UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

Towards Uncovering the True Use of Unlabeled Data in Machine Learning

Emanuele Sansone

Advisor

Prof. Francesco G. B. De Natale

Università degli Studi di Trento

March 2018

Abstract

Knowing how to exploit unlabeled data is a fundamental problem in machine learning. This dissertation provides contributions in different contexts, including semi-supervised learning, positive unlabeled learning and representation learning. In particular, we ask (i) whether is possible to learn a classifier in the context of limited data, (ii) whether is possible to scale existing models for positive unlabeled learning, and (iii) whether is possible to train a deep generative model with a single minimization problem.

Semi-supervised learning, positive unlabeled learning, unsupervised learning, deep generative models, feedforward neural networks, Plummer autoencoders.

Acknowledgements

I have been thinking for long time whether to write the acknowledgements in this dissertation, since I believe that this is somehow out of my nature. Finally, I have decided to do that, because I consider this as an opportunity to thank explicitly the people that have been around me in these four years.

First and foremost, I want to thank Francesco De Natale, my advisor, for giving me the freedom to choose the topic that I like and never putting pressure on me. This is not as beautiful as it sounds, because I believe that freedom comes at the cost of higher responsibility. Therefore, I thank Francesco for giving the chance of improving my personal skills. Furthermore, I have to be grateful for his availability and his capability to listen to me and providing useful feedbacks. This has been a real lesson for me.

I want to thank also Nicola Conci, because he has been the first person to propose me to do the PhD and I cannot forget that.

I thank people from the MediaLab group for the working atmosphere and for being patient with me, especially when I had to run many simulations on GPU. I imagine that this has been quite annoying for them.

I thank Quoc-Tin Phan, Lam Thi Ngoc Tran and Duc-Tien Dang-Nguyen, my friends, for the time spent together and the long discussions we had in this long period. For sure, I will miss them.

Finally, I want to thank my family (my father, my mother, my sister and Lolly). If I'm here today, it's due to them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This manuscript consists of a collection of works from multiple areas of machine learning, including semi-supervised learning, positive unlabeled learning and representation learning. The main goal is to provide insights on the exploitation of unlabeled data to improve performance in classification tasks. From a wider perspective, this is closely related to the understanding of how humans are able to process the whole amount of information from their sensory inputs, to learn new concepts and make decisions. This is indeed one of the essential steps to pursue general artificial intelligence.

Each chapter provides an answer to a different aspect of the same problem. The presentation of each work follows the chronological order of the doctoral study, which corresponds also to an increasing level of contribution towards achieving general artificial intelligence. In particular, we provide an answer to the following research questions:

1. Chapter 2. How is it possible to learn in the context of limited data?

2. Chapter 3. Is it possible to exploit unlabeled data for improving one class classifiers and how is it possible to scale?

3. Chapter 4. Is it possible to convert the traditional mini-max problem of generative adversarial networks into a single minimization problem?

Answering to the first question can be beneficial to applications where data collection is very expensive. Consider for example the problem in medicine/biology of identifying connections between genetic profiles of patients and tumoral pathologies, for which it is difficult to collect a large number of clinical samples. Chapter 2 proposes a fully-Bayesian model capable to exploit the information provided by class labels together with statistical information extracted by clusters. This allows to cope with the limited amount of data and allows to boost the classification/clustering performance. The model is also applied to a challenging real-world problem of subgroup discovery in breast cancer, identifying highly promising subgroups.

Chapter 3 considers the problem of positive unlabeled learning, where supervision is given only for a single class of interest. This arises from applications like retrieval, outlier detection and open set recognition. We propose an algorithm that reduces the computational complexity of existing solutions, without sacrificing the generalization performance. We prove theoretically that the algorithm converges to the same solution obtained by state-of-the-art methods. As a consequence, the algorithm can be applied to a larger variety of real-world problems involving positive unlabeled learning.

Chapter 4 considers the problem of unsupervised learning, analyzing how to estimate a probability density function in high-dimensional feature spaces. The problem is formulated as the minimization of a novel cost function, which enables for (i) stable training and (ii) for convergence to optimal solutions. The proposed theory is validated through experimental comparisons against several methods from the families of generative adversarial networks and autoencoder-based models.

We conclude this dissertation with some final remarks on future research directions.

Chapter 2

Semi-Supervised Learning with Limited Data

We propose a novel parametric Bayesian model for the problem of semi-supervised classification and clustering. Standard approaches of semi-supervised classification can recognize classes but cannot find groups of data. On the other hand, semi-supervised clustering techniques are able to discover groups of data but cannot find the associations between clusters and classes. The proposed model can classify and cluster samples simultaneously, allowing the analysis of data in the presence of an unknown number of classes and/or an arbitrary number of clusters per class. Experiments on synthetic and real world data show that the proposed model compares favourably to state-of-the-art approaches for semi-supervised clustering and that the discovered clusters can help to enhance classification performance, even in cases where the cluster and the low density separation assumptions do not hold. We finally show that when applied to a challenging real-world problem of subgroup discovery in breast cancer, the method is capable of maximally exploiting the limited information available and identifying highly promising subgroups.

2.1 Background

Semi-supervised learning (SSL) is a well-known area of machine learning. The main idea is to exploit both unlabeled data to increase predictive performance of supervised models. This is motivated by the fact that labeled data are usually expensive to collect and unlabeled data may aid to learning. The SSL field encompasses both semi-supervised classification and semi-supervised clustering [1]. In the former case, the goal is predicting the labels of unlabeled data based on few observed labeled samples, and smoothness assumptions are typically used in developing methods. The latter task aims at finding clusters in data subject to some given supervised constraints, defined usually as must- and cannot-link between instances.

Discriminative approaches, like Semi-Supervised SVM [2] or Laplacian SVM [3], provide among the best performance in semi-supervised classification. This is because they focus on minimizing an objective function based on classification error, by directly learning the mapping function between the sample and the class space. As a drawback, they cannot provide precise information about intra-class variabilities, since they do not estimate the class-conditional densities. On the other hand, generative approaches learn the joint probability density function over inputs and labels and, while usually not as accurate in classification [1], allow one to model both inter- and intra-class structures.

Concerning semi-supervised clustering, existing algorithms are able to discover the patterns of input data, but they strongly rely on the assumption that clusters have a direct correspondence with the structure of classes, the so-called cluster assumption [1]. However, there are many real-world situations where this assumption does not hold [4]. In fact, if labeled classes split up in different sub-clusters or if several classes cannot be distinguished leading to one larger cluster, then all existing approaches fail. As we will see later on in the experiments, the cluster assumption is also not guaranteed when feature dimensionality reduction is applied to the data.

Existing SSL approaches typically focus on either classification or clustering. However, many real-world applications requires to *jointly* address both tasks by classifying data and identifying groups within each class. Medicine is a paradigmatic example of this requirement. Many diseases are characterized by symptoms for which the discrimination between healthy and pathological cases is often hard, due to the lack of complete understanding of the pathology. Moreover, since the signs of each disease may assume multiple forms, discriminating between the healthy and pathological conditions is not sufficient, and identifying also the different forms of the disease becomes crucial [5].

Based on all these considerations, we introduce a unified generative framework based on a mixture of factor analysers that jointly performs classification and reveals the hidden structure of data by estimating the modes and the factors of the class-conditional densities.[1] The framework only relies on the manifold assumption [1] and is thus able to deal with cases where the cluster assumption is not valid. Experiments on synthetic and real world data show that the proposed model compares favourably to state-of-the-art approaches for semi-supervised clustering and that the discovered clusters can help to enhance classification performance. We show also that the proposed model is designed to exploit maximally the limited available information and that it is particularly suited to applications where the collection of new data is very expensive, like in the case of breast

[1]Code available at `https://github.com/emsansone/Classtering`

cancer samples.

2.2 Related Work

Some other works based on finite mixture models are similar to ours, but differ in the kind of assumptions made. The work in [6] proposes a Gaussian mixture model that integrates the information about the presence/absence of labels to perform new class discovery. The model assumes that each cluster has a distribution over labels, but no information about the correspondences between classes and clusters is added to the generative model, thus making it dependent on the cluster assumption. In the experimental section we will see that this assumption is quite limiting for many cases. The work in [7] proposes a finite mixture model for semisupervised classification. In their generative model, labeled samples are conditioned on unlabeled ones in order to ensure that, during the inference stage, the propagation of labels through the unlabeled samples respects the smoothness assumption. The authors apply the method also to the unsupervised learning setting, in particular to perform density estimation. Nevertheless, the experimental evaluation highlights the limitations of the method in this kind of setting, where the results are frequently worse than performance obtained by standard unsupervised techniques. In the context of semi-supervised clustering, the works in [8, 9, 10, 11] have addressed the problem of constraint propagation proposing solutions that fulfill both the contraints and the smoothness requirement. Like the other works in semi-supervised classification, they haven't considered that the problem of label/constraint propagation may be due to the violation of the cluster assumption. The recent work in [12] is probably the closest to ours. The method introduces a finite mixture model able to deal with an arbitrary number of clusters and classes. The learning is performed by optimizing an objective characterized by the log-likelihood function weighted by a term penalizing the violation of the must- and cannot-link constraints. Furthermore, a hard assignment between clusters and classes determines a partitioning of the feature space in which the majority of the constraints is satisfied. In our approach, instead, the assignment between clusters and classes is soft. This is essential for modelling the uncertainty of assignment due to the small amount of supervised information. Furthermore, the method is tested on datasets characterized by only few dozens of features.

The authors in [13] have recently proposed a unified framework that combines deep neural networks with generative models. The neural network learns an embedding of data and the generative models performs classification based on this new representation. The combination of these two parts is obtained by definining a single probabilistic graphical model that achieves good classification performance even when compared to discriminative approaches. Nevertheless, the framework is not designed to perform clustering and is based on the assumption that there exists a data representation for which the cluster assumption is valid. Furthermore, the use of a deep neural network requires generally large datasets for training, besides having to choose the proper architecture, making the framework not suitable to applications with limited number of samples.

2.3 Proposed Solution

We start by introducing a fully-supervised model and then extend it to the semi-supervised case. Given a set of i.i.d. observations $Y = \{\boldsymbol{y}_n\}_{n=1}^N$, where $\mathbf{y}_n \in \mathbb{R}^d$, and the respective set of labels $C = \{c_n\}_{n=1}^N$, where c_n specifies that $\mathbf{y}_n$ belongs to one among K predefined classes, the goal is to learn the underlying distribution generating the observations, namely the class-conditional densities. In particular, if we assume that the densities can be approximated by a Gaussian mixture and that high-dimensional data vectors lie approximately on a lower dimensional subspace, then we can model the data distribution as a mixture of factor analysers (MFA).

In the MFA model, if a factor analyser s_n is given (s_n is an indicator variable identifying one among S factors), then each sample $\boldsymbol{y}_n$ is described through the following linear elation

$$\boldsymbol{y}_n = \mathbf{\Lambda}_{s_n}\mathbf{x}_n + \boldsymbol{\mu}_{s_n} + \boldsymbol{\xi}$$

where $\mathbf{x}_n \in \mathbb{R}^k$ is a latent vector distributed according to a Gaussian density with zero-mean and covariance equal to the identity matrix, $\mathbf{\Lambda}_{s_n} \in \mathbb{R}^{d\times k}$ and $\boldsymbol{\mu}_{s_n} \in \mathbb{R}^d$ are respectively the factor loading matrix and the bias of factor analyser s_n, and $\boldsymbol{\xi} \in \mathbb{R}^d$ is the noise distributed according to a normal density with diagonal covariance matrix defined by $\mathbf{\Psi}$. From this, it is not difficult to show that each sample $\boldsymbol{y}_n$ can be generated by sampling a Gaussian density with mean value equal to $\boldsymbol{\mu}_{s_n}$ and covariance matrix equal to $\mathbf{\Lambda}_{s_n}\mathbf{\Lambda}_{s_n}^T + \mathbf{\Psi}$ [14]. As a consequence, the MFA model can be equivalently interpreted as a Gaussian mixture. In this case, the vector of the mixing proportions is defined by the latent vector $\boldsymbol{\pi} \in [0,1]^S$.

It is worth noting that $\mathbf{\Lambda}_{s_n}$ incorporates information about the local dimensionality of component s_n, while $\mathbf{\Psi}$ models the variability of data inside that component, namely the noise variance. Parameters $\boldsymbol{\mu}_{s_n}$ and $\mathbf{\Lambda}_{s_n}$ are treated as random variables, such that inference is performed by averaging over the ensemble of models and therefore model complexity is automatically taken into account.

The MFA model is an unsupervised method that simultaneously addresses the problem of clustering and the problem of local dimensionality reduction. Supervision can be incorporated into this model by introducing for each sample $\boldsymbol{y}_n$ a pair of independent latent

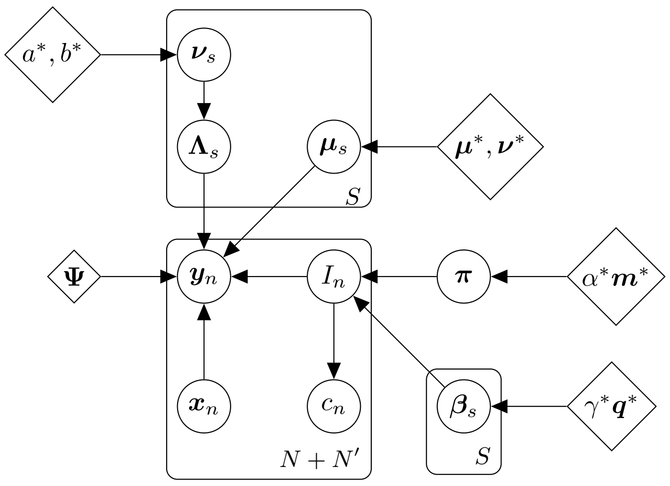


Figure 2.1: Representation of the supervised MFA model as a directed acyclic graph.

variables $I_n \doteq (s_n, l_n)$, where s_n is the above-mentioned cluster indicator, while l_n is the class indicator.[2] I_n takes into account all $S \times K$ possible combinations between the two indicators. It is worth saying that these combinations are not equally probable. In fact, if we assume that a cluster is associated more likely to one class, then some combinations of clusters and classes tend to appear more often than others. The mixing proportions for variable l_n are therefore defined by the set of random vectors $B = \{\boldsymbol{\beta}_s\}_{s=1}^{S}$, where each K-dimensional $\boldsymbol{\beta}_s$ is governed by a Dirichlet prior. This means that estimating the distribution over B is equivalent to learning the probabilistic associations between clusters and classes.

The complete set of conditional distributions and priors of our model is summarized by the following relations:

$$
\begin{aligned}
p(\boldsymbol{x}_n) &\doteq \mathcal{N}(0, \mathbf{I}) \\
p(c_n|I_n) &\doteq \delta(c_n - l_n) \\
p(\boldsymbol{\beta}_s|\gamma^* \boldsymbol{q}^*) &\doteq Dir(\gamma^* \boldsymbol{q}^*) \\
p(\boldsymbol{\pi}|\alpha^* \boldsymbol{m}^*) &\doteq Dir(\alpha^* \boldsymbol{m}^*) \\
p(\boldsymbol{\Lambda}_s|\boldsymbol{\nu}_s) &\doteq \prod_{j=1}^{k} \mathcal{N}(0, \mathbf{I}/\boldsymbol{\nu}_s(j)) \\
p(I_n|\boldsymbol{\pi}, \{\boldsymbol{\beta}_s\}_{s=1}^{S}) &\doteq \boldsymbol{\pi}(s_n)\boldsymbol{\beta}_{s_n}(l_n) \\
p(\boldsymbol{\mu}_s|\boldsymbol{\mu}^*, \boldsymbol{\nu}^*) &\doteq \mathcal{N}(\boldsymbol{\mu}^*, diag(\boldsymbol{\nu}^*)^{-1})
\end{aligned}
$$

[2]In our case, there is no distinction between l_n and c_n. Nevertheless, we keep these two variables separate. This is helpful for modelling scenarios with multiple and/or noisy labels. In these cases, l_n is the hidden true label, while c_n is the label provided by the annotator.

$$p(\boldsymbol{\nu}_s|a^*, b^*) \doteq \prod_{j=1}^{k} Gamma(\boldsymbol{\nu}_s(j)|a^*, b^*)$$

where $\mathbf{I}$ is the identity matrix, $\delta(\cdot)$ is the delta function and $\boldsymbol{\nu}_s$ is a k-dimensional vector whose elements govern the columns of $\boldsymbol{\Lambda}_s$. The mechanism known as automatic relevance determination (ARD) is used to improve the task of dimensionality reduction [15]. $a^*, b^*, \boldsymbol{\Psi}, \boldsymbol{\mu}^*, \boldsymbol{\nu}^*, \gamma^*\boldsymbol{q}^*$ are the hyperparameters of the model. In Figure 2.1, we show the graphical representation of our probabilistic model.

In the next two sections, we see how to apply this probabilistic graphical model to the semi-supervised scenario. In particular, a variational approximation of the log-likelihood function over input data and labels is derived in order to make inference computationally tractable. The unlabeled data are therefore taken into account by simply adding their contribution to the estimated lower bound. Then, we show how to predict the labels of unseen data.

