{i-1}$ with u_i and $h_i(\cdot; \theta_i)$ being respectively the output and the applied transformation function of i-th BD layer within the BD network. Note that in this thesis we use such shortcuts only for the experiments of this section. Furthermore, after training is finished, we convert the inferred $\bar{F}$ back into an image format, producing an *inferred image* I'.

In Figures 13.24 and 13.25 we show *inferred images* for several input images. As can be seen, there is high resemblance between both input I and inferred I'. That is, PSO-LDE succeeded to accurately infer densities even with a very complicated surface. Note that each *image-based* density was inferred by using identical hyper-parameters, that are the same as in the rest of our experiments (except for NN structure where shortcuts were applied). Additional parameter-tuning per a specific input image will most probably improve the produced herein results.

13.5 PDF Estimation via PSO - Joint Over Poses and CNN Features

Here we consider the joint density estimation task in context of robotics domain, over data considered in [62]. Specifically, we learn the joint $\mathbb{P}(Z, X)$ where robot pose $X \in \mathbb{R}^4$ has the form $\{x, y, pitch, yaw\}$ and $Z \in \mathbb{R}^{10}$ is a set of CNN features observed at X. The inferred dataset was generated from Unreal Engine [139] where we sampled uniformly poses $\{X_i\}$, retrieved a camera measurement I_i at each X_i, computed CNN logits f_i for each I_i using Inception-v3 [134], and finally retrieved 10 pre-specified features from f_i to construct Z_i. A Gaussian parametric estimation of this particular $\mathbb{P}(Z, X)$ was reported to have a relatively poor accuracy due to the intricate structure of ground truth distribution [62].

The technical setup for this task is very similar to the one described in Section 13.1, with few differences. Overall size of training and testing datasets was 100000 and 70000 respectively. BD architecture was used to represent $f_\theta(X)$, with $N_L = 8$, $N_B = 100$ and $S_B = 64$, and with Elu non-linearity activation function. Further, here we use negative log-likelihood

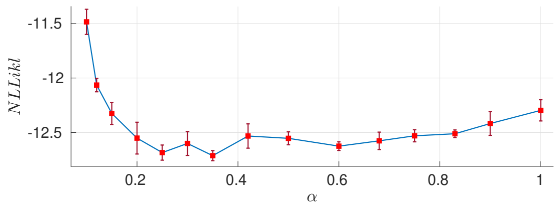


Figure 13.26: Evaluation of PSO-LDE for density estimation of CNN features. For different values of hyper-parameter α, $NLLikl$ is reported, along with the empirical standard deviation.

$NLLikl = -\frac{1}{N}\sum_{i=1}^{N}\log\bar{\mathbb{P}}_{\theta}(Z_i, X_i)$ as our performance metric, since we do not have access to ground truth pdf required to compute $LSQR$; $\{Z_i, X_i\}_{i=1}^{N}$ are $N = 70000$ testing samples.

First we apply the same baselines from Section 13.2.2. The achieved $NLLikl$ by score matching, MADE and MAF is, respectively, -7.056 ± 0.23, -11.55 ± 0.085 and -12.076 ± 0.01. Like previously, the score matching model was explicitly normalized via importance sampling. Furthermore, we inferred the considered model also via Gaussian-Mixture-Model with k components (GMM-k). The achieved $NLLikl$ was -6.74 ± 0.06 and -10.01 ± 0.03 for GMM-50 and GMM-300 respectively.

Further, PSO-LDE with various α was applied. The above learned GMM was chosen as *down* distribution in this scenario. We separate $3 \cdot 10^5$ learning iterations into two stages, 80000 and 220000, where during the first stage GMM-50 was used as $\mathbb{P}^D$, and during the second stage - GMM-300. Likewise, during the first stage we kept the learning rate constant 0.0035, and the applied batch size was $N_U = N_D = 600$. For second stage the learning rate was $1e-5$ and the batch size - $N_U = N_D = 100$. Such a 2-stage process empirically showed better performance.

Importantly, $NLLikl$ measures a statistical discrepancy only when the estimated model is properly normalized. Yet, as was seen above, PSO-LDE models are only approximately normalized. Hence, in order to compute $NLLikl$ for $\bar{\mathbb{P}}_{\theta}(Z, X)$ inferred via PSO-LDE, we can use its normalized form $\hat{\mathbb{P}}_{\theta}(Z, X) = \bar{\mathbb{P}}_{\theta}(Z, X)/TI$, with its total integral TI being calculated/approximated via importance sampling. Yet, such normalization destroys approximation power of PSO-LDE model, since as observed empirically, while $\bar{\mathbb{P}}_{\theta}$ can have a large TI, it is still very close to the ground truth pdf on local scale. A better normalization method, for which PSO-LDE is less sensitive, is to represent the normalized version of $\bar{\mathbb{P}}_{\theta}(Z, X)$ as $\hat{\mathbb{P}}_{\theta,C}(Z, X) = \min\Big[C \cdot \mathbb{P}_{GMM}(Z, X), \bar{\mathbb{P}}_{\theta}(Z, X)\Big]$, where $C > 0$ is tuned so that TI of $\hat{\mathbb{P}}_{\theta,C}(Z, X)$ is equal to 1, and where $\mathbb{P}_{GMM}$ is the GMM-300 that was used as *down* density. Such normalization can be considered as sort of a prior over joint $\mathbb{P}(Z, X)$ that requires inferred $\hat{\mathbb{P}}_{\theta,C}(Z, X)$ to be close to $\mathbb{P}_{GMM}$. In our experiments we used this form to compute $NLLikl$ for PSO-LDE methods. Importantly, such normalization is only required for comparison with other baselines, since the comparison of normalized and *unnormalized* models is a very challenging task in its own [78].

In Figure 13.26 we can see $NLLikl$ for various α. Best performance was achieved by $\alpha = 0.35$, with $NLLikl = -12.7 \pm 0.046$, which is significantly better than all state-of-the-art baselines. Furthermore, similarly to Figure 13.2a also here we observe that PSO-LDE with small α (here around $\alpha = 0.35$) has a better accuracy w.r.t. NCE (i.e. $\alpha = 1$) that achieved

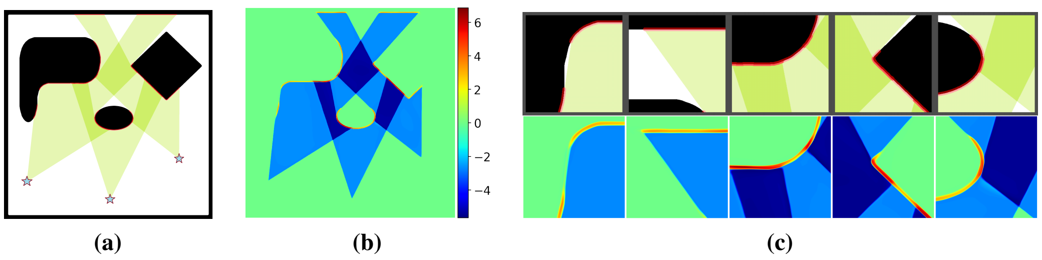


Figure 13.27: Probabilistic occupancy mapping scenario 1. (a) Environment with black obstacles and 3 lidar scans (in green). Laser hit points are in red. (b) NN output $f_\theta(X)$, approximating $J(X)$. (c) Zoomed-in map parts.

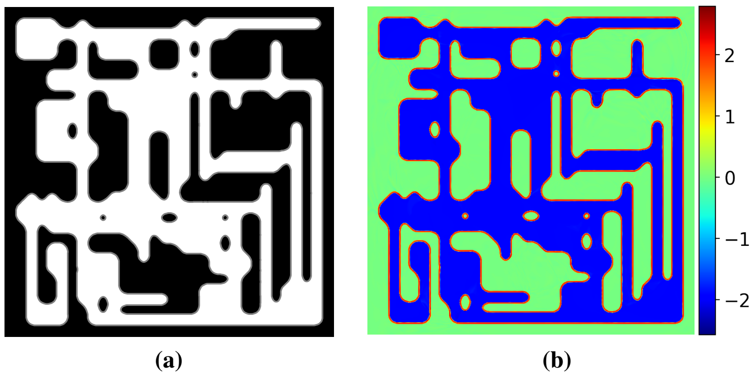


Figure 13.28: Probabilistic occupancy mapping scenario 2. (a) Environment with black obstacles and white free space. (b) NN output $f_\theta(X)$, approximating $J(X)$.

$NLLikl = -12.3 \pm 0.1$.

13.6 Probabilistic Occupancy Mapping

Finally, we deploy PSO to model a statistical occupancy map, according to the setup in Section 10.5. Here, we consider two scenarios. In the first, we have a room with 3 lidar scans, see Figure 13.27a. Points $\{X_i^{\mathbf{f}}\}$ are sampled uniformly from each scan cone (in green), whereas $\{X_i^{\mathbf{o}}\}$ are taken to be laser hits (in red). At the convergence PSO produced $f_\theta(X)$ in Figure 13.27b, which provides a continuous occupancy information about the environment. Specifically, the learned model returns zero for unobserved areas, depicting an information lack there. In areas of scans it returns positive values for occupied areas, and negative values - for unoccupied. Moreover, the magnitude of values is larger for areas that were observed twice, which we can use as a confidence measure of our knowledge when we plan autonomous navigation through the area.

In the second scenario a larger area with more details, taken from [8], is considered, see Figure 13.28. Here we assume that the entire environment was observed, with points $\{X_i^{\mathbf{f}}\}$ being uniformly sampled from the entire white area of Figure 13.28a, and $\{X_i^{\mathbf{o}}\}$ being uniformly sampled on the boundary between white and black regions. In Figure 13.28b we can see that the learned occupancy model produces very accurate information about the real environment.

Overall, in the above simplistic scenarios we showed that PSO can accurately infer a continuous probabilistic occupancy map. In future work we shall compare the proposed method to other alternatives in this domain [96, 111, 121].

The technical setup for the above learning task was very similar to the one described in Section 13.1, with few differences.

Scenario 1 Overall size of training dataset was $N^{\mathbf{o}} = 7347$ and $N^{\mathbf{f}} = 3.5 \cdot 10^6$. Note that "free" locations are sampled uniformly from scanned areas, and we can construct a dataset of these points as large as we want. FC architecture was used to represent $f_\theta(X)$, with 8 layers of size 256 each, and with Elu non-linearity activation function. Optimization was performed for 300000 iterations, and took around 30 minutes to run on a GeForce GTX 1080 Ti GPU card. Size of mini batches was 300.

Scenario 2 Overall size of training dataset was $N^{\mathbf{o}} = 674563$ and $N^{\mathbf{f}} = 4.3 \cdot 10^6$. FC architecture was used to represent $f_\theta(X)$, with 10 layers of size 256 each, and with Elu non-linearity activation function. Optimization was performed for 300000 iterations, and took around one hour to run on a GeForce GTX 1080 Ti GPU card. Size of mini batches was 1000.

CHAPTER 14

Neural Spectrum Alignment

Understanding expressiveness and generalization of deep models is essential for robust performance of NNs and PSO methods. Recently, the optimization analysis for a general NN architecture was related to the model kernel $g_\theta(X, X') \triangleq \nabla_\theta f_\theta(X)^T \cdot \nabla_\theta f_\theta(X')$ [54], which we also related to PSO performance in sections 3 and 7. Properties of this *gradient similarity* kernel, a.k.a. NTK, govern NN expressivity level, generalization and convergence rate. Under various considered conditions [54, 71], this NN kernel converges to its steady state and is invariant along the entire optimization, which significantly facilitates the analyses of DL theory [7, 11, 15, 41, 54, 71].

Yet, in a typical realistic setting the *gradient similarity* kernel is far from being constant, as we empirically demonstrate in this chapter. Moreover, its spectrum undergoes a very specific change during training, aligning itself towards the target function that is learned by NN. This kernel adaptation in its turn improves the optimization convergence rate, by decreasing a norm of the target function within the corresponding RKHS [7]. Furthermore, these *gradient similarity* dynamics can also explain the expressive superiority of deep NNs over more shallow models. Hence, we argue that understanding the *gradient similarity* of NNs beyond its time-invariant regime is a must for full comprehension of NN expressiveness power.

To encourage the onward theoretical research of the kernel, herein we report several strong empirical phenomena and trends of its dynamics. To the best of our knowledge, these trends neither were yet reported nor they can be explained by DL theory developed so far. We argue that accounting for the presented below phenomena can lead to a more complete learning theory of complex hierarchical models like modern NNs.

To this end, in this chapter we perform an empirical investigation of FC NN, its *gradient similarity* kernel and the corresponding Gramian at training data points during the entire period of a typical learning process. Our main empirical contributions below are:

- We show that Gramian serves as a NN memory, with its *top* eigenvectors changing to align with the learned target function. This improves the optimization performance since the convergence rate along kernel *top* eigenvectors is typically higher.

- During the entire optimization NN output is located inside a sub-space spanned by these *top* eigenvectors, making the eigenvectors to be a basis functions of NN.

- Deeper NNs demonstrate a stronger alignment, which may explain their expressive superiority. In contrast, shallow wide NNs with a similar number of parameters achieve a significantly lower alignment level and a worse optimization performance.

- We show additional trends in kernel dynamics as a consequence of learning rate decay. Specifically, after each decay the information about the target function, that is gathered inside *top* eigenvectors, is spread to a bigger number of *top* eigenvectors. Likewise, kernel eigenvalues grow after each learning rate drop, and an eigenvalue-learning-rate product is kept around the same value for the entire optimization.

- Experiments over various FC architectures and real-world datasets are performed. Likewise, several *supervised* and *unsupervised* learning algorithms and number of popular optimizers were evaluated. All experiments showed the mentioned above spectrum alignment.

This chapter is structured as follows. In Section 14.1 we define necessary notations for first-order NN dynamics. In Section 14.2 we relate *gradient similarity* with Fisher information matrix (FIM) of NN and in Section 14.3 we provide more insight about NN dynamics on L2 loss example. In Section 14.4 the work related to NTK and kernel alignment is described, and in Section 14.5 we present our main empirical study. Later, the experiment outcome is summarized in Section 14.6. Further, additional derivations placed in the Appendix. Finally, more visual illustrations of NN spectrum and additional experiments can be found in [63].

14.1 Notations for Alignment Experiment

Consider a NN $f_\theta(X) : \mathbb{R}^n \to \mathbb{R}$ with a parameter vector θ, a typical sample loss ℓ and an empirical loss L, training samples $D = \left[\boldsymbol{\mathcal{X}} = \{X^i \in \mathbb{R}^n\}_{i=1}^N, \boldsymbol{\mathcal{Y}} = \{Y^i \in \mathbb{R}\}_{i=1}^N\right]$ and the loss gradient $\nabla_\theta L$:

$$L(\theta, D) = \frac{1}{N}\sum_{i=1}^{N} \ell\left[X^i, Y^i, f_\theta(X^i)\right], \quad \nabla_\theta L(\theta, D) = \frac{1}{N}\sum_{i=1}^{N} \ell'\left[X^i, Y^i, f_\theta(X^i)\right] \cdot \nabla_\theta f_\theta(X^i). \tag{14.1}$$

The above formulation can be extended to include *unsupervised* PSO methods by eliminating labels $\boldsymbol{\mathcal{Y}}$ from the equations, and introducing *down* samples $\{X_i^D\}_{i=1}^{N^D}$ instead. Further, techniques with a model $f_\theta(X)$ returning multidimensional outputs are out of scope for this thesis, to simplify the formulation.

Consider a GD optimization with learning rate δ, where parameters change at each discrete optimization time t as $d\theta_t \triangleq \theta_{t+1} - \theta_t = -\delta \cdot \nabla_\theta L(\theta_t, D)$. Further, a model output change at any X according to first-order Taylor approximation is:

$$df_{\theta_t}(X) \triangleq f_{\theta_{t+1}}(X) - f_{\theta_t}(X) = d\theta_t^T \cdot \int_0^1 \nabla_\theta f_{\theta_s}(X) ds \approx$$
$$\approx \nabla_\theta f_{\theta_t}(X)^T \cdot d\theta_t = -\frac{\delta}{N} \sum_{i=1}^{N} g_t(X, X^i) \cdot \ell' \left[X^i, Y^i, f_{\theta_t}(X^i) \right], \quad (14.2)$$

where $\theta_s \triangleq (1-s)\theta_t + s\theta_{t+1}$ and $\int_0^1 \nabla_\theta f_{\theta_s}(X) ds$ is a gradient averaged over the straight line between θ_t and θ_{t+1}. Further, $g_t(X, X') \triangleq \nabla_\theta f_{\theta_t}(X)^T \cdot \nabla_\theta f_{\theta_t}(X')$ is a *gradient similarity* - the dot-product of gradients at two different input points also known as NTK [54], and where $\ell' \left[X^i, Y^i, f_{\theta_t}(X^i) \right] \triangleq \nabla_{f_\theta} \ell \left[X^i, Y^i, f_{\theta_t}(X^i) \right]$.

In this section we mainly focus on optimization dynamics of f_θ at training points. To this end, define a vector $\bar{f}_t \in \mathbb{R}^N$ with i-th entry being $f_{\theta_t}(X^i)$. According to Eq. (14.2) the discrete-time evolution of f_θ at testing and training points follows:

$$df_{\theta_t}(X) \approx -\frac{\delta}{N} \cdot g_t(X, \boldsymbol{\mathcal{X}}) \cdot \bar{m}_t, \quad d\bar{f}_t \triangleq \bar{f}_{t+1} - \bar{f}_t \approx -\frac{\delta}{N} \cdot G_t \cdot \bar{m}_t, \quad (14.3)$$

where $G_t \triangleq g_t(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}})$ is a $N \times N$ Gramian with entries $G_t(i,j) = g_t(X^i, X^j)$ and $\bar{m}_t \in \mathbb{R}^N$ is a vector with the i-th entry being $\ell' \left[X^i, Y^i, f_{\theta_t}(X^i) \right]$. Likewise, denote eigenvalues of G_t, sorted in decreasing order, by $\{\lambda_i^t\}_{i=1}^N$, with $\lambda_{max}^t \triangleq \lambda_1^t$ and $\lambda_{min}^t \triangleq \lambda_N^t$. Further, notate the associated orthonormal eigenvectors by $\{\bar{\upsilon}_i^t\}_{i=1}^N$. Note that $\{\lambda_i^t\}_{i=1}^N$ and $\{\bar{\upsilon}_i^t\}_{i=1}^N$ also represent estimations of eigenvalues and eigenfunctions of the kernel $g_t(X, X')$ (see Appendix K for more details). Below we will refer to large and small eigenvalues and their associated eigenvectors by *top* and *bottom* terms respectively.

Eq. (14.3) describes the first-order dynamics of GD learning, where $\bar{m}_t$ is a functional derivative of any considered loss L, and the global optimization convergence is typically associated with it becoming a zero vector, due to Euler-Lagrange equation of L. Further, G_t translates a movement in θ-space into a movement in a space of functions defined on $\boldsymbol{\mathcal{X}}$.

14.2 Relation to Fisher Information Matrix

NN Gramian can be written as $G_t = A_t^T A_t$ where A_t is $|\theta| \times N$ Jacobian matrix with i-th column being $\nabla_\theta f_{\theta_t}(X^i)$. Moreover, $F_t = A_t A_t^T$ is known as the empirical FIM of NN[1] [58, 97, 101] that approximates the second moment of model gradients $\frac{1}{N} F_t \approx \mathbb{E}_X \left[\nabla_\theta f_{\theta_t}(X) \nabla_\theta f_{\theta_t}(X)^T \right]$. Since F_t is dual of G_t, both matrices share same non-zero eigenvalues $\{\lambda_i^t \neq 0\}$. Furthermore, for each λ_i^t the respectful eigenvector $\bar{\omega}_i^t$ of F_t is associated with appropriate $\bar{\upsilon}_i^t$ - they are left and right singular vectors of A_t respectively. Moreover, change of θ_t along the direction $\bar{\omega}_i^t$ causes a change to $\bar{f}_t$ along $\bar{\upsilon}_i^t$ (see Appendix M for the proof). Therefore, spectrums of G_t and F_t describe principal directions in function space and θ-space respectively, according to which

[1]In some papers [117] FIM is also referred to as a Hessian of NN, due to the tight relation between F_t and the Hessian of the loss. See Appendix L for more details

$\bar{f}_t$ and θ_t are changing during the optimization. Based on the above, in Section 14.4 we relate some known properties of F_t towards G_t.

14.3 Analysis of L2 Loss For Constant Gramian

To get more insight into Eq. (14.3), we will consider L2 loss with $\ell\left[X^i, Y^i, f_\theta(X^i)\right] = \frac{1}{2}\left[f_\theta(X^i) - Y^i\right]^2$. In such a case we have $\bar{m}_t = \bar{f}_t - \bar{y}$, with $\bar{y}$ being a vector of labels. Assuming G_t to be fixed along the optimization (see Section 14.4 for justification), NN dynamics can be written as (see the Appendix N for a proper derivation):

$$\bar{f}_t = \bar{f}_0 - \sum_{i=1}^{N}\left[1 - \left[1 - \frac{\delta}{N}\lambda_i\right]^t\right] < \bar{\upsilon}_i, \bar{m}_0 > \bar{\upsilon}_i, \qquad (14.4)$$

$$\bar{m}_t = \sum_{i=1}^{N}\left[1 - \frac{\delta}{N}\lambda_i\right]^t < \bar{\upsilon}_i, \bar{m}_0 > \bar{\upsilon}_i. \qquad (14.5)$$

Further, dynamics of $f_{\theta_t}(X)$ at testing point X appear in the Appendix O since they are not the main focus of this thesis. Under the stability condition $\delta < \frac{2N}{\lambda_{max}}$, the above equations can be viewed as a transmission of a signal from $\bar{m}_0 = \bar{f}_0 - \bar{y}$ into our model $\bar{f}_t$ - at each iteration $\bar{m}_t$ is decreased along each $\{\bar{\upsilon}_i : \lambda_i \neq 0\}$ since $\lim_{t\to\infty}\left[1 - \frac{\delta}{N}\lambda_i\right]^t = 0$. Furthermore, the same information decreased from $\bar{m}_t$ in Eq. (14.5) is appended to $\bar{f}_t$ in Eq. (14.4).

Hence, in case of L2 loss and for a constant Gramian matrix, conceptually GD transmits information packets from the residual $\bar{m}_t$ into our model $\bar{f}_t$ along each axis $\bar{\upsilon}_i$. Further, $s_i^t \triangleq 1 - |1 - \frac{\delta}{N}\lambda_i|$ governs a speed of information flow along $\bar{\upsilon}_i$. Importantly, note that for a high learning rate (i.e. $\delta \approx \frac{2N}{\lambda_{max}}$) the information flow is slow for directions $\bar{\upsilon}_i$ with both very large and very small eigenvalues λ_i, since in former the term $1 - \frac{\delta}{N}\lambda_i$ is close to -1 whereas in latter - to 1. Yet, along with the learning rate decay, performed during a typical optimization, s_i^t for very large λ_i is increased. However, the speed along a direction with small λ_i is further decreasing with the decay of δ. As well, in case $\lambda_{min} > 0$, at the convergence $t \to \infty$ we will get from Eq. (14.4)-Eq. (14.5) the global minima convergence: $\bar{f}_\infty = \bar{f}_0 - \bar{m}_0 = \bar{y}$ and $\bar{m}_\infty = \bar{0}$.

Under the above setting, there are two important key observations. First, due to the restriction over δ in practice the information flow along small λ_i can be prohibitively slow in case a conditional number $\frac{\lambda_{max}}{\lambda_{min}}$ is very large. This implies that for a faster convergence it is desirable for NN to have many eigenvalues as close as possible to its λ_{max} since this will increase a number of directions in the function space where information flow is fast. Second, if $\bar{m}_0$ (or $\bar{y}$ if $\bar{f}_0 \approx 0$) is contained entirely within *top* eigenvectors, small eigenvalues will not affect the convergence rate at all. Hence, the higher alignment between $\bar{m}_0$ (or $\bar{y}$) and *top* eigenvectors may dramatically improve overall convergence rate. The above conclusions and their extensions towards the testing loss are proved in formal manner in [7, 99] for two-layer NNs. Further, the generalization is also shown to be dependent on the above alignment. In Section 14.5 we support these conclusions experimentally.

14.4 Work Related to Model Kernel

First-order NN dynamics can be understood by solving the system in Eq. (14.3). However, its solution is highly challenging due to two main reasons - non-linearity of $\bar{m}_t$ w.r.t. $\bar{f}_t$ (except for the L2 loss) and intricate and yet not fully known time-dependence of Gramian G_t. Although *gradient similarity* $g_t(X, X')$ and corresponding G_t achieved a lot of recent attention in DL community [54, 71], their properties are still investigated mostly only for limits under which G_t becomes time-constant. The first work in this direction was done in [54] where $g_t(X, X')$ was proven to converge to Neural Tangent Kernel (NTK) in infinite width limit. Similarly, in [71] G_0 was shown to accurately explain NN dynamics when θ_t is nearby θ_0 during the entire optimization. The considered case of constant Gramian facilitates solution of Eq. (14.3), as demonstrated in Section 14.3, which otherwise remains intractable. Moreover, GD over NN with constant Gramian/kernel is identical to kernel methods where optimization is solved via kernel gradient descent [54], and hence theoretical insights from kernel learning can be extrapolated towards NNs.

Yet, in practical-sized NNs the spectrum of G_t is neither constant nor it is similar to its initialization. Recent several studies explored its adaptive dynamics [23, 145, 148], although most of the work was done for single or two layer NNs. Further, in [25, 49] mathematical expressions for NTK dynamics were developed for a general NN architecture. Likewise, in the Appendix P we derive similar dynamics for the Gramian G_t. Yet, the above derivations produce intricate equations and it is not straightforward to explain the actual behavior of G_t along the optimization, revealed in this thesis. Particularly, in Section 14.5 we empirically demonstrate that *top* spectrum of G_t is dramatically affected by the learning task at hand, aligning itself with the target function. To the best of our knowledge, the presented NN kernel trends were not investigated in such detail before.

Further, many works explore properties of FIM F_t both theoretically and empirically [37, 58, 99, 117]. Specifically, most of these works come to conclusion that in typical NN an absolute majority of FIM eigenvalues are close to zero, with only small part of them being significantly strong. According to Section 14.2 the same is also true about eigenvalues of G_t. Furthermore, in [7, 99] authors showed for networks with a single hidden layer that NN learnability strongly depends on alignment between labels vector $\bar{y}$ and *top* eigenvectors of G_t. Intuitively, it can be explained by fast convergence rate along $\bar{v}_i$ with large λ_i vs impractically slow one along directions with small λ_i, as was shortly described in Section 14.3. Due to most of the eigenvalues being very small, the alignment between $\bar{y}$ and *top* eigenvectors of G_t defines the optimization performance. Moreover, in [99] authors also noted the increased aforementioned alignment comparing NN at start and end of the training. This observation was shortly made for ResNet convolutional NN architecture, and in Section 14.5 we empirically investigate this alignment for FC architecture, in comprehensive manner for various training tasks.