2.3.1 Variational Approximation

By defining $\mathcal{H} \doteq \{\boldsymbol{x}_n, I_n\}$ as the set of hidden variables and $\Theta \doteq \{\boldsymbol{\pi}, \{\boldsymbol{\beta}_s, \boldsymbol{\Lambda}_s, \boldsymbol{\mu}_s, \boldsymbol{\nu}_s\}_{s=1}^{S}\}$ as the set of parameters, we can express the log-likelihood function over Y and C as

$$\ln p(Y, C) = \ln \int d\Theta p(\Theta) \int d\mathcal{H} p(Y, C, \mathcal{H}|\Theta)$$

and by exploiting the conditional dependencies defined by the probabilistic graphical model we obtain that

$$\ln p(Y,C) = \ln \int d\Theta p(\Theta) \prod_{n=1}^{N} \sum_{s_n=1}^{S} \sum_{l_n=1}^{K} p(I_n|\Theta) p(c_n|I_n) \int d\boldsymbol{x}_n p(\boldsymbol{x}_n) p(\boldsymbol{y}_n|\Theta, \boldsymbol{x}_n, I_n, \boldsymbol{\Psi}) \quad (2.1)$$

Since the integrals in (2.1) are computationally and analytically intractable, we employ a standard approach to solve the Bayesian integration based on the variational approximation [16]. In practice, by introducing some auxiliary distributions for both the parameters and the hidden variables and by applying the Jensen's inequality, it is possible to obtain a lower bound on the log-likelihood over Y and C, namely

$$\begin{aligned}\ln p(Y, C) \geq & \int d\boldsymbol{\pi} q(\boldsymbol{\pi}) \ln \frac{p(\boldsymbol{\pi}|\alpha^*\boldsymbol{m}^*)}{q(\boldsymbol{\pi})} + \sum_{s=1}^{S} \int d\boldsymbol{\beta}_s q(\boldsymbol{\beta}_s) \ln \frac{p(\boldsymbol{\beta}_s|\gamma^*\boldsymbol{q}^*)}{q(\boldsymbol{\beta}_s)} \\ & + \sum_{s=1}^{S} \int d\boldsymbol{\nu}_s q(\boldsymbol{\nu}_s) \left[\ln \frac{p(\boldsymbol{\nu}_s|a^*, b^*)}{q(\boldsymbol{\nu}_s)} + \int d\tilde{\boldsymbol{\Lambda}}_s q(\tilde{\boldsymbol{\Lambda}}_s) \ln \frac{p(\tilde{\boldsymbol{\Lambda}}_s|\boldsymbol{\nu}_s, \boldsymbol{\mu}^*, \boldsymbol{\nu}^*)}{q(\tilde{\boldsymbol{\Lambda}}_s)} \right] \\ & + \sum_{n=1}^{N} \sum_{s_n=1}^{S} \sum_{l_n=1}^{K} q(I_n) \left[\int d\boldsymbol{\pi} q(\boldsymbol{\pi}) \int d\boldsymbol{\beta}_{s_n} q(\boldsymbol{\beta}_{s_n}) \ln \frac{p(I_n|\boldsymbol{\pi}, \boldsymbol{\beta}_{s_n})}{q(I_n)} \right.\end{aligned}$$

$$+ \int d\boldsymbol{x}_n q(\boldsymbol{x}_n|I_n) \ln \frac{p(\boldsymbol{x}_n)}{q(\boldsymbol{x}_n|I_n)} + \ln p(c_n|I_n) \int d\tilde{\boldsymbol{\Lambda}}_{s_n} q(\tilde{\boldsymbol{\Lambda}}_{s_n}) \int d\boldsymbol{x}_n q(\boldsymbol{x}_n|I_n)$$
$$\cdot \ln p(\boldsymbol{y}_n|\tilde{\boldsymbol{\Lambda}}_{s_n}, \boldsymbol{x}_n, I_n, \boldsymbol{\Psi}) \Bigg] \doteq \mathcal{F}\Big(\mathcal{Q}\Big) \tag{2.2}$$

where $\mathcal{Q}$ is the set of all auxiliary distributions, namely $q(\boldsymbol{\pi})$, $\{q(\boldsymbol{\beta}_s), q(\boldsymbol{\nu}_s), q(\tilde{\boldsymbol{\Lambda}}_s)\}_{s=1}^{S}$, $\{q(I_n), q(\boldsymbol{x}_n|I_n)\}_{n=1}^{N}$, and $\tilde{\boldsymbol{\Lambda}}_s$ represents the concatenation between $\boldsymbol{\Lambda}_s$ and $\boldsymbol{\mu}_s$. By maximizing the functional $\mathcal{F}$, the lower bound is guaranteed to monotonically increase [17] and can be used as an approximation of the log-likelihood function over Y and C. Furthermore, the functional $\mathcal{F}$ is used to compare models with different number of factor analysers in order to perform automatic model selection and choose the proper value of S.

The model can be further extended to perform semi-supervised classification by introducing the set of unlabeled observations $Y' = \{\boldsymbol{y}'_m\}_{m=1}^{N'}$ and by averaging over all possible labels. The extended log-likelihood function is therefore approximated following the same procedure in (2.2), namely

$$\ln p(Y, Y', C) \geq \mathcal{F}\Big(\mathcal{Q}\Big) + \sum_{m=1}^{N'} \sum_{s_m=1}^{S} \sum_{l_m=1}^{K} q(I_m) \Bigg[\int d\boldsymbol{\pi} q(\boldsymbol{\pi}) \int d\boldsymbol{\beta}_{s_m} q(\boldsymbol{\beta}_{s_m}) \ln \frac{p(I_m|\boldsymbol{\pi}, \boldsymbol{\beta}_{s_m})}{q(I_m)}$$
$$+ \int d\boldsymbol{x}_m q(\boldsymbol{x}_m|I_m) \ln \frac{p(\boldsymbol{x}_m)}{q(\boldsymbol{x}_m|I_m)} + \int d\tilde{\boldsymbol{\Lambda}}_{s_m} q(\tilde{\boldsymbol{\Lambda}}_{s_m}) \int d\boldsymbol{x}_m q(\boldsymbol{x}_m|I_m)$$
$$\cdot \ln p(\boldsymbol{y}'_m|\tilde{\boldsymbol{\Lambda}}_{s_m}, \boldsymbol{x}_m, I_m, \boldsymbol{\Psi}) \Bigg] \tag{2.3}$$

which is equivalent to (2.2) except for the last three addends, that represent the contribution of unlabeled samples to the the lower bound.

2.3.2 Posterior Inference and Prediction

Posteriors over parameters and hidden variables are estimated by optimizing the functional in (2.3). The optimization is performed by taking the functional derivatives of (2.3) with respect to all auxiliary distributions $q(\cdot)$ and equating them to zero. Similarly, the hyperparameters of the model are estimated by simply taking the derivatives of the lower bound in (2.3) with respect to $a^*, b^*, \boldsymbol{\Psi}, \boldsymbol{\mu}^*, \boldsymbol{\nu}^*, \gamma^* \boldsymbol{q}^*$. This operation is equivalent to performing a maximum likelihood estimation, where the true log-likelihood function is replaced by its lower bound. Iterative updates of the auxiliary distributions and of the hyperparameters guarantee to monotonically and maximally increase the lower bound in (2.3), as shown in [17].

After the optimization is completed, the model can be used to predict the labels of new observed samples. In fact, if we define $D = Y \cup Y'$ as the set of data used for

Table 2.1: Experimental datasets.

Datasets	G50C	CAKE	TOES	IRIS	USPS	ISOLET
Classes	2	2	2	3	3	2
Features	50	2	2	4	256	617
Instances	550	1000	1000	150	1918	3119

training the model and $Y'' = \{\boldsymbol{y}''_j\}_{j=1}^M$ as the set of test data, then the new labels can be estimated by maximizing the log-likelihood function conditioned on D and C. During the maximization, $\ln p(Y''|D,C)$ can be approximated by replacing the true parameter posterior with the estimated auxiliary distribution over the parameters, namely

$$\begin{aligned}\ln p(Y''|D,C) &= \ln \int d\Theta p(\Theta|D,C) \int d\mathcal{H} p(Y'',\mathcal{H}|\Theta,D,C) \\ &\approx \ln \int d\Theta q(\Theta) \int d\mathcal{H} p(Y'',\mathcal{H}|\Theta,D,C) \qquad (2.4)\end{aligned}$$

Integrals in (2.4) are computationally intractable. Similarly to the (2.2) and (2.3) cases, we thus look for a tractable lower bound on $\ln p(Y''|D,C)$

$$\begin{aligned}\ln p(Y''|D,C) \geq &\sum_{j=1}^{M} \sum_{s_j=1}^{S} \sum_{l_j=1}^{K} q(I_j) \Bigg[\int d\boldsymbol{\pi} q(\boldsymbol{\pi}) \int d\boldsymbol{\beta}_{s_j} q(\boldsymbol{\beta}_{s_j}) \ln \frac{p(I_j|\boldsymbol{\pi},\boldsymbol{\beta}_{s_j})}{q(I_j)} \\ &+ \int d\boldsymbol{x}_j q(\boldsymbol{x}_j|I_j) \ln \frac{p(\boldsymbol{x}_j)}{q(\boldsymbol{x}_j|I_j)} \\ &+ \int d\tilde{\boldsymbol{\Lambda}}_{s_j} q(\tilde{\boldsymbol{\Lambda}}_{s_j}) \int d\boldsymbol{x}_j q(\boldsymbol{x}_j|I_j) \ln p(\boldsymbol{y}''_j|\tilde{\boldsymbol{\Lambda}}_{s_j},\boldsymbol{x}_j,I_j,\boldsymbol{\Psi}) \Bigg] \qquad (2.5)\end{aligned}$$

Note that (2.5) is similar to the last three addends of (2.3). In this case, we are only interested in estimating the labels of test data and this is performed by taking the functional derivatives of (2.5) with respect to $q(I_j)$ for $j = 1, \ldots, M$.

2.4 Experiments

In order to assess the performance of the proposed model and compare it with state-of-the-art approaches, we performed experiments on three artificial and three real world datasets. Table 2.1 summarizes their properties.

The first synthetic dataset, G50C, is inspired by [18]. Data are generated from two standard normal densities located in a 50-dimensional space, such that the Bayes error is 5%. In this case, each class is represented by only one Gaussian. In the second dataset, CAKE, data are uniformly distributed according to a two-dimensional round shape. Two orthogonal decision functions are used to discriminate between the two classes in order to make them non-linearly separable. The Gaussian and the cluster assumptions do not hold in this case. The third dataset, TOES, represents the case where class-conditional densities are characterised by multiple clusters. Samples are drawn independently from a two-dimensional density composed by five Gaussians, two for the first class and three for the second class. The two classes have the same prior, resulting into a balanced number of samples per class. The different number of clusters per class is useful to analyse how unlabeled data influence the decision boundary. Figures 2.3(a) and 2.4(a) show the representation of the CAKE and the TOES datasets respectively.

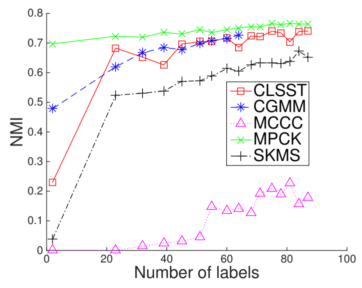
The real world datasets consist of two-class and multi-class problems from the UCI repository. The IRIS dataset contains data belonging to three different classes of iris plants. One of the three classes is not linearly separable from the others. The second real world data set, USPS, represents a well-known benchmark for hadwritten digits recognition. In our experiments, we only used samples belonging to the categories of digits 3, 8 and 9, which are among the most difficult classes to recognize [19]. In order to deal with real-valued vectors, normalized histograms are used as feature descriptors. Finally, the ISOLET dataset contains high-dimensional data for the spoken letter recognition task. In our case, the first three subsets of the whole collection were considered. Similarly to [3], we decided to classify the first 13 letters of the English alphabet from the last 13.

For the USPS and ISOLET datasets, we first apply a state-of-the-art technique for unsupervised dimensionality reduction, called t-SNE [20]. The motivation for the choice of t-SNE relies on the capability of visualizing high-dimensional datasets in a two or three-dimensional map without loosing too much information about the local and the global structure of data. Compared to other existing techniques, like Sammon mapping, Isomap and Locally Linear Embedding, t-SNE provides significantly better performance, especially in the data visualization task.

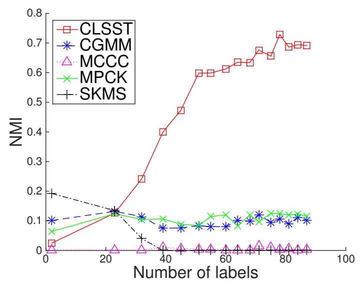
2.4.1 Semi-Supervised Clustering

For each dataset, the number of labeled instances is varied between 0 and 90 samples per class. For each of these configurations, 20 different datasets are generated by random sampling. To adhere to the problem of semi-supervised clustering, labeled samples are then converted into a balanced number of must- and cannot-link constraints following the same procedure of [21]. Performance is measured in terms of the normalized mutual information (NMI) using the true labels as gold standard.

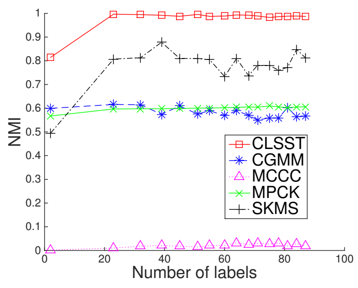
We compare our method, called Classtering (CLSST for short), with four state-of-the-art approaches. The first method proposed in [22] is based on the integration of supervised constraints into a Gaussian mixture model (CGMM). The second method (MCCC) is the



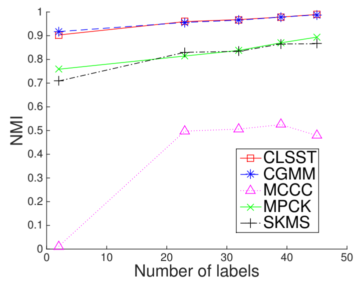
(a) G50C



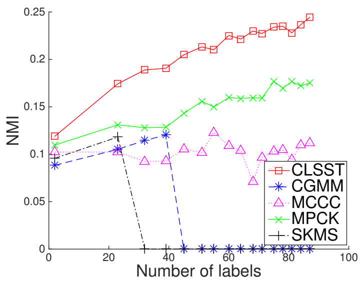
(b) CAKE



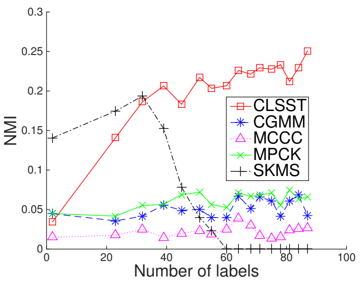
(c) TOES



(d) IRIS



(e) USPS



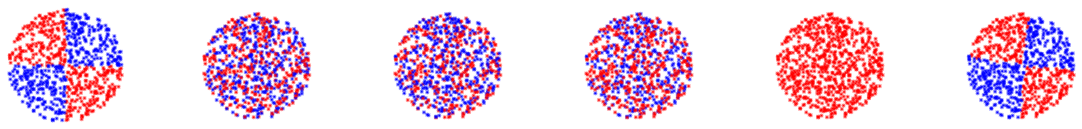
(f) ISOLET

Figure 2.2: Experimental results for semi-supervised clustering (the higher the better).

recent work proposed in [23], where the problem is formulated as a matrix completion task. The third method in [24] is based on an extension of the k-means algorithm (MPCK), where constraints and metric learning are incorporated into the objective function to enhance the performance. The last method is the semi-supervised kernel mean shift (SKMS) proposed in [21], where data are first mapped into a higher dimensional space and then clustered by the mean shift algorithm.

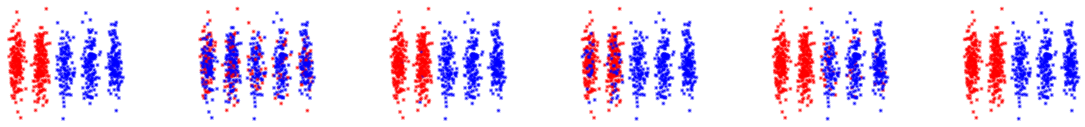
For all competitors, the parameters are chosen from a finite grid set such that the best performance are always considered. In particular, the tradeoff parameter C for MCCC is chosen from the set $\{0.1, 1, 10, 100, 1000\}$, while the regularization parameter γ of SKMS is chosen from the range $\{10, 100, 1000\}$. It is important to mention that the number of clusters for MCCC, MPCK and CGMM is equal to four in CAKE and five in TOES, while it is chosen to be equal to the number of classes in all other datasets, as done in [21, 22, 24, 23]. It is also worth noting that our algorithm does not require to set any parameter manually, since all hyperparameters are learnt automatically during the training procedure.[3] Figure 2.2 shows the results obtained over all datasets. CLSST clearly outperforms all competitors in all cases, except for the G50C dataset. In this case, the data are generated from a distribution of two Gaussians with identity covariance

[3]Except for the dimensionality of the latent variables x_n, which is always set to a low value ($k = 2$ for G50C, CAKE and TOES and $k = 3$ for IRIS, USPS and ISOLET).



(a) Labels (b) MCCC (c) MPCK (d) CGMM (e) SKMS (f) CLSST

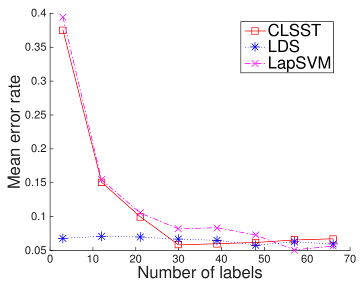
Figure 2.3: Estimated labels for semi-supervised clustering on CAKE dataset (87 labels per class)



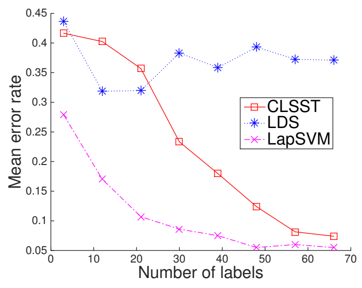
(a) Labels (b) MCCC (c) MPCK (d) CGMM (e) SKMS (f) CLSST

Figure 2.4: Estimated labels for semi-supervised clustering on TOES dataset (87 labels per class)

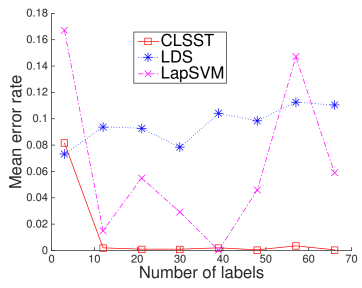
matrices. Authors in [25] prove mathematically that the k-means algorithm is equivalent to performing an EM algorithm on a mixture of Gaussians under the assumption of identity covariance matrices and uniform mixture priors, which clearly motivates why MPCK, that is k-means-based, achieves very good performance. The gap with respect to the results obtained by CLSST on G50C are mainly due to the fact that, while in CLSST the parameters of the Gaussians are assumed to be random variables, in MPCK it is assumed that there only exists a unique combination of true parameters. It is worth mentioning that CLSST and CGMM are both algorithms based on Gaussian mixtures. In fact, when considering cases characterized by one cluster per class, namely the G50C and the IRIS datasets, the performance of both methods are almost equivalent. When the cluster assumption does not hold, viz in the CAKE dataset, it is clearly visible that all methods, except CLSST, fail. The same holds when considering multiple clusters per class, i.e. the TOES dataset. A representative example of the results obtained on CAKE and TOES can be visualized more intuitively in Figure 2.3 and Figure 2.4. A thorough analysis of the results obtained on the USPS and the ISOLET datasets performed by data visualization indicates that in both cases we have a superposition of two effects, namely the absence of validity of the cluster assumption and the presence of multiple clusters per class, which explains why results are qualitatively similar to those obtained on the CAKE and the TOES datasets.



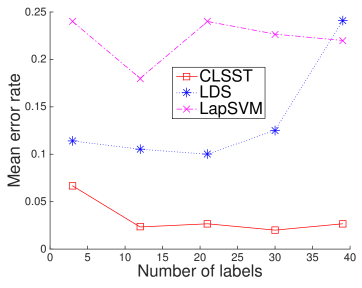
(a) G50C



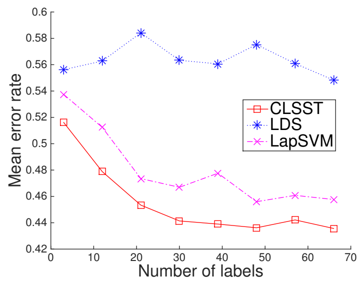
(b) CAKE



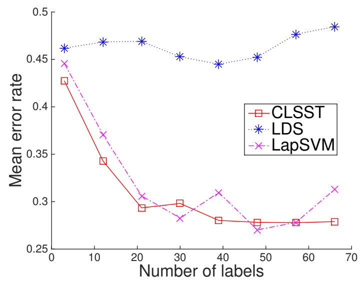
(c) TOES



(d) IRIS



(e) USPS



(f) ISOLET

Figure 2.5: Experimental results for semi-supervised classification (the lower the better).

2.4.2 Semi-Supervised Classification

For each dataset, 5-fold cross-validation is used to split data into training and test parts. For each training split, the number of labeled instances is varied between 0 and 70 samples per class. 10 different datasets are then generated by random sampling. In these experiments, we provide estimates of the generalization error. Performance are measured in terms of the error rate, since all datasets have a balanced number of samples per class.

Our method is compared against two state-of-the-art approaches. The first method [26] is based on the low-density separation assumption (LDS), which is the equivalent supervised form of the cluster assumption. A nearest-neighbor graph is used to compute the kernel matrix of an SVM. The second method proposed in [3] is an extension of the SVM framework, namely the Laplacian SVM (LapSVM). In particular, a penalty term is added to the objective function to take into account the marginal distribution of unlabeled data.

For each training dataset, hyperparameters for the two competitors are selected from a finite grid using an inner 3-fold cross-validation procedure. For LDS, ρ and C are respectively chosen from $\{1, 2, 4, 8\}$ and $\{0.1, 1, 10, 100\}$. In LapSVM, γ_A and γ_I are chosen from the same range, namely $\{0.005, 0.045, 05\}$. The σ value is chosen for both methods in the range $\{0.1, 1, 10\}$ in a transductive setting on the entire training set.

Table 2.2: Results on breast cancer dataset evaluated in terms of IGP measure. Clusters are ordered by decreasing IGP value.

Cluster	1	2	3	4	5	6	7	8	9	10
[**27**]	0.824	0.810	0.728	0.709	0.687	0.646	0.602	0.582	0.507	0.448
CLSST*	0.927	0.864	0.790	0.687	0.684	0.679	0.678	0.600	0.597	0.557
CLSST$^+$	0.912	0.838	0.793	0.773	0.762	0.583	0.575	-	-	-

CLSST*: fixed number of components S.

CLSST$^+$: variable number of components S.

Figure 2.5 shows the results obtained over all data sets.

CLSST outperforms all other approaches in almost all cases, except for the G50C and the CAKE datasets. G50C is in fact the perfect scenario for approaches relying on the low density separation assumption, as it was previously seen in the clustering setting for the methods based on the cluster assumption. This motivates why LDS provides good performance even when the number of labeled samples is small.

In the CAKE dataset, CLSST performs slightly worse than LapSVM. This is due to the fact that the Gaussian mixture density is not able to fit properly with the uniform distribution of unlabeled samples. The bias decreases as the number of labeled samples increases. In contrast, LapSVM is able to control the negative effect of the unlabeled samples even when the number of labeled data is small by assigning a higher value to the classification error term in the objective function. In all other cases, it is evident that CLSST achieves better performance than its competitors. This can be explained by the fact that our model is really flexible in estimating the class-conditional densities and the gained information about these distributions provides an effective way to fully exploit the unlabeled samples and increase the classification accuracy.

2.4.3 Subgroup Discovery in Breast Cancer

We finally tested our algorithm on a challenging real-world problem consisting in the identification of subgroups in breast cancer samples. A recent extensive study [27] analysed about 2,000 clinically annotated primary breast cancers collected from various sources and identified 10 novel subgroups with varying degrees of confidence. The authors used a subset of 997 samples as discovery set to identify clusters, and the remaining 995 ones as validation set to evaluate robustness of the detected clusters. Clustering was done with a joint latent variable model [28] on a set of 754 gene expression profiles. Reproducibility of clustering was measured in terms of in-group proportion (IGP) [29], which is the proportion of samples in a group whose nearest neighbours are also in the same group, after assigning samples in the validation set to the clusters in the discovery set.

Characterizing tumors in terms of subclasses is a crucial step in order to understand their behaviour and variability, and there is extensive literature addressing this task and proposing various classication schemes. Five “intrinsic” subtypes of human breast tumors have been identified in early studies [30] and termed Luminal A, Luminal B, HER2-Enriched (HER2-E), Basal-like and normal. The PAM50 gene is typically used [31] for gene expression-based subtyping in these five groups. Most of the 10 clusters identified in [27] contain samples belonging to multiple subtypes.

What we investigate here is whether incorporating intrinsic subtype classification as class labeling can produce a clustering with improved generalization capability, as measured by IGP. We first reduce data to 50 features using PCA in order to alleviate the problem of redundant features and then apply CLSST to discover the clusters. In this particular setting, the algorithm discovers seven groups achieving an averaged IGP of 74.8%, with a minimum value of 57.5% and a maximum of 91.2%. After this, we investigate a second setting, where we run CLSST by fixing the number of clusters to ten, in order to have a fair comparison with the results reported in [28]. In this second configuration, the obtained IGP scores range from a minimum of 55.7% to a maximum of 92.7% with a mean value equal to 70.6%. In both settings, the performance are better than those obtained in [28], where the IGP values span from a minimum of 44.8% to a maximum of 82.4% with a mean value equal to 65.4%. The performance improvement is on average greater than 5%, indicating that our algorithm successfully exploits the supervised information in performing group discovery. Table 2.2 reports the complete set of results for [28] and for CLSST in the two settings, with clusters ordered by decreasing IGP value.

With this study we are not claiming that the clusters we found are more biologically relevant than those identified by the original method, as this would require in-depth analyses and extensive validations, which are out of the scope of this work. Nonetheless, we believe that the obtained results are promising and highlight the potential of the method in discovering structure in data.

Chapter 3

Scalability in Positive Unlabeled Learning

Positive unlabeled (PU) learning is useful in various practical situations, where there is a need to learn a classifier for a class of interest from an unlabeled data set, which may contain anomalies as well as samples from unknown classes. The learning task can be formulated as an optimization problem under the framework of statistical learning theory. Recent studies have theoretically analyzed its properties and generalization performance, nevertheless, little effort has been made to consider the problem of scalability, especially when large sets of unlabeled data are available. In this work we propose a novel scalable PU learning algorithm (USMO – Unlabeled data in Sequential Minimal Optimization) that is theoretically proven to provide the optimal solution, while showing superior computational and memory performance. Experimental evaluation confirms the theoretical evidence and shows that the proposed method can be successfully applied to a large variety of real-world problems involving PU learning.

3.1 Background

PU learning refers to the task of learning a binary classifier from only positive and unlabeled data [32]. This classification problem arises in various practical situations, such as:

- Retrieval [33], where the goal is to find samples in an unlabeled data set similar to user-provided ones.
- Inlier-based outlier detection [34], where the goal is to detect outliers from an unlabeled data set, based on inlier samples.

- One-vs-rest classification [35], where the negative class is too diverse, thus being difficult to collect and label enough negative samples.
- Open set recognition [36], where testing classes are unknown at training time and the exploitation of unlabeled data may help learning more robust concepts.

Naive approaches are proposed to address PU learning. In particular, it is possible to distinguish between solutions that heuristically identify reliable negative samples from unlabeled data and use them to train a standard binary classifier, and solutions based on binary classifiers using all unlabeled data as negative. The former are heavily dependent on heuristics, while the latter suffer the problem of wrong label assignment.

The recent works in [37] and [38] formulate PU learning as an optimization problem under the framework of statistical learning theory [39]. Both works theoretically analyse the problem, deriving generalization error bounds and studying the optimality of the obtained solutions. Even though these methods are theoretically grounded, they are not scalable. In this work we present a method that provides better scalability, while maintaining the optimality of the above approaches for what concerns the generalization. In particular, starting from the formulation of the convex optimization problem in [38], we derive an algorithm that requires significantly lower memory and computation, while being proven to converge to the optimal solution.

3.2 Related Work

PU learning is well known in the machine learning community, as it is used in a variety of tasks such as matrix completion [40], multi-view learning [41], and semi-supervised learning [42]. It is also applied in data mining to classify data streams [43] or time series [44] and to detect events, like co-occurrences, in graphs [45].

PU learning approaches can be classified in two broad categories, according to the use of unlabeled data: two-stage and single-stage approaches. The former extract a set of reliable negative samples from unlabeled data and use them, together with the available positive data, to train a binary classifier [46, 47, 48, 49]. This first step is heuristic and strongly influences the final result. The latter regard **all unlabeled data as negative samples**. Positive and negative data are then used to train different classifiers based on SVM [50, 32], neural networks [51] or kernel density estimators [52]. These approaches suffer the problem of **wrong label assignment**, whose effect depends on the proportion of positive samples in the unlabeled dataset. We will see later, in the discussion about theoretical studies of PU learning, how critical this issue is. For the moment, we focus on analyzing the relations of PU learning with one-class classification (OCC) and semi-supervised learning, which allows us drawing some clear boundaries between these tasks and highlighting the novelty of this work.

3.2.1 Comparison with One-Class Classification

The main goal of OCC is to estimate the support of data distribution, which is extremely useful in unsupervised learning, especially in high-dimensional feature spaces, where it is very difficult to perform density estimation. OCC is applied to many real-world problems, such as anomaly/novelty detection (see [53] for a recent survey and definition of anomaly detection and [54, 55] for novelty detection). Other possible applications of OCC include author verification in text documents [56], document retrieval [33], and collaborative filtering in social networks [57].

Authors of [58, 59] are among the first to develop OCC algorithms.[1] In particular, [58] proposes a classifier that finds the hyperplane separating data from the origin with the maximum margin, while [59] proposes a classifier that finds the mimimum radius hypersphere enclosing data. Despite the difference between these two approaches, [58] proves that, for translation-invariant kernels such as the Gaussian kernel, they obtain the same solutions. Extensions of these two pioneering works, falling in the category of kernel methods, were proposed a few years later. [61] modifies the model of [58] by incorporating a small training set of anomalies and using the centroid of such set as the reference point to find the hyperplane. [62] proposes a strategy to automatically select the hyperparameters defined in [59] to increase the usability of the framework. Rather than repelling samples from a specific point, as in [58, 61], [63] suggests a strategy to attract samples towards the centroid, solving a linear programming problem that minimizes the average output of the target function computed on the training samples. Authors in [64] propose a similar strategy based on linear programming, where data are represented in a similarity/dissimilarity space. The framework is well suited for OCC applications involving strings, graphs or shapes. Solutions different from kernel methods are also proposed. To mention a few, [65] proposes a neural network-based approach, where the goal is to learn the identity function. New samples are fed into the network and tested against their corresponding outputs. The test sample is considered as part of the class of interest only when the input and the output of the network are similar. [66] proposes a one-class nearest neighbour, where a test sample is accepted as a member of the target class only when the distance from its neighbours is comparable to their local density. **It is worth noting that most of the works in OCC focus on increasing classification performance, rather than improving scalability**. This is arguably motivated by the fact that it is usually difficult to collect large amounts of training samples

[1]More precisely, the term OCC was coined in 1996 [60].

for the concept/class of interest. Solutions to improve classification performance rely on classical strategies such as ensemble methods [67], bagging [68] or boosting [69]. Authors in [70] argue that existing one-class classifiers fail when dealing with mixture distributions. Accordingly, they propose a multi-class classifier exploiting the supervised information of all classes of interest to refine support estimation.

A promising solution to improve OCC consists of exploiting unlabeled data, which are in general largely available. As discussed in [52], standard OCC algorithms are not designed to use unlabeled data, thus making the implicit assumption that they are uniformly distributed over the support of nominal distribution, which does not hold in general. The recent work in [71] proves that, under some simple conditions,[2] large amounts of unlabeled data can boost OCC even compared to fully supervised approaches. Furthermore, unlabeled data allow building OCC classifiers in the context of open set recognition [36], where it is essential to learn robust concepts/functions. Since the primary goal of PU learning is to exploit this unsupervised information, **it can be regarded as a generalization of OCC** [72], in the sense that it can manage unlabeled data coming from more general distributions than the uniform one.

3.2.2 Comparison with Semi-Supervised Learning

The idea of exploiting unlabeled data in semi-supervised learning was originally proposed in [73]. Earlier studies do not thoroughly explain why unlabeled data can be beneficial. Authors of [74] are among the first to analyze this aspect from a generative perspective. In particular, they assume that data are distributed according to a mixture of Gaussians and show that the class posterior distribution can be decomposed in two terms, one depending on the class labels and the other on the mixture components. The two terms can be estimated using labeled and unlabeled data, respectively, thus improving the performance of the learnt classifier. [75] extends this analysis assuming that data can be correctly described by a more general class of parametric models. The authors show that if both the class posterior distribution and the data prior distribution are dependent on model parameters, unlabeled examples can be exploited to learn a better set of parameters. Thus, the key idea of semi-supervised learning is to exploit the distributional information contained in unlabeled samples.

Many approaches have been proposed. The work in [1] provides a taxonomy of semi-supervised learning algorithms. In particular, it is possible to distinguish five types of approaches: generative approaches (see, e.g., [76]) exploit unlabeled data to better estimate the class-conditional densities and infer the unknown labels based on the learnt model; low-density separation methods (see, e.g., [26]) look for decision boundaries that

[2]The conditions are based on class prior and size of positive (class of interest) and negative (the rest) data.

correctly classify labeled data and are placed in regions with few unlabeled samples (the so called low-density regions); graph-based methods (see, e.g., [3]) exploit unlabeled data to build a similarity graph and then propagate labels based on smoothness assumptions; dimensionality reduction methods (see, e.g., [13]) use unlabeled samples for representation learning and then perform classification on the learnt feature representation; and disagreement-based methods (discussed in [77]) exploit the disagreement among multiple base learners to learn more robust ensemble classifiers.

The scalability issue is largely studied in the context of semi-supervised learning. For example, the work in [78] proposes a framework to solve a mixed-integer programming problem, which runs multiple times the SVM algorithm. State-of-art implementations of SVMs (see, e.g., LIBSVM [79]) are based mainly on decomposition methods [80], like our proposed approach. Other semi-supervised approaches use approximations of the fitness function to simplify the optimization problem (see [81, 82]).

Both semi-supervised and PU learning exploit unlabeled data to learn better classifiers. Nevertheless, substantial differences hold that make semi-supervised learning not applicable to PU learning tasks. An important aspect is that most semi-supervised learning algorithms assume that unlabeled data originated from a set of known classes (closed set environment), thus not coping with the presence of unknown classes in training and test sets (open set environment). To the best of our knowledge, only few works (see [83]) propose semi-supervised methods able to handle such situations. Another even more relevant aspect is that **semi-supervised learning cannot learn a classifier when only one known class is present**, since it requires at least two known classes to calculate the decision boundary. On the contrary, recent works show that it is possible to apply PU learning to solve semi-supervised learning tasks, even in the case of open set environment [42, 71].

3.2.3 Theoretical Studies about PU learning

Inspired by the seminal work [84] and by the first studies on OCC [59, 58], the work in [85] and its successive extension [86] are the first to define and theoretically analyze the problem of PU learning. In particular, they propose a framework based on the statistical query model [84] to theoretically assess the classification performance and to derive algorithms based on decision trees. The authors study the problem of learning functions characterized by monotonic conjuctions, which are particularly useful in text classification, where documents can be represented as binary vectors that model the presence/absence of words from an available dictionary. Instead of considering binary features, [87] proposes a Naive-Bayes classifier for categorical features in noisy environments. The work assumes the attribute independence, which eases the estimate of class-conditional densities in high-dimensional spaces, but is limiting as compared to discriminative approaches, directly focusing on classification and not requiring density estimations [88].

As already mentioned at the beginning of this section, PU learning studies can be roughly classified in two-stage and single stage approaches. The former are based on heuristics to select a set of reliable negative samples from the unlabeled data and are not theoretically grounded, while the latter are subject to the problem of wrong label assignment. In order to understand how this issue is critical, let consider the theoretical result of consistency presented in [52]. [3] For any set of classifiers $\mathcal{F}$ of Vapnik-Chervonenkis (VC) dimension V and any $\delta > 0$, there exists a constant c such that the following bounds hold with probability $1 - \delta$:

$$\begin{aligned} FNR(f) - FNR(f^*) &\leq c\epsilon_n, \\ FPR(f) - FPR(f^*) &\leq \frac{c}{1-\pi}(\epsilon_n + \epsilon_p) \end{aligned}$$

where FPR, FNR are the false positive/negative rates, $f \in \mathcal{F}$ is the function obtained by the above-mentioned strategy, $f^* \in \mathcal{F}$ is the optimal function having access to the ground truth, π is the positive class prior, $\epsilon_{\cdot} = \sqrt{\frac{V\log(\cdot) - \log(\delta)}{\cdot}}$ and p and n are the number of positive and unlabeled samples, respectively. In particular, if one considers a simple scenario where the feature space is $\mathbb{R}^{100}$ and $V = 101$ (in the case of linear classifiers), it is possible to learn a classifier such that, with probability of 90%, the performance does not deviate from the optimal values for more than 10% (which is equivalent to setting $\delta, \epsilon_p, \epsilon_n = 0.1$). This is guaranteed when both positive and unlabeled sets consist of at least 10^5 training samples each. This is impractical in real world applications, since collecting and labelling so many data is very expensive. The effect of wrong label assignment is even more evident for larger values of positive class prior.

Recently, [37, 38] proposed two frameworks based on the statistical learning theory [39]. These works are free from heuristics, do not suffer the problem of wrong label assignment, and provide theoretical bounds on the generalization error. Another theoretical work is the one in [40], which specifically addresses the matrix completion problem, motivated by applications like recovering friendship relations in social networks based on few observed connections. This work however is unable to deal with the more general problem of binary PU learning, and formulates the optimization problem using the squared loss, which is known to be subobtimal according to the theoretical findings in [37, 38]. Overall, the analysis of the literature in the field makes evident the lack of **theoretically-grounded PU learning approaches with good scalability properties**.

[3]It is rewritten to be more consistent with the notation in this paper.

3.3 PU Learning Formulation

Assume that we are given a training dataset $D_b = \{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in X, y_i \in Y\}_{i=1}^m$, where $X \subseteq \mathbb{R}^d$, $Y = \{-1, 1\}$ and each pair of samples in D_b is drawn independently from the same unknown joint distribution $\mathcal{P}$ defined over X and Y. The goal is to learn a function f that maps the input space X into the class set Y, known as the binary classification problem. According to statistical learning theory [39], the function f can be learnt by minimizing the risk functional $\mathcal{R}$, namely

$$\begin{aligned}\mathcal{R}(f) =& \sum_{y\in Y}\int \ell(f(\mathbf{x}), y)\mathcal{P}(\mathbf{x}, y)d\mathbf{x} \\ =& \pi\int \ell(f(\mathbf{x}), 1)\mathcal{P}(\mathbf{x}|y = 1)d\mathbf{x} + (1-\pi)\int \ell(f(\mathbf{x}), -1)\mathcal{P}(\mathbf{x}|y = -1)d\mathbf{x}\end{aligned} \tag{3.1}$$

where π is the positive class prior and ℓ is a loss function measuring the disagreement between the prediction and the ground truth for sample x, viz. $f(\mathbf{x})$ and y, respectively.