Furthermore, the picture of information flow from Section 14.3 also explains what target functions are more "easy" to learn. The *top* eigenvectors of G_t typically contain low-frequency signal, which was discussed in [7] and proved in [11] for data uniformly distributed on a hypersphere. In its turn, this explains why low-frequency target functions are learned significantly faster as reported in [7, 110, 152]. Combined with early stopping such behavior is used by DL community as a regularization to prevent fitting high-frequency signal affiliated with noise; this can also be considered as an instance of commonly known Landweber iteration algorithm [66]. We support findings of [11] also in our experiments below, additionally revealing that for a general case the eigenvectors/eigenfunctions of the *gradient similarity* are not spherical harmonics considered in [11].

Finally, in context of kernel methods a lot of effort was done to learn the kernels themselves [31, 141, 144, 146]. The standard 2-stage procedure is to first learn the kernel and latter combine it with the original kernel algorithm, where the first stage can involve search for a kernel whose kernel matrix is strongly aligned with the label vector $\bar{y}$ [31, 144], and the second is to solve a data fitting task (e.g. L2 regression problem) over RKHS defined by the new kernel. Such 2-stage adaptive-kernel methods demonstrated an improved accuracy and robustness compared to techniques with pre-defined kernel [31, 141, 146]. In our experiments we show that NNs exhibit a similar alignment of $g_t(X, X')$ during the optimization, and hence can be viewed as an adaptive-kernel method where both kernel learning and data fitting proceed in parallel.

14.5 Experiments

In this section we empirically study Gramian dynamics along the optimization process. Our main goal here is to illustrate the alignment nature of the *gradient similarity* kernel and verify various deductions made in Section 14.3 under a constant-Gramian setting for a real learning case. To do so in detailed and intuitive manner, we focus our experiments on 2D dataset where visualization of kernel eigenfunctions is possible. We perform a simple regression optimization of FC network via GD, where a learning setup[2] is similar to common conventions applied by DL practitioners. **All** empirical conclusions are also validated for high-dimensional real-world data, which can be found in [63].

Setup To provide a better intuition, we specifically consider a regression of the target function $y(X)$ with $X \in [0,1]^2 \subseteq \mathbb{R}^2$ depicted in Figure 14.1a. We approximate this function with Leaky-Relu FC network via L2 loss, using $N = 10000$ training points sampled uniformly from $[0,1]^2$ (see Figure 14.1c). Training dataset is normalized to an empirical mean 0 and a standard deviation 1. NN contains 6 layers with 256 neurons each, with $|\theta| = 264193$, that was initialized via Xavier initialization [29]. Such large NN size was chosen to specifically satisfy an over-parametrized regime $|\theta| \gg N$, typically met in DL community. Further, learning rate δ starts at 0.25 and is decayed twice each 10^5 iterations, with the total optimization duration being $6 \cdot 10^5$. At convergence $f_\theta(X)$ gets very close to its target, see Figure 14.1b. Additionally, in Figure 14.1d we show that first-order dynamics in Eq. (14.3) describe around 90 percent of the change in NN output along the optimization, leaving another 10 for higher-order Taylor terms.

[2]Related code can be accessed via a repository `https://bit.ly/2kGVHhG`

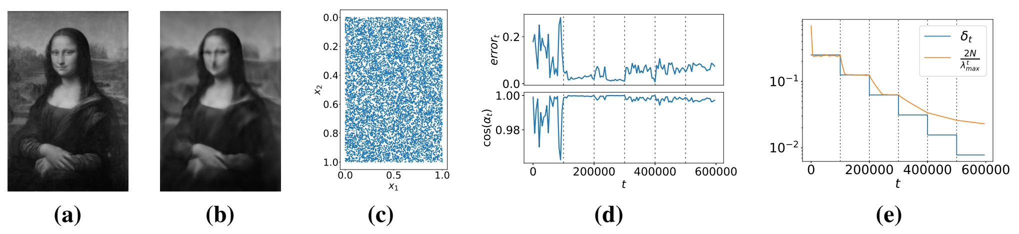


Figure 14.1: (a) Mona Lisa target function for a regression task. (b) NN $f_\theta(X)$ at convergence. (c) 10^4 sampled training points. (d) Accuracy of first order dynamics in Eq. (14.3). Depicted is $error_t = \frac{\|d\bar{f}_t - d\tilde{f}_t\|}{\|d\bar{f}_t\|}$, where $d\tilde{f}_t = -\frac{\delta_t}{N} \cdot G_t \cdot \bar{m}_t$ is the first-order approximation of a real differential $d\bar{f}_t \triangleq \bar{f}_{t+1} - \bar{f}_t$; $\cos(\alpha_t)$ is cosine of an angle between $d\tilde{f}_t$ and $d\bar{f}_t$. As observed, Eq. (14.3) explains roughly 90% of NN change. (e) Learning rate δ_t and its upper stability boundary $\frac{2N}{\lambda^t_{max}}$ along the optimization. We empirically observe a relation $\lambda^t_{max} \propto \frac{1}{\delta_t}$.

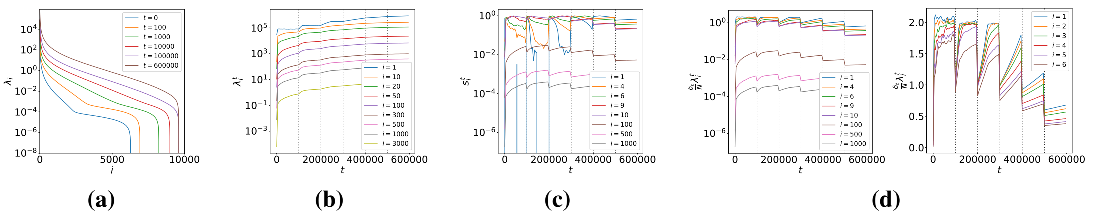


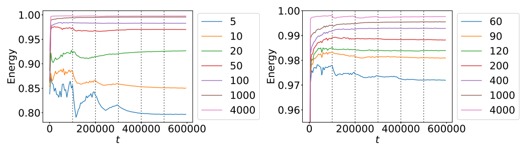
Figure 14.2: (a) Eigenvalues $\{\lambda^t_i\}_{i=1}^N$ for different t. (b) Individual eigenvalues along t. As observed, eigenvalues monotonically grow along t, with growing boost at times of the learning rate drop. (c) The information flow speed s^t_i discussed in Section 14.3 for several *top* eigenvectors. For first 8 eigenvectors, roughly, this speed is increased at learning rate drop. (d) $\frac{\delta_t}{N}\lambda^t_i$ along time t, for various i.

Further, we compute G_t and its spectrum along the optimization, and thoroughly analyze them below.

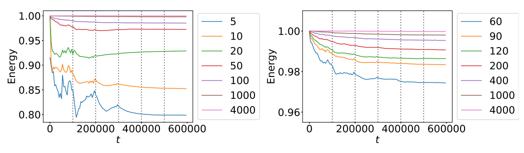
Eigenvalues In Figures 14.2a-14.2b it is shown that each eigenvalue is monotonically increasing along t. Moreover, at learning rate decay there is an especial boost in its growth. Since $\frac{\delta_t}{N}\lambda^t_i$ also defines a speed of movement in θ-space along one of FIM eigenvectors (see Section 14.2), such behavior of eigenvalues suggests an existence of mechanism that keeps a roughly constant movement speed of θ within $\mathbb{R}^{|\theta|}$. To do that, when δ_t is reduced, this mechanism is responsible for increase of $\{\lambda^t_i\}_{i=1}^N$ as a compensation. This is also supported by Figure 14.2d where each $\frac{\delta_t}{N}\lambda^t_i$ is balancing, roughly, around the same value along the entire optimization. Furthermore, in Figure 14.1e it is clearly observed that an evolution of λ^t_{max} stabilizes[3] only when it reaches value of $\frac{2N}{\delta_t}$, further supporting the above hypothesis.

Neural Spectrum Alignment Notate by $\cos\left[\alpha_t\left(\bar{\phi}, k\right)\right] \triangleq \sqrt{\frac{\sum_{i=1}^k <\bar{\upsilon}^t_i, \bar{\phi}>^2}{\|\bar{\phi}\|_2^2}}$ the cosine of an angle $\alpha_t\left(\bar{\phi}, k\right)$ between an arbitrary vector $\bar{\phi}$ and its projection to the sub-space of $\mathbb{R}^N$ spanned by $\{\bar{\upsilon}^t_i\}_{i=1}^k$. Further, $E_t(\bar{\phi}, k) \triangleq \cos^2\left[\alpha_t\left(\bar{\phi}, k\right)\right]$ can be considered as a *relative energy* of $\bar{\phi}$, the percentage of its energy $\left\|\bar{\phi}\right\|_2^2$ located inside *span* $\left(\{\bar{\upsilon}^t_i\}_{i=1}^k\right)$. In our experiments we will

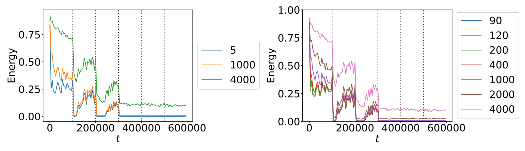
[3]Trend $\lambda^t_{max} \rightarrow \frac{2N}{\delta_t}$ was consistent in FC NNs for a wide range of initial learning rates, number of layers and neurons, and various datasets (see [63]), making it an interesting venue for a future theoretical investigation



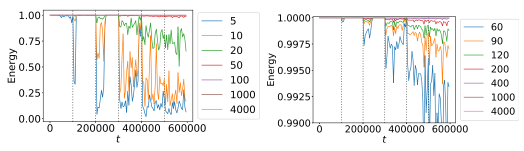
(a)



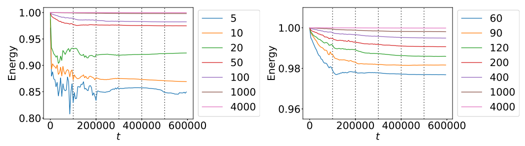
(b)



(c)



(d)



(e)

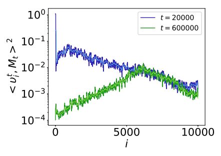
Figure 14.3: (a) For different k, relative energy of the label vector $\bar{y}$ in *top* k eigenvectors of G_t, $E_t(\bar{y}, k)$, along the optimization time t. (b) Relative energy of NN output, $E_t(\bar{f}_t, k)$. (c) Relative energy of the residual, $E_t(\bar{m}_t, k)$. (d) Relative energy of the differential $d\bar{f}_t = -\frac{\delta_t}{N} \cdot G_t \cdot \bar{m}_t$, $E_t(d\bar{f}_t, k)$. (e) Relative energy of NN output, $E_t(\bar{f}_t^{test}, k)$, with both G_t and $\bar{f}_t^{test}$ computed at 10^4 testing points. Dashed vertical lines depict time t at which learning rate δ was decayed (see Figure 14.1e).

use $E_t(\bar{\phi}, k)$ as an alignment metric between $\bar{\phi}$ and $\{\bar{\upsilon}_i^t\}_{i=1}^k$. Further, we evaluate alignment of G_t with $\bar{y}$ instead of $\bar{m}_0$ since $\bar{f}_0$ is approximately zero in the considered FC networks.

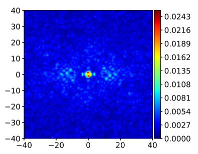
In Figure 14.3a we depict relative energy of the label vector $\bar{y}$ in *top* k eigenvectors of G_t, $E_t(\bar{y}, k)$. As observed, 20 *top* eigenvectors of G_t contain 90 percent of $\bar{y}$ for almost all t. Similarly, 200 *top* eigenvectors of G_t contain roughly 98 percent of $\bar{y}$, with rest of eigenvectors being practically orthogonal w.r.t. $\bar{y}$. That is, G_t aligns its *top* spectrum towards the ground truth target function $\bar{y}$ almost immediately after starting of training, which improves the convergence rate since the information flow is fast along *top* eigenvectors as discussed in Section 14.3 and proved in [7, 99].

Further, we can see that for $k < 400$ the relative energy $E_t(\bar{y}, k)$ is decreasing after each decay of δ, yet for $k > 400$ it keeps growing along the entire optimization. Hence, the *top* eigenvectors of G_t can be seen as NN memory that is learned/tuned toward representing the target $\bar{y}$, while after each learning rate drop the learned information is spread more evenly among a higher number of different *top* eigenvectors.

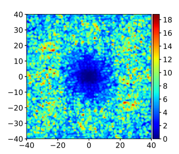
Likewise, in Figure 14.3b we can see that NN outputs vector $\bar{f}_t$ is located entirely in a few hundreds of *top* eigenvectors. In case we consider G_t to be constant, such behavior can be explained by Eq. (14.3) since each increment of $\bar{f}_t$, $d\bar{f}_t$, is also located almost entirely within *top* 60 eigenvectors of G_t (e.g. see $E_t(d\bar{f}_t, 60)$ in Figure 14.3d). Yet, for a general NN with a time-dependent kernel the theoretical justification for the above empirical observation is currently missing. Further, similar relation is observed also at points outside of $\boldsymbol{\mathcal{X}}$ (see Figure 14.3e), leading to the empirical conclusion that *top* eigenfunctions of *gradient similarity* $g_t(X, X')$ are the basis functions of NN $f_\theta(X)$.



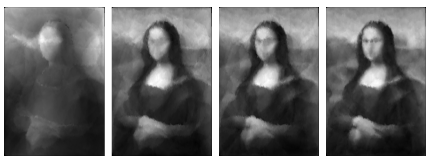
(a)



(b)



(c)



(d)

Figure 14.4: (a) Spectral projections of the residual $\bar{m}_t$, $< \bar{\upsilon}_i^t, \bar{m}_t >^2$, at $t = 20000$ and $t = 600000$; (b) and (c) Fourier Transform of $\bar{m}_t$ at $t = 20000$ and $t = 600000$ respectively. The high frequency is observed to be dominant in (c). (d) a linear combination $\bar{f}_{t,k} \triangleq \sum_{i=1}^{k} < \bar{\upsilon}_i^t, \bar{f}_t > \bar{\upsilon}_i^t$ of first $k = \{10, 100, 200, 500\}$ eigenvectors at $t = 600000$. Each vector $\bar{f}_{t,k}$ was interpolated from training points $\{X^i\}_{i=1}^N$ to entire $[0, 1]^2$ via a linear interpolation.

Residual Dynamics Further, a projection of the residual $\bar{m}_t$ onto *top* eigenvectors, shown in Figure 14.3c, is decreasing along t, supporting Eq. (14.5). Particularly, we can see that at $t = 600000$ only 10% of $\bar{m}_t$'s energy is located inside *top* 4000 eigenvectors, and thus at the optimization end 90% of its energy is inside *bottom* eigenvectors. Moreover, in Figure 14.4a we can observe that the projection of $\bar{m}_t$ along *bottom* 5000 eigenvectors almost does not change during the entire optimization. This may be caused by two main reasons - the slow convergence rate associated with *bottom* eigenvectors and a single-precision floating-point (float32) format used in our simulation. The latter can prevent the information flow along the *bottom* spectrum due to the numerical precision limit. No matter the case, we empirically observe that the information located in the *bottom* spectrum of G_t was not learned, even for a relatively long optimization process (i.e. 600000 iterations). Furthermore, since this spectrum part is also associated with high-frequency information [11], $\bar{m}_t$ at $t = 600000$ comprises mostly the noise, which is also evident from Figures 14.4b-14.4c.

Moreover, we can also observe in Figure 14.3c a special drop of $E_t(\bar{m}_t, k)$ at times of δ decrease. This can be explained by the fact that a lot of $\bar{m}_t$'s energy is located inside first several $\{\bar{\upsilon}_i^t\}$ (see $E_t(\bar{m}_t, 5)$ in Figure 14.3c). When learning rate is decreased, the information flow speed $s_i^t \triangleq 1 - |1 - \frac{\delta_t}{N} \lambda_i^t|$, discussed in Section 14.3, is actually increasing for a few *top* eigenvectors (see Figure 14.2c). That is, terms $\frac{\delta_t}{N} \lambda_i^t$, being very close to 2 before δ's decay, are getting close to 1 after, as seen in Figure 14.2d. In its turn this accelerates the information flow along these first $\{\bar{\upsilon}_i^t\}$, as described in Eq. (14.4)-(14.5), leading also to a special descend of $E_t(\bar{m}_t, k)$ and of the training loss in Figure 14.7b.

Eigenvectors We further explore $\{\bar{\upsilon}_i^t\}$ in a more illustrative manner, to produce a better intuition about their nature. In Figure 14.4d a linear combination of several *top* eigenvectors at $t = 600000$ is presented, showing that with only 100 vectors we can accurately approximate the NN output.

Furthermore, in Figure 14.5 several eigenvectors are interpolated to entire $[0, 1]^2$. We can see that *top* $\{\bar{\upsilon}_i^t\}$ obtained visual similarity with various parts of Mona Lisa image and indeed can be seen as basis functions of $f_\theta(X)$ depicted in Figure 14.1b. Likewise, we also demonstrate the Fourier Transform of each $\bar{\upsilon}_i^t$. As observed, the frequency of the contained information is higher for smaller eigenvalues, supporting conclusions of [11]. More eigenvectors are depicted

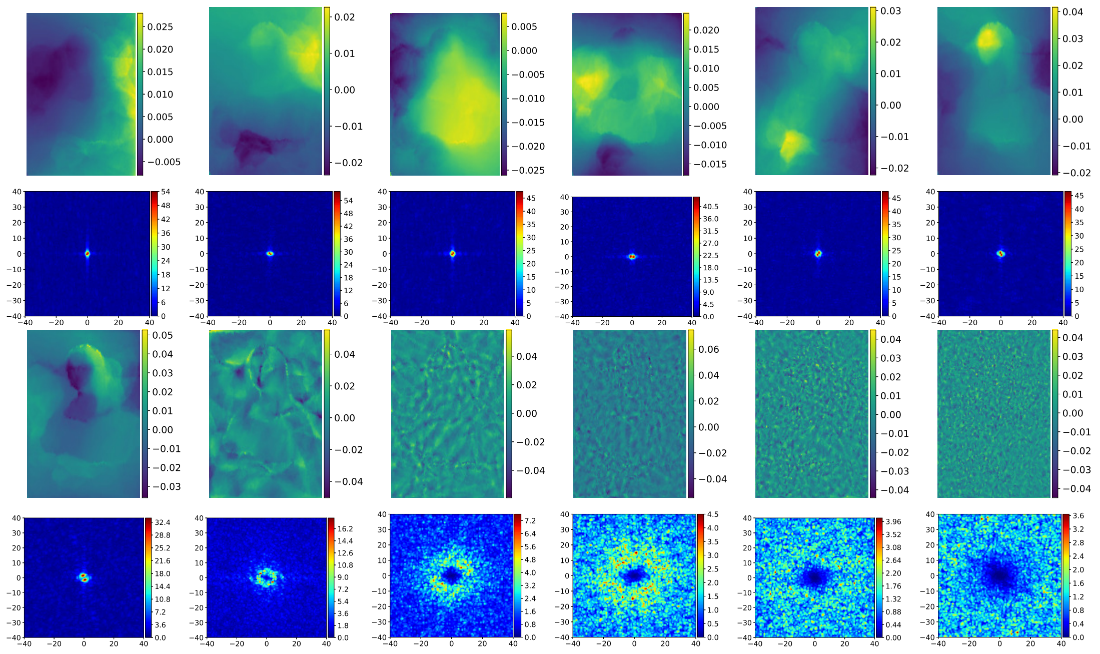


Figure 14.5: Eigenvectors of Gramian G_t at $t = 600000$. First two rows: from left-to-right, 6 first eigenvectors and their Fourier Transforms (see the Appendix Q for details). Last two rows: 10-th, 100-th, 500-th, 1000-th, 2000-th and 4000-th eigenvectors, and their Fourier Transforms. As observed, a frequency of signal inside of each eigenvector increases when moving from large to small eigenvalue.

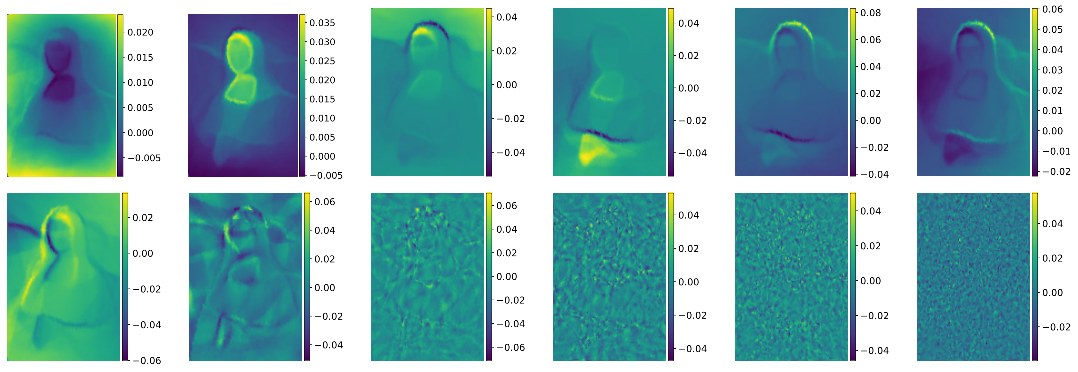
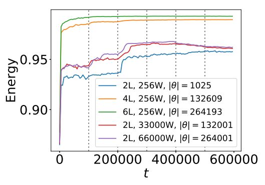
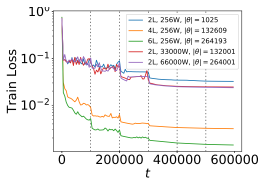


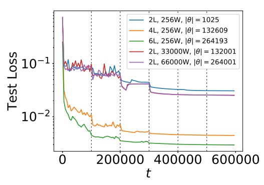
Figure 14.6: First line: from left-to-right, 6 first eigenvectors of Gramian G_t at $t = 20000$. Second line: 10-th, 100-th, 500-th, 1000-th, 2000-th and 4000-th eigenvectors.



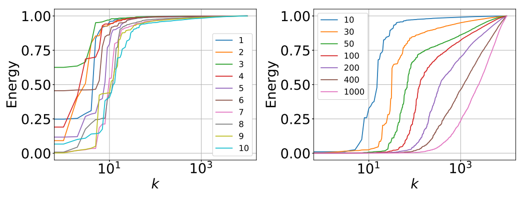
(a)



(b)



(c)



(d)

Figure 14.7: (a) For NNs with a different number of layers and of neurons, relative energy of the label vector $\bar{y}$ in *top* 400 eigenvectors of G_t, $E_t(\bar{y}, 400)$, along the optimization time t; (b) training loss and (c) testing loss of these models. **L** and **W** stand for number of layers and number of neurons respectively. (d) For different i, relative energy of $\bar{\upsilon}_i^{600000}$ in spectrum of G_{20000}, $E_{20000}(\bar{\upsilon}_i^{600000}, k)$, as a function of k, with horizontal axes being log-scaled. As seen, 10 first *top* eigenvectors at final time $t = 600000$ are located also in the *top* spectrum of G_{20000}, hence the *top* Gramian spectrum was preserved along the optimization. Yet, *bottom* eigenvectors are significantly less stable.

in [63].

Likewise, in Figure 14.6 same eigenvectors are displayed at $t = 20000$. At this time the visual similarity between each one of first eigenvectors and the target function in Figure 14.1a is much stronger. This can be explained by the fact that the information about the target function within G_t is spread from first few towards higher number of *top* eigenvectors after each learning rate drop, as was described above. Hence, before the first drop at $t = 100000$ this information is mostly gathered within first few $\{\bar{\upsilon}_i^t\}$ (see also $E_t(\bar{y}, 10)$ in Figure 14.3a).

Alignment and NN Depth / Width Here we further study how a width and a depth of NN affect the alignment between G_t and the ground truth signal $\bar{y}$. To this purpose, we performed the optimization under the identical setup, yet with NNs containing various numbers of layers and neurons. In Figure 14.7a we can see that in deeper NN *top* eigenvectors of G_t aligned more towards $\bar{y}$ - the relative energy $E_t(\bar{y}, 400)$ is higher for a larger depth. This implies that more layers, and the higher level of non-linearity produced by them, yield a better alignment between G_t and $\bar{y}$. In its turn this allows NN to better approximate a given target function, as shown in Figures 14.7b-14.7c, making it more expressive for a given task. Moreover, in evaluated 2-layer NNs, with an increase of neurons and parameters the alignment rises only marginally.

Spectrum Preservation Next, we examine how stable are eigenvectors of G_t along t. For this we explore the relative energy of G_{600000}'s eigenvectors, final eigenvectors of the optimization, within spectrum of G_{20000}. Note that we compare spectrums at $t = 600000$ and $t = 20000$ to skip first several thousands of iterations since during this bootstrap period the change of G_t is highly notable.

In Figure 14.7d we depict $E_{20000}(\bar{\upsilon}_i^{600000}, k)$ as a function of k, for various $\{\bar{\upsilon}_i^{600000}\}$. As observed, 10 first *top* eigenvectors of G_{600000} are also located in the *top* spectrum of G_{20000} - the function $E_{20000}(\bar{\upsilon}_i^{600000}, k)$ is almost 1 for even relatively small k. Hence, the *top* Gramian spectrum was preserved, roughly, along the performed optimization. Further, eigenvectors of smaller eigenvalues (i.e. with higher indexes i) are significantly less stable, with large amount of their energy widely spread inside *bottom* eigenvectors of G_{20000}. Moreover, we can see a clear trend that with higher i the associated eigenvector is less preserved.

Scope of Analysis The above empirical analysis was repeated under numerous different settings and can be found in [63]. We evaluated various FC architectures, with and without shortcuts between the layers and including various activation functions. Likewise, optimizers GD, stochastic GD and Adam were tested on problems of regression (L2 loss) and density estimation (NCE [39]). Additionally, various high-dimensional real-world datasets were tested, including MNIST and CIFAR100. **All** experiments exhibit the same alignment nature of kernel towards the learned target function.

14.6 Summary

In this chapter we empirically revealed that during GD *top* eigenfunctions of *gradient similarity* kernel change to align with the target function $y(X)$ learned by NN $f_\theta(X)$, and hence can be considered as a NN memory tuned during the optimization to better represent $y(X)$. This alignment is significantly higher for deeper NNs, whereas a NN width has only a minor effect on it. Moreover, the same *top* eigenfunctions represent a *neural spectrum* - the $f_\theta(X)$ is a linear combination of these eigenfunctions during the optimization. As well, we showed various trends of the kernel dynamics as a result of the learning rate decay, accounting for which we argue may lead to a further progress in DL theory. The considered herein optimization scenarios include various *supervised* and *unsupervised* losses over various high-dimensional datasets, optimized via several different optimizers. Several variants of FC architecture were evaluated.