In PU learning, the training set is split into two parts: a set of samples $D_p = \{\mathbf{x}_i \in X\}_{i=1}^p$ drawn from the positive class and a set of unlabeled samples $D_n = \{\mathbf{x}_i \in X\}_{i=1}^n$ drawn from both positive and negative classes. The goal is the same of the binary classification problem, but this time the supervised information is available only for one class. The learning problem can be still formulated as a risk minimization. In fact, since $\mathcal{P}(\mathbf{x}) = \pi\mathcal{P}(\mathbf{x}|y = 1) + (1-\pi)\mathcal{P}(\mathbf{x}|y = -1)$, (3.1) can be rewritten in the following way:

$$\begin{aligned}\mathcal{R}(f) =& \pi\int \ell(f(\mathbf{x}), 1)\mathcal{P}(\mathbf{x}|y = 1)d\mathbf{x} \\ &+ (1-\pi)\int \ell(f(\mathbf{x}), -1)\frac{\mathcal{P}(\mathbf{x}) - \pi\mathcal{P}(\mathbf{x}|y = 1)}{1-\pi}d\mathbf{x} \\ =& \pi\int \tilde{\ell}(f(\mathbf{x}), 1)\mathcal{P}(\mathbf{x}|y = 1)d\mathbf{x} + \int \ell(f(\mathbf{x}), -1)\mathcal{P}(\mathbf{x})d\mathbf{x}\end{aligned} \tag{3.2}$$

where $\tilde{\ell}(f(\mathbf{x}), 1) = \ell(f(\mathbf{x}), 1) - \ell(f(\mathbf{x}), -1)$ is called the **composite loss** [38].

The risk functional in (3.2) cannot be minimized since the distributions are unknown. In practice, one considers the empirical risk functional in place of (3.2), where expectation integrals are replaced with the empirical mean estimates computed over the available training data, namely

$$\mathcal{R}_{emp}(f) = \frac{\pi}{p}\sum_{\mathbf{x}_i\in D_p}\tilde{\ell}(f(\mathbf{x}_i), 1) + \frac{1}{n}\sum_{\mathbf{x}_i\in D_n}\ell(f(\mathbf{x}_i), -1) \tag{3.3}$$

The minimization of $\mathcal{R}_{emp}$ is in general an ill-posed problem. A regularization term is usually added to $\mathcal{R}_{emp}$ to restrict the solution space and to penalize complex solutions. The learning problem is then stated as an optimization task:

$$f^* = \arg\min_{f\in\mathcal{H}_k}\left\{\mathcal{R}_{emp}(f) + \lambda\|f\|^2_{\mathcal{H}_k}\right\} \tag{3.4}$$

where λ is a positive real parameter weighting the relative importance of the regularizer with respect to the empirical risk and $\|\cdot\|_{\mathcal{H}_k}$ is the norm associated with the function space $\mathcal{H}_k$. In this case, $\mathcal{H}_k$ refers to the Reproducing Kernel Hilbert Space (RKHS) associated with its Mercer kernel $k : X \times X \to \mathbb{R}$.[4] We can then enunciate the representer theorem for PU learning (proof in Supplementary Material):

Representer Theorem 1. *Given the training set $D = D_p \cup D_n$ and the Mercer kernel k associated with the RKHS $\mathcal{H}_k$, any minimizer $f^* \in \mathcal{H}_k$ of (3.4) admits the following representation*

$$f^*(\mathbf{x}) = \sum_{\mathbf{x}_i \in D} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$$

where $\alpha_i \in \mathbb{R}$ for all i.

This theorem shows that it is possible to learn functions defined on possibly-infinite dimensional spaces, i.e., the ones induced by the kernel k, but depending only on a finite number of parameters α_i. Thus, training focuses on learning such restricted set of parameters. Another important aspect is that the representer theorem does not say anything about the uniqueness of the solution, as it only says that every minimum solution has the same parametric form. In other words, different solutions have different values of parameters. The uniqueness of the solution is guaranteed only when the empirical risk functional in (3.4) is convex. A proper selection of the loss function is then necessary to fulfill this condition. Authors in [38] analysed the properties of loss functions for the PU learning problem, showing that a necessary condition for convexity is that the composite loss function in (3.3) is affine. This requirement is satisfied by some loss functions, such as the squared loss, the modified Huber loss, the logistic loss, the perceptron loss, and the double Hinge loss. The latter ensures the best generalization performance [38]. [5] Moreover, the comparison with non-convex loss functions [37, 38] suggests to use the double Hinge loss for the PU learning problem, with a twofold advantage: ensuring that the obtained solution is globally optimal, and allowing the use of convex optimization theory to perform a more efficient training.

These considerations, together with the result stated by the representer theorem, allow us rewriting problem (3.4) in an equivalent parametric form. In particular, defining $\boldsymbol{\alpha} \in \mathbb{R}^{(p+n)}$ as the vector of alpha values, $\boldsymbol{\xi} \in \mathbb{R}^n$ as the vector of slack variables, $\mathbf{K} \in \mathbb{R}^{(p+n)\times(p+n)}$ as the Gram matrix computed using the training set D, and considering, without loss of generality, target functions in the form $f(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x}, \mathbf{x}_i) + \beta$, where β is the bias of f, it is possible to derive the following optimization problem (derivation in

[4]For an overview of RKHS and their properties, see the work in [89]

[5]Double Hinge loss can be considered as the equivalent of Hinge loss function for the binary classification problem.

Supplementary Material):

$$\begin{aligned}
&\min_{\boldsymbol{\alpha},\boldsymbol{\xi},\beta}\Big\{-c_1\tilde{\mathbf{1}}^T\mathbf{K}\boldsymbol{\alpha} - c_1\tilde{\mathbf{1}}^T\mathbf{1}\beta + c_2\mathbf{1}_n^T\xi + \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}\Big\}\\
&\text{s.t. } \boldsymbol{\xi} \succeq \mathbf{0}_n,\\
&\quad\ \ \boldsymbol{\xi} \succeq \mathbf{U}\mathbf{K}\boldsymbol{\alpha} + \beta\mathbf{1}_n,\\
&\quad\ \ \boldsymbol{\xi} \succeq \frac{1}{2}\mathbf{1}_n + \frac{1}{2}\mathbf{U}\mathbf{K}\boldsymbol{\alpha} + \frac{\beta}{2}\mathbf{1}_n
\end{aligned} \tag{3.5}$$

where $\tilde{\mathbf{1}} = [1,\dots,1,0,\dots,0]^T$ is a vector of size $p+n$ with p non-zero entries, $\mathbf{1}$ and $\mathbf{1}_n$ are all-ones vectors of size $p+n$ and n, respectively, $\mathbf{U}$ is a $n\times(p+n)$ matrix obtained through the concatenation of a $n\times p$ null matrix and an identity matrix of size n, $\succeq$ is an element-wise operator, $c_1 = \frac{\pi}{2\lambda p}$ and $c_2 = \frac{1}{2\lambda n}$.

The equivalent dual problem of (3.5) is more compactly expressed as:

$$\begin{aligned}
&\min_{\boldsymbol{\sigma},\boldsymbol{\delta}}\left\{\frac{1}{2}\boldsymbol{\sigma}^T\mathbf{U}\mathbf{K}\mathbf{U}^T\boldsymbol{\sigma} - c_1\tilde{\mathbf{1}}^T\mathbf{K}\mathbf{U}^T\boldsymbol{\sigma} - \frac{1}{2}\mathbf{1}_n^T\boldsymbol{\delta}\right\}\\
&\text{s.t. } \mathbf{1}^T\left[c_1\tilde{\mathbf{1}} - \mathbf{U}^T\boldsymbol{\sigma}\right] = 0,\\
&\quad\ \ \boldsymbol{\sigma} + \frac{1}{2}\boldsymbol{\delta} \preceq c_2\mathbf{1}_n,\\
&\quad\ \ \boldsymbol{\sigma} - \frac{1}{2}\boldsymbol{\delta} \succeq \mathbf{0}_n,\\
&\quad\ \ \mathbf{0}_n \preceq \boldsymbol{\delta} \preceq c_2\mathbf{1}_n
\end{aligned} \tag{3.6}$$

where $\boldsymbol{\sigma},\boldsymbol{\delta} \in \mathbb{R}^n$ and are related to the Lagrange multipliers introduced during the derivation of the dual formulation (see Supplementary Material for details).

Due to linearity of constraints in (3.5), Slater's condition is trivially satisfied[6], thus strong duality holds. This means that (3.6) can be solved in place of (3.5) to get the primal solution. The optimal $\boldsymbol{\alpha}$ can be obtained from one of the stationarity conditions used during the Lagrangian formulation (details in Supplementary Material), namely using the following relation

$$\boldsymbol{\alpha} = c_1\tilde{\mathbf{1}} - \mathbf{U}^T\boldsymbol{\sigma} \tag{3.7}$$

Note that the bias β has to be considered separately, since problem (3.6) does not give any information on how to compute it (this will be discussed in the next section).

It is to be pointed out that (3.6) is a quadratic programming (QP) problem that can be solved by existing numerical QP optimization libraries. Nevertheless, it is memory inefficient, as it requires storing the Gram matrix, which scales quadratically with the number of training samples. Thus, modern computers cannot manage to solve (3.6) even for a

[6]See, e.g., [90]

few thousands samples. A question therefore arises: **is it possible to efficiently find an exact solution to problem (3.6) without storing the whole Gram matrix?**

3.4 Proposed Solution

In order to avoid the storage problem, we propose an iterative algorithm that converts problem (3.6) into a sequence of smaller QP subproblems associated to subsets of training samples (working sets), which require the computation and temporary storage of much smaller Gram matrices. Regarding the speed of convergence, we observe experimentally that the runtime scales sub-quadratically with the size of the training set (see the experimental section for a thorough discussion). As a consequence, our algorithm finds the exact optimal solution with a lower computational complexity. Each iteration of USMO is made of three steps: selection of the working set S, computation of the Gram matrix for samples associated to indices in S, solution of a QP subproblem, where only terms depending on S are considered. Details are provided in Algorithm 1. It is to be mentioned that in principle this strategy allows decreasing the storage requirement at the expense of a heavier computation. In fact, the same samples may be selected multiple times over iterations, thus requiring recomputing matrices $\boldsymbol{K}_{SS}, \boldsymbol{K}_{SP}$ and $\boldsymbol{K}_{S\bar{S}}$. We will see later how to deal with this inefficiency. Another important aspect is that at each iteration only few parameters are updated (those indexed by the working set S, namely $\boldsymbol{\sigma}_S^k$ and $\boldsymbol{\delta}_S^k$), while the others are kept fixed. Here, we consider a working set of size two, as this allows solving the QP subproblem (3.8) analytically, without the need for further optimization. This is discussed in the next subsection.

3.4.1 QP Subproblem

We start by considering the following Lemma (proof in Supplementary Material):

Lemma 1. *Given $S = \{i, j\}$, any optimal solution $\boldsymbol{\sigma}_S^* = [\sigma_i^* \, \sigma_j^*]^T$, $\boldsymbol{\delta}_S^* = [\delta_i^* \, \delta_j^*]^T$ of the QP subproblem (3.8) has to satisfy the following condition: $\forall u : \boldsymbol{x}_u \in S \wedge 0 \leq \delta_u^* \leq c_2$ either $\sigma_u^* = c_2 - \frac{\delta_u^*}{2}$ or $\sigma_u^* = \frac{\delta_u^*}{2}$.*

This tells us that the optimal solution $(\boldsymbol{\sigma}_S^*, \boldsymbol{\delta}_S^*)$ assumes a specific form and can be calculated by searching in a smaller space. In particular, four subspaces can be identified for search :

$$\text{Case 1: } \sigma_i^k = c_2 - \frac{\delta_i^k}{2} \;\wedge\; \sigma_j^k = c_2 - \frac{\delta_j^k}{2} \;\wedge\; 0 \leq \delta_i^k, \delta_j^k \leq c_2,$$

$$\text{Case 2: } \sigma_i^k = c_2 - \frac{\delta_i^k}{2} \;\wedge\; \sigma_j^k = \frac{\delta_j^k}{2} \;\wedge\; 0 \leq \delta_i^k, \delta_j^k \leq c_2,$$

Algorithm 1 General USMO algorithm

```
1: $k \leftarrow 1$.
2: Initialize $(\boldsymbol{\sigma}^1, \boldsymbol{\delta}^1)$.
3: while $(\boldsymbol{\sigma}^k, \boldsymbol{\delta}^k)$ is not a stationary point of (3.6) do
4:     Select the working set $S \subset U = \{u : \boldsymbol{x}_u \in D_n\}$ with $|S| = 2$.
5:     Compute $\boldsymbol{K}_{SS}$, $\boldsymbol{K}_{SP}$ and $\boldsymbol{K}_{S\bar{S}}$ where $P = \{u : \boldsymbol{x}_u \in D_p\}$ and $\bar{S} = U \backslash S$.
6:     Solve
```

$$\begin{aligned} \min_{\boldsymbol{\sigma}_S^k, \boldsymbol{\delta}_S^k} & \left\{ \frac{1}{2} (\boldsymbol{\sigma}_S^k)^T \boldsymbol{K}_{SS} \boldsymbol{\sigma}_S^k + \boldsymbol{e}^T \boldsymbol{\sigma}_S^k - \frac{1}{2} \mathbf{1}^T \boldsymbol{\delta}_S^k \right\} \\ \text{s.t. } & \mathbf{1}^T \boldsymbol{\sigma}_S^k = c_1 p - \mathbf{1}^T \boldsymbol{\sigma}_{\bar{S}}^k, \\ & \boldsymbol{\sigma}_S^k + \frac{1}{2} \boldsymbol{\delta}_S^k \preceq c_2 \mathbf{1}, \\ & \boldsymbol{\sigma}_S^k - \frac{1}{2} \boldsymbol{\delta}_S^k \succeq \mathbf{0}, \\ & \mathbf{0} \preceq \boldsymbol{\delta}_S^k \preceq c_2 \mathbf{1} \end{aligned} \tag{3.8}$$

where

$$\boldsymbol{e} = \boldsymbol{K}_{S\bar{S}} \boldsymbol{\sigma}_{\bar{S}}^k - c_1 \boldsymbol{K}_{SP} \mathbf{1}_p$$

and

$$\tilde{\boldsymbol{K}} = \begin{bmatrix} \boldsymbol{K}_{PP} & \boldsymbol{K}_{PS} & \boldsymbol{K}_{P\bar{S}} \\ \boldsymbol{K}_{SP} & \boldsymbol{K}_{SS} & \boldsymbol{K}_{S\bar{S}} \\ \boldsymbol{K}_{\bar{S}P} & \boldsymbol{K}_{\bar{S}S} & \boldsymbol{K}_{\bar{S}\bar{S}} \end{bmatrix}, \ \tilde{\boldsymbol{\sigma}}^k = \begin{bmatrix} \boldsymbol{\sigma}_S^k \\ \boldsymbol{\sigma}_{\bar{S}}^k \end{bmatrix}, \ \tilde{\boldsymbol{\delta}}^k = \begin{bmatrix} \boldsymbol{\delta}_S^k \\ \boldsymbol{\delta}_{\bar{S}}^k \end{bmatrix}$$

$\tilde{\boldsymbol{K}}, \tilde{\boldsymbol{\sigma}}^k$ and $\tilde{\boldsymbol{\delta}}^k$ are permutations of $\boldsymbol{K}, \boldsymbol{\sigma}^k$ and $\boldsymbol{\delta}^k$, respectively. In general, $\boldsymbol{K}_{VW}$ is used to denote a matrix containing rows of $\boldsymbol{K}$ indexed by elements in set V and columns of $\boldsymbol{K}$ indexed by elements in set W.

```
7:     $(\boldsymbol{\sigma}_S^{k+1}, \boldsymbol{\delta}_S^{k+1}) \leftarrow (\boldsymbol{\sigma}_S^k, \boldsymbol{\delta}_S^k)$.
8:     $k \leftarrow k + 1$.
9: end while
```

Table 3.1: Equations and Conditions Used to Solve the Four QP Subproblems.

Case	**Equations**
1	$\sigma_j^k = (a^k(k(\boldsymbol{x}_i, \boldsymbol{x}_i) - k(\boldsymbol{x}_i, \boldsymbol{x}_j)) + e_1 - e_2)\eta$
2	$\sigma_j^k = (a^k(k(\boldsymbol{x}_i, \boldsymbol{x}_i) - k(\boldsymbol{x}_i, \boldsymbol{x}_j)) + e_1 - e_2 + 2)/\eta$
3	$\sigma_j^k = (a^k(k(\boldsymbol{x}_i, \boldsymbol{x}_i) - k(\boldsymbol{x}_i, \boldsymbol{x}_j)) + e_1 - e_2 - 2)/\eta$
4	$\sigma_j^k = (a^k(k(\boldsymbol{x}_i, \boldsymbol{x}_i) - k(\boldsymbol{x}_i, \boldsymbol{x}_j)) + e_1 - e_2)/\eta$
Case	**Conditions**
1	$\max\{c_2/2, a^k - c_2\} \leq \sigma_j^k \leq \min\{c_2, a^k - c_2/2\}$
2	$\max\{0, a^k - c_2\} \leq \sigma_j^k \leq \min\{c_2/2, a^k - c_2/2\}$
3	$\max\{c_2/2, a^k - c_2/2\} \leq \sigma_j^k \leq \min\{c_2, a^k\}$
4	$\max\{0, a^k - c_2/2\} \leq \sigma_j^k \leq \min\{c_2/2, a^k\}$
Note:	$a^k = c_1 p - \mathbf{1}^T \boldsymbol{\sigma}_{\bar{S}}^k$, $\boldsymbol{e} = [e_1, e_2]^T$ and $\eta = k(\boldsymbol{x}_i, \boldsymbol{x}_i) + k(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

$$\text{Case 3: } \sigma_i^k = \frac{\delta_i^k}{2} \wedge \sigma_j^k = c_2 - \frac{\delta_j^k}{2} \wedge 0 \leq \delta_i^k, \delta_j^k \leq c_2,$$
$$\text{Case 4: } \sigma_i^k = \frac{\delta_i^k}{2} \wedge \sigma_j^k = \frac{\delta_j^k}{2} \wedge 0 \leq \delta_i^k, \delta_j^k \leq c_2 \tag{3.9}$$

Then, in order to solve the QP subproblem (3.8), one can solve four optimization subproblems, where the objective function is the same as (3.8), but the inequality constraints of (3.8) are simplified to (3.9). Each of these subproblems can be expressed as an optimization of **just one variable**, by exploiting the equality constraints of both (3.8) and (3.9). It can therefore be solved analytically, without the need for further optimization. Table 3.1 reports the equations used to solve the four subproblems (we omit the derivation, which is straightforward), where σ_j^k is computed for all four cases. All other variables, namely σ_i^k, δ_i^k and δ_j^k, can be obtained in a second phase by simply exploiting the equality constraints in (3.8) and (3.9).

These equations do not guarantee that the inequalities in (3.9) are satisfied. To verify this, one can rewrite these inequalities as equivalent conditions of only σ_j^k (by exploiting the equality constraints in (3.8) and (3.9)), and check σ_j^k against them, as soon as all σ_j^k are available. If these conditions are violated, a proper clipping is applied to σ_j^k to restore feasibility. Table 3.1 summarizes the checking conditions.

Finally, the minimizer of the QP subproblem (3.8) can be obtained by retaining only the solution with the lowest level of objective. At each iteration, the output of the algorithm is both optimal and feasible for the QP subproblem (3.8). The question now is: when is it also optimal for the problem (3.6)?