The above revealed behavior leads to several implications. First, our empirical study suggests that the high approximation power of deep models is produced by the above alignment capability of the *gradient similarity*, since the learning along its *top* eigenfunctions is considerably faster. Furthermore, it also implies that the family of functions that a NN can approximate (in reasonable time) is limited to functions within the *top* spectrum of the kernel. Recently, it was proved in [7,11,99]. Thus, it leads to the next main question - how the NN architecture and optimization hyper-parameters affect this spectrum, and what is their optimal configuration for learning a given function $y(X)$. Moreover, NN dynamics behavior beyond first-order Taylor expansion is still unexplored. We shall leave it for a future research.

CHAPTER 15

Conclusions and Future Work

In this thesis we contributed a new algorithm family, *Probabilistic Surface Optimization* (PSO), that allows to learn numerous different statistical functions of given data, including (conditional) density estimation and ratios between two unknown pdfs of two given datasets. In our work we found a new perspective to view a model as a representation of a virtual physical surface, which is pushed by the PSO algorithm *up* and *down* via gradient descent (GD) optimization updates. Further, the equilibrium at each point, that is, when *up* and *down* forces are point-wise equal, ensures that the converged surface satisfies PSO *balance state*, where the ratio of the frequency components is equal to the opposite ratio of the analytical components. In Section 5 we saw that such formulation yields infinitely many estimation approaches to learn almost any function of the density ratio. Moreover, it generalizes numerous existing works, like *energy* and *unnormalized* models as also critics of GAN approaches. Likewise, we showed that f-divergence and Bregman divergence based techniques (e.g. the cross-entropy loss from the image classification domain) are also instances of PSO, applying the same physical forces over the model surface.

We provided a thorough analysis of the PSO functional implicitly employed during the optimization, describing its equilibrium for a wide diapason of settings. Furthermore, we derived the sufficient conditions over PSO *magnitude* functions under which the equilibrium is stable. We likewise related PSO to Legendre-Fenchel transform, demonstrating that its convergence is an inverse of the *magnitude* ratio, with their *primitives* being convex-conjugate of each other. This resembles the relationship between Langrangian and Hamiltonian mechanics, opening interesting future directions to connect control and learning theories.

Furthermore, we systematically modulated the set of all PSO instances into various subgroups, providing a useful terminology for a future PSO study. Additionally, along this paper we described several possible parameterizations of PSO family, with each having its own benefits. Concretely, PSO can be represented/parametrized via a pair of *magnitudes* $\{M^U, M^D\}$ which leads to the geometrical/physical force perspective. Such angle brings many insights and is the central focus of this work. Likewise, we can parametrize PSO by $\{\widetilde{M}^U, \widetilde{M}^D\}$ that leads to the *PSO functional*, which may be viewed as an energy of the optimized physical system. Further, the polar parametrization $\{c_r, c_\angle\}$ in Section 5.3 permits for an easier feasibility verification. Lastly, $\{\phi^c, G\}$ described in Section 6.3 allows to connect PSO methods with f-divergence between *up* and *down* densities.

Moreover, the **main** goal behind this work is to introduce a novel universal way in forging new statistical techniques and corresponding objective functions. Due to simplicity and intuitiveness of the presented PSO principles, this new framework allows for an easy derivation of new statistical approaches, which, in turn, is highly useful in many different domains. Depending on the target function required by a specific application, a data analyst can select suitable *magnitude* functions according to the PSO *balance state*, and simply employ them inside the general PSO loss. Along this thesis we demonstrated a step-by-step derivation of several such new approaches.

Likewise, herein we investigated the reason for high resemblance between the statistical model inference and physics over virtual surfaces. We showed that during the optimization, a change of model output at any point (the height change of the virtual surface at the point) is equal to the model kernel (a.k.a. NTK, [54]) between this point and the optimized training point. Following from this, the optimization can be viewed as pushes at training points performed via some employed sticks, whose shape is described by the kernel. We analyzed this kernel's properties (e.g. the shape of the pushing sticks) and their impact over the convergence of PSO algorithm. Specifically, we showed that its bandwidth corresponds to the flexibility/expressiveness of the model - with a narrower kernel it is possible to push the surface towards various target forms, making it more elastic.

Further, the bandwidth of the model kernel can be viewed as a hyper-parameter that controls the estimation bias-variance tradeoff. We empirically investigated both *underfitting* and *overfitting* scenarios that can occur in PSO. In our experiments we showed that the wide bandwidth is correlated with a sub-optimal optimization performance in case of a large training dataset. Moreover, if it is too narrow and in case of a small training dataset, the surface converges to peaks around the training points and also produce a poor target approximation. Thus, the optimal kernel bandwidth depends on the number of available training points, which agrees with existing analysis of KDE methods.

Furthermore, we showed that the model kernel serves as a metric over function space during PSO estimation procedure, which agrees with already existing NTK literature [54]. Namely, its eigenfunctions associated with largest eigenvalues define directions inside the space of functions where propagation/movement is fast, and vice versa. Moreover, our empirical analysis of NTK during the learning process showed a particular dynamics pattern where top eigenfunctions are aligned towards the target function. Such surprising behavior allows to easily propagate towards the global minima of the optimization and is overall extremely beneficial, which may explain why NN-based models typically produce more accurate results compared to RKHS-based models whose kernel is constant.

Lastly, we applied PSO to learn data log-density, proposing several new PSO instances for this purpose, including PSO *log density estimators* (PSO-LDE). Additionally, we presented a new NN block-diagonal architecture that allowed us to significantly reduce the bandwidth of the model kernel and to extremely increase an approximation accuracy. In our experiments we showed how the above methods can be used to perform precise pdf inference of multi-modal 20D data, getting a superior accuracy over other state-of-the-art baselines. Importantly, in an infinite dataset setting we also empirically revealed a connection between the point-wise error and gradient norm at the point, which in theory can be used for measuring a model uncertainty.

15.1 Future Research Directions

Along this thesis we remarked many possible research directions to further enhance PSO estimation techniques. The current solution is still very new and many of its aspects require additional attention and further study. Below is the list of research topics that shall be addressed in the future:

- Formulation of PSO framework as an arbitrary flow: Currently our approach is based on an existence of *PSO functional* which is minimized during the optimization, and therefore PSO can be viewed as a gradient flow of this functional. Yet, we can extend it to a flow which does not correspond to any objective function. This will allow us to reduce some of the ”sufficient” conditions over *magnitude* functions that were derived in Section 4.1. Likewise, it can lead to even a more general estimation framework with a higher practical applicability, and to extensive theoretical implications.

- Search for best density estimator: In the context of density estimation, currently we learn multiple models for different values of PSO-LDE hyper-parameter α and choose the one with the highest performance metric. Yet, such brute-force procedure is computationally very expensive. It is important to understand the exact connection between α and the produced log-pdf estimation, and also to provide a more intelligent way to choose α based on properties of the given data. Likewise, a more thorough exploration of all PSO instances for log-pdf inference is required.

- Robust statistics: The above topic can be extended to a search for the most optimal PSO instance of any considered target function. Since PSO framework allows us to generate an infinite number of various PSO instances to approximate a specific function, the natural question to ask is which one should we pick. The answer is currently unclear, and the goal here is to categorize different instances by their statistic robustness properties and to find the optimal one. This topic also includes questions of what is the optimality and how the most optimal PSO instance is related to properties of the model kernel.

- Model kernel impact: Although we analyzed the effect of some kernel properties, the entire and full understating of the kernel impact is still missing. It is important to understand the precise relation between an accuracy of PSO estimators and various properties of the model kernel. This topic also includes the analysis of PSO convergence rates (w.r.t. a number of GD iterations), generalization bounds (w.r.t. a number of training points), and the impact of $g_\theta(X, X')$ in small dataset regime.

- Model kernel beyond GD: The above topics and the corresponding analysis must be extended beyond a simple full-batch GD optimization. In practice a mini-batch setting combined with various optimizers (e.g. Adam [59]) is common. Therefore, to fully understand the real learning process and its properties, it is important to extend the idea of the model kernel from GD optimizer towards other optimizers, and also to account for difference between ”mini” and ”full” batch regimes.
- Control properties of the kernel via NN architecture: Once we answer the above questions and once we know what are the most desired properties for kernel to have, the next natural endeavor is to construct a model with such kernel. Hence, we want to understand how to control $g_\theta(X, X')$ via NN design. Although this topic was partly addressed in this thesis (the BD architecture allowed us to reduce kernel bandwidth and to improve estimation accuracy), it is still very far from being solved. A promising direction to solve this topic is to first understand why NTK alignment happens, which may produce us with a set of tools to adjust/control $g_\theta(X, X')$. Research in this direction may guide us to better NN architectures and new methods to control the bias-variance tradeoff.
- Better model regularization: One of the most difficult issues to handle when applying PSO in practice is its *overfitting* behavior. As was shown in Section 12, when the applied model is overly flexible, the practical outcome will be the spikes at the training points. Furthermore, this behavior is even more extreme in high-dimensional small dataset setting. Here our goal is to propose efficient and theoretically-motivated regularization methods that reduce the above over-flexibility problem.

APPENDIX A

Proof of Lemmas 14 and 15

A.1 Lemma 14

Consider the setting of Section 7.1. Further, notate the model Hessian as $\mathcal{H}_\theta(X) \equiv \nabla_{\theta\theta} f_\theta(X)$. Using the gradient of $L_{PSO}(f_\theta)$ defined in Eq. (3.5), the second derivative of $L_{PSO}(f_\theta)$ w.r.t. θ is:

$$\begin{aligned}
\nabla_{\theta\theta} L_{PSO}(f_\theta) = &- \mathop{\mathbb{E}}_{X\sim\mathbb{P}^U} \left[M^{U\prime}\left[X, f_\theta(X)\right] \cdot \mathcal{I}_\theta(X,X) + M^U\left[X, f_\theta(X)\right] \cdot \mathcal{H}_\theta(X)\right] + \\
&+ \mathop{\mathbb{E}}_{X\sim\mathbb{P}^D} \left[M^{D\prime}\left[X, f_\theta(X)\right] \cdot \mathcal{I}_\theta(X,X) + M^D\left[X, f_\theta(X)\right] \cdot \mathcal{H}_\theta(X)\right] = \\
= &- \mathop{\mathbb{E}}_{X\sim\mathbb{P}^U} M^{U\prime}\left[X, f_\theta(X)\right] \cdot \mathcal{I}_\theta(X,X) - \mathop{\mathbb{E}}_{X\sim\mathbb{P}^U} M^U\left[X, f_\theta(X)\right] \cdot \mathcal{H}_\theta(X) + \\
&+ \mathop{\mathbb{E}}_{X\sim\mathbb{P}^D} M^{D\prime}\left[X, f_\theta(X)\right] \cdot \mathcal{I}_\theta(X,X) + \mathop{\mathbb{E}}_{X\sim\mathbb{P}^D} M^D\left[X, f_\theta(X)\right] \cdot \mathcal{H}_\theta(X).
\end{aligned} \tag{A.1}$$

Likewise, at θ^* we also have:

$$\begin{aligned}
&- \mathop{\mathbb{E}}_{X\sim\mathbb{P}^U} M^U\left[X, f_{\theta^*}(X)\right] \cdot \mathcal{H}_{\theta^*}(X) + \mathop{\mathbb{E}}_{X\sim\mathbb{P}^D} M^D\left[X, f_{\theta^*}(X)\right] \cdot \mathcal{H}_{\theta^*}(X) = \\
&= - \mathop{\mathbb{E}}_{X\sim\mathbb{P}^U} M^U\left[X, f^*(X)\right] \cdot \mathcal{H}_{\theta^*}(X) + \mathop{\mathbb{E}}_{X\sim\mathbb{P}^D} M^D\left[X, f^*(X)\right] \cdot \mathcal{H}_{\theta^*}(X) = \\
&= \int \left[-\mathbb{P}^U(X) \cdot M^U\left[X, f^*(X)\right] + \mathbb{P}^D(X) \cdot M^D\left[X, f^*(X)\right]\right] \cdot \mathcal{H}_{\theta^*}(X) dX = 0,
\end{aligned} \tag{A.2}$$

where the last row is true because f^* satisfies PSO *balance state*.

Therefore, we have

$$\begin{aligned}
\mathcal{H} \equiv \nabla_{\theta\theta} L_{PSO}(f_{\theta^*}) = \\
= - \mathop{\mathbb{E}}_{X\sim\mathbb{P}^U} M^{U\prime}\left[X, f_{\theta^*}(X)\right] \cdot \mathcal{I}_{\theta^*}(X,X) + \mathop{\mathbb{E}}_{X\sim\mathbb{P}^D} M^{D\prime}\left[X, f_{\theta^*}(X)\right] \cdot \mathcal{I}_{\theta^*}(X,X) = \\
= - \mathop{\mathbb{E}}_{X\sim\mathbb{P}^U} M^{U\prime}\left[X, f^*(X)\right] \cdot \mathcal{I}_{\theta^*}(X,X) + \mathop{\mathbb{E}}_{X\sim\mathbb{P}^D} M^{D\prime}\left[X, f^*(X)\right] \cdot \mathcal{I}_{\theta^*}(X,X).
\end{aligned} \tag{A.3}$$

Observe also that only $\mathcal{I}_{\theta^*}$ terms depend on the parameter vector θ^*.

Additionally, $\mathcal{H}$ has an another form:

$$\mathcal{H} = \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \frac{M^U\left[X, f^*(X)\right]}{T'\left[X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}\right]} \cdot \mathcal{I}_{\theta^*}(X, X), \tag{A.4}$$

where $T'(X, z) \triangleq \frac{\partial T(X,z)}{\partial z}$ is a first derivative of the considered PSO convergence $T(X, z)$. This can be derived as follows.

First, since T and $R \equiv \frac{M^D}{M^U}$ are inverse functions, derivative of R can be computed via derivative of T as:

$$\frac{\partial R(X, s)}{\partial s} = \frac{1}{T'(X, R(X, s))}. \tag{A.5}$$

Observe that due to PSO *balance state* $R(X, f^*(X)) = \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)}$ we have

$$\left.\frac{\partial R(X, s)}{\partial s}\right|_{s=f^*(X)} = \frac{1}{T'(X, R(X, f^*(X)))} = \frac{1}{T'(X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)})}. \tag{A.6}$$

Next, the expression inside integral of Eq. (A.3) is:

$$\begin{aligned}
&- \mathbb{P}^U(X) \cdot M^{U\prime}\left[X, f^*(X)\right] + \mathbb{P}^D(X) \cdot M^{D\prime}\left[X, f^*(X)\right] = \\
&= \mathbb{P}^D(X) \cdot M^{U\prime}\left[X, f^*(X)\right] \cdot \left[-\frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)} + \frac{M^{D\prime}\left[X, f^*(X)\right]}{M^{U\prime}\left[X, f^*(X)\right]}\right] = \\
&= \mathbb{P}^D(X) \cdot M^{U\prime}\left[X, f^*(X)\right] \cdot \left[-\frac{M^D\left[X, f^*(X)\right]}{M^U\left[X, f^*(X)\right]} + \frac{M^{D\prime}\left[X, f^*(X)\right]}{M^{U\prime}\left[X, f^*(X)\right]}\right] = \\
&= \mathbb{P}^D(X) \cdot M^U\left[X, f^*(X)\right] \cdot \frac{M^{D\prime}\left[X, f^*(X)\right] \cdot M^U\left[X, f^*(X)\right] -}{M^U\left[X, f^*(X)\right]^2} \\
&\quad \underline{-M^D\left[X, f^*(X)\right] \cdot M^{U\prime}\left[X, f^*(X)\right]} = \\
&= \mathbb{P}^D(X) \cdot M^U\left[X, f^*(X)\right] \cdot \frac{\partial \frac{M^D}{M^U}(X, s)}{\partial s}|_{s=f^*(X)} = \\
&= \mathbb{P}^D(X) \cdot M^U\left[X, f^*(X)\right] \cdot \frac{\partial R(X, s)}{\partial s}|_{s=f^*(X)} = \frac{\mathbb{P}^D(X) \cdot M^U\left[X, f^*(X)\right]}{T'(X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)})}.
\end{aligned} \tag{A.7}$$

Hence:

$$\mathcal{H} = \int \frac{\mathbb{P}^D(X) \cdot M^U\left[X, f^*(X)\right]}{T'(X, \frac{\mathbb{P}^U(X)}{\mathbb{P}^D(X)})} \cdot \mathcal{I}_{\theta^*}(X, X) dX, \tag{A.8}$$

from which Eq. (A.4) follows.

■

A.2 Lemma 15

Consider the empirical PSO gradient $\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)$ as defined in Eq. (3.1). Its uncentered variance is then:

$$\begin{aligned}
&\mathbb{E}\left[\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta) \cdot \nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)^T\right] = \\
&\quad = \frac{1}{(N^U)^2} \sum_{i,j=1}^{N^U} \mathbb{E}\left[M^U\left[X_i^U, f_\theta(X_i^U)\right] \cdot M^U\left[X_j^U, f_\theta(X_j^U)\right] \cdot \mathcal{I}_\theta(X_i^U, X_j^U)\right] + \\
&\quad + \frac{1}{(N^D)^2} \sum_{i,j=1}^{N^D} \mathbb{E}\left[M^D\left[X_i^D, f_\theta(X_i^D)\right] \cdot M^D\left[X_j^D, f_\theta(X_j^D)\right] \cdot \mathcal{I}_\theta(X_i^D, X_j^D)\right] - \\
&\quad - \frac{1}{N^U N^D} \sum_{i=1}^{N^U} \sum_{j=1}^{N^D} \mathbb{E}\left[M^U\left[X_i^U, f_\theta(X_i^U)\right] \cdot M^D\left[X_j^D, f_\theta(X_j^D)\right] \cdot \mathcal{I}_\theta(X_i^U, X_j^D)\right] - \\
&\quad - \frac{1}{N^U N^D} \sum_{i=1}^{N^U} \sum_{j=1}^{N^D} \mathbb{E}\left[M^U\left[X_i^U, f_\theta(X_i^U)\right] \cdot M^D\left[X_j^D, f_\theta(X_j^D)\right] \cdot \mathcal{I}_\theta(X_j^D, X_i^U)\right] = \\
&\quad = \frac{1}{(N^U)^2} \sum_{i=1}^{N^U} \mathbb{E}\left[M^U\left[X_i^U, f_\theta(X_i^U)\right] \cdot M^U\left[X_i^U, f_\theta(X_i^U)\right] \cdot \mathcal{I}_\theta(X_i^U, X_i^U)\right] + \\
&\quad + \frac{1}{(N^D)^2} \sum_{i=1}^{N^D} \mathbb{E}\left[M^D\left[X_i^D, f_\theta(X_i^D)\right] \cdot M^D\left[X_i^D, f_\theta(X_i^D)\right] \cdot \mathcal{I}_\theta(X_i^D, X_i^D)\right] + \\
&\quad + \frac{1}{(N^U)^2} \sum_{\substack{i,j=1 \\ i \neq j}}^{N^U} \mathbb{E}\left[M^U\left[X_i^U, f_\theta(X_i^U)\right] \cdot M^U\left[X_j^U, f_\theta(X_j^U)\right] \cdot \mathcal{I}_\theta(X_i^U, X_j^U)\right] + \\
&\quad + \frac{1}{(N^D)^2} \sum_{\substack{i,j=1 \\ i \neq j}}^{N^D} \mathbb{E}\left[M^D\left[X_i^D, f_\theta(X_i^D)\right] \cdot M^D\left[X_j^D, f_\theta(X_j^D)\right] \cdot \mathcal{I}_\theta(X_i^D, X_j^D)\right] - \\
&\quad - \frac{1}{N^U N^D} \sum_{i=1}^{N^U} \sum_{j=1}^{N^D} \mathbb{E}\left[M^U\left[X_i^U, f_\theta(X_i^U)\right] \cdot M^D\left[X_j^D, f_\theta(X_j^D)\right] \cdot \mathcal{I}_\theta(X_i^U, X_j^D)\right] - \\
&\quad - \frac{1}{N^U N^D} \sum_{i=1}^{N^U} \sum_{j=1}^{N^D} \mathbb{E}\left[M^U\left[X_i^U, f_\theta(X_i^U)\right] \cdot M^D\left[X_j^D, f_\theta(X_j^D)\right] \cdot \mathcal{I}_\theta(X_j^D, X_i^U)\right]. \qquad (A.9)
\end{aligned}$$

Denote:

$$\bar{\mu}_\theta^U \triangleq \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} M^U\left[X, f_\theta(X)\right] \cdot \nabla_\theta f_\theta(X), \quad \bar{\mu}_\theta^D \triangleq \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} M^D\left[X, f_\theta(X)\right] \cdot \nabla_\theta f_\theta(X). \qquad (A.10)$$

According to Eq. (A.9), we have:

$$\begin{aligned}
&\mathbb{E}\left[\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta) \cdot \nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)^T\right] = \\
&= \frac{1}{N^U} \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} \left[M^U[X, f_\theta(X)]^2 \cdot \mathcal{I}_\theta(X,X)\right] + \frac{1}{N^D} \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \left[M^D[X, f_\theta(X)]^2 \cdot \mathcal{I}_\theta(X,X)\right] + \\
&\qquad + \frac{N^U-1}{N^U} \bar{\mu}_\theta^U \cdot (\bar{\mu}_\theta^U)^T + \frac{N^D-1}{N^D} \bar{\mu}_\theta^D \cdot (\bar{\mu}_\theta^D)^T - \bar{\mu}_\theta^U \cdot (\bar{\mu}_\theta^D)^T - \bar{\mu}_\theta^D \cdot (\bar{\mu}_\theta^U)^T = \\
&\qquad = \frac{1}{N^U} \left[\mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} \left[M^U[X, f_\theta(X)]^2 \cdot \mathcal{I}_\theta(X,X)\right] - \bar{\mu}_\theta^U \cdot (\bar{\mu}_\theta^U)^T\right] + \\
&\qquad + \frac{1}{N^D} \left[\mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \left[M^D[X, f_\theta(X)]^2 \cdot \mathcal{I}_\theta(X,X)\right] - \bar{\mu}_\theta^D \cdot (\bar{\mu}_\theta^D)^T\right] + \\
&\qquad\qquad + \bar{\mu}_\theta^U \cdot (\bar{\mu}_\theta^U)^T + \bar{\mu}_\theta^D \cdot (\bar{\mu}_\theta^D)^T - \bar{\mu}_\theta^U \cdot (\bar{\mu}_\theta^D)^T - \bar{\mu}_\theta^D \cdot (\bar{\mu}_\theta^U)^T. \qquad (A.11)
\end{aligned}$$

Further, the outer product of $\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)$'s expected value $\mathbb{E}\left[\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)\right] = -\bar{\mu}_\theta^U + \bar{\mu}_\theta^D$ is:

$$\mathbb{E}\left[\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)\right] \cdot \mathbb{E}\left[\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)\right]^T = \bar{\mu}_\theta^U \cdot (\bar{\mu}_\theta^U)^T + \bar{\mu}_\theta^D \cdot (\bar{\mu}_\theta^D)^T - \bar{\mu}_\theta^U \cdot (\bar{\mu}_\theta^D)^T - \bar{\mu}_\theta^D \cdot (\bar{\mu}_\theta^U)^T, \qquad (A.12)$$

and hence:

$$\begin{aligned}
\operatorname{Var}\left[\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)\right] &= \frac{1}{N^U} \left[\mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} \left[M^U[X, f_\theta(X)]^2 \cdot \mathcal{I}_\theta(X,X)\right] - \bar{\mu}_\theta^U \cdot (\bar{\mu}_\theta^U)^T\right] + \\
&+ \frac{1}{N^D} \left[\mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \left[M^D[X, f_\theta(X)]^2 \cdot \mathcal{I}_\theta(X,X)\right] - \bar{\mu}_\theta^D \cdot (\bar{\mu}_\theta^D)^T\right]. \qquad (A.13)
\end{aligned}$$

Next, using relations $N^U = \frac{\tau}{\tau+1} N$ and $N^D = \frac{1}{\tau+1} N$, we can write the above variance as:

$$\begin{aligned}
&\operatorname{Var}\left[\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta)\right] = \frac{1}{N}\left[\frac{\tau+1}{\tau} \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} \left[M^U[X, f_\theta(X)]^2 \cdot \mathcal{I}_\theta(X,X)\right] + \right. \\
&\left. + [\tau+1] \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \left[M^D[X, f_\theta(X)]^2 \cdot \mathcal{I}_\theta(X,X)\right] - \frac{\tau+1}{\tau} \bar{\mu}_\theta^U \cdot (\bar{\mu}_\theta^U)^T - [\tau+1] \bar{\mu}_\theta^D \cdot (\bar{\mu}_\theta^D)^T\right]. \qquad (A.14)
\end{aligned}$$

Further, due to the identical support assumption $\mathbb{S}^U \equiv \mathbb{S}^D$ we also have $\bar{\mu}_{\theta^*}^U = \bar{\mu}_{\theta^*}^D$:

$$\begin{aligned}
&\mathbb{E}\left[\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_{\theta^*})\right] = -\bar{\mu}_{\theta^*}^U + \bar{\mu}_{\theta^*}^D = \\
&\quad = - \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} M^U[X, f^*(X)] \cdot \nabla_\theta f_{\theta^*}(X) + \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} M^D[X, f^*(X)] \cdot \nabla_\theta f_{\theta^*}(X) = \\
&= \int \left[-\mathbb{P}^U(X) \cdot M^U[X, f^*(X)] + \mathbb{P}^D(X) \cdot M^D[X, f^*(X)]\right] \cdot \nabla_\theta f_{\theta^*}(X) dX = 0, \qquad (A.15)
\end{aligned}$$

where the last row is true because f^* satisfies PSO *balance state*. Hence, the following is also true:

$$\bar{\mu}_{\theta^*}^U \cdot (\bar{\mu}_{\theta^*}^U)^T = \bar{\mu}_{\theta^*}^D \cdot (\bar{\mu}_{\theta^*}^D)^T = \bar{\mu}_{\theta^*}^U \cdot (\bar{\mu}_{\theta^*}^D)^T = \bar{\mu}_{\theta^*}^D \cdot (\bar{\mu}_{\theta^*}^U)^T. \qquad (A.16)$$

Therefore, Var $\left[\nabla_\theta \hat{L}^{N^U,N^D}_{PSO}(f_\theta)\right]$ (Eq. (A.14)) at θ^* is Var $\left[\nabla_\theta \hat{L}^{N^U,N^D}_{PSO}(f_{\theta^*})\right] = \frac{1}{N}\mathcal{J}$ with:

$$\begin{aligned}\mathcal{J} = \frac{\tau+1}{\tau} \mathop{\mathbb{E}}_{X\sim\mathbb{P}^U} \left[M^U\left[X, f^*(X)\right]^2 \cdot \mathcal{I}_{\theta^*}(X,X)\right] + \\ + (\tau+1) \mathop{\mathbb{E}}_{X\sim\mathbb{P}^D} \left[M^D\left[X, f^*(X)\right]^2 \cdot \mathcal{I}_{\theta^*}(X,X)\right] - \\ - \frac{(\tau+1)^2}{\tau} \mathop{\mathbb{E}}_{\substack{X\sim\mathbb{P}^U \\ X'\sim\mathbb{P}^D}} M^U\left[X, f^*(X)\right] \cdot M^D\left[X', f^*(X')\right] \cdot \mathcal{I}_{\theta^*}(X,X'), \quad \text{(A.17)}\end{aligned}$$

where we applied identity from Eq. (A.16). Observe that only $\mathcal{I}_{\theta^*}$ terms depend on the parameter vector θ^*. Likewise, the term next to $\frac{(\tau+1)^2}{\tau}$ is actually $\bar{\mu}^U_{\theta^*} \cdot (\bar{\mu}^D_{\theta^*})^T$, and it can be substituted by either of $\{\bar{\mu}^U_{\theta^*} \cdot (\bar{\mu}^U_{\theta^*})^T; \bar{\mu}^D_{\theta^*} \cdot (\bar{\mu}^D_{\theta^*})^T; \bar{\mu}^D_{\theta^*} \cdot (\bar{\mu}^U_{\theta^*})^T\}$.

■

APPENDIX **B**

Proof of Theorem 16

First, we prove the stepping stone lemmas.

B.1 Lemmata

Lemma 33. *Denote $N \triangleq N^U + N^D$ and $\tau \triangleq \frac{N^U}{N^D}$, and assume τ to be a strictly positive, finite and constant scalar. Then $\sqrt{N} \cdot \nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_{\theta^*}) \xrightarrow{d} \mathcal{N}(0, \mathcal{J})$, convergence in distribution along with $N \to \infty$, where $\mathcal{J}$ is defined by Lemma 15.*

Proof. Define:

$$\Sigma_\theta^U \triangleq \operatorname*{Var}_{X\sim\mathbb{P}^U} M^U[X, f_\theta(X)] \cdot \nabla_\theta f_\theta(X) = \mathop{\mathbb{E}}_{X\sim\mathbb{P}^U}\Big[M^U[X, f_\theta(X)]^2 \cdot \mathcal{I}_\theta(X,X)\Big] - \bar{\mu}_\theta^U \cdot (\bar{\mu}_\theta^U)^T, \tag{B.1}$$

$$\Sigma_\theta^D \triangleq \operatorname*{Var}_{X\sim\mathbb{P}^D} M^D[X, f_\theta(X)] \cdot \nabla_\theta f_\theta(X) = \mathop{\mathbb{E}}_{X\sim\mathbb{P}^D}\Big[M^D[X, f_\theta(X)]^2 \cdot \mathcal{I}_\theta(X,X)\Big] - \bar{\mu}_\theta^D \cdot (\bar{\mu}_\theta^D)^T, \tag{B.2}$$

where $\bar{\mu}_\theta^U$ and $\bar{\mu}_\theta^D$ are defined in Eq. (A.10).

Consider the average $\frac{1}{N^U}\sum_{i=1}^{N^U} M^U[X_i^U, f_{\theta^*}(X_i^U)] \cdot \nabla_\theta f_{\theta^*}(X_i^U)$. It contains i.i.d. random vectors with mean $\bar{\mu}_{\theta^*}^U$ and variance $\Sigma_{\theta^*}^U$. Using a multivariate Central Limit Theorem (CLT) on the considered average, we have:

$$\bar{a} \equiv \sqrt{N^U}\left[\frac{1}{N^U}\sum_{i=1}^{N^U} M^U[X_i^U, f_{\theta^*}(X_i^U)] \cdot \nabla_\theta f_{\theta^*}(X_i^U) - \bar{\mu}_{\theta^*}^U\right] \xrightarrow{d} \mathcal{N}(0, \Sigma_{\theta^*}^U). \tag{B.3}$$

In similar manner, we also have:

$$\bar{b} \equiv \sqrt{N^D}\left[\frac{1}{N^D}\sum_{i=1}^{N^D} M^D[X_i^D, f_{\theta^*}(X_i^D)] \cdot \nabla_\theta f_{\theta^*}(X_i^D) - \bar{\mu}_{\theta^*}^D\right] \xrightarrow{d} \mathcal{N}(0, \Sigma_{\theta^*}^D). \tag{B.4}$$

Therefore, the linear combination also has a convergence in distribution:

$$-\sqrt{\frac{\tau+1}{\tau}} \cdot \bar{a} + \sqrt{\tau+1} \cdot \bar{b} \xrightarrow{d} \mathcal{N}(0, \Sigma), \quad \Sigma \triangleq \frac{\tau+1}{\tau} \cdot \Sigma^U_{\theta^*} + (\tau+1) \cdot \Sigma^D_{\theta^*}, \qquad \text{(B.5)}$$

where $\sqrt{\frac{\tau+1}{\tau}}$ and $\sqrt{\tau+1}$ are finite constant scalars, and where the convergence happens along with $N^U \rightarrow \infty$ and $N^D \rightarrow \infty$. Note that this implies the convergence in $N \rightarrow \infty$, since the latter leads to $\{N^U \rightarrow \infty, N^D \rightarrow \infty, \min(N^U, N^D) \rightarrow \infty\}$ due to τ being fixed and finite.

Further, using relations $N^U = \frac{\tau}{\tau+1}N$ and $N^D = \frac{1}{\tau+1}N$, the above linear combination is equal to:

$$-\sqrt{\frac{\tau+1}{\tau}} \cdot \bar{a} + \sqrt{\tau+1} \cdot \bar{b} = \sqrt{N}\Big[-\frac{1}{N^U}\sum_{i=1}^{N^U} M^U\left[X^U_i, f_{\theta^*}(X^U_i)\right] \cdot \nabla_\theta f_{\theta^*}(X^U_i) + \bar{\mu}^U_{\theta^*} +$$
$$+\frac{1}{N^D}\sum_{i=1}^{N^D} M^D\left[X^D_i, f_{\theta^*}(X^D_i)\right] \cdot \nabla_\theta f_{\theta^*}(X^D_i) - \bar{\mu}^D_{\theta^*}\Big] = \sqrt{N} \cdot \nabla_\theta \hat{L}^{N^U,N^D}_{PSO}(f_{\theta^*}), \quad \text{(B.6)}$$

where we used $\bar{\mu}^U_{\theta^*} = \bar{\mu}^D_{\theta^*}$ from Eq. (A.15). Thus, we have $\sqrt{N} \cdot \nabla_\theta \hat{L}^{N^U,N^D}_{PSO}(f_{\theta^*}) \xrightarrow{d} \mathcal{N}(0, \Sigma)$.

Finally, we have:

$$\Sigma = \frac{\tau+1}{\tau} \cdot \left[\mathop{\mathbb{E}}_{X \sim \mathbb{P}^U}\left[M^U\left[X, f_{\theta^*}(X)\right]^2 \cdot \mathcal{I}_{\theta^*}(X, X)\right] - \bar{\mu}^U_{\theta^*} \cdot (\bar{\mu}^U_{\theta^*})^T\right] +$$
$$+ (\tau+1) \cdot \left[\mathop{\mathbb{E}}_{X \sim \mathbb{P}^D}\left[M^D\left[X, f_{\theta^*}(X)\right]^2 \cdot \mathcal{I}_{\theta^*}(X, X)\right] - \bar{\mu}^D_{\theta^*} \cdot (\bar{\mu}^D_{\theta^*})^T\right]. \quad \text{(B.7)}$$

Using Eq. (A.16) we conclude $\mathcal{J} \equiv \Sigma$.

■

B.2 Proof of Theorem

Define $\hat{\theta}_{N^U,N^D} = \arg\min_{\theta\in\Theta} \hat{L}^{N^U,N^D}_{PSO}(f_\theta)$ and $\theta^* = \arg\min_{\theta\in\Theta} L_{PSO}(f_\theta)$. Since assumptions of Theorem 13 are also the assumptions of Theorem 16, f_{θ^*} satisfies PSO *balance state* and that $\hat{\theta}_{N^U,N^D} \xrightarrow{p} \theta^*$ when $\min(N^U, N^D) \rightarrow \infty$.

Further, note that first order conditions (FOCs) are also satisfied $\nabla_\theta \hat{L}^{N^U,N^D}_{PSO}(f_{\hat{\theta}_{N^U,N^D}}) = 0$. Assuming that $\hat{L}^{N^U,N^D}_{PSO}(f_\theta)$ is continuously differentiable w.r.t. θ, we can apply mean-value theorem on FOCs:

$$\nabla_\theta \hat{L}^{N^U,N^D}_{PSO}(f_{\hat{\theta}_{N^U,N^D}}) = \nabla_\theta \hat{L}^{N^U,N^D}_{PSO}(f_{\theta^*}) + \nabla_{\theta\theta} \hat{L}^{N^U,N^D}_{PSO}(f_{\bar{\theta}}) \cdot \left[\hat{\theta}_{N^U,N^D} - \theta^*\right] = 0, \quad \text{(B.8)}$$

where $\bar{\theta}$ is located on a line segment between $\hat{\theta}_{N^U,N^D}$ and θ^*. Given the estimation consistency $\hat{\theta}_{N^U,N^D} \xrightarrow{p} \theta^*$, the definition of $\bar{\theta}$ implies $\bar{\theta} \xrightarrow{p} \theta^*$.

The identity in Eq. (B.8) can be rewritten as:

$$\sqrt{N} \cdot \left[\hat{\theta}_{N^U,N^D} - \theta^*\right] = - \left[\nabla_{\theta\theta} \hat{L}_{PSO}^{N^U,N^D}(f_{\bar{\theta}})\right]^{-1} \cdot \sqrt{N} \cdot \nabla_{\theta} \hat{L}_{PSO}^{N^U,N^D}(f_{\theta^*}), \tag{B.9}$$

where we used a notation $N \triangleq N^U + N^D$. Using CLT in Lemma 33 we have:

$$\sqrt{N} \cdot \nabla_{\theta} \hat{L}_{PSO}^{N^U,N^D}(f_{\theta^*}) \xrightarrow{d} \mathcal{N}(0, \mathcal{J}). \tag{B.10}$$

Next, $\nabla_{\theta\theta} \hat{L}_{PSO}^{N^U,N^D}(f_{\bar{\theta}})$ has a form:

$$\nabla_{\theta\theta} \hat{L}_{PSO}^{N^U,N^D}(f_{\bar{\theta}}) = -\frac{1}{N^U} \sum_{i=1}^{N^U} \nabla_{\theta\theta} \widetilde{M}^U \left[X_i^U, f_{\bar{\theta}}(X_i^U)\right] + \frac{1}{N^D} \sum_{i=1}^{N^D} \nabla_{\theta\theta} \widetilde{M}^D \left[X_i^D, f_{\bar{\theta}}(X_i^D)\right]. \tag{B.11}$$

Using the uniform law of large numbers (LLN) and $\bar{\theta} \xrightarrow{p} \theta^*$, we get:

$$\frac{1}{N^U} \sum_{i=1}^{N^U} \nabla_{\theta\theta} \widetilde{M}^U \left[X_i^U, f_{\bar{\theta}}(X_i^U)\right] \xrightarrow{p} \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} \nabla_{\theta\theta} \widetilde{M}^U \left[X, f_{\theta^*}(X)\right], \tag{B.12}$$

$$\frac{1}{N^D} \sum_{i=1}^{N^D} \nabla_{\theta\theta} \widetilde{M}^D \left[X_i^D, f_{\bar{\theta}}(X_i^D)\right] \xrightarrow{p} \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \nabla_{\theta\theta} \widetilde{M}^D \left[X, f_{\theta^*}(X)\right], \tag{B.13}$$

and hence

$$\begin{aligned} \nabla_{\theta\theta} \hat{L}_{PSO}^{N^U,N^D}(f_{\bar{\theta}}) &\xrightarrow{p} - \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} \nabla_{\theta\theta} \widetilde{M}^U \left[X, f_{\theta^*}(X)\right] + \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \nabla_{\theta\theta} \widetilde{M}^D \left[X, f_{\theta^*}(X)\right] = \\ &= \nabla_{\theta\theta} \left[- \mathop{\mathbb{E}}_{X \sim \mathbb{P}^U} \widetilde{M}^U \left[X, f_{\theta^*}(X)\right] + \mathop{\mathbb{E}}_{X \sim \mathbb{P}^D} \widetilde{M}^D \left[X, f_{\theta^*}(X)\right] \right] = \nabla_{\theta\theta} L_{PSO}(f_{\theta^*}) = \mathcal{H}, \end{aligned} \tag{B.14}$$

with $\mathcal{H}$ being defined by Lemma 14. Applying the Continuous Mapping Theorem, we get:

$$\left[\nabla_{\theta\theta} \hat{L}_{PSO}^{N^U,N^D}(f_{\bar{\theta}})\right]^{-1} \xrightarrow{p} \mathcal{H}^{-1}. \tag{B.15}$$

Further, we apply Slutzky theorem on Eqs. (B.10)-(B.15) to get:

$$- \left[\nabla_{\theta\theta} \hat{L}_{PSO}^{N^U,N^D}(f_{\bar{\theta}})\right]^{-1} \cdot \sqrt{N} \cdot \nabla_{\theta} \hat{L}_{PSO}^{N^U,N^D}(f_{\theta^*}) \xrightarrow{d} -\mathcal{H}^{-1} \mathcal{N}(0, \mathcal{J}) = \mathcal{N}(0, \mathcal{H}^{-1} \mathcal{J} \mathcal{H}^{-1}). \tag{B.16}$$

Note that $\min(N^U, N^D) \to \infty$ is required for the convergence (see Lemma 33). This limit is identical to $N \to \infty$ due to assumption that τ is fixed and constant.

Therefore, we get:

$$\sqrt{N} \cdot \left[\hat{\theta}_{N^U,N^D} - \theta^*\right] \xrightarrow{d} \mathcal{N}(0, \mathcal{H}^{-1} \mathcal{J} \mathcal{H}^{-1}), \tag{B.17}$$

where the convergence in distribution is achieved along with $N \to \infty$. Further, all the regulatory assumptions of the theorem are required for application of CLT, LLN, and satisfaction of FOCs.

See theorem 3.1 in [90] for the more technical exposition.

■

APPENDIX C

Proof of Theorem 18

The proof for $df_\theta(X)$'s expected value is trivial. Its covariance is derived as following:

$$\mathbb{E}\left[df_\theta(X)\cdot df_\theta(X')\right] = \delta^2\cdot\mathbb{E}\left[\nabla_\theta f_\theta(X)^T\cdot\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)\cdot\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)^T\cdot\nabla_\theta f_\theta(X')\right] =$$
$$= \delta^2\cdot\nabla_\theta f_\theta(X)^T\cdot\mathbb{E}\left[\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)\cdot\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)^T\right]\cdot\nabla_\theta f_\theta(X'), \quad \text{(C.1)}$$

$$\mathbb{E}\left[df_\theta(X)\right] = -\delta\cdot\nabla_\theta f_\theta(X)^T\cdot\mathbb{E}\left[\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)\right], \quad \text{(C.2)}$$

$$\text{Cov}\left[df_\theta(X), df_\theta(X')\right] = \mathbb{E}\left[df_\theta(X)\cdot df_\theta(X')\right] - \mathbb{E}\left[df_\theta(X)\right]\cdot\mathbb{E}\left[df_\theta(X')\right] =$$
$$= \delta^2\cdot\nabla_\theta f_\theta(X)^T\cdot\mathbb{E}\left[\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)\cdot\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)^T\right]\cdot\nabla_\theta f_\theta(X') -$$
$$-\delta^2\cdot\nabla_\theta f_\theta(X)^T\cdot\mathbb{E}\left[\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)\right]\cdot\mathbb{E}\left[\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)\right]^T\cdot\nabla_\theta f_\theta(X') =$$
$$= \delta^2\cdot\nabla_\theta f_\theta(X)^T\cdot\Bigg[\mathbb{E}\left[\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)\cdot\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)^T\right] -$$
$$-\mathbb{E}\left[\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)\right]\cdot\mathbb{E}\left[\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)\right]^T\Bigg]\cdot\nabla_\theta f_\theta(X') =$$
$$= \delta^2\cdot\nabla_\theta f_\theta(X)^T\cdot\text{Var}\left[\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)\right]\cdot\nabla_\theta f_\theta(X'), \quad \text{(C.3)}$$

where Var $\left[\nabla_\theta\hat{L}_{PSO}^{N^U,N^D}(f_\theta)\right]$ was proven to have a form in Eq. (A.14). Observe that it is proportional to $\frac{1}{N}$ where $N = N^U + N^D$.

■

APPENDIX D

Proof of Theorem 19

Assume that first order conditions (FOCs) were satisfied, $\nabla_\theta \hat{L}_{PSO}^{N^U,N^D}(f_\theta) = 0$. Then we have:

$$\frac{1}{N^U}\sum_{i=1}^{N^U} M^U\left[X_i^U, f_\theta(X_i^U)\right]\cdot\nabla_\theta f_\theta(X_i^U) = \frac{1}{N^D}\sum_{i=1}^{N^D} M^D\left[X_i^D, f_\theta(X_i^D)\right]\cdot\nabla_\theta f_\theta(X_i^D). \quad \text{(D.1)}$$

Consider a specific $\mathbb{P}^U$'s training sample $X \equiv X_j^U$ with $j \in \{1, \dots, N^U\}$. Multiplying the above expression by $\nabla_\theta f_\theta(X)^T$, we get

$$M^U\left[X, f_\theta(X)\right]\cdot g_\theta(X,X) + \sum_{i,i\neq j}^{N^U} M^U\left[X_i^U, f_\theta(X_i^U)\right]\cdot g_\theta(X, X_i^U) =$$
$$= \frac{N^U}{N^D}\sum_{i=1}^{N^D} M^D\left[X_i^D, f_\theta(X_i^D)\right]\cdot g_\theta(X, X_i^D). \quad \text{(D.2)}$$

Kernel g_θ is non-negative due to boundedness assumed in Eq. (7.11). Since M^U is likewise assumed to be non-negative, we obtain an inequality:

$$M^U\left[X, f_\theta(X)\right]\cdot g_\theta(X,X) \leq \frac{N^U}{N^D}\sum_{i=1}^{N^D} M^D\left[X_i^D, f_\theta(X_i^D)\right]\cdot g_\theta(X, X_i^D). \quad \text{(D.3)}$$

Next, we divide by $g_\theta(X, X)$:

$$M^U\left[X, f_\theta(X)\right] \leq \frac{N^U}{N^D}\sum_{i=1}^{N^D} M^D\left[X_i^D, f_\theta(X_i^D)\right]\cdot r_\theta(X, X_i^D) \leq$$
$$\leq \frac{N^U}{N^D}\sum_{i=1}^{N^D} M^D\left[X_i^D, f_\theta(X_i^D)\right]\cdot \exp\left[-\frac{d(X, X_i^D)}{h_{max}}\right] \equiv \alpha, \quad \text{(D.4)}$$

where in the last part we applied the assumed bounds over r_θ.

Denote the inverse function of $M^U\left[X, s\right]$ by $(M^U)^{-1}\left[X, z\right]$. Since M^U is assumed to be strictly decreasing, then so is its inverse $(M^U)^{-1}$. Further, apply $(M^U)^{-1}$ on both sides of Eq. (D.4):

$$(M^U)^{-1}\left[X, M^U\left[X, f_\theta(X)\right]\right] = f_\theta(X) \geq (M^U)^{-1}\left[X, \alpha\right], \tag{D.5}$$

where we reversed the inequality since the applied function is strictly decreasing.

Next, observe that $0 \leq \alpha \leq \infty$ due to the assumed non-negativity of M^D. Further, for $h_{max} \to 0$ we also have $\alpha \to 0$ - for zero bandwidth α goes also to zero. Moreover, due to its properties $(M^U)^{-1}[X, \alpha]$ is strictly decreasing for $\alpha \in [0, \infty]$. Hence, $(M^U)^{-1}[X, \alpha] \to \max_{\alpha' \in [0,\infty]}(M^U)^{-1}[X, \alpha']$ along with $\alpha \to 0$.

Further note that the range of $(M^U)^{-1}$ is the subset of values within $\mathbb{R}$ that $f_\theta(X)$ can have. That is, $(M^U)^{-1}$ and f_θ share their range. Assuming that this range is entire $\mathbb{R}$ or its positive part $\mathbb{R}_{>0}$, extended to contain ∞, we will have $\max_{\alpha' \in [0,\infty]}(M^U)^{-1}[X, \alpha'] = \infty$.

To conclude, we have that $f_\theta(X) \geq (M^U)^{-1}[X, \alpha]$, where for $\alpha \to 0$ this lower bound behaves as $(M^U)^{-1}[X, \alpha] \to \infty$.

■

APPENDIX **E**

Proof of Theorem 20

First, we prove the stepping stone lemmas.

E.1 Lemmata

Lemma 34. *Consider the relative model kernel r_θ defined in Eq. (7.10), and assume it to be bounded as in Eq. (7.11). Then $|r_\theta(X_1, X) - r_\theta(X_2, X)| \leq \epsilon\,[X_1, X_2, X]$ with $\epsilon\,[X_1, X_2, X] \triangleq 1 - \exp\left[-\frac{1}{h_{min}} d(X_1, X_2)\right] \cdot \exp\left[-\frac{1}{h_{min}} \max\left[d(X_1, X), d(X_2, X)\right]\right]$.*

Proof. Consider two scenarios: $r_\theta(X_1, X) \geq r_\theta(X_2, X)$ and $r_\theta(X_1, X) < r_\theta(X_2, X)$. In the first case we have:

$$|r_\theta(X_1, X) - r_\theta(X_2, X)| = r_\theta(X_1, X) - r_\theta(X_2, X) \leq 1 - \exp\left[-\frac{1}{h_{min}} d(X_2, X)\right] \leq$$
$$\leq 1 - \exp\left[-\frac{1}{h_{min}} d(X_1, X_2)\right] \cdot \exp\left[-\frac{1}{h_{min}} d(X_1, X)\right] \triangleq c_1, \quad \text{(E.1)}$$

where in the second row we used a triangle inequality $d(X_2, X) \leq d(X_1, X_2) + d(X_1, X)$.

Similarly, in the second case $r_\theta(X_1, X) < r_\theta(X_2, X)$ we will obtain:

$$|r_\theta(X_1, X) - r_\theta(X_2, X)| = r_\theta(X_2, X) - r_\theta(X_1, X) \leq$$
$$\leq 1 - \exp\left[-\frac{1}{h_{min}} d(X_1, X_2)\right] \cdot \exp\left[-\frac{1}{h_{min}} d(X_2, X)\right] \triangleq c_2. \quad \text{(E.2)}$$

Next, we combine the two cases:

$$|r_\theta(X_1, X) - r_\theta(X_2, X)| \leq \max(c_1, c_2) =$$
$$= 1 - \exp\left[-\frac{1}{h_{min}} d(X_1, X_2)\right] \cdot \exp\left[-\frac{1}{h_{min}} \max\left[d(X_1, X), d(X_2, X)\right]\right]. \quad \text{(E.3)}$$

∎

Lemma 35. *Consider the relative model kernel r_θ defined in Eq. (7.10), and assume it to be bounded as in Eq. (7.11). Then* $\left|\frac{df_\theta(X_1)}{g_\theta(X_1,X_1)} - \frac{df_\theta(X_2)}{g_\theta(X_2,X_2)}\right| \le \varepsilon_1(X_1,X_2)$ *with:*

$$\varepsilon_1(X_1,X_2) = \delta \cdot \left[\frac{1}{N^U}\sum_{i=1}^{N^U} |M^U[X_i^U, f_\theta(X_i^U)]| \cdot \epsilon[X_1,X_2,X_i^U] + \right.$$
$$\left. + \frac{1}{N^D}\sum_{i=1}^{N^D} |M^D[X_i^D, f_\theta(X_i^D)]| \cdot \epsilon[X_1,X_2,X_i^D]\right]. \quad \text{(E.4)}$$

Proof. According to Eq. (7.5) we have:

$$\left|\frac{df_\theta(X_1)}{g_\theta(X_1,X_1)} - \frac{df_\theta(X_2)}{g_\theta(X_2,X_2)}\right| = \delta \cdot \left|\frac{1}{N^U}\sum_{i=1}^{N^U} M^U[X_i^U, f_\theta(X_i^U)] \cdot [r_\theta(X_1,X_i^U) - r_\theta(X_2,X_i^U)] - \right.$$
$$\left. - \frac{1}{N^D}\sum_{i=1}^{N^D} M^D[X_i^D, f_\theta(X_i^D)] \cdot [r_\theta(X_1,X_i^D) - r_\theta(X_2,X_i^D)]\right| \le$$
$$\le \delta \cdot \frac{1}{N^U}\sum_{i=1}^{N^U} |M^U[X_i^U, f_\theta(X_i^U)]| \cdot |r_\theta(X_1,X_i^U) - r_\theta(X_2,X_i^U)| +$$
$$+ \delta \cdot \frac{1}{N^D}\sum_{i=1}^{N^D} |M^D[X_i^D, f_\theta(X_i^D)]| \cdot |r_\theta(X_1,X_i^D) - r_\theta(X_2,X_i^D)| \le$$
$$\le \delta \cdot \frac{1}{N^U}\sum_{i=1}^{N^U} |M^U[X_i^U, f_\theta(X_i^U)]| \cdot \epsilon[X_1,X_2,X_i^U] +$$
$$+ \delta \cdot \frac{1}{N^D}\sum_{i=1}^{N^D} |M^D[X_i^D, f_\theta(X_i^D)]| \cdot \epsilon[X_1,X_2,X_i^D], \quad \text{(E.5)}$$

where in the last part we applied Lemma 34.

■

Lemma 36. *Consider the relative model kernel r_θ defined in Eq. (7.10), and assume it to be bounded as in Eq. (7.11). Then* $\left|\frac{df_\theta(X)}{g_\theta(X,X)}\right| \le \varepsilon_2$ *with:*

$$\varepsilon_2 = \delta \cdot \left[\frac{1}{N^U}\sum_{i=1}^{N^U} |M^U[X_i^U, f_\theta(X_i^U)]| + \frac{1}{N^D}\sum_{i=1}^{N^D} |M^D[X_i^D, f_\theta(X_i^D)]|\right]. \quad \text{(E.6)}$$

Proof.

$$\begin{aligned}
&\left|\frac{df_\theta(X)}{g_\theta(X,X)}\right| = \\
&= \delta \cdot \left| \frac{1}{N^U} \sum_{i=1}^{N^U} M^U\left[X_i^U, f_\theta(X_i^U)\right] \cdot r_\theta(X, X_i^U) - \frac{1}{N^D} \sum_{i=1}^{N^D} M^D\left[X_i^D, f_\theta(X_i^D)\right] \cdot r_\theta(X, X_i^D) \right| \leq \\
&\leq \delta \cdot \frac{1}{N^U} \sum_{i=1}^{N^U} \left|M^U\left[X_i^U, f_\theta(X_i^U)\right]\right| \cdot r_\theta(X, X_i^U) + \delta \cdot \frac{1}{N^D} \sum_{i=1}^{N^D} \left|M^D\left[X_i^D, f_\theta(X_i^D)\right]\right| \cdot r_\theta(X, X_i^D) \leq \\
&\leq \delta \cdot \frac{1}{N^U} \sum_{i=1}^{N^U} \left|M^U\left[X_i^U, f_\theta(X_i^U)\right]\right| + \delta \cdot \frac{1}{N^D} \sum_{i=1}^{N^D} \left|M^D\left[X_i^D, f_\theta(X_i^D)\right]\right|, \quad \text{(E.7)}
\end{aligned}$$

■

where in the last part we used $r_\theta(X, X') \leq 1$.