3.4.2 Optimality Conditions

A problem of any optimization algorithm is to determine the stop condition. In Algorithm 1, the search of the solution is stopped as soon as some stationarity conditions are met. These conditions, called Karush-Kuhn-Tucker (KKT) conditions, represent the certificates of optimality for the obtained solution. In case of (3.6) they are both necessary and sufficient conditions, since the objective is convex and the constraints are affine functions [91]. More in detail, an optimal solution has to satisfy the following relations:

$$\begin{aligned}
&\frac{\partial F(\boldsymbol{\sigma},\boldsymbol{\delta})}{\partial \sigma_u} - \beta = -\lambda_u + \mu_u, \\
&\frac{\partial F(\boldsymbol{\sigma},\boldsymbol{\delta})}{\partial \delta_u} = -\frac{\lambda_u}{2} - \frac{\mu_u}{2} - \xi_u + \eta_u, \\
&\lambda_u\Big(\sigma_u + \frac{\delta_u}{2} - c_2\Big) = 0, \\
&\mu_u\Big(\frac{\delta_u}{2} - \sigma_u\Big) = 0, \\
&\xi_u\Big(\delta_u - c_2\Big) = 0, \\
&\eta_u \delta_u = 0, \\
&\lambda_u, \mu_u, \xi_u, \eta_u \geq 0
\end{aligned} \tag{3.10}$$

and this is valid for any component of the optimal solution, namely $\forall u : \boldsymbol{x}_u \in D_n$. In (3.10) $F(\boldsymbol{\sigma},\boldsymbol{\delta})$ is used as an abbreviation of the objective function of (3.6), while $\beta, \lambda_u, \mu_u, \xi_u, \eta_u$ are the Lagrange multipliers introduced to deal with the constraints in (3.6). These conditions can be rewritten more compactly as:

$$\begin{aligned}
&0 \leq \delta_u {<} c_2 \;\wedge\; \sigma_u{=}\frac{\delta_u}{2} \;\Rightarrow\; \frac{\partial F(\boldsymbol{\sigma},\boldsymbol{\delta})}{\partial \sigma_u}{-}\beta \geq 1 \;\Rightarrow\; f(\boldsymbol{x}_u) \leq -1, \\
&0 \leq \delta_u {<} c_2 \;\wedge\; \sigma_u{=}c_2{-}\frac{\delta_u}{2} \;\Rightarrow\; \frac{\partial F(\boldsymbol{\sigma},\boldsymbol{\delta})}{\partial \sigma_u}{-}\beta \leq -1 \;\Rightarrow\; f(\boldsymbol{x}_u) \geq 1, \\
&\delta_u{=}c_2 \;\wedge\; \sigma_u{=}\frac{c_2}{2} \;\Rightarrow\; -1 \leq \frac{\partial F(\boldsymbol{\sigma},\boldsymbol{\delta})}{\partial \sigma_u}{-}\beta \leq 1 \;\Rightarrow\; -1 \leq f(\boldsymbol{x}_u) \leq 1
\end{aligned} \tag{3.11}$$

In order to derive both (3.10) and (3.11), one can follow a strategy similar to the proof of Lemma 1. It is easy to verify that $\frac{\partial F(\boldsymbol{\sigma},\boldsymbol{\delta})}{\partial \sigma_u} = -f(\boldsymbol{x}_u) + \beta$. Thus, (3.11) provides also conditions on the target function f. From now on, we will refer to (3.11) as the **optimality conditions**, to distinguish from approximate conditions used to deal with numerical approximations of calculators. To this aim, the **$\boldsymbol{\tau}-$optimality conditions** are introduced, namely:

$$0 \leq \delta_u {<} c_2 \;\wedge\; \sigma_u{=}\frac{\delta_u}{2} \;\Rightarrow\; \frac{\partial F(\boldsymbol{\sigma},\boldsymbol{\delta})}{\partial \sigma_u}{-}\beta \geq 1 - \frac{\tau}{2} \;\Rightarrow\; f(\boldsymbol{x}_u) \leq -1 + \frac{\tau}{2},$$

$$0 \leq \delta_u < c_2 \wedge \sigma_u = c_2 - \frac{\delta_u}{2} \Rightarrow \frac{\partial F(\boldsymbol{\sigma}, \boldsymbol{\delta})}{\partial \sigma_u} - \beta \leq -1 + \frac{\tau}{2} \Rightarrow f(\boldsymbol{x}_u) \geq 1 - \frac{\tau}{2},$$
$$\delta_u = c_2 \wedge \sigma_u = \frac{c_2}{2} \Rightarrow -1 - \frac{\tau}{2} \leq \frac{\partial F(\boldsymbol{\sigma}, \boldsymbol{\delta})}{\partial \sigma_u} - \beta \leq 1 + \frac{\tau}{2} \Rightarrow -1 - \frac{\tau}{2} \leq f(\boldsymbol{x}_u) \leq 1 + \frac{\tau}{2} \quad (3.12)$$

where τ is a real-positive scalar used to perturb the optimality conditions.

By introducing the sets $D_1(\boldsymbol{\sigma}, \boldsymbol{\delta}) = \left\{\boldsymbol{x}_u \in D_n : 0 \leq \delta_u < c_2 \wedge \sigma_u = \frac{\delta_u}{2}\right\}$, $D_2(\boldsymbol{\sigma}, \boldsymbol{\delta}) = \left\{\boldsymbol{x}_u \in D_n : 0 \leq \delta_u < c_2 \wedge \sigma_u = c_2 - \frac{\delta_u}{2}\right\}$ and $D_3(\boldsymbol{\sigma}, \boldsymbol{\delta}) = \left\{\boldsymbol{x}_u \in D_n : 0 < \delta_u \leq c_2 \wedge \left(\sigma_u = \frac{\delta_u}{2} \vee \sigma_u = c_2 - \frac{\delta_u}{2}\right)\right\}$ and the quantities $m_1^{max}(\boldsymbol{\sigma}, \boldsymbol{\delta}) = \max_{\boldsymbol{x}_u \in D_1} f(\boldsymbol{x}_u)$, $m_2^{min}(\boldsymbol{\sigma}, \boldsymbol{\delta}) = \min_{\boldsymbol{x}_u \in D_2} f(\boldsymbol{x}_u)$, $m_3^{min}(\boldsymbol{\sigma}, \boldsymbol{\delta}) = \min_{\boldsymbol{x}_u \in D_3} f(\boldsymbol{x}_U)$ and $m_3^{max}(\boldsymbol{\sigma}, \boldsymbol{\delta}) = \max_{\boldsymbol{x}_u \in D_3} f(\boldsymbol{x}_u)$, called the **most critical values**, it is possible to rewrite conditions (3.12) in the following equivalent way:

$$\begin{aligned}
&m_1^{max}(\boldsymbol{\sigma}, \boldsymbol{\delta}) - m_3^{min}(\boldsymbol{\sigma}, \boldsymbol{\delta}) \leq \tau, \\
&m_3^{max}(\boldsymbol{\sigma}, \boldsymbol{\delta}) - m_2^{min}(\boldsymbol{\sigma}, \boldsymbol{\delta}) \leq \tau, \\
&m_1^{max}(\boldsymbol{\sigma}, \boldsymbol{\delta}) - m_2^{min}(\boldsymbol{\sigma}, \boldsymbol{\delta}) + 2 \leq \tau
\end{aligned} \quad (3.13)$$

Apart from being written more compactly than (3.12), conditions (3.13) have the advantage that they can be computed without knowing the bias β. Due to the dependence on $\boldsymbol{\sigma}$ and $\boldsymbol{\delta}$, m_1^{max}, m_2^{min}, m_3^{min} m_3^{max} need to be tracked and computed at each iteration in order to check $\tau-$ optimality and to decide whether to stop the algorithm.

3.4.3 Working Set Selection

A natural choice for selecting the working set is to look for pairs violating the $\tau-$ optimality conditions. In particular,

Definition 1. *Any pair* $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ *from* D_n *is a* ***violating pair****, if and only if it satisfies the following relations:*

$$\begin{aligned}
&\boldsymbol{x}_i \in D_1, \boldsymbol{x}_j \in D_3 \Rightarrow f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j) > \tau, \\
&\boldsymbol{x}_i \in D_3, \boldsymbol{x}_j \in D_1 \Rightarrow f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j) < -\tau, \\
&\boldsymbol{x}_i \in D_2, \boldsymbol{x}_j \in D_3 \Rightarrow f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j) < -\tau, \\
&\boldsymbol{x}_i \in D_3, \boldsymbol{x}_j \in D_2 \Rightarrow f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j) > \tau, \\
&\boldsymbol{x}_i \in D_1, \boldsymbol{x}_j \in D_2 \Rightarrow f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j) + 2 > \tau, \\
&\boldsymbol{x}_i \in D_2, \boldsymbol{x}_j \in D_1 \Rightarrow f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j) - 2 < -\tau
\end{aligned} \quad (3.14)$$

Conditions (3.13) are not satisfied as long as violating pairs are found. Therefore, the algorithm keeps looking for violating pairs and use them to improve the objective function until $\tau-$optimality is reached.

The search of violating pairs as well as the computation of the most critical values go hand in hand in the optimization and follow two different approaches. The former consists of finding violating pairs and computing the most critical values based only on a subset of samples called the **non-bound** set, namely $D_n^- = (D_1 \cap D_3) \cup (D_2 \cap D_3)$, [7] while the latter consists of looking for violating pairs based on the whole set D_n by scanning all samples one by one. In this second approach, the most critical values are updated using the non-bound set together with the current examined sample. Only when all samples are examined, it is possible to check conditions (3.13), since the most critical values correspond to the original definition. The algorithm keeps using the first approach until $\tau-$optimality for the non-bound set is reached, after that the second approach is used. This process is repeated until the $\tau-$optimality for the whole set D_n is achieved. On the one hand, checking these conditions only on the non-bound set is very efficient but does not ensure the global τ optimality; on the other hand, the use of the whole unlabeled set is much more expensive, while ensuring the global τ optimality

The motivation of having two different approaches for selecting the violating pairs and computing the most critical values is to enhance efficiency in computation. This will be clarified in the next subsection.

3.4.4 Function Cache and Bias Update

Recall that each iteration of the USMO algorithm is composed by three main operations, namely: the working set selection, the resolution of the associated QP subproblem, and the update of the most critical values based on the obtained solution. It is interesting to note that all these operations require to compute the target function $f(\boldsymbol{x}_u)$ for all $\boldsymbol{x}_u$ belonging either to the non-bound set or to the whole set D_n, depending on the approach selected by that iteration. In fact, the stage of working set selection requires to evaluate conditions in (3.14) for pairs of samples depending on f; the equations used to solve the QP subproblem, shown in Table 3.1, depend on vector $\boldsymbol{e} = \boldsymbol{K}_{S\bar{S}}\boldsymbol{\sigma}_{\bar{S}}^{k} - c_1\boldsymbol{K}_{SP}\mathbf{1}_p + \boldsymbol{K}_{SS}\boldsymbol{\sigma}_{S}^{k-1} - \boldsymbol{K}_{SS}\boldsymbol{\sigma}_{S}^{k-1} = -[f(\boldsymbol{x}_i) - \beta, f(\boldsymbol{x}_j) - \beta]^T - \boldsymbol{K}_{SS}\boldsymbol{\sigma}_{S}^{k-1}$, which is also influenced by f; finally, the computation of the most critical values requires to evaluate the target function f. Therefore, it is necessary to define a strategy that limits the number of times the target function is evaluated at each iteration. This can be achieved by considering the fact that the algorithm performs most of the iterations on samples in the non-bound set, while the whole set is used mainly to check if $\tau-$optimality is reached, and then the values of the target function for those samples can be stored in a cache, called the **function cache**. Since usually $|D_n^-| \ll |D_n|$, storing $f(\boldsymbol{x}_u)$ for all $\boldsymbol{x}_u \in D_n^-$ is a cheap operation,

[7]The term *non-bound* comes from the fact that $0 < \delta_u < c_2$ for all $\boldsymbol{x}_u \in D_n$.

which allows to save a huge amount of computation, thus increasing the computational efficiency.

At each iteration the function cache has to be updated in order to take into account the changes occured at some of the entries of vectors $\boldsymbol{\sigma}$ and $\boldsymbol{\delta}$, or equivalently at some of the entries of $\boldsymbol{\alpha}$. By defining $\mathcal{F}^k(\boldsymbol{x}_u)$ as the function cache for sample $\boldsymbol{x}_u$ at iteration k, it is possible to perform the update operation using the following relation:

$$\mathcal{F}^k(\boldsymbol{x}_u) = \mathcal{F}^{k-1}(\boldsymbol{x}_u) + (\alpha_i^k - \alpha_u^{k-1})k(\boldsymbol{x}_i, \boldsymbol{x}_u) + (\alpha_j^k - \alpha_j^{k-1})k(\boldsymbol{x}_j, \boldsymbol{x}_u) \tag{3.15}$$

Since all operations at each iteration are invariant with respect to β (because they require to evaluate differences between target function values), β can be computed at the end of the algorithm, namely when $\tau-$optimality is reached. By exploiting the fact that the inequalities in (3.11) become simply equalities for samples in the non-bound set, meaning that the target function evaluated at those samples can assume only two values, 1 or -1,[8] it is possible to compute β for each of these samples in the following way:

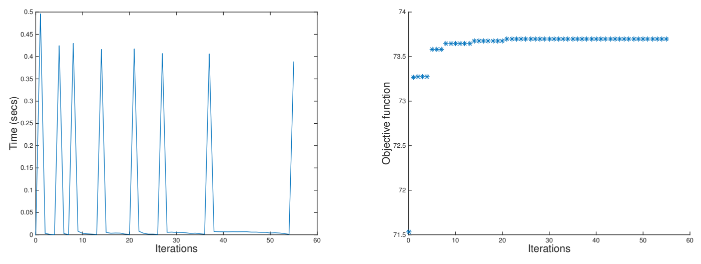
$$\beta_u = \begin{cases} -1 - \mathcal{F}(\boldsymbol{x}_u), & \forall \boldsymbol{x}_u \in D_1 \cap D_3 \\ 1 - \mathcal{F}(\boldsymbol{x}_u), & \forall \boldsymbol{x}_u \in D_2 \cap D_3 \end{cases} \tag{3.16}$$

The final β can be computed by averaging of (3.16) over all samples in the non-bound set, in order to reduce the effect of wrong label assignment.

3.4.5 Initialization

As previously mentioned, the proposed algorithm is characterized by iterations focusing either on the whole training data set or on a smaller set of non-bound samples. The formers are computationally more expensive, not only because a larger amount of samples is involved in the optimization, but also because the algorithm has to perform many evaluation operations, which can be skipped in the latter case, thanks to the exploitation of the function cache. An example of this is provided in Figure 3.1: the chart on the left plots the time required by the algorithm to complete the corresponding set of iterations. Peaks correspond to cases where the whole training data set is considered, while valleys represent the cases where iterations are performed over the non-bound samples. The chart on the right plots the overall objective score vs. the different sets of iterations. Jumps are associated with cases where all training samples are involved in the optimization. As soon as the algorithm approaches convergence, the contribution of the first kind of iterations becomes less and less relevant. Therefore, the selection of a good starting point is important to limit the number of iterations requested over the whole unlabeled

[8]The same principle holds for conditions (3.12), but in this case the inequalities are defined over arbitrary small intervals centered at 1 and -1 rather than being equalities.



(a) (b)

Figure 3.1: (a) plot of training time over iterations, (b) learning curve expressed in terms of objective function.

dataset. Given the convex nature of the problem, any suboptimal solution achievable with a low complexity can be used as a starting point. In our method, we propose the following heuristic procedure. Labeled samples are used to train a one-class SVM [92], that is in turn used to rank the unlabeled samples according to their value of estimated target function. From this ordered list it is possible to identify groups of samples that can be associated with the cases in (3.11). In particular, we identify five groups of samples corresponding to the following five cases:

$$\begin{aligned}
&\delta_u^{(1)} = 0 \,\wedge\, \sigma_u^{(1)} = 0,\\
&0 < \delta_u^{(2)} < c_2 \,\wedge\, \sigma_u = \frac{\delta_u^{(2)}}{2},\\
&\delta_u^{(3)} = c_2 \,\wedge\, \sigma_u^{(3)} = \frac{c_2}{2},\\
&0 < \delta_u^{(4)} < c_2 \,\wedge\, \sigma_u^{(4)} = c_2 - \frac{\delta_u}{2},\\
&\delta_u^{(5)} = 0 \,\wedge\, \sigma_u^{(5)} = c_2
\end{aligned} \tag{3.17}$$

The size of each group as well as the initial parameters for cases in (3.17) can be computed by solving an optimization problem, whose objective function is defined starting from the equality constraint in (3.6). In particular, by defining n_1, n_2, n_3, n_4, n_5 as the sizes of the groups for the different cases and by assuming that $n_1 = (1-\pi)an$, $n_2 = bn$, $n_3 = (1-a-b-c)n$, $n_4 = cn$, $n_5 = \pi an$, where $a, b, c \in \mathbb{R}^+$, and that the parameters for the second and the fourth cases in (3.17), namely $\sigma_u^{(2)}$ and $\sigma_u^{(4)}$, are the same, the optimization problem can be formulated in the following way:

$$\min_{\substack{\sigma_u^{(2)}, \sigma_u^{(4)},\\ a,b,c}} \left\{ c_1 p - bn\sigma_u^{(2)} - cn\sigma_u^{(4)} - c_2 n \left[\pi a + \frac{1-a-b-c}{2} \right] \right\}^2$$

$$\text{s.t. } 0 \le a + b + c \le 1 - \frac{1}{n},$$

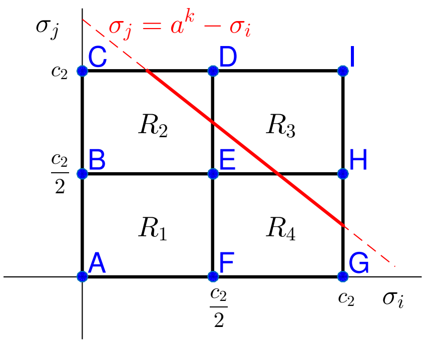


Figure 3.2: Subdivision of the feasible region in the plane defined by the variables σ_i and σ_j. The red solid line represents the feasible region including the equality constraint in (3.8).

$$\begin{aligned}
&\max\left\{\frac{1}{\pi n}, \frac{1}{(1-\pi)n}\right\} \leq a \leq \min\left\{\frac{1}{1-\pi}, \frac{1}{\pi}\right\}, \\
&\frac{1}{n} \leq b, c \leq \frac{\log(n)}{n}, \\
&0 < \sigma_u^{(2)} < \frac{c_2}{2} \wedge \frac{c_2}{2} < \sigma_u^{(4)} < c_2
\end{aligned} \tag{3.18}$$

where the constraints in (3.18) can be obtained by imposing $n_1 + n_2 + n_3 + n_4 + n_5 = n$ and $1 \leq n_2, n_3, n_4, n_5 \leq n$. Furthermore, we decide to have some upper bounds for b and c to limit the size of the initial non-bound set.

In practice, after ranking the unlabeled samples through the one-class SVM and solving the optimization problem in (3.18), the initial solution is obtained by assigning to each sample the value of parameters corresponding to the case that sample belongs to. For example, if the samples are ranked in ascending order, then the first n_1 samples in the list have $\sigma_u = 0$ and $\delta_u = 0$, the next n_2 samples have $\sigma_u = \sigma_u^{(2)}$ and $\delta_u = 2\sigma_u$ and the others follow the same strategy.

3.4.6 Theoretical Analysis

In this section, we present the main theoretical result, namely, we prove that Algorithm 1 converges to a $\tau-$optimal solution.

It is important to recall that each iteration of USMO requires to solve an optimization subproblem, that depends on a single variable. In particular, if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ correspond to the selected pair of points at one iteration, then the solution space corresponds to a line lying in the two-dimensional plane defined by the variables σ_i and σ_j. The feasible region in that plane can be subdivided into four parts, as defined according to Figure 3.2. These regions are considered closed sets, therefore including boundary points, like edges and corners. To consider only the interior of any set U, we use the notation int U. Based on these considerations, it is possible to prove the following lemma.

Lemma 2. *Let the vector $\boldsymbol{z}' = [\boldsymbol{\sigma}'; \boldsymbol{\delta}']$ be in the feasible set of (3.6) and $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ be a violating pair at point $\boldsymbol{z}'$. Let also $\boldsymbol{z}^* = [\boldsymbol{\sigma}^*; \boldsymbol{\delta}^*]$ be the solution obtained after applying one iteration of the Algorithm 1 using the working set $S = \{i, j\}$ and starting from $\boldsymbol{z}'$. Then, the following results hold:*

(a) $\boldsymbol{z}^* \neq \boldsymbol{z}'$,

(b) *After the minimization step, $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is no more a violating pair,*

(c) $(\sigma_i^*,\sigma_j^*) \in \mathit{int}\ R_1 \cup \mathit{int}\ R_3 \ \Rightarrow\ f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) = 0$,

(d) $(\sigma_i^*,\sigma_j^*) \in \mathit{int}\ R_2 \ \Rightarrow\ f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) = 2$,

(e) $(\sigma_i^*,\sigma_j^*) \in \mathit{int}\ R_4 \ \Rightarrow\ f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) = -2$,

(f) $(\sigma_i^*,\sigma_j^*) \in BE \Rightarrow 0 \leq f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) \leq 2$,

(g) $(\sigma_i^*,\sigma_j^*) \in DE \Rightarrow 0 \leq f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) \leq 2$,

(h) $(\sigma_i^*,\sigma_j^*) \in EF \Rightarrow -2 \leq f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) \leq 0$,

(i) $(\sigma_i^*,\sigma_j^*) \in EH \Rightarrow -2 \leq f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) \leq 0$,

(l) $(\sigma_i^*,\sigma_j^*) \in AB \cup DI \Rightarrow f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) \geq 0$,

(m) $(\sigma_i^*,\sigma_j^*) \in AF \cup HI \Rightarrow f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) \leq 0$,

(n) $(\sigma_i^*,\sigma_j^*) \in BC \cup CD \Rightarrow f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) \geq 2$,

(o) $(\sigma_i^*,\sigma_j^*) \in FG \cup GH \Rightarrow f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) \leq -2$,

(p) $F(\boldsymbol{\sigma}', \boldsymbol{\delta}') - F(\boldsymbol{\sigma}^*, \boldsymbol{\delta}^*) > \dfrac{\tau}{2\sqrt{2}} \|\boldsymbol{\sigma}' - \boldsymbol{\sigma}^*\|_2$

where $f_{\boldsymbol{z}^*}$ represents the target function with coefficients α_i computed according to (3.7) using $\boldsymbol{z}^*$ (see Figure 3.2).

Proof. Note that the feasible region for the QP subproblem (3.8) is a portion of line with negative slope lying on the plane defined by variables σ_i and σ_j (see Figure 3.2). Thus, any point (σ_i, σ_j) on this line can be expressed using the following relationship:

$$\begin{aligned} \sigma_i &= \sigma_i' + t, \\ \sigma_j &= \sigma_j' - t \end{aligned} \tag{3.19}$$

where $t \in \mathbb{R}$. In particular, if $t = 0$, then $(\sigma_i, \sigma_j) \equiv (\sigma_i', \sigma_j')$ and, if $t = t^*$, then $(\sigma_i, \sigma_j) \equiv (\sigma_i^*, \sigma_j^*)$.

Considering (3.19) and the fact that $\delta_{\boldsymbol{x_i}}=2\sigma_i \wedge \delta_j=2\sigma_j$ when $(\sigma_i, \sigma_j) \in R_1$, $\delta_{\boldsymbol{x_i}}=2\sigma_i \wedge \delta_j=2c_2 - 2\sigma_j$ when $(\sigma_i, \sigma_j) \in R_2$, $\delta_{\boldsymbol{x_i}}=c_2 - 2\sigma_i \wedge \delta_j=c_2 - 2\sigma_j$ when $(\sigma_i, \sigma_j) \in R_3$ and $\delta_{\boldsymbol{x_i}}=c_2 - 2\sigma_i \wedge \delta_j=2\sigma_j$ when $(\sigma_i, \sigma_j) \in R_4$, it is possible to rewrite the objective in (3.8) as a function of t, namely:

$$\begin{aligned}\phi(t) =& \frac{1}{2}(\sigma_i' + t)^2 k(\boldsymbol{x}_i, \boldsymbol{x}_i) + \frac{1}{2}(\sigma_j' - t)^2 k(\boldsymbol{x}_j, \boldsymbol{x}_j) \\ &+ (\sigma_i' + t)(\sigma_j' - t) k(\boldsymbol{x}_i, \boldsymbol{x}_j) + h_{\boldsymbol{z}}(t)\end{aligned} \tag{3.20}$$

where $h_{\boldsymbol{z}}$ is a function defined in the following way:

$$h_{\boldsymbol{z}}(t) = \begin{cases} (e_1-1)(\sigma_i'+t)+(e_2-1)(\sigma_j'-t), & (\sigma_i,\sigma_j)\in R_1, \\ (e_1-1)(\sigma_i'+t)+(e_2+1)(\sigma_j'-t)-c_2, & (\sigma_i,\sigma_j)\in R_2, \\ (e_1+1)(\sigma_i'+t)+(e_2+1)(\sigma_j'-t)-2c_2, & (\sigma_i,\sigma_j)\in R_3, \\ (e_1+1)(\sigma_i'+t)+(e_2-1)(\sigma_j'-t)-c_2, & (\sigma_i,\sigma_j)\in R_4 \end{cases}$$

Note that $\frac{d^2\phi(t)}{dt^2} = k(\boldsymbol{x}_i, \boldsymbol{x}_i) + k(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$ (k is a Mercer kernel), meaning that (3.20) is convex.

If $(\sigma_i^*, \sigma_j^*) \in \text{int } R_1$, then (σ_i^*, σ_j^*) is the minimum and $\frac{d\phi(t^*)}{dt} = 0$. Since $\frac{d\phi(t^*)}{dt} = f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) = 0$, the first and the second conditions in (3.14), which are the only possibilities to have a violating pair, are not satisfied. Therefore, $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is not violating at point $\boldsymbol{z}^*$, but it is violating at $\boldsymbol{z}'$, implying that $\boldsymbol{z}^* \neq \boldsymbol{z}'$. The same situation holds for $(\sigma_i^*, \sigma_j^*) \in \text{int } R_3$ and this proves statement (c). [9] Statements (d) and (e) can be proven in the same way, considering that the admissible conditions to have a violating pair are the first, the fourth and the fifth conditions for the former case and the second, the third and the sixth ones for the latter case.

If $(\sigma_i^*,\sigma_j^*) \in BE$, there are two possibilities to compute the derivative depending on the position of (σ_i',σ_j'), namely approaching (σ_i^*,σ_j^*) from the bottom or from the top of the constraint line. In the first case, the derivative is identified by $\frac{d\phi(t^*)}{dt^-}$, while in the second case by $\frac{d\phi(t^*)}{dt^+}$. Since (σ_i^*,σ_j^*) is the minimum and due to the convexity of function $\phi(t)$, $\frac{d\phi(t^*)}{dt^-} \geq 0$ and $\frac{d\phi(t^*)}{dt^+} \leq 0$. Furthermore, it is easy to verify that $\frac{d\phi(t^*)}{dt^-} = f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i)$ and $\frac{d\phi(t^*)}{dt^+} = f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) - 2$. By combining these results, we obtain that $0 \leq f_{\boldsymbol{z}^*}(\boldsymbol{x}_j) - f_{\boldsymbol{z}^*}(\boldsymbol{x}_i) \leq 2$. This, compared with the first condition in (3.14), guarantees that $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is not a violating pair at $\boldsymbol{z}^*$ and therefore that $\boldsymbol{z}^* \neq \boldsymbol{z}'$. The same strategy can be applied to derive statements (g)-(o).

For the sake of notation compactness, we use $\phi'(t)$ to identify both the classical and the directional derivatives of $\phi(t)$, viz. $\frac{d\phi(t)}{dt}$, $\frac{d\phi(t^*)}{dt^-}$ and $\frac{d\phi(t^*)}{dt^+}$, respectively. Therefore, it is possible to show that $\phi(t) = \phi(0) + \phi'(0)t + \frac{\phi''(0)}{2}t^2$. Furthermore, due to the convexity of $\phi(t)$, we have that

$$\begin{aligned}\phi'(0) < 0 \Rightarrow t_q \geq t^* > 0, \\ \phi'(0) > 0 \Rightarrow t_q \leq t^* < 0\end{aligned} \tag{3.21}$$

[9]In this case, the admissible conditions for violation are the second and the third conditions in (3.14).

where $t_q = -\frac{\phi'(0)}{\phi''(0)}$ is the unconstrained minimum of $\phi(t)$. From all these considerations, we can derive the following relation:

$$\phi(t^*) \leq \phi(0) + \frac{\phi'(0)}{2} t^* \tag{3.22}$$

In fact, if $\phi''(0) = 0$, then (3.22) trivially holds. If $\phi''(0) > 0$, then

$$\phi(t^*) - \phi(0) = \frac{\phi'(0)}{2} t^* \left(\frac{2t_q - t^*}{t_q} \right) \leq \frac{\phi'(0)}{2} t^* \tag{3.23}$$

where the last inequality of (3.23) is valid because $\left(\frac{2t_q - t^*}{t_q} \right) \geq 1$, by simply applying (3.21).

Note also that (3.19) can be used to derive the following result, namely:

$$\|\boldsymbol{\sigma}' - \boldsymbol{\sigma}^*\|_2 = |t^*|\sqrt{2} \tag{3.24}$$

By combining (3.23) and (3.24) and considering that conditions (3.14) can be compactly rewritten as $|\phi'(0)| > \tau$, we obtain that

$$\begin{aligned} \phi(0) - \phi(t^*) &\geq -\frac{\phi'(0)}{2} t^* = \frac{|\phi'(0)|}{2} |t^*| \\ &> \frac{\tau}{2} |t^*| = \frac{\tau}{2\sqrt{2}} \|\boldsymbol{\sigma}' - \boldsymbol{\sigma}^*\|_2 \end{aligned} \tag{3.25}$$

Finally, statement (p) is obtained from (3.25), by taking into account that $\phi(t^*) = F(\boldsymbol{\sigma}^*, \boldsymbol{\delta}^*)$ and $\phi(0) = F(\boldsymbol{\sigma}', \boldsymbol{\delta}')$. □

Lemma 2 states that each iteration of Algorithm 1 generates a solution that is $\tau-$optimal for the indices in the working set S.

The convergence of USMO to a $\tau-$optimal solution can be proven by contradiction by assuming that the algorithm proceeds indefinitely. This is equivalent to assume that $(\boldsymbol{x}_{i^k}, \boldsymbol{x}_{j^k})$ is violating $\forall k \geq 0$, where (i^k, j^k) represents the pair of indices selected at iteration k.

Since $\{F(\boldsymbol{\sigma}^k, \boldsymbol{\delta}^k)\}$ is a decreasing sequence (due to the fact that $\boldsymbol{z}^k \neq \boldsymbol{z}^{k+1}$ $\forall k \geq 0$[10] and that the algorithm minimizes the objective function at each iteration) and bounded below (due to the existence of an unknown global optimum), it is convergent. By exploiting this fact and by considering that $\frac{2\sqrt{2}}{\tau}[F(\boldsymbol{\sigma}^k, \boldsymbol{\delta}^k) - F(\boldsymbol{\sigma}^{k+l}, \boldsymbol{\delta}^{k+l})] > \|\boldsymbol{\sigma}^k - \boldsymbol{\sigma}^{k+l}\|_2$, $\forall k, l \geq 0$, which can be obtained from (p) of Lemma 2 by applying l times the triangle inequality, it is possible to conclude that $\{\boldsymbol{\sigma}^k\}$ is a Cauchy sequence. Therefore, since the sequence lies also in a closed feasible set, it is convergent. In other words, we have that $\boldsymbol{\sigma}^k \to \bar{\boldsymbol{\sigma}}$ for $k \to \infty$, meaning that Algorithm 1 produces **a convergent sequence of points**. Now, it is important to understand if this sequence converges to a $\tau-$optimal solution.

[10] Statement (a) of Lemma 2.

First of all, let us define the set of indices that are encountered/selected by the algorithm infinitely many times:

$$I_\infty = \{(\mu,\nu) : \exists\{k_t\} \subset \{k\}, (i^{k_t}, j^{k_t}) = (\mu,\nu), \forall t \in \mathbb{N}\} \tag{3.26}$$

$\{k_t\}$ is therefore a subsequence of $\{k\}$. It is also important to mention that since the number of iterations is infinite and the number of samples is finite, I_∞ cannot be an empty set. Based on this consideration, we define $\boldsymbol{v}_{\mu\nu}$ as the vector, whose elements are the entries at position μ and ν of a general vector $\boldsymbol{v}$, and provide the following lemma.

Lemma 3. *Assume $(\mu,\nu) \in I_\infty$ and let $\{k_t\}$ be the sequence of indices for which $(i^{k_t}, j^{k_t}) = (\mu,\nu)$. Then,*

(a) $\forall\epsilon > 0, \exists\hat{t} > 0 : \forall t \geq \hat{t},\ \|\boldsymbol{\sigma}^{k_t}_{\mu\nu} - \bar{\boldsymbol{\sigma}}_{\mu\nu}\| < \epsilon$ *and* $\|\boldsymbol{\sigma}^{k_t+1}_{\mu\nu} - \bar{\boldsymbol{\sigma}}_{\mu\nu}\| < \epsilon$,

(b) $f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\mu) - f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\nu) > \tau \Rightarrow f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\mu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\nu) \geq \tau$,

(c) $f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\mu) - f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\nu) < -\tau \Rightarrow f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\mu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\nu) \leq -\tau$,

(d) $f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\mu) - f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\nu) > \tau - 2 \Rightarrow f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\mu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\nu) \geq \tau - 2$,

(e) $f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\mu) - f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\nu) < -\tau + 2 \Rightarrow f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\mu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\nu) \leq -\tau + 2$

where $f_{\boldsymbol{\sigma}^{k_t}}$, $f_{\bar{\boldsymbol{\sigma}}}$ represent the target function with coefficients α_i computed according to (3.7) using $\boldsymbol{\sigma}^{k_t}$ and $\bar{\boldsymbol{\sigma}}$, respectively.

Proof. Since $\{\boldsymbol{\sigma}^k\}$ is convergent and $\{k_t\}$, $\{k_t + 1\}$ are subsequences of $\{k\}$, $\{\boldsymbol{\sigma}^{k_t}\}$ and $\{\boldsymbol{\sigma}^{k_t+1}\}$ are also convergent sequences. In other words, $\exists\hat{t} > 0$ such that $\|\boldsymbol{\sigma}^{k_t} - \bar{\boldsymbol{\sigma}}\| < \epsilon$ and $\|\boldsymbol{\sigma}^{k_t+1} - \bar{\boldsymbol{\sigma}}\| < \epsilon$. Furthemore, $\|\boldsymbol{\sigma}^{k_t} - \bar{\boldsymbol{\sigma}}\| \geq \|\boldsymbol{\sigma}^{k_t}_{\mu\nu} - \bar{\boldsymbol{\sigma}}_{\mu\nu}\|$ and $\|\boldsymbol{\sigma}^{k_t+1} - \bar{\boldsymbol{\sigma}}\| \geq \|\boldsymbol{\sigma}^{k_t+1}_{\mu\nu} - \bar{\boldsymbol{\sigma}}_{\mu\nu}\|$. By combining these two results, we obtain statement (a).

Concerning statement (b), we have that $f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\mu) - f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\nu) > \tau$. Furthermore, from convergence of $\{\boldsymbol{\sigma}^{k_t}\}$ and continuity of f, we obtain that $\forall\epsilon > 0$, $\exists\tilde{t} \geq \hat{t}$: $\forall t \geq \tilde{t}$, $-\epsilon \leq f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\mu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\mu) \leq \epsilon$ and $-\epsilon \leq f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\nu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\nu) \leq \epsilon$, meaning that both $\{f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\mu)\}$ and $\{f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\nu)\}$ are convergent. Therefore, $f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\mu) - f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\nu) > \tau$ can be rewritten as

$$f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\mu) - f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\nu) + f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\mu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\mu) + f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\nu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\nu) > \tau$$

and by applying the information about the convergence of both $\{f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\mu)\}$ and $\{f_{\boldsymbol{\sigma}^{k_t}}(\boldsymbol{x}_\nu)\}$, we get that

$$f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\mu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\nu) > \tau - 2\epsilon$$

which is valid $\forall\epsilon > 0$ and therefore proves statement (b). All other statements, namely (c)-(e), can be proven using the same approach. $\square$

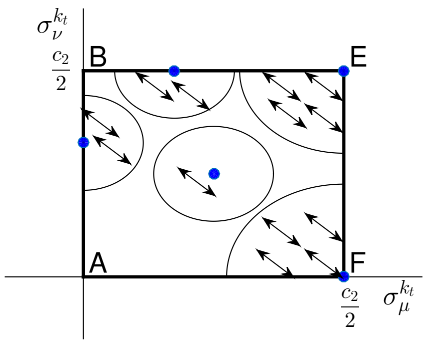


Figure 3.3: Example of transitions performed by a minimization step of Algorithm 1 for different locations of $\bar{\boldsymbol{\sigma}}_{\mu,\nu}$ (highlighted by blue points) and for sufficiently large number of iterations.

Lemma 3 states some conditions about the final target function and also states that the sequence output by Algorithm 1, after a sufficiently large number of iterations, is enclosed in a ball centered at $\bar{\boldsymbol{\sigma}}$. This aspect is shown in Figure 3.3 for R_1 and for different possible locations of $\bar{\boldsymbol{\sigma}}_{\mu,\nu}$. The same picture shows also the possible transitions that may happen at each iteration. In particular, we see that for $\bar{\boldsymbol{\sigma}}_{\mu,\nu}$ lying on corners and edges, different kinds of transitions exist. In fact, we find transitions from border to border, transitions from border to inner points and viceversa, and transitions from inner points to inner points. These are indetified as *bd* $\rightarrow$ *bd*, *bd* $\rightarrow$ *int*, *int* $\rightarrow$ *bd* and *int* $\rightarrow$ *int*, respectively. Note that for $\bar{\boldsymbol{\sigma}}_{\mu,\nu}$ not lying on borders, *int* $\rightarrow$ *int* is the only available kind of transition. Based on these considerations, it is possible to prove the following lemma.