E.2 Proof of Theorem

Observe that:

$$\begin{aligned}
&\left|\frac{df_\theta(X_1)}{g_\theta(X_1,X_1)} - \frac{df_\theta(X_2)}{g_\theta(X_2,X_2)}\right| = \\
&= \frac{1}{g_\theta(X_1,X_1)} \cdot \left| df_\theta(X_1) - df_\theta(X_2) - \frac{g_\theta(X_1,X_1) - g_\theta(X_2,X_2)}{g_\theta(X_2,X_2)} \cdot df_\theta(X_2) \right|. \quad \text{(E.8)}
\end{aligned}$$

Applying Lemma 35, we have:

$$\left| df_\theta(X_1) - df_\theta(X_2) - \frac{g_\theta(X_1,X_1) - g_\theta(X_2,X_2)}{g_\theta(X_2,X_2)} \cdot df_\theta(X_2) \right| \leq g_\theta(X_1,X_1) \cdot \varepsilon_1(X_1,X_2). \quad \text{(E.9)}$$

Using the reverse triangle inequality $||x| - |y|| \leq |x - y|$ on left part of the above equation, we get:

$$\left| |df_\theta(X_1) - df_\theta(X_2)| - \left| \frac{g_\theta(X_1,X_1) - g_\theta(X_2,X_2)}{g_\theta(X_2,X_2)} \cdot df_\theta(X_2) \right| \right| \leq g_\theta(X_1,X_1) \cdot \varepsilon_1(X_1,X_2). \quad \text{(E.10)}$$

Next, we check the above inequality under two possible scenarios:

1) $|df_\theta(X_1) - df_\theta(X_2)| \geq \left| \frac{g_\theta(X_1,X_1) - g_\theta(X_2,X_2)}{g_\theta(X_2,X_2)} \cdot df_\theta(X_2) \right|$**:** Here we have:

$$|df_\theta(X_1) - df_\theta(X_2)| \leq g_\theta(X_1,X_1) \cdot \varepsilon_1(X_1,X_2) + \left| \frac{g_\theta(X_1,X_1) - g_\theta(X_2,X_2)}{g_\theta(X_2,X_2)} \cdot df_\theta(X_2) \right|. \quad \text{(E.11)}$$

2) $|df_\theta(X_1) - df_\theta(X_2)| < \left| \frac{g_\theta(X_1,X_1) - g_\theta(X_2,X_2)}{g_\theta(X_2,X_2)} \cdot df_\theta(X_2) \right|$**:** In such case Eq. (E.11) is trivially satisfied.

Hence, Eq. (E.11) is satisfied always and therefore:

$$|df_\theta(X_1) - df_\theta(X_2)| \leq g_\theta(X_1, X_1) \cdot \varepsilon_1(X_1, X_2) + |g_\theta(X_1, X_1) - g_\theta(X_2, X_2)| \cdot \left| \frac{df_\theta(X_2)}{g_\theta(X_2, X_2)} \right| \leq$$
$$\leq g_\theta(X_1, X_1) \cdot \varepsilon_1(X_1, X_2) + |g_\theta(X_1, X_1) - g_\theta(X_2, X_2)| \cdot \varepsilon_2 \equiv \varepsilon_3(X_1, X_2), \quad \text{(E.12)}$$

where we applied Lemma 36.

Further, using definitions of ε_1 and ε_2 we get:

$$\varepsilon_3(X_1, X_2) = g_\theta(X_1, X_1) \cdot \delta \cdot \left[\frac{1}{N^U} \sum_{i=1}^{N^U} |M^U [X_i^U, f_\theta(X_i^U)]| \cdot \epsilon [X_1, X_2, X_i^U] + \right.$$
$$\left. + \frac{1}{N^D} \sum_{i=1}^{N^D} |M^D [X_i^D, f_\theta(X_i^D)]| \cdot \epsilon [X_1, X_2, X_i^D] \right] +$$
$$+ |g_\theta(X_1, X_1) - g_\theta(X_2, X_2)| \cdot \delta \cdot \left[\frac{1}{N^U} \sum_{i=1}^{N^U} |M^U [X_i^U, f_\theta(X_i^U)]| + \frac{1}{N^D} \sum_{i=1}^{N^D} |M^D [X_i^D, f_\theta(X_i^D)]| \right] =$$
$$= \delta \cdot g_\theta(X_1, X_1) \cdot \left[\frac{1}{N^U} \sum_{i=1}^{N^U} |M^U [X_i^U, f_\theta(X_i^U)]| \cdot \left[\epsilon [X_1, X_2, X_i^U] + \frac{|g_\theta(X_1, X_1) - g_\theta(X_2, X_2)|}{g_\theta(X_1, X_1)} \right] + \right.$$
$$\left. + \frac{1}{N^D} \sum_{i=1}^{N^D} |M^D [X_i^D, f_\theta(X_i^D)]| \cdot \left[\epsilon [X_1, X_2, X_i^D] + \frac{|g_\theta(X_1, X_1) - g_\theta(X_2, X_2)|}{g_\theta(X_1, X_1)} \right] \right]. \quad \text{(E.13)}$$

Define $\nu_\theta(X_1, X_2, X) \triangleq \epsilon [X_1, X_2, X] + \frac{|g_\theta(X_1,X_1) - g_\theta(X_2,X_2)|}{g_\theta(X_1,X_1)}$. Then, the combination of Eq. (E.12) and Eq. (E.13) will lead to:

$$|df_\theta(X_1) - df_\theta(X_2)| \leq \delta \cdot g_\theta(X_1, X_1) \cdot \left[\frac{1}{N^U} \sum_{i=1}^{N^U} |M^U [X_i^U, f_\theta(X_i^U)]| \cdot \nu_\theta(X_1, X_2, X_i^U) + \right.$$
$$\left. + \frac{1}{N^D} \sum_{i=1}^{N^D} |M^D [X_i^D, f_\theta(X_i^D)]| \cdot \nu_\theta(X_1, X_2, X_i^D) \right]. \quad \text{(E.14)}$$

■

APPENDIX **F**

Proof of Softmax Cross-Entropy being Instance of PSO

Here we will derive the cross-entropy loss combined with a $Softmax$ layer, typically used in the image classification domain, via PSO principles, showing it to be another instance of PSO. For this we define our training dataset as a set of pairs $\{X_i, Y_i\}_{i=1}^N$ where $X_i \in \mathbb{R}^n$ is a data point of an arbitrary dimension n (e.g. image) and Y_i is its label - a discrete number that takes values from $\{1, \ldots, C\}$ with C being the number of classes. Number of samples for each class is denoted by $\{N_1, \ldots, N_C\}$, with $\sum_{j=1}^C N_j = N$. For the classification task we assume that each sample pair is i.i.d. sampled from an unknown density $\mathbb{P}(X, Y) = \mathbb{P}(X) \cdot \mathbb{P}(Y|X)$. Our goal is to enforce the output of $Softmax$ layer to converge to the unknown conditional $\mathbb{P}(Y|X)$. To this end, define a model f_θ that returns C dimensional output $f_\theta(X) \in \mathbb{R}^C$, with its j-th entry denoted by $f_{\theta,j}(X)$. Further, $Softmax$ transformation $h_\theta(X)$ is defined as:

$$h_{\theta,j}(X) = \frac{\exp f_{\theta,j}(X)}{\sum_{k=1}^C \exp f_{\theta,k}(X)} = \frac{\exp f_{\theta,j}(X)}{\|\exp f_\theta(X)\|_1}, \tag{F.1}$$

which yields properties $h_{\theta,j}(X) \geq 0$ and $\sum_k h_{\theta,k}(X) = 1$. We aim for $h_{\theta,j}(X)$ to converge to $\mathbb{P}(Y = j|X)$ - the probability of X's label to be j. Each $f_{\theta,j}(X)$ will be considered as an independent surface in PSO framework, which we will push to the equilibrium where

$$\frac{\exp f_{\theta,j}(X)}{\|\exp f_\theta(X)\|_1} = \mathbb{P}(Y = j|X), \tag{F.2}$$

by optimizing the corresponding loss $L_{PSO}^j(f_{\theta,j})$, with total minimized loss being defined as $L_{PSO}(f_\theta) = \sum_{j=1}^C L_{PSO}^j(f_{\theta,j})$. That is, the described below minimization of $L_{PSO}(f_\theta)$ will consist of solving C PSO problems in parallel.

PSO over $f_{\theta,j}(X)$ via $L_{PSO}^j(f_{\theta,j})$: Consider a typical PSO estimation, where $\mathbb{P}(X|Y = j)$ serves as *up* density $\mathbb{P}^U$ and $\mathbb{P}(X|Y \neq j)$ - as *down* density $\mathbb{P}^D$. Sample batch from $\mathbb{P}(X|Y = j)$

is obtained by fetching samples with label $Y = j$; data points from $\mathbb{P}(X|Y \neq j)$ will be the rest of samples. Note also that the identity $\frac{\mathbb{P}(X|Y=j)}{\mathbb{P}(X|Y\neq j)} = \frac{\mathbb{P}(Y\neq j)}{\mathbb{P}(Y=j)} \cdot \frac{\mathbb{P}(Y=j|X)}{1-\mathbb{P}(Y=j|X)}$ holds, due to below derivation:

$$\frac{\mathbb{P}(X|Y \neq j)}{\mathbb{P}(X|Y = j)} = \frac{\mathbb{P}(X, Y \neq j) \cdot \mathbb{P}(Y = j)}{\mathbb{P}(X, Y = j) \cdot \mathbb{P}(Y \neq j)} = \frac{\mathbb{P}(Y = j)}{\mathbb{P}(Y \neq j)} \cdot \frac{\sum_{k \neq j} \mathbb{P}(X, Y = k)}{\mathbb{P}(X, Y = j)} =$$
$$= \frac{\mathbb{P}(Y = j)}{\mathbb{P}(Y \neq j)} \cdot \frac{\left[\sum_{k=1}^{C} \mathbb{P}(X, Y = k)\right] - \mathbb{P}(X, Y = j)}{\mathbb{P}(X, Y = j)} = \frac{\mathbb{P}(Y = j)}{\mathbb{P}(Y \neq j)} \cdot \left[\frac{\mathbb{P}(X)}{\mathbb{P}(X, Y = j)} - 1\right] =$$
$$= \frac{\mathbb{P}(Y = j)}{\mathbb{P}(Y \neq j)} \cdot \frac{1 - \mathbb{P}(Y = j|X)}{\mathbb{P}(Y = j|X)}. \quad \text{(F.3)}$$

Considering PSO *balance state*, we are looking for a pair of *magnitudes* $\{M_j^U, M_j^D\}$ that for the below system:

$$\frac{M_j^D\left[X, f_\theta(X)\right]}{M_j^U\left[X, f_\theta(X)\right]} = \frac{\mathbb{P}(X|Y = j)}{\mathbb{P}(X|Y \neq j)}, \quad \text{(F.4)}$$

will produce a solution at Eq. (F.2). That is, denoting $\frac{M_j^D[X,s]}{M_j^U[X,s]}$ by $R(X, s) : \mathbb{R}^n \times \mathbb{R}^C \to \mathbb{R}$ and using the identity in Eq. (F.3), we are looking for the transformation R s.t. the solution $f_\theta(X)$ of:

$$R\left[X, f_\theta(X)\right] = \frac{\mathbb{P}(Y \neq j)}{\mathbb{P}(Y = j)} \cdot \frac{\mathbb{P}(Y = j|X)}{1 - \mathbb{P}(Y = j|X)}, \quad \text{(F.5)}$$

will satisfy Eq. (F.2). Assuming that R has a form $R\left[X, f_\theta(X)\right] = \bar{R}\left[\frac{\exp f_{\theta,j}(X)}{\|\exp f_\theta(X)\|_1}\right]$, the above is equivalent to find the transformation $\bar{R}(s) : \mathbb{R} \to \mathbb{R}$ s.t. the solution s of a system:

$$\bar{R}(s) = \frac{\mathbb{P}(Y \neq j)}{\mathbb{P}(Y = j)} \cdot \frac{\mathbb{P}(Y = j|X)}{1 - \mathbb{P}(Y = j|X)} \quad \text{(F.6)}$$

is $\mathbb{P}(Y = j|X)$. Thus, it can be easily identified as $\bar{R}(s) = \frac{\mathbb{P}(Y\neq j)}{\mathbb{P}(Y=j)} \cdot \frac{s}{1-s}$. From this we conclude that $R\left[X, f_\theta(X)\right] = \frac{\mathbb{P}(Y\neq j)}{\mathbb{P}(Y=j)} \cdot \frac{\exp f_{\theta,j}(X)}{\|\exp f_\theta(X)\|_1 - \exp f_{\theta,j}(X)}$ and that *magnitudes* must satisfy:

$$\frac{M_j^D\left[X, f_\theta(X)\right]}{M_j^U\left[X, f_\theta(X)\right]} = \frac{\mathbb{P}(Y \neq j)}{\mathbb{P}(Y = j)} \cdot \frac{\exp f_{\theta,j}(X)}{\|\exp f_\theta(X)\|_1 - \exp f_{\theta,j}(X)}. \quad \text{(F.7)}$$

Specifically, we will choose them to be:

$$M_j^U\left[X, f_\theta(X)\right] = \mathbb{P}(Y = j) \cdot \frac{\|\exp f_\theta(X)\|_1 - \exp f_{\theta,j}(X)}{\|\exp f_\theta(X)\|_1} = \mathbb{P}(Y = j) \cdot \frac{\sum_{k=1,k\neq j}^{C} \exp f_{\theta,k}(X)}{\|\exp f_\theta(X)\|_1}, \quad \text{(F.8)}$$

$$M_j^D\left[X, f_\theta(X)\right] = \mathbb{P}(Y \neq j) \cdot \frac{\exp f_{\theta,j}(X)}{\|\exp f_\theta(X)\|_1} \quad \text{(F.9)}$$

where the denominator $\|\exp f_\theta(X)\|_1$ serves as a normalization factor that enforces $\{M_j^U, M_j^D\}$ to be between 0 and 1. Such normalization is only one from many possible, yet this choice will eventually yield the popular softmax cross-entropy loss.

Using the above setting to define $L^j_{PSO}(f_{\theta,j})$, its gradient can be written as

$$\nabla_\theta L^j_{PSO}(f_{\theta,j}) = - \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X|Y=j)} M^U_j \left[X, f_\theta(X)\right] \cdot \nabla_\theta f_{\theta,j}(X) + \\ + \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X|Y \neq j)} M^D_j \left[X, f_\theta(X)\right] \cdot \nabla_\theta f_{\theta,j}(X). \quad \text{(F.10)}$$

Gradient-based optimization via the above expression will lead to Eq. (F.2). Also, in practice $\mathbb{P}(Y = j)$ inside M^U_j can be approximated as $\frac{N_j}{N}$, and $\mathbb{P}(Y \neq j)$ inside M^D_j - as $\frac{N-N_j}{N}$.

PSO over multiple surfaces via $L_{PSO}(f_\theta)$: Further, combining all losses together into $L_{PSO}(f_\theta) = \sum_{j=1}^C L^j_{PSO}(f_{\theta,j})$ will produce $\nabla_\theta L_{PSO}(f_\theta) = \sum_{j=1}^C \nabla_\theta L^j_{PSO}(f_{\theta,j})$:

$$\nabla_\theta L_{PSO}(f_\theta) = \sum_{j=1}^C \Bigg[- \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X|Y=j)} M^U_j \left[X, f_\theta(X)\right] \cdot \nabla_\theta f_{\theta,j}(X) + \\ + \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X|Y \neq j)} M^D_j \left[X, f_\theta(X)\right] \cdot \nabla_\theta f_{\theta,j}(X) \Bigg]. \quad \text{(F.11)}$$

The above expression is also the gradient of softmax cross-entropy, which can be shown as follows. First, note that the second term can be rewritten as:

$$\mathop{\mathbb{E}}_{X \sim \mathbb{P}(X|Y \neq j)} M^D_j \left[X, f_\theta(X)\right] \cdot \nabla_\theta f_{\theta,j}(X) = \\ = \int \mathbb{P}(X, Y \neq j) \cdot \frac{\exp f_{\theta,j}(X)}{\left\| \exp f_\theta(X) \right\|_1} \cdot \nabla_\theta f_{\theta,j}(X) dX = \\ = \int \left[\sum_{k=1,k\neq j}^C \mathbb{P}(X, Y = k) \right] \cdot \frac{\exp f_{\theta,j}(X)}{\left\| \exp f_\theta(X) \right\|_1} \cdot \nabla_\theta f_{\theta,j}(X) dX = \\ = \sum_{k=1,k\neq j}^C \mathbb{P}(Y = k) \cdot \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X|Y=k)} \frac{\exp f_{\theta,j}(X)}{\left\| \exp f_\theta(X) \right\|_1} \cdot \nabla_\theta f_{\theta,j}(X). \quad \text{(F.12)}$$

Then, $\nabla_\theta L_{PSO}(f_\theta)$ is equal to:

$$\nabla_\theta L_{PSO}(f_\theta) = \sum_{j=1}^C \Bigg[- \mathbb{P}(Y = j) \cdot \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X|Y=j)} \frac{\sum_{k=1,k\neq j}^C \exp f_{\theta,k}(X)}{\left\| \exp f_\theta(X) \right\|_1} \cdot \nabla_\theta f_{\theta,j}(X) + \\ + \sum_{k=1,k\neq j}^C \mathbb{P}(Y = k) \cdot \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X|Y=k)} \frac{\exp f_{\theta,j}(X)}{\left\| \exp f_\theta(X) \right\|_1} \cdot \nabla_\theta f_{\theta,j}(X) \Bigg] = \\ = - \sum_{j=1}^C \mathbb{P}(Y = j) \cdot \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X|Y=j)} \frac{\sum_{k=1,k\neq j}^C \exp f_{\theta,k}(X)}{\left\| \exp f_\theta(X) \right\|_1} \cdot \nabla_\theta f_{\theta,j}(X) + \\ + \sum_{j=1}^C \mathbb{P}(Y = j) \cdot \mathop{\mathbb{E}}_{X \sim \mathbb{P}(X|Y=j)} \sum_{k=1,k\neq j}^C \frac{\exp f_{\theta,k}(X)}{\left\| \exp f_\theta(X) \right\|_1} \cdot \nabla_\theta f_{\theta,k}(X), \quad \text{(F.13)}$$

where the second equality can be verified by examining coefficients of each $\nabla_\theta f_{\theta,j}(X)$ before and after "=". Further:

$$\nabla_\theta L_{PSO}(f_\theta) = -\sum_{j=1}^{C} \mathbb{P}(Y=j) \cdot \mathop{\mathbb{E}}_{X\sim\mathbb{P}(X|Y=j)} \left[\frac{\|\exp f_\theta(X)\|_1 - \exp f_{\theta,j}(X)}{\|\exp f_\theta(X)\|_1} \cdot \nabla_\theta f_{\theta,j}(X) - \right.$$
$$\left. - \sum_{k=1,k\neq j}^{C} \frac{\exp f_{\theta,k}(X)}{\|\exp f_\theta(X)\|_1} \cdot \nabla_\theta f_{\theta,k}(X) \right], \quad \text{(F.14)}$$

with the expression in brackets being derivative of $\log \frac{\exp f_{\theta,j}(X)}{\|\exp f_\theta(X)\|_1}$ w.r.t. θ.

Concluding from above, $L_{PSO}(f_\theta)$ with the above gradient can be written as:

$$L_{PSO}(f_\theta) = -\sum_{j=1}^{C} \mathbb{P}(Y=j) \cdot \mathop{\mathbb{E}}_{X\sim\mathbb{P}(X|Y=j)} \log \frac{\exp f_{\theta,j}(X)}{\|\exp f_\theta(X)\|_1} =$$
$$= -\int \sum_{j=1}^{C} \left[\mathbb{P}(X, Y=j) \cdot \log \frac{\exp f_{\theta,j}(X)}{\|\exp f_\theta(X)\|_1} \right] dX =$$
$$= -\mathop{\mathbb{E}}_{X,Y\sim\mathbb{P}(X,Y)} \log \frac{\exp f_{\theta,Y}(X)}{\|\exp f_\theta(X)\|_1}. \quad \text{(F.15)}$$

with its empirical version being:

$$L_{PSO}(f_\theta) \approx -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp f_{\theta,Y_i}(X_i)}{\|\exp f_\theta(X_i)\|_1}. \quad \text{(F.16)}$$

The above loss is known in Machine Learning community as softmax cross-entropy loss. Therefore, we can conclude that PSO over multiple surfaces $\{f_{\theta,j}(X)\}_{j=1}^{C}$ with *magnitudes* in Eqs. (F.8)-(F.9) corresponds to cross-entropy when $\mathbb{P}(X|Y=j)$ and $\mathbb{P}(X|Y\neq j)$ serve as *up* and *down* densities respectively. Yet, according to the PSO principles the *magnitudes* in Eqs. (F.8)-(F.9) are not the only choice for such convergence. In fact, we can change the norm within the denominator of $M^U(\cdot)$ and $M^D(\cdot)$ to any L-p norm, since the denominator term is eventually canceled out and since its actual role is to bound outputs of *magnitude* functions. Similarly to what we observed in our experiments about the PSO-LDE (see Sections 8.2 and 13), different norms (the α value in context of PSO-LDE) can have smoother dynamics and produce a smaller approximation error.

■

APPENDIX G

Differential approximation

In this section we will empirically justify our approximation in Eq. (7.5), where we assumed that the surface *differential*, caused by GD update of weights θ, can be approximated via its first-order Taylor expansion.

For this purpose we performed a single iteration of GD optimization and measured the real and the estimated *differentials* at train and test points as following. First, points $D = \{X_i\}_{i=1}^{2000}$ were sampled from $\mathbb{P}^U$ density, which is *Columns* distribution from Section 13.2, where $X_i \in \mathbb{R}^n$ with $n = 20$. Further, we performed a single GD iteration of the following loss:

$$L(\theta, D) = -\frac{1}{1000} \sum_{i=1}^{1000} f_\theta(X_i) + \frac{1}{1000} \sum_{i=1001}^{2000} f_\theta(X_i), \tag{G.1}$$

where $f_\theta(X)$ is a FC network depicted in Figure 11.1a, with overall 4 layers of size 1024 each. Next, we measured the surface height $f_\theta(X)$ at two points X_{train} and X_{test} before and after GD update, where $X_{train} \in D$ and $X_{test} \notin D$. We performed this procedure for a range of learning rate values and thus obtained the real *differential* $df_\theta(X)$ at X_{train} and X_{test} as a function of δ (see Figure G.1). Likewise, we calculated the *approximated differential* $\bar{df}(X)$ at X_{train} and X_{test} using first-order Taylor expansion as:

$$\begin{aligned} \bar{df}(X) &= \frac{\delta}{1000} \cdot \nabla_\theta f_\theta(X)^T \cdot \Big[\sum_{i=1}^{1000} \nabla_\theta f_\theta(X_i) - \sum_{i=1001}^{2000} \nabla_\theta f_\theta(X_i) \Big] = \\ &= \frac{\delta}{1000} \cdot \Big[\sum_{i=1}^{1000} g_\theta(X, X_i) - \sum_{i=1001}^{2000} g_\theta(X, X_i) \Big], \end{aligned} \tag{G.2}$$

where θ is taken at time before GD update.

In Figures G.1a and G.1b we can see the calculated *differentials* for both X_{train} and X_{test}, respectively. In both figures the real *differential* (blue line) and the estimated *differential* (red line) become very close to each other for $\delta < 0.01$. Note that for the most part of a typical optimization process δ satisfies this criteria.

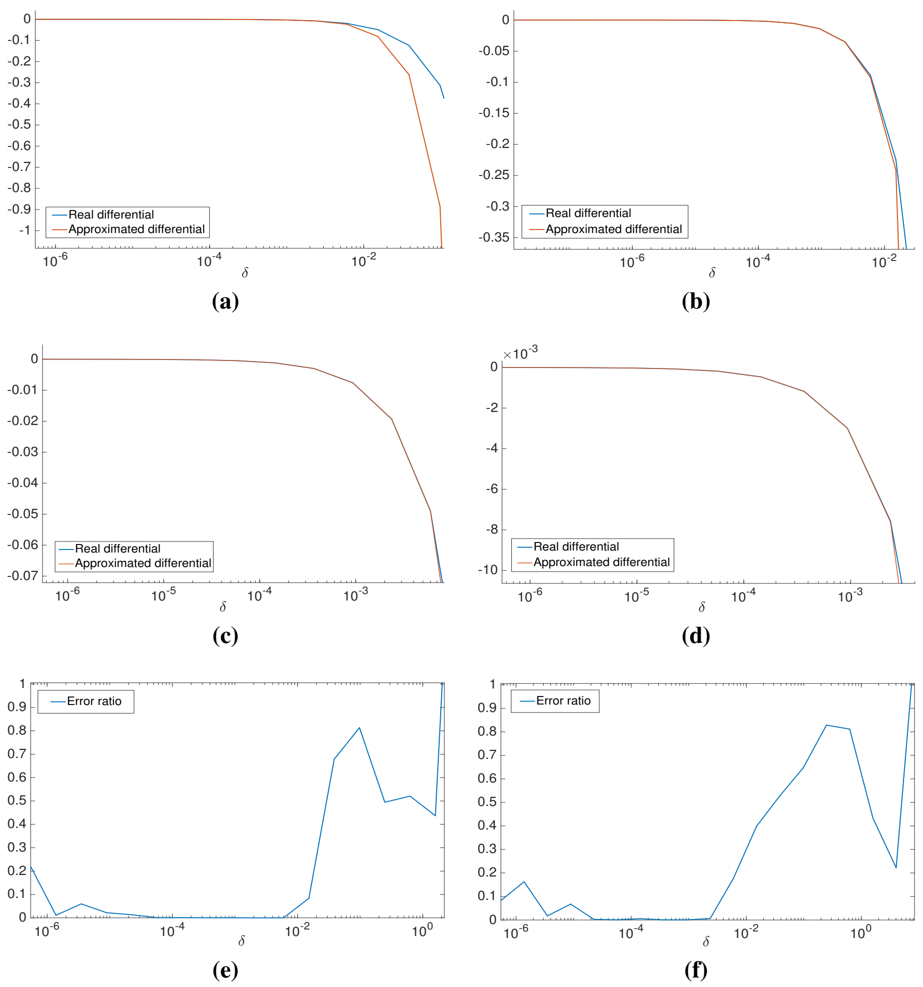


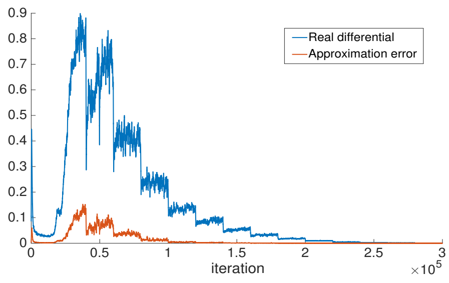
Figure G.1: Real and approximated *differentials* for the training point X_{train} (a)-(c)-(e) and the testing point X_{test} (b)-(d)-(f). (a)-(b) Real *differential* (blue line) vs approximated *differential* (red line), as a function of the learning rate δ; (c)-(d) Zoom-in of (a)-(b); (e)-(f) Error ratio, defined in Eq. (G.3).