Lemma 4. *Let (μ,ν), $\{k_t\}$, $\hat{t}$ and ϵ be defined according to Lemma 3. Then, $\exists \bar{t} \geq \hat{t}$ such that $\forall t \geq \bar{t}$ and for sequence $\{k_t\}$ the only allowed transitions are int $\rightarrow$ bd and bd $\rightarrow$ bd.*

Proof. Consider region R_1 and $(\bar{\sigma}_\mu, \bar{\sigma}_\nu) \in \text{int } R_1$. Then, the only admissible type of transitions for this case is *int* $\rightarrow$ *int*. Therefore, based on statement (c) of Lemma 2 (and thanks also to statement (a) of Lemma 3), we obtain that $\forall t \geq \bar{t}$, $f_{\boldsymbol{\sigma}^{k_t+1}}(\boldsymbol{x}_\mu) - f_{\boldsymbol{\sigma}^{k_t+1}}(\boldsymbol{x}_\nu) = 0$. By exploiting this fact, the continuity of f and the convergence of $\{\boldsymbol{\sigma}^{k_t+1}\}$, it is possible to show that

$$f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\mu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\nu) = 0 \tag{3.27}$$

Furthermore, since $(\boldsymbol{x}_\mu, \boldsymbol{x}_\nu)$ is a violating pair at all iterations and $\forall t \geq \bar{t}$, $\boldsymbol{\sigma}_{\mu\nu}^{k_t} \in \text{int } R_1$ (due to statement (a) of Lemma 3), $(\boldsymbol{x}_\mu, \boldsymbol{x}_\nu)$ has to satisfy conditions (b) or (c) of Lemma 3. These conditions are in contradiction with (3.27), meaning that the *int* $\rightarrow$ *int* transition is not allowed in this case.

Consider now region R_1 and $(\bar{\sigma}_\mu, \bar{\sigma}_\nu) \in E$, or equivalently $(\bar{\sigma}_\mu, \bar{\sigma}_\nu) \in A$. This time, the potential transitions are *bd* $\rightarrow$ *bd*, *bd* $\rightarrow$ *int*, *int* $\rightarrow$ *bd* and *int* $\rightarrow$ *int*. Nevertheless, it is always possible to define a subsequence containing only either *int* $\rightarrow$ *int* or *bd* $\rightarrow$ *int* and obtain therefore conclusions similar to the previous case. In fact, both *int* $\to$ *int* and *bd* $\to$ *int* are not allowed transitions.

The same results can be obtained in a similar way for other edges, corners of R_1 as well as for points in R_3, upon selection of the proper conditions in Lemma 2.

Consider now region R_2 and $(\bar{\sigma}_\mu, \bar{\sigma}_\nu) \in \text{int } R_2$. The only admissible transition in this case is *int* $\to$ *int*. From statement (d) of Lemma 2, we have that $\forall t \geq \bar{t}$, $f_{\boldsymbol{\sigma}^{k_t+1}}(\boldsymbol{x}_\mu) - f_{\boldsymbol{\sigma}^{k_t+1}}(\boldsymbol{x}_\nu) = -2$ and, from the continuity of f and the convergence of $\{\boldsymbol{\sigma}^{k_t+1}\}$, it is possible to show that

$$f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\mu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\nu) = -2 \tag{3.28}$$

Furthermore, since $(\boldsymbol{x}_\mu, \boldsymbol{x}_\nu)$ is a violating pair at all iterations and $\forall t \geq \bar{t}$, $\boldsymbol{\sigma}^{k_t}_{\mu\nu} \in \text{int } R_2$, $(\boldsymbol{x}_\mu, \boldsymbol{x}_\nu)$ has to satisfy conditions (b) or (d) of Lemma 3. These conditions are in contradiction with (3.28), meaning that the *int* $\to$ *int* transition is not valid.

For all corners and edges of R_2, as well as for all points in R_4, it is possible to show that *int* $\to$ *int* and *bd* $\to$ *int* are not valid transitions. The proof is similar to the previous cases. Therefore, the only admissible transitions after a sufficiently large number of iterations are *int* $\to$ *bd* and *bd* $\to$ *bd*. □

It is interesting to note that each transition *int* $\to$ *bd* increases the number of components of $\boldsymbol{\sigma}$ belonging to borders of the four regions, by one or two, while each transition *bd* $\to$ *bd* leaves it unchanged. Since this number is bounded by n, transition *int* $\to$ *bd* cannot appear infinitely many times. Therefore, $\exists t^* \geq \bar{t}$, $\forall t \geq t^*$, *bd* $\to$ *bd* is the only valid transition.

Note that *bd* $\to$ *bd* may happen only when $(\bar{\sigma}_\mu, \bar{\sigma}_\nu)$ is located at some specific corners of the feasible region, namely corners A or E for region R_1, corners B or C for region R_2, corners E or I for region R_3 and corners F or H for region R_4. For all cases, it is possible to define a subsequence that goes only from a vertical to a horizontal border and a subsequence that goes only from a horizontal to a vertical border. Without loss of generality, we can consider a specific case, namely $(\bar{\sigma}_\mu, \bar{\sigma}_\nu) \in A$. Note that for the first subsequence, $f_{\boldsymbol{\sigma}^{k_t+1}}(\boldsymbol{x}_\mu) - f_{\boldsymbol{\sigma}^{k_t+1}}(\boldsymbol{x}_\nu) < -\tau$, since $(\boldsymbol{x}_\mu, \boldsymbol{x}_\nu)$ has to be a violating pair in order not to stop the iterations and therefore, from statement (c) of Lemma 3, $f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\mu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\nu) < -\tau$. For the second subsequence, $f_{\boldsymbol{\sigma}^{k_t+1}}(\boldsymbol{x}_\mu) - f_{\boldsymbol{\sigma}^{k_t+1}}(\boldsymbol{x}_\nu) > \tau$ and consequently $f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\mu) - f_{\bar{\boldsymbol{\sigma}}}(\boldsymbol{x}_\nu) > \tau$. This leads to a contradiction which holds $\forall(\mu, \nu) \in I_\infty$. Therefore, the assumption that Algorithm 1 proceeds indefinitely is not verified. In other words, there exists an iteration at which the algorithm stops because a $\tau-$optimal solution is obtained.

Table 3.2: Characteristics of datasets (small scale datasets on the left and large scale datasets on the right).

Datasets	**# Instances**	**# Features**	**Datasets**	**# Instances**	**# Features**
Australian	690	42	*Bank-marketing*	28000	20
Clean1	476	166	*Adult*	32562	123
Diabetes	768	8	*Statlog (shuttle)*	43500	9
Heart	270	9	*Mnist*	60000	784
Heart-statlog	270	13	*Poker-hand*	1000000	10
House	232	16			
House-votes	435	16			
Ionosphere	351	33			
Isolet	600	51			
Krvskp	3196	36			
Liverdisorders	345	6			
Spectf	349	44			

3.5 Experiments

In this section, comprehensive evaluations are presented to demonstrate the effectiveness of the proposed approach. USMO is compared with [38] and [37]. The three methods have been implemented in MATLAB, to ensure fair comparison. [11] The method in [38] solves problem (3.6) using the MATLAB built-in function *quadprog*, combined with the second-order primal-dual interior point algorithm [93], while the method in [37] solves problem (3.4) with the ramp loss function using the *quadprog* function combined with the concave-convex procedure [94]. Experiments were run on a PC with 16 2.40 GHz cores and 64GB RAM. A collection of 17 real-world datasets from the UCI repository was used, 12 of which contain hundreds or thousands of samples, while the remaining 5 are significantly bigger. Table 3.2 shows some of their statistics.

Since USMO and [38] solve the same optimization problem, we first verify that **both achieve the same solution**. We consider the F-measure in a transductive setting, to assess the generalization performance on all small-scale datasets and under different configurations of hyper-parameters and kernel functions. In particular, we consider different values of λ, viz. 0.0001, 0.001, 0.01, 0.1, using linear and Gaussian kernels. [12] In these experiments, only 20% of positive samples are labeled. Tables 3.3-3.4 show the results

[11]USMO source code is available at `https://github.com/emsansone/USMO`.

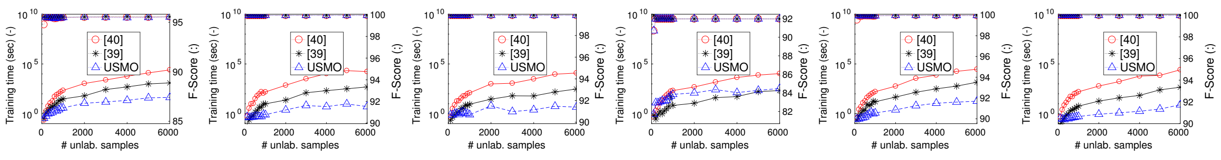
[12]The positive class prior π is set to the class proportion in the training datasets. Methods like [95, 52, 96] can be used to estimate it.

Table 3.3: Comparative results (F-measure) on different small-scale datasets and on different values of hyperparameters using the linear kernel. 20% of positive examples are labeled, while the remaining are unlabeled.

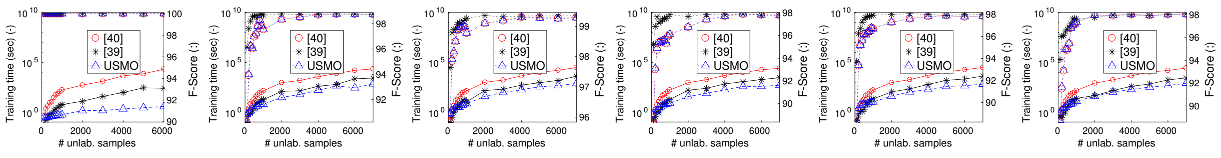
	$\lambda = 0.0001$				$\lambda = 0.001$				$\lambda = 0.01$				$\lambda = 0.1$			
Datasets	Init	[37]	[38]	USMO	Init	[37]	[38]	USMO	Init	[37]	[38]	USMO	Init	[37]	[38]	USMO
Australian	61.8	61.5	67.9	68.3	61.8	68.8	67.7	67.6	61.8	63.4	69.0	69.3	61.8	58.5	70.0	70.2
Clean1	60.6	81.6	70.5	70.2	63.8	76.2	65.5	65.6	64.6	81.9	73.3	73.0	66.2	80.4	77.9	75.6
Diabetes	41.8	73.3	70.0	70.1	41.4	71.5	71.2	71.1	37.6	77.9	78.0	79.3	6.5	82.3	82.3	82.3
Heart	46.8	60.8	60.1	59.4	49.0	63.4	60.7	60.0	57.2	64.9	66.0	66.0	75.6	75.6	75.6	75.6
Heart-statlog	63.3	65.1	54.5	54.5	63.8	58.1	55.4	55.2	66.9	61.2	53.8	53.8	75.6	67.5	61.7	60.2
House	49.0	57.9	59.9	59.9	49.8	66.7	59.0	59.0	49.8	57.9	57.4	56.8	58.4	51.8	64.6	64.0
House-votes	59.3	51.6	59.6	59.6	60.0	56.6	60.0	59.9	66.9	59.3	57.1	57.2	71.8	51.3	61.7	62.0
Ionosphere	19.3	59.9	65.1	65.1	19.3	74.9	71.5	72.0	19.3	71.9	72.6	73.7	22.3	75.5	75.2	75.2
Isolet	98.3	77.1	91.8	93.7	98.1	78.0	93.3	93.3	98.1	81.1	94.7	94.7	98.0	86.8	95.5	95.5
Krvskp	57.0	78.4	81.1	81.1	57.3	78.7	79.6	79.9	61.1	75.8	82.9	82.9	68.8	75.0	80.6	80.6
Liverdisorders	54.5	56.0	55.7	56.0	60.0	58.1	63.9	63.4	69.0	68.8	68.8	68.8	68.8	68.8	68.8	68.8
Spectf	56.4	58.0	73.5	73.5	60.4	66.3	72.9	72.3	58.1	79.9	80.6	80.6	79.2	81.1	81.1	81.1
Avg.	55.7±18.1	65.1±10.0	**67.5±10.9**	**67.6±11.3**	57.0±18.1	**68.1±7.9**	**68.4±10.4**	**68.3±10.6**	59.2±18.9	70.3±8.9	**71.2±11.9**	**71.3±12.1**	62.7±24.9	71.2±11.9	**74.6±9.9**	**74.3±10.0**

Table 3.4: Comparative results (F-measure) on different small-scale datasets and on different values of hyperparameters using the Gaussian kernel (scale parameter equal to 1). 20% of positive examples are labeled, while the remaining are unlabeled.

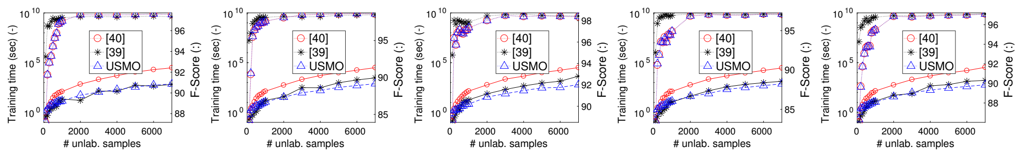
	$\lambda = 0.0001$				$\lambda = 0.001$				$\lambda = 0.01$				$\lambda = 0.1$			
Datasets	Init	[37]	[38]	USMO	Init	[37]	[38]	USMO	Init	[37]	[38]	USMO	Init	[37]	[38]	USMO
Australian	67.0	64.2	64.2	64.7	67.0	64.2	70.6	70.6	67.0	57.2	59.4	59.4	67.0	0.0	0.0	0.0
Clean1	28.7	80.1	77.6	77.6	44.0	80.6	81.7	81.7	78.3	78.0	76.6	76.6	76.4	76.4	76.4	76.4
Diabetes	58.8	74.1	70.0	70.0	58.4	71.3	71.2	70.6	52.6	80.7	80.1	80.1	0.0	82.3	82.3	82.3
Heart	46.5	66.7	58.9	58.9	47.7	64.1	59.8	59.8	63.8	70.3	70.9	70.9	75.6	75.6	75.6	75.6
Heart-statlog	30.7	70.3	53.5	53.5	31.5	65.1	56.7	56.7	49.6	60.6	56.4	56.4	75.6	75.6	75.6	75.6
House	38.5	63.9	56.5	56.5	37.1	68.8	61.1	60.1	28.2	59.1	56.8	54.2	74.0	74.0	74.0	74.0
House-votes	64.2	29.7	61.2	61.2	64.1	54.3	59.2	59.4	67.4	61.1	57.5	57.5	71.8	71.8	71.8	71.8
Ionosphere	55.6	52.6	65.3	65.3	56.3	68.2	74.3	74.1	58.4	58.7	65.7	65.7	72.7	74.2	74.2	74.2
Isolet	0.0	73.3	75.3	75.3	0.0	73.8	75.7	75.7	0.0	74.1	76.4	76.4	0.0	73.3	75.9	75.9
Krvskp	52.0	73.9	82.4	82.5	59.4	77.2	84.3	84.1	44.2	73.2	73.2	73.2	73.2	73.2	73.2	73.2
Liverdisorders	44.3	60.6	57.0	56.2	49.5	63.7	62.2	62.5	68.8	68.8	68.8	68.8	68.8	68.8	68.8	68.8
Spectf	61.5	38.0	74.0	74.0	80.0	62.4	72.9	72.1	69.0	81.1	81.1	81.1	81.1	81.1	81.1	81.1
Avg.	45.7±19.1	62.3±15.2	**66.3±9.4**	**66.3±9.5**	49.6±20.5	67.8±7.2	**69.2±9.2**	**68.9±9.2**	53.9±21.7	**68.6±9.0**	**68.6±9.3**	**68.4±9.6**	61.4±28.9	**68.9±22.0**	**69.1±22.1**	**69.1±22.1**



(a) Statlog 1/all (b) Statlog 2/all (c) Statlog 3/all (d) Statlog 4/all (e) Statlog 5/all (f) Statlog 6/all

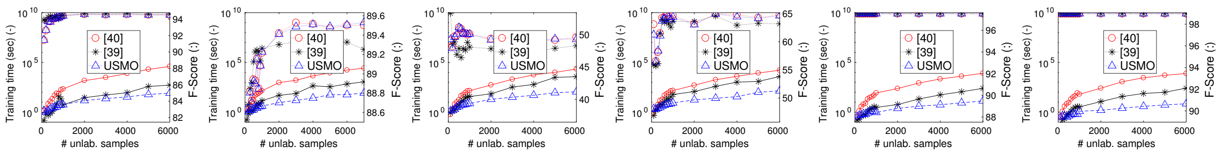


(g) Statlog 7/all (h) MNIST 0/all (i) MNIST 1/all (j) MNIST 2/all (k) MNIST 3/all (l) MNIST 4/all

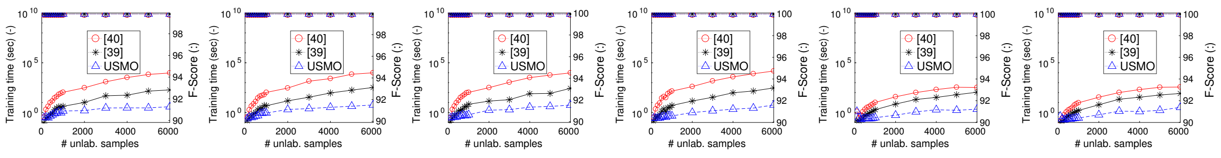


(m) MNIST 5/all (n) MNIST 6/all (o) MNIST 7/all (p) MNIST 8/all (q) MNIST 9/all

Figure 3.4: Comparative results on (a)-(g) Statlog (shuttle) and (h)-(q) MNIST datasets using the linear kernel ($\lambda = 0.01$). Each plot shows the training time against different number of unlabeled samples (100 positive samples) as well as the generalization performance on the test set.



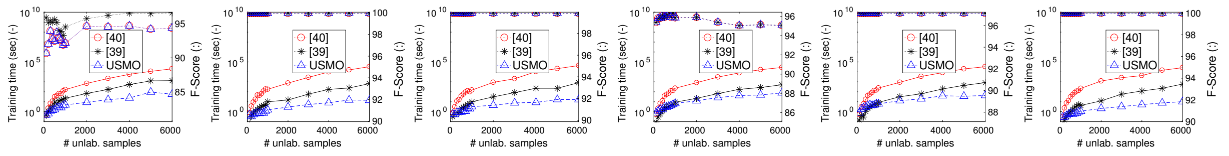
(a) Bank 1/all (b) Adult 1/all (c) POKER 0/all (d) POKER 1/all (e) POKER 2/all (f) POKER 3/all



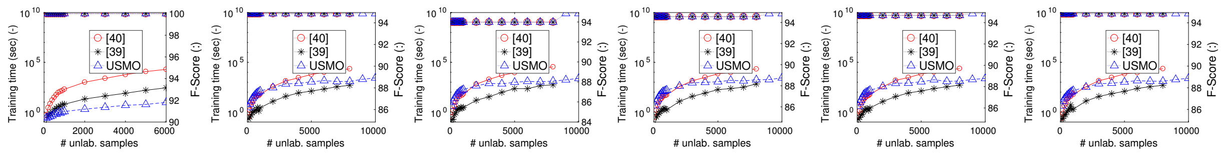
(g) POKER 4/all (h) POKER 5/all (i) POKER 6/all (j) POKER 7/all (k) POKER 8/all (l) POKER 9/all

Figure 3.5: Comparative results on (a) Bank-marketing, (b) Adult and (c)-(l) Poker-hand datasets using the linear kernel ($\lambda = 0.01$). Each plot shows the training time against different number of unlabeled samples (100 positive samples) as well as the generalization performance on the test set.