Further, in Figures G.1e (for X_{train}) and G.1f (for X_{test}) we can see the ratio:

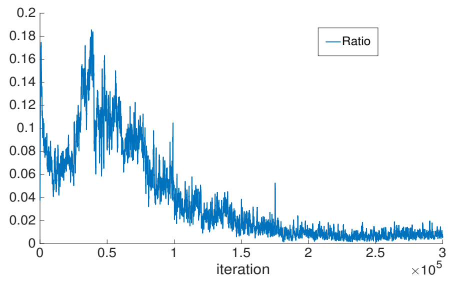
$$ratio = \left| df_\theta(X) - \bar{df}(X) \right| / \left| df_\theta(X) \right| , \tag{G.3}$$

which expresses an error $\left| df_\theta(X) - \bar{df}(X) \right|$ as the percentage from the real *differential*. As can be seen, for both X_{train} and X_{test} the error ratio is very low for $\delta < 0.01$ (under 10% for most part). Additionally, the error ratio slightly increases for a very small δ (around 10^{-6}). We speculate this to be a precision artifact, since the calculation of an approximated *differential* in Eq. (G.2) was done in a single-precision floating-point format (float32) and involved the multiplication by a very small number δ.

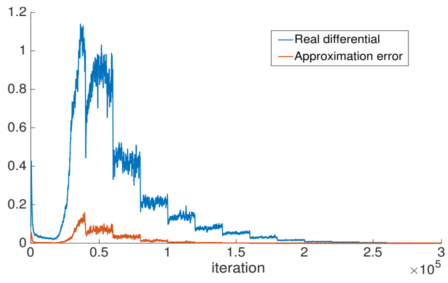
Additionally, we calculated the real and approximated *differentials* along the entire GD optimization process of PSO-LDE, where the same NN architecture was used as in the first experiment, and where the pdf inference was applied to *Columns* distribution from Section 13.2. Particularly, we trained a NN for 300000 iterations, while during each iteration we



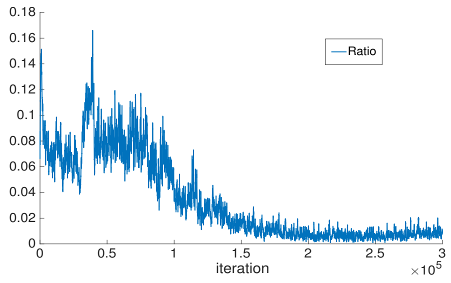
(a)



(b)



(c)



(d)

Figure G.2: The real *differential* and the approximation error during the training of PSO-LDE for two different testing points X_1 and X_2. (a)-(b) Results for X_1. (c)-(d) Results for X_2. (a)-(c) Blue line is an absolute value of the real *differential* at a specific point for each iteration time, smoothed via a moving mean with the window size 300; red line is the absolute value of a difference between the real *differential* and the approximated one, smoothed via a moving mean of the same window size. (b)-(d) Ratio between two lines in (a)-(c), can be seen as a moving mean version of Eq. (G.3) - an error as the percentage of the real *differential*.

computed the real and the approximated *differentials* for a specific test point X. We performed such simulation twice, for two different points and plotted their *differentials* in Figure G.2. In the left column, the blue line is the absolute value of the real *differential*, $|df_\theta(X)|$, and the red line is error $\left|df_\theta(X) - \bar{df}(X)\right|$, both smoothed via moving mean with window size 300. The right column shows the ratio between smoothed $\left|df_\theta(X) - \bar{df}(X)\right|$ and smoothed $|df_\theta(X)|$, $\left|df_\theta(X) - \bar{df}(X)\right| / |df_\theta(X)|$, which can be seen as the error percentage from the real *differential*. As shown in Figures G.2b and G.2d, this error percentage is less than 15% and for most part of the training is even lower. This trend is the same for both verified points, suggesting that the real *differential* indeed can be approximated very closely by the first-order Taylor expansion. In overall, above we showed that most of the surface change can be explained by the *gradient similarity* $g_\theta(X, X')$ in Eq. (7.5).

APPENDIX **H**

Weights Uncorrelation and Gradient Similarity Space

In this appendix we empirically demonstrate the relation between *gradient similarity* $g_\theta(X, X') = \nabla_\theta f_\theta(X)^T \cdot \nabla_\theta f_\theta(X')$ and Euclidean distance $d(X, X')$, and show how this relation changes along the optimization over NNs. Particularly, we observe empirically that during first several thousand iterations of a typical optimization the trend is achieved where high values of $g_\theta(X, X')$ are correlated with small values of $d(X, X')$ - the model kernel of NN obtains a local-support structure. Further, this trend is preserved during the rest part of the optimization. This behavior can be seen as an another motivation for the kernel bandwidth analysis made in Section 7.5 - the shape of $r_\theta(X, X') = \frac{g_\theta(X,X')}{g_\theta(X,X)}$ has some implicit particular bandwidth.

Global Evaluation We apply PSO-LDE with $\alpha = \frac{1}{4}$ on a BD model for the inference of *Columns* distribution defined in Eq. (13.1), where at different optimization iterations we plot output pairs of $g_\theta(X, X')$ and $d(X, X')$. The plots are constructed similarly to Figure 11.2b. Specifically, we sample 500 points $D^U = \{X_i^U\}$ and 500 points $D^D = \{X_i^D\}$ from $\mathbb{P}^U$ and $\mathbb{P}^D$ respectively. For each sample from $D = D^U \cup D^D$ we calculate the gradient $\nabla_\theta f_\theta(X)$. Further we compute Euclidean distance and the *gradient similarity* between every two points within D, producing $\frac{1000 \cdot 1001}{2}$ pairs of distance and similarity values. These values are plotted in Figure H.1.

Also, we compute a *relative* side-influence $r_\theta(X, X')$, defined in Eq. (7.10), for each pair of points in D. In Figure H.2 we construct a histogram of 10^6 obtained pairs $\{r_\theta(X_i, X_j), d(X_i, X_j)\}$.

As seen from the above figures, during first iterations the *gradient similarity* obtains a form where its values are monotonically decreasing with bigger Euclidean distance between the points. During next optimization iterations the self similarity $g_\theta(X, X)$ is growing by several orders of magnitudes. At the same time the side-similarity $g_\theta(X', X)$ for $X' \neq X$ is growing significantly slower and mostly stays centered around zero. In overall values of $g_\theta(X, X)$ are much higher than values of $g_\theta(X', X)$ for $X' \neq X$, implying that the model kernel has mostly a local influence/impact. Likewise, from Figure H.2 it is also clear that $r_\theta(X, X')$ for faraway

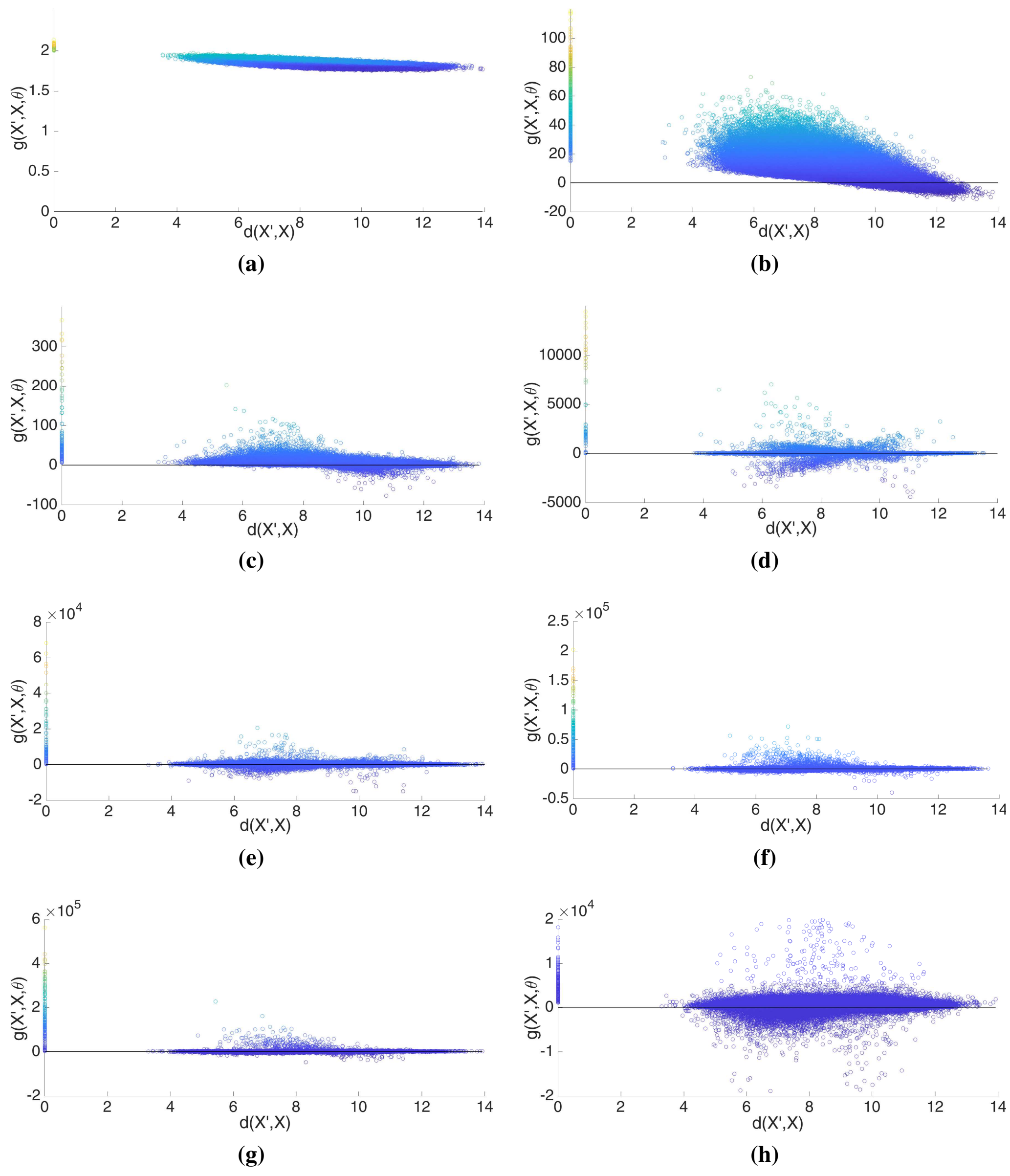


Figure H.1: Relation between values of *gradient similarity* $g_\theta(X', X)$ and values of Euclidean distance $d(X', X)$, along the optimization time t (iteration index). PSO-LDE with $\alpha = \frac{1}{4}$ is applied, where NN architecture is block-diagonal with 6 layers, number of blocks $N_B = 50$ and block size $S_B = 64$ (see Section 11.1). Each plot is constructed similarly to Figure 11.2b, see the main text for more details. Outputs from both $g_\theta(X', X)$ and $d(X', X)$ are demonstrated at different times; (a) $t = 0$, (b) $t = 100$, (c) $t = 3200$, (d) $t = 3400$, (e) $t = 6000$, (f) $t = 100000$ and (g) $t = 200000$. (h) Zoom-in of (g). As can be seen, self similarities $g_\theta(X, X)$, depicted at $d(X', X) = 0$, are high and increase during the optimization. The *side* similarities $g_\theta(X', X)$ for $X' \neq X$, depicted at $d(X', X) > 0$, are centered around zero at $t = 200000$ and are significantly lower than self similarities.

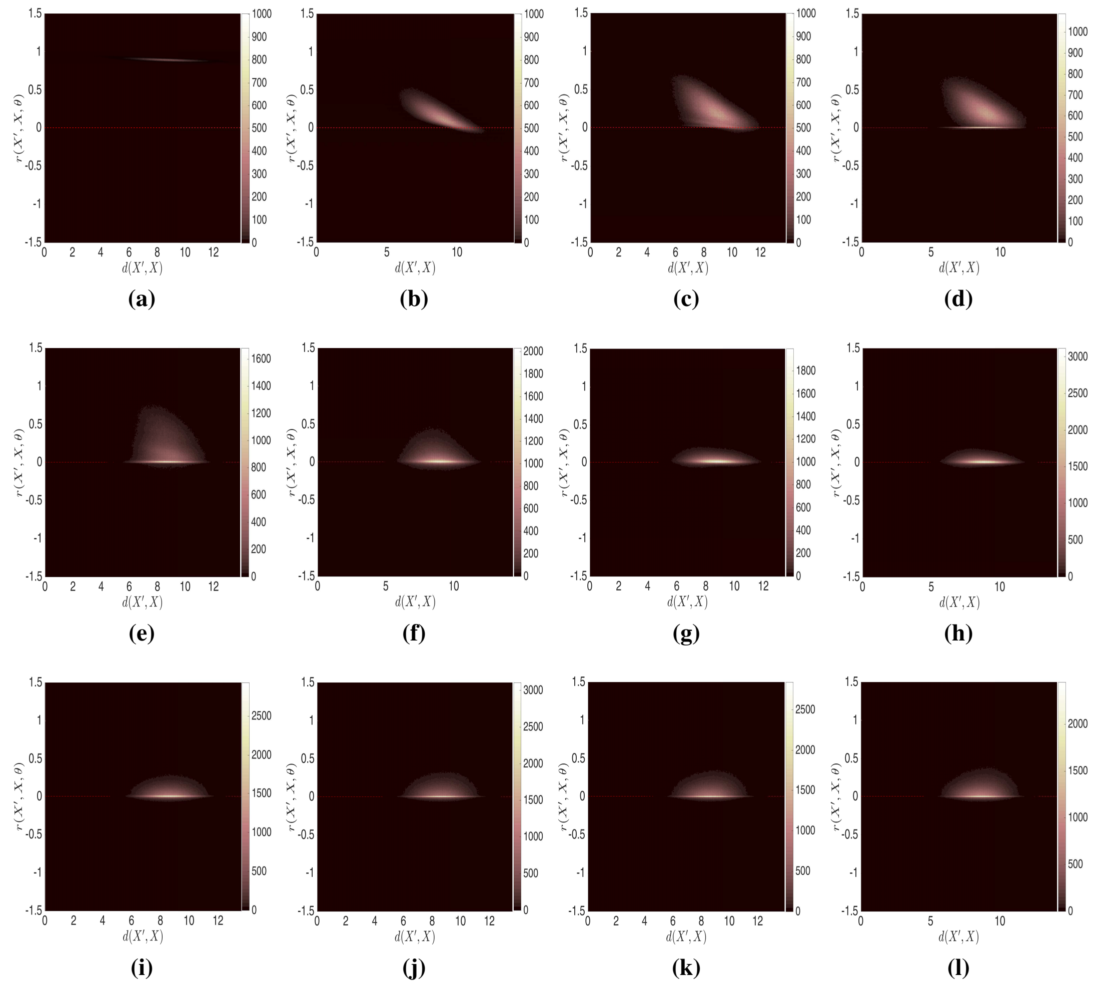
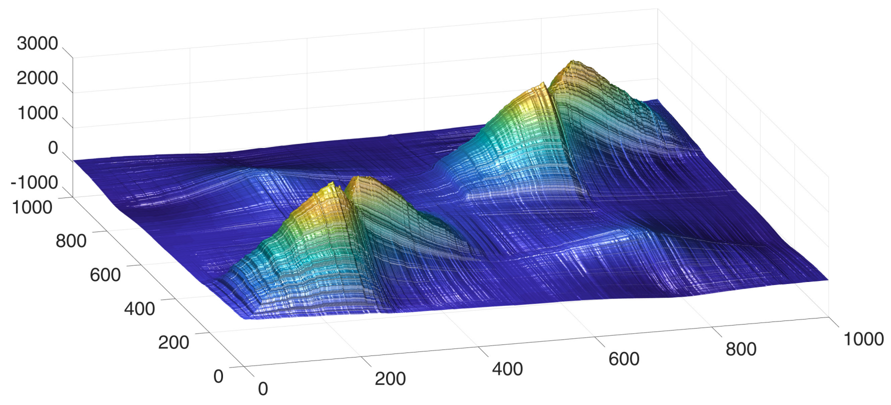


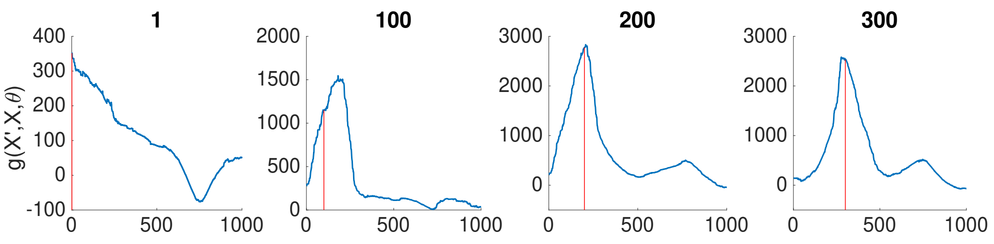
Figure H.2: Histograms of the relative *gradient similarity* $r_\theta(X', X)$ and the Euclidean distance $d(X', X)$ at different optimization times t, for the experiment in Figure H.1. At each time t we calculate a *relative* side-influence $r_\theta(X_i, X_j)$ and a Euclidean distance $d(X_i, X_j)$ for 10^6 point pairs and depict a histogram of obtained $\{r_\theta(X_i, X_j)\}$ and $\{d(X_i, X_j)\}$. The optimization time is (a) $t = 0$, (b) $t = 100$, (c) $t = 3200$, (d) $t = 3400$, (e) $t = 4000$, (f) $t = 6000$, (g) $t = 10000$, (h) $t = 29000$, (i) $t = 100000$, (j) $t = 150000$, (k) $t = 200000$ and (l) $t = 300000$. As observed, after $t = 10000$ the relative *gradient similarity* between far away regions is much smaller than 1, implying that there is a insignificant side-influence over height $f_\theta(X)$ at point X from other points that are far away from X.

points X and X' is near-zero during the most part of the optimization process (i.e. after 10000 iterations in this experiment). Thus, the corresponding bandwidth of r_θ can be bounded similarly to Eq. (7.11).

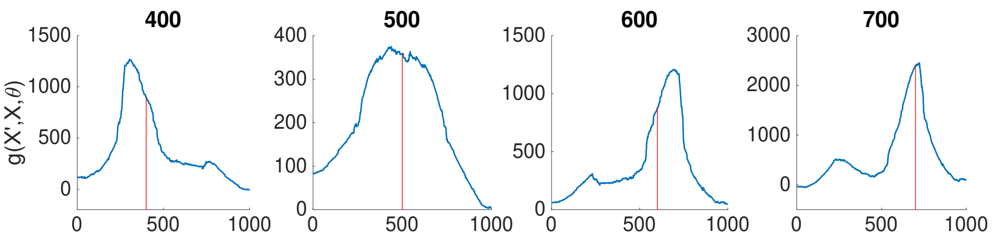
Local Evaluation Additionally, we performed a local evaluation of the above relation between *gradient similarity* and Euclidean distance. Particularly, after convergence we consider a path within a 20D input space, that starts at $S = [-1, \ldots, -1]$ and ends at $E = [1, \ldots, 1]$. We evenly discretized this path into 1000 middle points with which we form an ordered point set $D = \{S, \ldots, E\}$, with $|D| = 1000$. Afterwards, we calculate gradients $\nabla_\theta f_\theta(\cdot)$ at each point in D and construct the *Gramian* matrix G, with $G_{ij} = g_\theta(X_i, X_j)$. Note that the index of each point expresses also its location within the chosen point path, and the index difference $|i - j|$ also represents Euclidean distance between points X_i and X_j. In Figure H.3a this matrix G is



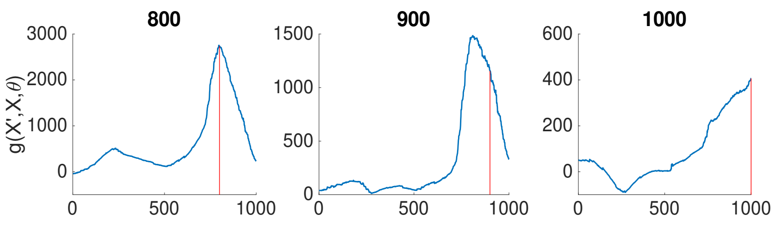
(a)



(b)



(c)



(d)

Figure H.3: Local relation between values of *gradient similarity* $g_\theta(X', X)$ and values of Euclidean distance $d(X', X)$ within BD architecture. The applied PSO method and model architecture are same as in Figure H.1. After convergence, we calculate gradient $\nabla_\theta f_\theta(X)$ along path within input space, $[-1, \ldots, -1] \longrightarrow [1, \ldots, 1]$, which is uniformly discretized via 1000 middle points. Afterwards, (a) the *Gramian* matrix G is constructed, with $G_{ij} = g_\theta(X_i, X_j)$ where X_i and X_j are i-th and j-th points along the path. Note that the index difference $|i - j|$ also represents Euclidean distance between points X_i and X_j. Further, i-th row of G contains similarities $g_\theta(X_i, \cdot)$ between point i and rest of points. (a)-(c) 1-th, 100-th, $\ldots$, 900-th and 1000-th rows of G are shown. Red line indicates i-th entry of i-th row where self similarity $g_\theta(X_i, X_i)$ is plotted. See more details in the main text.

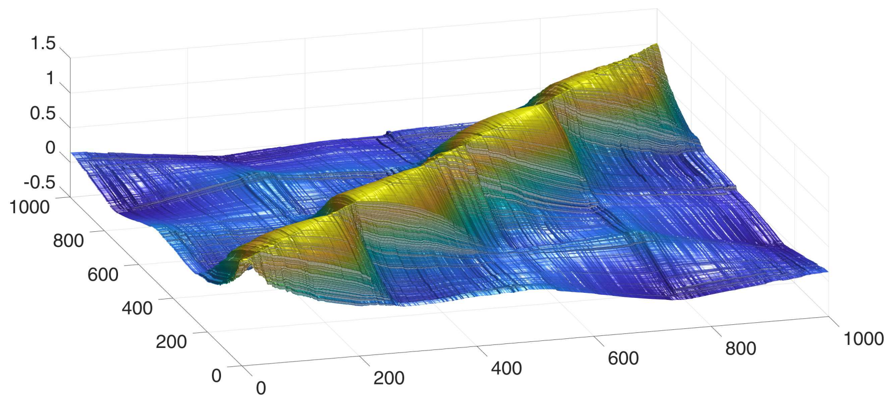
depicted, and here it can be observed that G's diagonal is very prominent. This again reasserts that the *gradient similarity* kernel has some local support induced by the implicit bandwidth.

Moreover, each row r_i inside G represents $g_\theta(X_i, \cdot)$ - side similarity between the point X_i and the rest of points in D (i.e. the chosen path). Note that the i-th entry of r_i represents self-similarity $g_\theta(X_i, X_i)$, while other entries represent the side-similarity from path points around X_i. Further, indexes of these other entries are related to the distance between the points and X_i, via the index difference $|i - j|$. Hence, first $i - 1$ entries of r_i represent first $i - 1$ points within the chosen path D before the i-th point, whereas the last $1000 - i$ entries of r_i represent points at the end of the path. In Figures H.3b-H.3d 11 different rows of G are depicted, where each row demonstrates *gradient similarity* around some path point X_i as a function of the second point index (and thus the distance between two points). Here we can see that $g_\theta(X_i, X_j)$ typically has a peak at $X_j = X_i$, and further smoothly decreases as we walk away from point X_i in any of the two path directions (towards S or E). Thus, here we see that *gradient similarity* has a bell-like behavior, returning high similarity for the same point and diminishing as the distance between the points grows. Yet, these "bells" are not centered, with some rows (e.g. 400-th in Figure H.3c) having peaks outside of the i-th entry. Nevertheless, in context of PSO such local behavior allows us to conclude that when any training point X is pushed by PSO loss, the force impact on the model surface is local, centered around the pushed X.

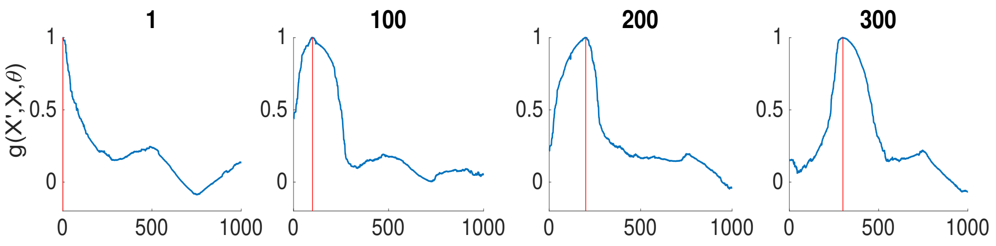
Further, in Figure H.4 we also plot the normalized *gradient similarity* $\bar{g}_\theta(X_i, X_j) \triangleq \frac{\nabla_\theta f_\theta(X_i)^T \cdot \nabla_\theta f_\theta(X_j)}{\|\nabla_\theta f_\theta(X_i)\| \cdot \|\nabla_\theta f_\theta(X_j)\|} = \cos\left[\angle\left[\nabla_\theta f_\theta(X_i), \nabla_\theta f(X_j)\right]\right]$ for the same setting as in Figure H.3. Here, we can see that $\bar{g}_\theta(X_i, X_j)$, which is a cosine of the angle between θ gradients at points X_i and X_j, has a more symmetrical and centered behavior compared with $g_\theta(X_i, X_j)$. That is, $\bar{g}_\theta(X_i, \cdot)$ has a peak when the second argument is equal to X_i, and it gradually decreases as the distance between the second argument and X_i increases. Moreover, the asymmetry that we observed in case of $g_\theta(X_i, X_j)$ (400-th and 600-th rows in Figure H.3c) is actually caused by the difference in a gradient norm at different points X_i and X_j. Specifically, in Figure H.3c we can see that peak of 400-th row is pushed towards the beginning of the path D, where X_{400} has neighbors with higher norm $\|\nabla_\theta f_\theta(X_j)\|$. In $\bar{g}_\theta(X_i, X_j)$ each gradient is normalized to have a unit norm, which eliminates the above asymmetry as observed in Figure H.4c. Hence, nearby points with a large gradient norm may affect the *gradient similarity* at a specific point and make it less symmetric/centered.

Note that in case the applied optimizer is GD, during the actual optimization the NN surface is pushed according to the $g_\theta(X_i, X_j)$ and not $\bar{g}_\theta(X_i, X_j)$. Therefore, various local asymmetries inside $g_\theta(X_i, X_j)$ may affect the optimization optimality. However, Adam optimizer [59], used in most of our experiments, with its adaptive moment estimation and normalization per each weight implicitly transforms the actual "pushing" kernel of the optimization. We speculate that in this case the actual information kernel is more similar to the normalized gradient similarity. We shall leave a detailed investigation of the model kernel under Adam optimization update rule for future work.

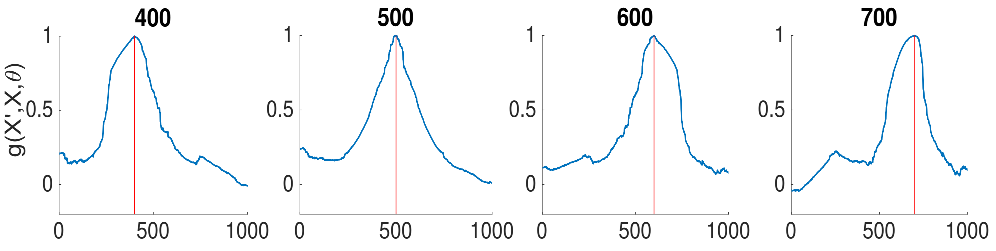
Overall, our experiments show that NN weights undergo some *uncorrelation* process during the first few thousands of iterations, after which the *gradient similarity* obtains properties



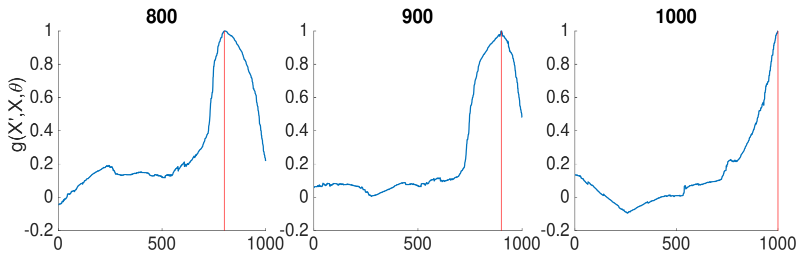
(a)



(b)



(c)



(d)

Figure H.4: Normalized gradient results within BD architecture: same results from Figure H.3, with *Gramian* matrix being normalized to $\bar{G} : \bar{G}_{ij} = \bar{g}_\theta(X_i, X_j) = \frac{\nabla_\theta f_\theta(X_i)^T \cdot \nabla_\theta f_\theta(X_j)}{\|\nabla_\theta f_\theta(X_i)\| \cdot \|\nabla_\theta f_\theta(X_j)\|}$. As observed, normalized *gradient similarity* $\bar{g}_\theta(X_i, X_j)$ is more symmetrical along both directions of a chosen path D compared with the regular *gradient similarity* $g_\theta(X_i, X_j)$ in Figure H.3.

approximately similar to a local-support kernel function. Specifically, an opposite relation is formed between the *gradient similarity* and Euclidean distance, where a higher distance is associated with a smaller similarity. This *uncorrelation* can also be viewed as gradients (w.r.t. θ) at different training points are becoming more and more linearly independent along the optimization, which as a result increases angles between gradient vectors. Hence, these gradients point to different directions inside the parameter space $\mathbb{R}^{|\theta|}$, decreasing the side-influence between the training points. Furthermore, this *uncorrelation* process was observed almost in each experiment, yet the radius of local support for the kernel $g_\theta(X', X)$ (that is, how fast *gradient similarity* is decreasing w.r.t. $d(X, X')$) is changing depending on the applied NN architecture (e.g. FC vs BD) and on the specific inferred density $\mathbb{P}^U$.

APPENDIX I

$LSQR$ Divergence

Here we will prove that $LSQR$ evaluation metric, considered in this thesis, is an actual statistical divergence. First, the log-pdf squared error (LSQR) divergence is defined as:

$$LSQR(\mathbb{P}, \mathbb{Q}) = \int \mathbb{P}(X) \cdot [\log \mathbb{P}(X) - \log \mathbb{Q}(X)]^2 \, dX, \tag{I.1}$$

where $\mathbb{P}$ is the pdf over a compact support $\Omega \subset \mathbb{R}^n$; $\mathbb{Q}$ is normalized or unnormalized model whose support is also Ω.

According to the definition of statistical divergences, LSQR must satisfy $\forall \mathbb{P}, \mathbb{Q} : LSQR(\mathbb{P}, \mathbb{Q}) \geq 0$ and $LSQR(\mathbb{P}, \mathbb{Q}) = 0 \Leftrightarrow \mathbb{P} = \mathbb{Q}$. Obviously, these two conditions are satisfied by Eq. (I.1). Hence, $LSQR(\mathbb{P}, \mathbb{Q})$ is the statistical divergence.

Importantly, we emphasize that $LSQR(\mathbb{P}, \mathbb{Q})$ measures a discrepancy between pdf $\mathbb{P}$ and model $\mathbb{Q}$ where the latter is allowed to be unnormalized. Therefore, it can be used to evaluate PSO-based methods since they are only approximately normalized. However, such evaluation is only possible when $\mathbb{P}$ is known analytically.

Further, in this thesis $LSQR$ is measured between the target density, defined as $\mathbb{P}^U$ in the thesis, and the pdf estimator $\bar{\mathbb{P}}_\theta$ produced by some method in the following way:

$$LSQR(\mathbb{P}^U, \bar{\mathbb{P}}_\theta) = \frac{1}{N} \sum_{i=1}^{N} \left[\log \mathbb{P}^U(X_i^U) - \log \bar{\mathbb{P}}_\theta(X_i^U) \right]^2, \tag{I.2}$$

where $\{X_i^U\}_{i=1}^N$ are testing points, sampled from $\mathbb{P}^U$, that were not involved in the estimation process of $\bar{\mathbb{P}}_\theta$.

APPENDIX **J**

Matrix A from definition of *Transformed Columns* Distribution

Matrix A was randomly generated under the constraint of having a determinant 1, to keep the volume of sampled points the same. Its generated entries are:

$$A = \begin{pmatrix}
0.190704135 & -0.103706818 & 0.287080085 & -0.224115607 & -0.0296220322 & -0.200017067 & 0.107420472 & 0.145939799 & -0.21151047 & 0.290260037 & -0.109926743 & -0.138214861 & 0.0739138407 & -0.173910764 & -0.158279581 & -0.138856972 & -0.512741096 & -0.0894244111 & -0.465075432 & 0.138203328 \\
0.254157669 & 0.00972120677 & -0.425381258 & -0.165311223 & -0.0732519109 & 0.316785766 & -0.0651314216 & -0.153534853 & -0.294111694 & -0.29775682 & -0.285308807 & 0.12228138 & -0.11072477 & -0.0955035066 & 0.00942833152 & -0.252106498 & -0.40791915 & 0.14810246 & 0.219848116 & 0.0170452831 \\
0.213837698 & 0.435577026 & -0.0250319656 & 0.297552176 & 0.14030724 & 0.17703815 & 0.179253182 & -0.0710653564 & 0.0507340489 & 0.235684257 & 0.33391508 & -0.40609493 & -0.197930666 & -0.322136609 & -0.146204612 & -0.164195869 & -0.0672063429 & 0.138970011 & 0.121959537 & -0.150499483 \\
0.229971825 & 0.235126465 & 0.158420112 & -0.0223857003 & 0.28726862 & 0.133325519 & -0.352231148 & 0.403451114 & -0.0161522847 & -0.0193998935 & 0.0455239109 & 0.162348247 & -0.108553457 & -0.126312424 & 0.354695129 & -0.18792775 & 0.101384726 & -0.360466001 & 0.0797850581 & 0.338497968 \\
0.168709235 & 0.0770914534 & -0.178165495 & -0.0661545928 & 0.323673824 & -0.216202087 & 0.475022027 & 0.129138946 & -0.0173273685 & -0.305472906 & -0.187205709 & -0.0359566427 & -0.216262061 & -0.0428909211 & -0.354871726 & 0.299579441 & 0.157803981 & -0.190573488 & 0.0280581785 & 0.281767192 \\
0.238265575 & 0.14326338 & 0.323225051 & 0.101182056 & 0.222068216 & -0.42470829 & -0.132128709 & -0.203895612 & -0.38640015 & -0.193494524 & 0.15340899 & 0.0919709633 & -0.150512414 & 0.257608882 & 0.182536036 & 0.223159059 & -0.177186352 & 0.198776911 & 0.130192805 & -0.201342323 \\
0.279579926 & -0.162797928 & -0.0586375005 & -0.211398563 & -0.178520507 & -0.0154862203 & -0.371463145 & 0.187233788 & -0.19506691 & 0.119455231 & -0.202696444 & -0.581504491 & -0.160227753 & 0.135365515 & -0.104123883 & 0.128700751 & 0.341221005 & 0.04776047 & 0.0956523695 & -0.105994479 \\
0.220148935 & -0.2672238 & 0.259200965 & -0.348342982 & 0.155930129 & 0.0194560055 & 0.136887538 & -0.264686829 & 0.243602026 & 0.117285157 & -0.002458813 & 0.241121126 & -0.423370778 & 0.01867544 & -0.0604651676 & -0.397344315 & 0.273889032 & 0.122122216 & 0.00146222648 & -0.125711392 \\
0.274834233 & 0.105359473 & 0.135585987 & -0.19681974 & -0.0573374634 & 0.272574082 & 0.0741237415 & 0.0122961299 & 0.331281502 & -0.33605766 & 0.271171768 & -0.205618232 & 0.213806318 & 0.535961439 & -0.103513592 & -0.0402908492 & -0.199964863 & -0.203164108 & -0.0657867077 & -0.0601941311 \\
0.191457811 & -0.234983092 & 0.180777146 & 0.293340161 & -0.0365900702 & 0.214595475 & 0.236285794 & 0.256954426 & 0.152660465 & 0.0502990912 & -0.34518931 & 0.0711801608 & -0.128073093 & -0.0785011303 & 0.222291071 & 0.252921604 & -0.174804068 & -0.217786875 & 0.114727637 & -0.492946359 \\
0.223304218 & -0.0257257131 & -0.413566391 & 0.172073687 & 0.326658323 & -0.251109479 & -0.188027609 & 0.334468985 & 0.161164411 & 0.150742708 & -0.0165671389 & 0.232902107 & 0.21926384 & 0.194760411 & -0.235966926 & -0.210659524 & -0.00475631985 & 0.172782694 & -0.148866241 & -0.300869538 \\
0.243388344 & 0.243683991 & -0.121589184 & -0.187437781 & 0.0578409285 & -0.27767391 & 0.118032949 & -0.346625601 & 0.292226942 & 0.243065097 & -0.348108205 & -0.141543891 & 0.37738745 & -0.0330943339 & 0.390702777 & 0.02656679 & 0.0292203222 & -0.0520266353 & 0.165078206 & 0.0596683021 \\
0.230141811 & 0.0247039796 & -0.160146528 & 0.276454478 & -0.576178718 & -0.329822403 & 0.25548108 & 0.14013065 & -0.0103114951 & -0.162178682 & 0.0992219866 & 0.012231234 & -0.189306474 & 0.0930990418 & 0.282716476 & -0.306277128 & 0.142988577 & -0.00844652753 & -0.179308874 & 0.106622196 \\
0.232700446 & -0.203358735 & -0.0404096623 & 0.0651155788 & -0.0876307509 & -0.191362905 & -0.350703084 & -0.292134625 & 0.189041139 & -0.344463408 & 0.182873652 & 0.00843615229 & 0.103973847 & -0.52808934 & -0.153969081 & 0.0920142117 & 0.0274541623 & -0.30781733 & -0.126023284 & -0.157587751 \\
0.199527977 & 0.204898606 & -0.0450516851 & -0.270830109 & -0.122669508 & 0.203103159 & 0.237732868 & 0.012741777 & -0.462819922 & 0.114896626 & 0.187819585 & 0.289174359 & 0.310962024 & -0.0974490092 & -0.00676534358 & 0.104883625 & 0.361388813 & -0.151993161 & -0.133748076 & -0.307757495 \\
0.175957009 & 0.114449497 & 0.0205901599 & -0.309807805 & -0.235656127 & 0.00557129199 & -0.0424724202 & 0.307742364 & 0.323261613 & -0.00822297356 & 0.171003077 & 0.20675015 & -0.0593725006 & -0.232062167 & 0.0481299972 & 0.415888833 & -0.102561506 & 0.524594092 & 0.0346260903 & 0.106458345 \\
0.223633992 & -0.408554226 & -0.309022911 & 0.131333656 & 0.253238692 & 0.245874156 & 0.0563174345 & -0.203397977 & -0.0592004594 & 0.179237363 & 0.275909961 & -0.0735710289 & -0.0681467794 & 0.0974804318 & 0.340180897 & 0.282145806 & -0.0050587548 & 0.0683927742 & -0.329991983 & 0.246896636 \\
0.215583387 & 0.212554612 & 0.289443614 & 0.366636519 & -0.034606719 & 0.274097732 & -0.127438501 & -0.179674304 & 0.0238811214 & -0.126780371 & -0.419167616 & 0.0954798128 & 0.126317847 & 0.0330201935 & -0.166833728 & 0.039242297 & 0.231007699 & 0.29173484 & -0.381428263 & 0.182896273 \\
0.210225414 & -0.42839288 & 0.239002756 & 0.174593297 & 0.0372987005 & -0.000156767089 & 0.183766314 & 0.132302659 & -0.124351715 & -0.0656208547 & 0.126841957 & -0.0486788409 & 0.471440044 & -0.112599645 & -0.0828678102 & -0.173846575 & 0.0871950404 & 0.210646003 & 0.459392924 & 0.242375288 \\
0.220671036 & 0.0270987729 & -0.0325243953 & 0.176295746 & -0.299408603 & -0.00284534072 & -0.133474574 & -0.186385408 & 0.0205827025 & 0.45209226 & 0.0583508015 & 0.321753008 & -0.120986377 & 0.217766867 & -0.359227919 & 0.180578462 & -0.115873733 & -0.281505687 & 0.296303031 & 0.245563455
\end{pmatrix}. \tag{J.1}$$

APPENDIX **K**

Relation between spectrums of $g_t(X, X')$ and its Gramian G_t

Consider N dataset points $\boldsymbol{\mathcal{X}} = \{X^i \in \mathbb{R}^d\}_{i=1}^N$ sampled from an arbitrary probability density function (pdf) $P(X)$. Further, consider a kernel $g_t(X, X')$ and the corresponding Gramian G_t defined on $\boldsymbol{\mathcal{X}}$, with $G_t(i, j) = g_t(X^i, X^j)$. Eigenvalues $\{\tilde{\lambda}_k\}_k$, sorted in decreasing order, and eigenfunctions $\{\tilde{v}_k(\cdot)\}_k$ of $g_t(\cdot, \cdot)$ w.r.t. $P(X)$ are defined as solutions of:

$$\tilde{\lambda}_k \cdot \tilde{v}_k(X) = \int g_t(X, X') \cdot \tilde{v}_k(X') \cdot P(X')dX'. \tag{K.1}$$

The integral in Eq. (K.1) can be approximated via a sampled approximation:

$$\int g_t(X, X') \cdot \tilde{v}_k(X') \cdot P(X')dX' \approx \frac{1}{N} \sum_{i=1}^{N} g_t(X, X^i) \cdot \tilde{v}_k(X^i), \tag{K.2}$$

with the RHS of the above expression converging to the LHS as $N \to \infty$ due to the law of large numbers.

Further, denote by $\bar{v}_k$ a $N \times 1$ vector whose i-th entry is $\tilde{v}_k(X^i)$. Combining Eq. (K.1) and Eq. (K.2), $\bar{v}_k$ can be written as:

$$\tilde{\lambda}_k \cdot \bar{v}_k = \frac{1}{N} G_t \cdot \bar{v}_k, \tag{K.3}$$

where we can see $\bar{v}_k$ to be eigenvector of G_t. Therefore, eigenvectors $\{\bar{v}_k\}_k$ of G_t can be considered as unbiased estimations of eigenfunctions $\{\tilde{v}_k(\cdot)\}_k$ at points in $\boldsymbol{\mathcal{X}}$. Note that the above sampled approximations are expected to be less accurate for larger indexes k since the corresponding $\tilde{v}_k(\cdot)$ will contain more high-frequency oscillations.

Furthermore, from Eq. (K.3) it is clear that each $\bar{v}_k$ is associated with the eigenvalue $\lambda_k = N \cdot \tilde{\lambda}_k$ of G_t. Hence, eigenvalues $\{\lambda_k\}_k$ of G_t can be considered as unbiased estimations of eigenfunctions $\{\tilde{\lambda}_k\}_k$, up to a multiplier N.

Likewise, $\tilde{v}_k(X)$ at an arbitrary point X can be estimated in a similar way, by combining

Eq. (K.1) and Eq. (K.2):

$$\tilde{\lambda}_k \cdot \tilde{\upsilon}_k(X) \approx \frac{1}{N} \sum_{i=1}^{N} g_t(X, X^i) \cdot \tilde{\upsilon}_k(X^i) \quad \Longrightarrow \quad \lambda_k \cdot \tilde{\upsilon}_k(X) \approx g_t(X, \boldsymbol{\mathcal{X}}) \cdot \bar{\upsilon}_k, \tag{K.4}$$

where $g_t(X, \boldsymbol{\mathcal{X}})$ is a row vector with $g_t(X, \boldsymbol{\mathcal{X}})_{(i)} = g_t(X, X^i)$. The above approximation is used in the Appendix O to derive NN dynamics at testing points.

APPENDIX **L**

Relation between FIM and Hessian of the Loss

Hessian of a typical loss in Eq. (14.1) can be written as:

$$H_t \triangleq \frac{\partial^2 L(\theta_t, D)}{\partial \theta^2} = \frac{1}{N} A_t D_t A_t^T + \frac{1}{N} \sum_{i=1}^{N} \ell' \left[X^i, Y^i, f_{\theta_t}(X^i) \right] \cdot \mathcal{H}_t(X^i), \tag{L.1}$$

where A_t is Jacobian matrix defined in Section 14.2, D_t is a diagonal matrix with $D_t(i,i) = \frac{\partial^2 \ell[X^i, Y^i, f_{\theta_t}(X^i)]}{\partial f_\theta^2}$ and $\mathcal{H}_t(X) \triangleq \frac{\partial^2 f_{\theta_t}(X)}{\partial \theta^2}$ is the model Hessian.

Further, in case of L2 loss we will have $D_t = I$ and

$$H_t = \frac{1}{N} F_t + \frac{1}{N} \sum_{i=1}^{N} \ell' \left[X^i, Y^i, f_{\theta_t}(X^i) \right] \cdot \mathcal{H}_t(X^i). \tag{L.2}$$

Finally, considering final stages of the optimization, the residual $\ell' \left[X^i, Y^i, f_{\theta_t}(X^i) \right] = f_{\theta_t}(X^i) - Y^i$ is approximately zero and hence the second term of Eq. (L.2) RHS can be neglected. Therefore, for L2 loss we will have $H_t \approx \frac{1}{N} F_t$.

Beyond L2 loss, a connection between FIM and the loss Hessian was also observed for the cross-entropy loss in [37]. Authors empirically observed that the loss gradient $\nabla_\theta L(\theta_t, D)$ converges very fast into a tiny subspace spanned by a few *top* eigenvectors of H_t. This suggests that *top* eigenvectors of H_t and F_t are tightly aligned and are spanning the same subspace of $\mathbb{R}^{|\theta|}$ also for cross-entropy case, as follows. Denote A_t's SVD as triplets $\{\sqrt{\lambda_i^t}, \bar{\omega}_i^t, \bar{\upsilon}_i^t\}_{i=1}^{N'}$ of ordered singular values, left and right singular vectors respectively, where N' is a number of non-zero singular values. Then, $\nabla_\theta L(\theta_t, D)$ can be written as:

$$\nabla_\theta L(\theta_t, D) = \frac{1}{N} A_t \cdot \bar{m}_t = \frac{1}{N} \left[\sum_{i=1}^{N'} \sqrt{\lambda_i^t} \cdot \bar{\omega}_i^t \cdot (\bar{\upsilon}_i^t)^T \right] \cdot \bar{m}_t = \frac{1}{N} \sum_{i=1}^{N'} \sqrt{\lambda_i^t} < \bar{\upsilon}_i^t, \bar{m}_t > \bar{\omega}_i^t. \tag{L.3}$$

Due to typical extremely fast decay of λ_i^t w.r.t. i, described along Chapter 14, $\nabla_\theta L(\theta_t, D)$ in the above expression can be roughly seen as a linear combination of only $\{\bar{\omega}_i^t\}$ associated with several *top* $\{\lambda_i^t\}$. Noting that these are also the *top* eigenvectors of F_t, we see that $\nabla_\theta L(\theta_t, D)$ is located in top-spectrum of F_t. Further, taking into account the empirical observation from [37], we can conclude from above that *top* eigenvectors of F_t and H_t are tightly aligned.

APPENDIX M

Movement of θ along FIM Eigenvector causes Movement of NN Output along Gramian Eigenvector

To understand the relation between FIM F_t and Gramian G_t more intuitively, here we show their dual connection in terms of how the movement along FIM eigenvector $\bar{\omega}_i^t$ in θ-space affects the movement in the function space. Specifically, consider $\bar{f}_t$ to be a vector of NN outputs at training points at optimization time t, similarly to the formulation in Section 14.1. Further, consider a movement of the model in θ-space from current θ_t to a new location $\theta_{t'} = \theta_t + \sqrt{\lambda_i^t} \cdot \bar{\omega}_i^t$ in direction $\bar{\omega}_i^t$ where $\sqrt{\lambda_i^t}$ is used as a step size. Then the $\bar{f}_{t'}$ at the new location can be approximated via first-order Taylor as:

$$\bar{f}_{t'} = \bar{f}_t + \sqrt{\lambda_i^t} \cdot A_t^T \cdot \bar{\omega}_i^t, \tag{M.1}$$

where A_t is Jacobian matrix defined in Section 14.2. Moreover, considering the singular value decomposition (SVD) of A_t, we can see that $\bar{f}_{t'} - \bar{f}_t = \lambda_i^t \cdot \bar{\upsilon}_i^t$. That is, walking in the direction $\bar{\omega}_i^t$ in θ-space changes NN outputs only along $\bar{\upsilon}_i^t$, according to first-order dynamics.

APPENDIX N

Dynamics of L2 Loss for a Fixed Gramian, at Training Points

Consider Eq. (14.3) with a fixed Gramian G whose eigenvalues and eigenvectors are $\{\lambda_i\}_{i=1}^N$ and $\{\bar{\upsilon}_i\}_{i=1}^N$ respectively. Define N' to be a number of non-zero eigenvalues. Likewise, consider the residual vector $\bar{m}_t = \bar{f}_t - \bar{y}$ whose first-order dynamics can be written as:

$$
\begin{aligned}
&d\bar{m}_t \triangleq \bar{m}_{t+1} - \bar{m}_t = \bar{f}_{t+1} - \bar{f}_t = d\bar{f}_t = -\frac{\delta}{N} \cdot G \cdot \bar{m}_t \quad \Longrightarrow \\
&\Longrightarrow \quad \bar{m}_{t+1} = \left[I - \frac{\delta}{N} \cdot G \right] \cdot \bar{m}_t \quad \Longrightarrow \quad \bar{m}_t = \sum_{i=1}^{N'} \left[1 - \frac{\delta}{N} \lambda_i \right]^t < \bar{\upsilon}_i, \bar{m}_0 > \bar{\upsilon}_i + \bar{m}_0^z,
\end{aligned}
\tag{N.1}
$$

where $\bar{m}_0^z$ is a projection of $\bar{m}_0$ to null-space of G, with $G \cdot \bar{m}_0^z = \bar{0}$.

Further, noting that:

$$
\sum_{j=0}^{t-1} \bar{m}_j = \sum_{i=1}^{N'} \frac{1 - \left[1 - \frac{\delta}{N} \lambda_i \right]^t}{\frac{\delta}{N} \lambda_i} < \bar{\upsilon}_i, \bar{m}_0 > \bar{\upsilon}_i + t \bar{m}_0^z,
\tag{N.2}
$$

the $\bar{f}_t$ can be then rewritten as:

$$
\begin{aligned}
\bar{f}_t = \bar{f}_0 + \sum_{j=0}^{t-1} d\bar{f}_j = \bar{f}_0 - \frac{\delta}{N} G \cdot \sum_{j=0}^{t-1} \bar{m}_j = \bar{f}_0 - \frac{\delta}{N} G \cdot \sum_{i=1}^{N'} \frac{1 - \left[1 - \frac{\delta}{N} \lambda_i \right]^t}{\frac{\delta}{N} \lambda_i} < \bar{\upsilon}_i, \bar{m}_0 > \bar{\upsilon}_i = \\
= \bar{f}_0 - \sum_{i=1}^{N'} \left[1 - \left[1 - \frac{\delta}{N} \lambda_i \right]^t \right] < \bar{\upsilon}_i, \bar{m}_0 > \bar{\upsilon}_i.
\end{aligned}
\tag{N.3}
$$

APPENDIX **O**

Dynamics of L2 Loss for a Fixed Gramian, at Testing Points

From Eq. (14.2) we can also derive dynamics of NN output at an arbitrary testing point X':

$$df_{\theta_t}(X') = f_{\theta_{t+1}}(X') - f_{\theta_t}(X') = -\frac{\delta}{N} g(X', \boldsymbol{\mathcal{X}}) \cdot \bar{m}_t, \tag{O.1}$$

where $g(X', \boldsymbol{\mathcal{X}}) \triangleq \nabla_\theta f_{\theta_t}(X')^T \cdot A_t$ is a row vector with $g(X', \boldsymbol{\mathcal{X}})_{(j)} = g(X', X^j)$. Moreover, similarly to Eq. (N.3) we get:

$$\begin{aligned} f_{\theta_t}(X') &= f_{\theta_0}(X') + \sum_{j=0}^{t-1} df_{\theta_j}(X') = f_{\theta_0}(X') - \frac{\delta}{N} g(X', \boldsymbol{\mathcal{X}}) \cdot \sum_{j=0}^{t-1} \bar{m}_j = \\ &= f_{\theta_0}(X') - \frac{\delta}{N} g(X', \boldsymbol{\mathcal{X}}) \cdot \left[\sum_{i=1}^{N'} \frac{1 - \left[1 - \frac{\delta}{N}\lambda_i\right]^t}{\frac{\delta}{N}\lambda_i} < \bar{\upsilon}_i, \bar{m}_0 > \bar{\upsilon}_i + t\bar{m}_0^z \right]. \end{aligned} \tag{O.2}$$

In case G is invertible (i.e. $\lambda_{min} > 0$), the above expression can also be written as $f_{\theta_t}(X') = f_{\theta_0}(X') - g(X', \boldsymbol{\mathcal{X}}) \cdot G^{-1} \cdot \left[I - \left[I - \frac{\delta}{N} \cdot G \right]^t \right] \cdot \bar{m}_0$; a very similar expression was previously derived in [71]. Likewise, considering the stability condition $\delta < \frac{2N}{\lambda_{max}}$, which is required for a proper optimization convergence $\lim_{t \to \infty} \left[1 - \frac{\delta}{N}\lambda_i\right]^t = 0$, at time $t = \infty$ we will have $f_{\theta_\infty}(X') = f_{\theta_0}(X') - g(X', \boldsymbol{\mathcal{X}}) \cdot G^{-1} \cdot \bar{m}_0$.

Furthermore, for a singular G Eq. (O.2) can be simplified via two methods, using a gradient at X' or eigenfunctions of the kernel $g(\cdot, \cdot)$.

Simplification via Gradient Observe that for $G = A_t^T \cdot A_t$ to be time-invariant it is necessary for gradients $\{\nabla_\theta f_{\theta_t}(X^i)\}_{i=1}^N$ at training points either to be constant along the optimization or rotating together via some time-variant rotation matrix R_t, $\nabla_\theta f_{\theta_t}(X^i) = R_t \cdot \nabla_\theta f_{\theta_0}(X^i)$ and $A_t = R_t \cdot A_0$. Such rotational behavior will lead to the required time-independence of $G = A_0^T \cdot R_t^T \cdot R_t \cdot A_0 = A_0^T \cdot A_0$. Similarly, for $g(X', \boldsymbol{\mathcal{X}})$ to be time-invariant the gradient $\nabla_\theta f_{\theta_t}(X')$

at the testing point must rotate with the same rotation R_t, $\nabla_\theta f_{\theta_t}(X') = R_t \cdot \nabla_\theta f_{\theta_0}(X')$.

Assuming the above gradient rotation, the row vector $g(X', \boldsymbol{\mathcal{X}})$ can be written as:

$$g(X', \boldsymbol{\mathcal{X}}) = \nabla_\theta f_{\theta_t}(X')^T \cdot A_t = \nabla_\theta f_{\theta_0}(X')^T \cdot R_t^T \cdot R_t \cdot A_0 = \nabla_\theta f_{\theta_0}(X')^T \cdot A_0. \quad \text{(O.3)}$$

Next, consider A_0's SVD as triplets $\{\sqrt{\lambda_i}, \bar{\omega}_i, \bar{\upsilon}_i\}_{i=1}^{N'}$ of ordered singular values, left and right singular vectors respectively, and denote $\nabla_\theta f_{\theta_0}(X') = \sum_{i=1}^{N'} a_i \cdot \sqrt{\lambda_i} \cdot \bar{\omega}_i$ for $a_i \triangleq \frac{<\bar{\omega}_i, \nabla_\theta f_{\theta_0}(X')>}{\sqrt{\lambda_i}}$. Using SVD properties of A_0, we get an identity $g(X', \boldsymbol{\mathcal{X}}) = \sum_{i=1}^{N'} a_i \cdot \lambda_i \cdot \bar{\upsilon}_i^T$, and we can rewrite $f_{\theta_t}(X')$ from Eq. (O.2) as (note that $\bar{m}_0^{\tilde{z}}$ is reduced since it is orthogonal to $\{\bar{\upsilon}_i : \lambda_i \neq 0\}$):

$$f_{\theta_t}(X') = f_{\theta_0}(X') - \sum_{i=1}^{N'} \left[1 - \left[1 - \frac{\delta}{N}\lambda_i\right]^t\right] a_i < \bar{\upsilon}_i, \bar{m}_0 >=$$
$$= f_{\theta_0}(X') - \sum_{i=1}^{N'} \left[1 - \left[1 - \frac{\delta}{N}\lambda_i\right]^t\right] \frac{1}{\sqrt{\lambda_i}} < \bar{\upsilon}_i, \bar{m}_0 >< \bar{\omega}_i, \nabla_\theta f_{\theta_0}(X') > . \quad \text{(O.4)}$$

Likewise, under the stability condition $\delta < \frac{2N}{\lambda_{max}}$, $f_{\theta_t}(X')$ at time $t = \infty$ can be expressed as:

$$f_{\theta_\infty}(X') = f_{\theta_0}(X') - \sum_{i=1}^{N'} \frac{1}{\sqrt{\lambda_i}} < \bar{\upsilon}_i, \bar{m}_0 >< \bar{\omega}_i, \nabla_\theta f_{\theta_0}(X') > . \quad \text{(O.5)}$$

Simplification via Kernel Eigenfunctions According to Eq. (K.4), a product $g(X', \boldsymbol{\mathcal{X}}) \cdot \bar{\upsilon}_i$ can be approximated by $\lambda_i \cdot \tilde{\upsilon}_i(X')$, with $\tilde{\upsilon}_i(\cdot)$ being an eigenfunction of $g(\cdot, \cdot)$. Using this approximation, Eq. (O.2) is reduced to:

$$f_{\theta_t}(X') \approx f_{\theta_0}(X') - \frac{\delta}{N} \cdot \left[\sum_{i=1}^{N'} \frac{1 - \left[1 - \frac{\delta}{N}\lambda_i\right]^t}{\frac{\delta}{N}} < \bar{\upsilon}_i, \bar{m}_0 > \tilde{\upsilon}_i(X') + \right.$$
$$\left. + t \cdot \sum_{i:\lambda_i = 0} \lambda_i < \bar{\upsilon}_i, \bar{m}_0 > \tilde{\upsilon}_i(X')\right] = f_{\theta_0}(X') - \sum_{i=1}^{N'} \left[1 - \left[1 - \frac{\delta}{N}\lambda_i\right]^t\right] < \bar{\upsilon}_i, \bar{m}_0 > \tilde{\upsilon}_i(X'), \quad \text{(O.6)}$$

which at time $t = \infty$ will converge to:

$$f_{\theta_\infty}(X') = f_{\theta_0}(X') - \sum_{i=1}^{N'} < \bar{\upsilon}_i, \bar{m}_0 > \tilde{\upsilon}_i(X'). \quad \text{(O.7)}$$

Intuition Eq. (O.4) and Eq. (O.6) describe first-order dynamics of NN output at a testing point. The intuition behind these expressions can be summarized as following. First, for standard NN initialization $f_{\theta_0}(X')$ is typically very close to be zero and can be neglected, leading to $\bar{m}_0 \approx -\bar{y}$. Like in Eq. (N.3), the inner-product term $< \bar{\upsilon}_i, \bar{m}_0 >$, independent of testing point X', defines which part of the signal contained in $\bar{m}_0$ is learned along each spectral direction. In general, $\left[1 - \frac{\delta}{N}\lambda_i\right]^t$ converges faster for large eigenvalues. Also, due to large λ_i being typically associated with $\bar{\upsilon}_i$ that contains a low-frequency signal, this leads to fast learning of low-frequency information and slow (sometimes infinitely slow) learning of high-frequency information. Further, the inner-product term $< \bar{\omega}_i, \nabla_\theta f_{\theta_0}(X') >$ in Eq. (O.4) or the eigenfunction $\tilde{\upsilon}_i(X')$ in Eq. (O.6), that are functions of X', determine amount of information along i-th spectral direction that is transferred into $f_{\theta_t}(X')$, basically describing the generalization behind Eq. (14.3) for a fixed Gramian G. Note that the convergence rate of $f_{\theta_t}(X')$ towards $f_{\theta_\infty}(X')$ is governed by how close terms $1 - \frac{\delta}{N}\lambda_i$ in Eq. (O.4) and Eq. (O.6) are to zero, similarly to the convergence rate of a system in Eq. (N.3). Hence, we expect f_{θ_t} to converge to its final state at both training and testing points with a similar speed.

APPENDIX P

First-order Change of G_t

Here we describe the first-order Taylor approximation of a change in G_t between sequential iterations of GD optimization. We theorize that the thorough analysis of below expressions will lead to the mathematical explanation required to understand evolution of G_t as also to better understanding of NN dynamics.

First, change of the Jacobian A_t, defined in Section 14.2, can be described as:

$$dA_t \triangleq A_{t+1} - A_t \approx -\frac{\delta}{N} \cdot W_t, \tag{P.1}$$

where W_t is $|\theta| \times N$ matrix with i-th column being $\mathcal{H}_t(X^i) \cdot A_t \cdot \bar{m}_t$, with $\mathcal{H}_t(X) \triangleq \frac{\partial^2 f_{\theta_t}(X)}{\partial \theta^2}$ being the model Hessian.

Hence, the change between $G_{t+1} = A_{t+1}^T \cdot A_{t+1}$ and $G_t = A_t^T \cdot A_t$ can be written as:

$$dG_t \triangleq G_{t+1} - G_t \approx -\frac{\delta}{N} \cdot \left[A_t^T \cdot W_t + W_t^T \cdot A_t\right] + \frac{\delta^2}{N^2} \cdot W_t^T \cdot W_t. \tag{P.2}$$

The last term can be neglected due to $\frac{\delta^2}{N^2}$ being significantly smaller than $\frac{\delta}{N}$, which leads to:

$$dG_t \triangleq G_{t+1} - G_t \approx -\frac{\delta}{N} \cdot \left[Q_t + Q_t^T\right], \tag{P.3}$$

where Q_t is $N \times N$ matrix whose i-th column is $A_t^T \cdot \mathcal{H}_t(X^i) \cdot A_t \cdot \bar{m}_t$.

Recently, similar expressions were reported by [25] (specifically, see Eq. (100-102)) and by [49].

APPENDIX Q

Computation Details of Fourier Transform

Here we provide more details on how Fourier Transform was calculated in our experiments. Consider a function $\varphi(X)$ and N dataset points $\boldsymbol{\mathcal{X}} = \{X^k \in \mathbb{R}^d\}_{k=1}^N$ sampled from an arbitrary pdf $P(X)$. Further, consider a $N \times 1$ vector $\bar{\varphi}$ with entries $\bar{\varphi}(k) = \varphi(X^k)$. Given $\bar{\varphi}$, we compute Fourier Transform $\hat{\varphi}(\xi)$ of a function $\varphi(X)$ at $\xi \in \mathbb{R}^d$ as following:

$$\hat{\varphi}(\xi) = \int \varphi(X) \cdot \exp\left[-2\pi i \cdot < \xi, X >\right] \cdot P(X) dX \approx$$
$$\approx \frac{1}{N} \sum_{k=1}^{N} \varphi(X^k) \cdot \exp\left[-2\pi i \cdot < \xi, X^k >\right] = \frac{1}{N} \bar{\varphi}^T \bar{\varepsilon}, \quad \text{(Q.1)}$$

where $\bar{\varepsilon}$ is a $N \times 1$ vector with entries $\bar{\varepsilon}(k) = \exp\left[-2\pi i \cdot < \xi, X^k >\right]$. Note that the above definition of Fourier Transform w.r.t. pdf $P(X)$ is identical to the common formulation without a term $P(X)$ inside, since in our experiments data distribution is $P(X) = 1$ (see "Setup" in Section 14.5).

In all our experiments we compute $\hat{\varphi}(\xi)$ for ξ taking values in $[-40, 40]^2$. Further, we present a frequency component $|\hat{\varphi}(\xi)|$ as an image.

To perform the above computation, we require sampled values $\bar{\varphi}$ of the analyzed function $\varphi(X)$. In case this function is the eigenfunction of *gradient similarity* kernel, the eigenvector of G_t approximates this eigenfunction' values at the training points, as is shown in the Appendix K. Hence, in this case the eigenvector of G_t serves as a vector $\bar{\varphi}$ in Eq. (Q.1). Likewise, the above calculation using the residual vector $\bar{m}_t$ can be considered as a Fourier Transform of a function $r(X) \triangleq f_{\theta_t}(X) - y(X)$.

Bibliography

[1] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

[2] Syed Mumtaz Ali and Samuel D Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1):131–142, 1966.

[3] T Amemiya. Asymptotic properties of extremum estimators. *Advanced econometrics, Harvard university press*, 1985.

[4] C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics*, 2:1152–1174, 1974.

[5] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[6] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[7] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019.

[8] Daniel Ashlock, Colin Lee, and Cameron McGuinness. Search-based procedural generation of maze-like levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):260–273, 2011.

[9] Adil M Bagirov, L Jin, N Karmitsa, A Al Nuaimat, and Napsu Sultanova. Subgradient method for nonconvex nonsmooth optimization. *Journal of Optimization Theory and applications*, 157(2):416–435, 2013.

[10] Leemon Baird, David Smalenberger, and Shawn Ingkiriwang. One-step neural network inversion with pdf learning and emulation. In *2005 IEEE International Joint Conference on Neural Networks, IJCNN'05*, volume 2, pages 966–971. IEEE, 2005.

[11] Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. *arXiv preprint arXiv:1906.00425*, 2019.

[12] James Vere Beck, Kevin David Cole, A Haji-Sheikh, and B Litkouhi. *Heat conduction using Green's functions*, volume 194. Hemisphere Publishing Corporation London, 1992.

[13] Ishmael Belghazi, Sai Rajeswar, Aristide Baratin, R Devon Hjelm, and Aaron Courville. Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062*, 2018.

[14] Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 400–406, 2000.

[15] Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. *arXiv preprint arXiv:1905.12173*, 2019.

[16] C.M. Bishop. Mixture density networks. Technical report, Aston University, Birmingham, 1994.

[17] Mathieu Blondel, André FT Martins, and Vlad Niculae. Learning with fenchel-young losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020.

[18] Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[19] Caffe. `caffe.berkeleyvision.org`.

[20] Gustavo Deco and Wilfried Brauer. Higher order statistical decorrelation without information loss. In *Advances in Neural Information Processing Systems (NIPS)*, pages 247–254, 1995.

[21] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[22] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[23] Xialiang Dou and Tengyuan Liang. Training neural networks as learning data-adaptive kernels: Provable representation and approximation benefits. *arXiv preprint arXiv:1901.07114*, 2019.

[24] Tarn Duong and Martin L Hazelton. Cross-validation bandwidth matrices for multivariate kernel density estimation. *Scandinavian Journal of Statistics*, 32(3):485–506, 2005.

[25] Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. *arXiv preprint arXiv:1909.11304*, 2019.

[26] Shinto Eguchi. Information divergence geometry and the application to statistical machine learning. In *Information theory and statistical learning*, pages 309–332. Springer, 2009.

[27] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *Intl. Conf. on Machine Learning (ICML)*, pages 881–889, 2015.

[28] Charles J Geyer and Elizabeth A Thompson. Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 657–699, 1992.

[29] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[30] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[31] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of machine learning research*, 12(Jul):2211–2268, 2011.

[32] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

[33] Dilan Görür and Carl Edward Rasmussen. Dirichlet process gaussian mixture models: Choice of the base distribution. *Journal of Computer Science and Technology*, 25(4):653–664, 2010.

[34] Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2007.

[35] Peter D Grünwald, A Philip Dawid, et al. Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory. *the Annals of Statistics*, 32(4):1367–1433, 2004.

[36] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5769–5779, 2017.

[37] Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

[38] Michael Gutmann and Jun-ichiro Hirayama. Bregman divergence as general framework to estimate unnormalized statistical models. *arXiv preprint arXiv:1202.3727*, 2012.

[39] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[40] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. of Machine Learning Research*, 13(Feb):307–361, 2012.

[41] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. Training dynamics of deep networks using stochastic gradient descent via neural tangent kernel. *arXiv preprint arXiv:1905.13654*, 2019.

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[43] Nils-Bastian Heidenreich, Anja Schindler, and Stefan Sperlich. Bandwidth selection for kernel density estimation: a review of fully automatic selectors. *AStA Advances in Statistical Analysis*, 97(4):403–433, 2013.

[44] G.E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[45] Geoffrey E Hinton. Products of experts. 1999.

[46] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[47] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of convex analysis*. Springer Science & Business Media, 2012.

[48] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[49] Jiaoyang Huang and Horng-Tzer Yau. Dynamics of deep neural networks and neural tangent hierarchy. *arXiv preprint arXiv:1909.08156*, 2019.

[50] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.

[51] Aapo Hyvarinen. Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables. *IEEE Transactions on neural networks*, 18(5):1529–1531, 2007.

[52] Aapo Hyvärinen. Some extensions of score matching. *Computational statistics & data analysis*, 51(5):2499–2512, 2007.

[53] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[54] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 8571–8580, 2018.

[55] Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. Perplexity - a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63, 1977.

[56] Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10(Jul):1391–1445, 2009.

[57] Takafumi Kanamori, Taiji Suzuki, and Masashi Sugiyama. Statistical analysis of kernel-based least-squares density-ratio estimation. *Machine Learning*, 86(3):335–367, 2012.

[58] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal statistics of fisher information in deep neural networks: mean field approach. *arXiv preprint arXiv:1806.01316*, 2018.

[59] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[60] D. Kopitkov and V. Indelman. No belief propagation required: Belief space planning in high-dimensional state spaces via factor graphs, matrix determinant lemma and re-use of calculation. *Intl. J. of Robotics Research*, 36(10):1088–1130, August 2017.

[61] D. Kopitkov and V. Indelman. Deep PDF: Probabilistic surface optimization and density estimation. *arXiv preprint arXiv:1807.10728*, 2018.

[62] D. Kopitkov and V. Indelman. Robot localization through information recovered from cnn classificators. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, October 2018.

[63] Dmitry Kopitkov and Vadim Indelman. Neural spectrum alignment: Empirical study. *arXiv preprint arXiv:1910.08720*, 2019.

[64] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[65] Matthieu Labeau and Alexandre Allauzen. Learning with noise-contrastive estimation: Easing training by learning to scale. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3090–3101, 2018.

[66] Louis Landweber. An iteration formula for fredholm integral equations of the first kind. *American journal of mathematics*, 73(3):615–624, 1951.

[67] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2011.

[68] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[69] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[70] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint*, 2016.

[71] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.

[72] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *NIPS*, pages 2378–2386, 2016.

[73] Qiang Liu and Dilin Wang. Learning deep energy models: Contrastive divergence vs. amortized mle. *arXiv preprint arXiv:1707.00797*, 2017.

[74] Cong Ma, Kaizheng Wang, Yuejie Chi, and Yuxin Chen. Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval, matrix completion, and blind deconvolution. *Foundations of Computational Mathematics*, pages 1–182, 2019.

[75] S. N. MacEachern and P. Muller. Estimating mixture of dirichlet process models. *Journal of Computational and Graphical Statistics*, 7:223–238, 1998.

[76] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Intl. Conf. on Computer Vision (ICCV)*, pages 2813–2821. IEEE, 2017.

[77] Hamed Masnadi-Shirazi and Nuno Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In *Advances in neural information processing systems*, pages 1049–1056, 2009.

[78] Takeru Matsuda, Masatoshi Uehara, and Aapo Hyvarinen. Information criteria for non-normalized models. *arXiv preprint arXiv:1905.05976*, 2019.

[79] G.J. McLachlan and K.E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, 1988.

[80] Aditya Menon and Cheng Soon Ong. Linking losses for density ratio and class-probability estimation. In *Intl. Conf. on Machine Learning (ICML)*, pages 304–313, 2016.

[81] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[82] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[83] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2265–2273, 2013.

[84] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

[85] Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

[86] Youssef Mroueh and Tom Sercu. Fisher gan. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2510–2520, 2017.

[87] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

[88] Hyunha Nam and Masashi Sugiyama. Direct density ratio estimation with convolutional neural networks with application in outlier detection. *IEICE TRANSACTIONS on Information and Systems*, 98(5):1073–1079, 2015.

[89] Amy Nesky and Quentin F Stout. Neural networks with block diagonal inner product layers. In *International Conference on Artificial Neural Networks*, pages 51–61. Springer, 2018.

[90] KW Newey and D McFadden. Large sample estimation and hypothesis. *Handbook of Econometrics, IV, Edited by RF Engle and DL McFadden*, pages 2112–2245, 1994.

[91] Jiquan Ngiam, Zhenghao Chen, Pang W Koh, and Andrew Y Ng. Learning deep energy models. In *International Conference on Machine Learning*, pages 1105–1112, 2011.

[92] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[93] XuanLong Nguyen, Martin J Wainwright, Michael I Jordan, et al. On surrogate loss functions and f-divergences. *The Annals of Statistics*, 37(2):876–904, 2009.

[94] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[95] Travis A O'Brien, Karthik Kashinath, Nicholas R Cavanaugh, William D Collins, and John P O'Brien. A fast and objective multidimensional kernel density estimation method: fastkde. *Computational Statistics & Data Analysis*, 101:148–160, 2016.

[96] Simon T O'Callaghan and Fabio T Ramos. Gaussian process occupancy maps for dynamic environments. In *Experimental Robotics*, pages 791–805. Springer, 2016.

[97] Yann Ollivier. Riemannian metrics for neural networks i: feedforward networks. *Information and Inference: A Journal of the IMA*, 4(2):108–153, 2015.

[98] Thomas R Osborn. Fast teaching of boltzmann machines with local inhibition. In *International Neural Network Conference*, pages 785–785. Springer, 1990.

[99] Samet Oymak, Zalan Fabian, Mingchen Li, and Mahdi Soltanolkotabi. Generalization guarantees for neural networks via harnessing the low-rank structure of the jacobian. *arXiv preprint arXiv:1906.05392*, 2019.

[100] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2338–2347, 2017.

[101] Hyeyoung Park, S-I Amari, and Kenji Fukumizu. Adaptive natural gradient learning algorithms for various stochastic models. *Neural Networks*, 13(7):755–764, 2000.

[102] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2:417, 2012.

[103] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[104] Henning Petzka, Asja Fischer, and Denis Lukovnicov. On the regularization of wasserstein gans. *arXiv preprint arXiv:1709.08894*, 2017.

[105] Miika Pihlaja, Michael Gutmann, and Aapo Hyvarinen. A family of computationally efficient and simple estimators for unnormalized statistical models. *arXiv preprint arXiv:1203.3506*, 2012.

[106] David Pollard. *A user's guide to measure theoretic probability*, volume 8. Cambridge University Press, 2002.

[107] Jose C Principe. *Information theoretic learning: Renyi's entropy and kernel perspectives*. Springer Science & Business Media, 2010.

[108] PyTorch. `pytorch.org`.

[109] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[110] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. *arXiv preprint arXiv:1806.08734*, 2018.

[111] Fabio Ramos and Lionel Ott. Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, 35(14):1717–1730, 2016.

[112] Mark D Reid and Robert C Williamson. Composite binary losses. *The Journal of Machine Learning Research*, 11:2387–2422, 2010.

[113] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[114] Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

[115] R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

[116] Basri Ronen, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. In *Advances in Neural Information Processing Systems*, pages 4761–4771, 2019.

[117] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[118] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2234–2242, 2016.

[119] Saeed Saremi, Arash Mehrjou, Bernhard Schölkopf, and Aapo Hyvärinen. Deep energy estimator networks. *arXiv preprint arXiv:1805.08306*, 2018.

[120] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[121] Ransalu Senanayake and Fabio Ramos. Building continuous occupancy maps with moving robots. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[122] Jayaram Sethuraman and Ram C Tiwari. Convergence of dirichlet measures and the interpretation of their parameter. In *Statistical decision theory and related topics III*, pages 305–315. Elsevier, 1982.

[123] Yi Shen. *Loss functions for binary classification and class probability estimation*. PhD thesis, University of Pennsylvania, 2005.

[124] Georgy Shevlyakov, Stephan Morgenthaler, and Alexander Shurygin. Redescending m-estimators. *Journal of Statistical Planning and Inference*, 138(10):2906–2917, 2008.

[125] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[126] Noah A Smith and Jason Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics, 2005.

[127] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, pages 194–281. The MIT press, Cambridge, MA, 1986.

[128] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[129] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[130] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012.

[131] Masashi Sugiyama, Taiji Suzuki, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008.

[132] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[133] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[134] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[135] TensorFlow. `www.tensorflow.org`.

[136] George R Terrell, David W Scott, et al. Variable kernel density estimation. *The Annals of Statistics*, 20(3):1236–1265, 1992.

[137] Masatoshi Uehara, Takeru Matsuda, and Fumiyasu Komaki. Analysis of noise contrastive estimation from the perspective of asymptotic variance. *arXiv preprint arXiv:1808.07983*, 2018.

[138] Masatosi Uehara, Issei Sato, Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. b-gan: Unified framework of generative adversarial networks. 2016.

[139] Unreal Engine. `www.unrealengine.com`.

[140] Benigno Uria, Iain Murray, and Hugo Larochelle. Rnade: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2175–2183, 2013.

[141] Manik Varma and Bodla Rakesh Babu. More generality in efficient multiple kernel learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1065–1072, 2009.

[142] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[143] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[144] Tinghua Wang, Dongyan Zhao, and Shengfeng Tian. An overview of kernel alignment and its applications. *Artificial Intelligence Review*, 43(2):179–192, 2015.

[145] Francis Williams, Matthew Trager, Claudio Silva, Daniele Panozzo, Denis Zorin, and Joan Bruna. Gradient dynamics of shallow univariate relu networks. *arXiv preprint arXiv:1906.07842*, 2019.

[146] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

[147] Sebastien C Wong, Adam Gatt, Victor Stamatescu, and Mark D McDonnell. Understanding data augmentation for classification: when to warp? *arXiv preprint arXiv:1609.08764*, 2016.

[148] Blake Woodworth, Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Kernel and deep regimes in overparametrized models. *arXiv preprint arXiv:1906.05827*, 2019.

[149] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995. IEEE, 2017.

[150] Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. Relative density-ratio estimation for robust distribution comparison. In *Advances in Neural Information Processing Systems (NIPS)*, pages 594–602, 2011.

[151] Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Zhang. Deep structured energy based models for anomaly detection. In *International Conference on Machine Learning*, pages 1100–1109, 2016.

[152] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[153] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[154] Zhiming Zhou, Yuxuan Song, Lantao Yu, Hongwei Wang, Weinan Zhang, Zhihua Zhang, and Yong Yu. Understanding the effectiveness of lipschitz-continuity in generative adversarial nets. *arXiv preprint arXiv:1807.00751*, 2018.

[155] Royce KP Zia, Edward F Redish, and Susan R McKay. Making sense of the legendre transform. *American Journal of Physics*, 77(7):614–622, 2009.