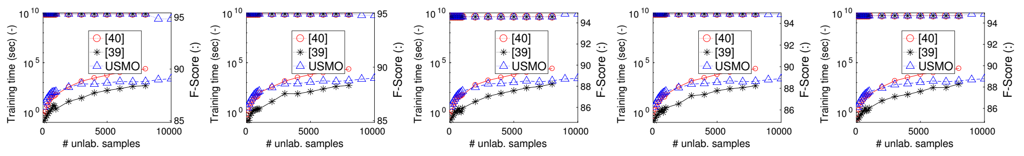
for linear and Gaussian kernels, respectively. Both algorithms achieve almost identical performance, with small differences due to numerical approximations. This fact confirms the results of the theoretical analysis of previous subsection, according to which USMO



(a) Statlog 1/all (b) Statlog 2/all (c) Statlog 3/all (d) Statlog 4/all (e) Statlog 5/all (f) Statlog 6/all

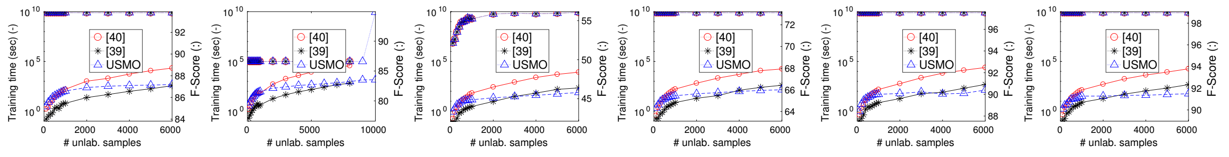


(g) Statlog 7/all (h) MNIST 0/all (i) MNIST 1/all (j) MNIST 2/all (k) MNIST 3/all (l) MNIST 4/all

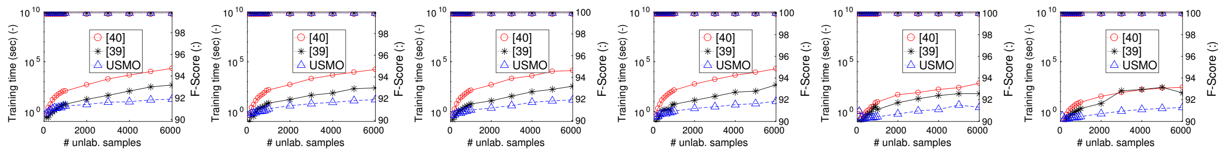


(m) MNIST 5/all (n) MNIST 6/all (o) MNIST 7/all (p) MNIST 8/all (q) MNIST 9/all

Figure 3.6: Comparative results on (a)-(g) Statlog (shuttle) and (h)-(q) MNIST datasets using the Gaussian kernel ($\lambda = 0.01$ and scale parameter equal to 1). Each plot shows the training time against different number of unlabeled samples (100 positive samples) as well as the generalization performance on the test set.



(a) Bank 1/all (b) Adult 1/all (c) POKER 0/all (d) POKER 1/all (e) POKER 2/all (f) POKER 3/all



(g) POKER 4/all (h) POKER 5/all (i) POKER 6/all (j) POKER 7/all (k) POKER 8/all (l) POKER 9/all

Figure 3.7: Comparative results on (a) Bank-marketing, (b) Adult and (c)-(l) Poker-hand datasets using the Gaussian kernel ($\lambda = 0.01$ and scale parameter equal to 1). Each plot shows the training time against different number of unlabeled samples (100 positive samples) as well as the generalization performance on the test set.

is guaranteed to converge to the same value of objective function obtained by [38]. In Tables 3.3 -3.4, we report the baseline performance obtained by applying only our proposed initialization (called Init).

Note that ramp loss [37] achieves on average inferior performance compared to USMO and [38]. Furthermore, it is influenced by the starting point due to a non-convex objective function, thus making double Hinge loss preferable in practical applications.

Secondly, we investigate **the complexity of USMO with respect to [37, 38]**. As to the storage requirements, USMO behaves linearly instead of quadratically as [37, 38]. Concerning the computational complexity, it can be easily found that each iteration has, in the worst case (i.e., an iteration over the whole unlabeled dataset), a complexity $O(|D_n|)$. As to the number of iterations, it is difficult to determine a theoretical limit and it has been experimentally observed over a large and variate set of tests that it is possible to establish a linear upper bound with very low slope (less than 40 iterations for 6000 samples). Therefore, we can state that a quadratic dependence represents a very conservative upper limit for the complexity of USMO. In particular, we measured the processing time of all methods for an increasing number of unlabeled samples and with different kernel functions. Figures 3.4 and 3.5 show elapsed time and generalization performance with the linear kernel, while Figures 3.6 and 3.7 show the results achieved with a Gaussian kernel. In most cases, and especially for linear kernel, USMO outperforms all competitors. For Gaussian kernel and few unlabeled samples USMO may require higher computation than ramp loss [37], however, its performance consistently increases with the number of unlabeled samples and its lower storage requirements allow using it also when other methods run out of memory (see results for MNIST in Figure 3.6).

Chapter 4

Deep Generative Models

Performing density estimation in high-dimensional feature spaces is a well-known problem in machine learning. This work shows that it is possible to formulate the optimization problem as a minimization and use the representational power of neural networks to learn very complex densities. A theoretical bound on the estimation error is given when dealing with finite number of samples. The proposed theory is corroborated by extensive experiments on different datasets and compared against several existing approaches from the families of generative adversarial networks and autoencoder-based models.

4.1 Background

Deep generative models, like autoregressive models [97], generative adversarial networks [98] and variational autoencoders [99], represent very promising research directions for solving the problem of density estimation. Each of these families have their own limitations. The performance of existing autoregressive models based on recurrent neural networks degrade as the number of features (e.g the resolution of images) in data grows. Generative adversarial networks are difficult to train due to the mini-max nature of the optimization problem and therefore it becomes difficult to ensure global convergence of algorithms without making very hard assumptions, like having networks with infinite capacity. The performance of variational autoencoders are strongly dependent on the choice of posterior density with direct consequence on the quality of the learnt density.

In this work, we show that it is possible to cast the problem of density estimation as a minimization problem, thus overcoming the difficulties encountered during the mini-max optimization in generative adversarial networks. This allows to exploit existing optimization routines for neural networks and converge to global optimal solutions. Furthemore, we provide finite-sample bounds on the estimation error of the true density. The whole framework is validated through extensive experimental analysis on a variety of synthetic and real-world datasets by comparing the proposed model against several state-of-the-art methods, including generative adversarial networks, adversarial autoencoders and variational autoencoders.

4.2 Related work

The most promising research directions for generative models are autoregressive models, generative adversarial networks (GANs) and variational autoencoders (VAEs).

Autoregressive models [97] convert the problem of density estimation into a sequence problem based on the fact that the unknown density function can be expressed as a product of conditional distributions. Recurrent neural networks [100] are generally used to learn these conditional distributions. The main drawback is due to the difficulty of recurrent neural networks in catching long-term dependencies. Therefore, performance tend to degrade as the length of the sequences increase. Another limitation of these models is the absence of a latent representation of data.

Generative adversarial networks [98] cast the problem of density estimation as a mini-max game between two neural networks, namely a discriminator, that tries to distinguish between true and generated samples, and a generator, that tries to produce samples similar to the true ones, to fool the discriminator. They have the reputation of being difficult to train and also require careful design of network architectures [101]. Some of the most known issues are (i) the problem of vanishing gradients [102], which happens when the output of the discriminator is saturated, because true and generated data are perfectly classified, and no more gradient information is provided to the generator, (ii) the problem of mode collapse [103], which happens when the samples from the generator collapse to a single point corresponding to the maximum output value of the discriminator, and (iii) the problem of instability associated with the failure of convergence, which is due to the intrinsic nature of the mini-max problem.

Plethora of solutions exist in the literature and therefore we focus only on the most relevant ones. To solve the problem of vanishing gradients, the original paper of GANs [98] proposes to use a different objective for the generator, called the $-\log D$ alternative. The improved version from the same authors [104] proposes another objective for the generator which is based on feature matching, namely comparing the mean statistics of true and generated data on features extracted by the discriminator. The work of [105] incorporates multiple discriminators into the original framework, to ensure that gradient continually flows. Arjowski et al. [102] analyzes the problem of vanishing gradients from a theoretical perspective and provides a solution, which consists of adding instance noise during training to increase the attraction force between the generated and the true data manifold.

To solve the problem of mode collapse and more generally the problem of instability, authors in [106] introduce a regularizer in the optimization objective, that acts as a form of consensus between the discriminator and the generator and favors the convergence of training. Nevertheless, only local convergence is guaranteed. Metz et al. [103] propose an unrolled optimization of the discriminator. The update of the generator is computed by backpropagating the gradient through multiple updates of the discriminator and generated samples are not any more attracted to a single data point. Zhao et al. [107] formulate the problem differently. In particular, they consider the discriminator as an energy function, assigning low scores to samples on true data manifold and high scores to samples on generated manifold, and use an autoencoder as discriminator network. The authors show experimentally that the proposed model leads to better stability during training. The work of [108] considers the application of image generation and introduce a sort of curriculum in the training of GANs. A progressive growing of the network capacity and of the image resolution allows to stabilize the training. Their model is able to generate images at a very high resolution. Nevertheless, the method is heavily based on heuristics and more research effort is required in order to formalize the approach and generalize it to other domains.

GANs can be formulated also as a divergence minimization problem. The seminal paper of Goodfellow et al. [98] shows that, under the assumption of optimal discriminator, the generator objective is equivalent to computing the Jensen Shannon divergence between the true and the generated density. Authors [109] extend the analysis to a broader families of divergences, called $f-$divergences. They compare the different measures from this class and then provide some experimental insights on which divergence to choose for natural images. The work of [110] defines some pathological examples, for which many divergences, including the Jensen Shannon, yield to suboptimal solutions for the generator, and therefore propose to use the Wasserstein distance. A heuristic based on weight clipping is used to constrain the critic[1] to lie in the class of $1-$Lipschitz functions. The follow-up paper of [111] substitute this heuristic with a gradient penalty.

Another research direction for GANs consists on using integral probability metrics [112] as optimization objective. In particular, the maximum mean discrepancy [113] can be used to measure the distance between $p_{\mathbf{X}}$ and $q_{\mathbf{X}}$ and train the generator network. The general problem is formulated in the following way:

$$\inf_{g\in\mathcal{G}}\sup_{f\in\mathcal{F}}\left\{E_{\mathbf{x}\sim p_{\mathbf{X}}}[f(\mathbf{x})]-E_{\mathbf{x}\sim q_{\mathbf{X}}}[f(\mathbf{x})]\right\}$$

In generative moment matching networks [114, 115] $\mathcal{F}$ is a RKHS, which is induced by the Gaussian kernel. One drawback of these models is due to the fact that no maximization

[1]Namely, the discriminator in the traditional formulation of GANs.

is performed over $\mathcal{F}$ and the resulting solutions are suboptimal. Another limitation is due to the fact that the similarity scores associated with the kernel function are directly computed in the sample space. Therefore, the performance degrade as the dimensionality of the feature space increases [116]. The work of [117] introduces an encoding function to represent data in a more compact way and distances are computed in the latent representation, thus solving the problem of dimensionality. Authors [118] propose to extend the maximum mean discrepancy and include also covariance statistics to ensure better stability.

A common drawback of GANs is the lack of a latent representation of data. Authors [119] add an autoencoder network to the original framework for reconstructing part of the latent code. The identical works of [120] and [121] propose to add an encoding function together with the generator and perform an adversarial game to ensure that the joint density on the input/output of the generator agrees with the joint density of the input/output of the encoder. They prove that the optimal solution is achieved when the generator and the encoder are invertible. In practice, they fail to guarantee the convergence to that solution due to the adversarial nature of the game. Authors [122] extend the previous works by explicitly imposing the invertibility condition. They achieve this by adding a term to the generator objective that computes the reconstruction error on the latent space. Adversarial autoencoders [123] are similar to these approaches with the only differences that the estimation of the reconstruction error is performed in the sample space, while the adversarial game is performed only in the latent space.

It is important to mention that all of these works are based on a mini-max problem, while our method solves a simple minimization problem, for which it is possible to achieve global convergence.

The most related work to ours is the one proposed by [124]. The authors optimize a similar objective and use the extended version of the Plummer kernel. The computation of distances is performed directly in the sample space and performance degrade as the number of features increases. Another drawback is that their model does not allow to infer a latent representation of data.

Variational autoencoders [99] represent another family of implicit generative models. The framework is based on maximizing the log-likelihood of training data. In order to efficiently sample from the prior and therefore estimate the log-likelihood, the authors introduce an inference network which gives information about the most likely regions in the latent space to sample from. Nevertheless, the performance of the model are strongly dependent on the capability of the inference network to approximate the true posterior density. For this reason, the work of [125] proposes an adversarial game to enforce this condition.

$$\begin{tikzcd}
 & & p_{\mathbf{Z}}(\mathbf{z}) \arrow[d, leftrightarrow] & & \\
p_{\mathbf{X}}(\mathbf{x}) \arrow[d, phantom, "\wr"] & & q_{\mathbf{Z}}(\mathbf{z}) \arrow[d, phantom, "\wr"] & & q_{\mathbf{Y}}(\mathbf{y}) \arrow[d, phantom, "\wr"] \\
\mathbf{x} \arrow[r] & \boxed{f(\cdot)} \arrow[r] & \mathbf{z} \arrow[r] & \boxed{g(\cdot)} \arrow[r] & \mathbf{y}
\end{tikzcd}$$

Figure 4.1: Visualization of the encoding and decoding functions with associated densities.

4.3 Proposed Solution

This section deals with the problem of density estimation. The goal is to estimate the unknown density function $p_{\mathbf{X}}(\mathbf{x})$, whose support is defined by $\Omega_{\mathbf{x}} \subset \mathbb{R}^d$.

We start by providing the overall idea of learning the density underlying the training data, then we formulate the optimization problem used in our approach and finally we analyze the properties of the objective function and provide a bound on the estimation error.

We consider two continuous functions $f : \Omega_{\mathbf{x}} \to \Omega_{\mathbf{z}}$ and $g : \Omega_{\mathbf{z}} \to \Omega_{\mathbf{x}}$, where $\Omega_{\mathbf{z}} \subseteq \mathbb{R}^h$ and h is equal to the intrinsic dimensionality of $\Omega_{\mathbf{x}}$. Furthermore, we consider that $g(f(\mathbf{x})) = \mathbf{x}$ for every $\mathbf{x} \in \Omega_{\mathbf{x}}$, namely that g is the left inverse for f on domain $\Omega_{\mathbf{x}}$. In this work, f and g are neural networks parameterized by vectors $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$, respectively. f is called the encoding function, taking a random input $\mathbf{x}$ with density $p_{\mathbf{X}}(\mathbf{x})$ and producing a random vector $\mathbf{z}$ with density $q_{\mathbf{Z}}(\mathbf{z})$, while g is the decoding function taking $\mathbf{z}$ as input and producing the random vector $\mathbf{y}$ distributed according to $q_{\mathbf{Y}}(\mathbf{y})$. Note that, $p_{\mathbf{X}}(\mathbf{x}) = q_{\mathbf{Y}}(\mathbf{y})$, since $\mathbf{y} = g(\mathbf{z}) = g(f(\mathbf{x})) = \mathbf{x}$ for every $\mathbf{x} \in \Omega_{\mathbf{x}}$. This is already a density estimator, but it has the drawback that in general $q_{\mathbf{Z}}(\mathbf{z})$ and $q_{\mathbf{Y}}(\mathbf{y})$ cannot be written in closed form. Now, define $p_{\mathbf{Z}}(\mathbf{z})$ an arbitrary density with support $\Omega_{\mathbf{z}}$, that has a closed form. Our goal is to guarantee that $q_{\mathbf{Z}}(\mathbf{z}) = p_{\mathbf{Z}}(\mathbf{z})$ on the whole support, while maintaining $g(f(\mathbf{x})) = \mathbf{x}$ for every $\mathbf{x} \in \Omega_{\mathbf{x}}$. This allows us to use the decoding function as a generator and produce samples distributed according to $p_{\mathbf{X}}(\mathbf{x})$. Therefore, the problem of density estimation in a high-dimensional feature space is converted into a problem of estimation in a lower dimensional vector space, thus overcoming the curse of dimensionality. Figure 4.1 provides a graphical interpretation for these concepts.

The objective of our minimization problem is defined as follows:

$$\mathcal{L}(f, g) = \int_{\Omega_{\mathbf{x}}} \|\mathbf{x} - g(f(\mathbf{x}))\|^2 p_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} + \lambda \int_{\Omega_{\mathbf{z}}} \int_{\Omega_{\mathbf{z}}} \phi(\mathbf{z}) \phi(\mathbf{z}') k(\mathbf{z}, \mathbf{z}') d\mathbf{z} d\mathbf{z}' \tag{4.1}$$

where $\phi(\mathbf{z}) = p_{\mathbf{Z}}(\mathbf{z}) - q_{\mathbf{Z}}(\mathbf{z})$, $k(\cdot, \cdot)$ is a positive definite kernel and $\lambda > 0$ is a regularization parameter used to normalize the two addends.

Note that the first term in (4.1) reaches its global minimum when the encoding and the decoding functions are invertible on support $\Omega_{\mathbf{x}}$, while the second term in (4.1) is globally optimal when $q_{\mathbf{Z}}(\mathbf{z})$ equals $p_{\mathbf{Z}}(\mathbf{z})$ (see Lemma 5). Therefore, the global minimum of (4.1) satisfies our initial requirements and the optimal solution corresponds to the case where $q_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{X}}(\mathbf{x})$.

The following lemma proves that the global minimum of the second term in (4.1) is achieved when $q_{\mathbf{Z}}(\mathbf{z})$ equals $p_{\mathbf{Z}}(\mathbf{z})$ (proof in the supplementary material).

Lemma 5. *Given $k : \Omega_{\mathbf{z}} \times \Omega_{\mathbf{z}} \rightarrow \mathbb{R}$ a symmetric positive definite kernel, then:*

(a) there exists a unique Hilbert space $\mathcal{H}$ of real-valued functions over $\Omega_{\mathbf{z}}$, for which k is a reproducing kernel. $\mathcal{H}$ is therefore a Reproducing kernel Hilbert Space (RKHS).

(b) For all $\ell \in \mathcal{H}$

$$\int_{\Omega_{\mathbf{z}}}\int_{\Omega_{\mathbf{z}}} \phi(\mathbf{z})\phi(\mathbf{z}')k(\mathbf{z},\mathbf{z}')d\mathbf{z}d\mathbf{z}' = MMD^2(p_{\mathbf{Z}}, q_{\mathbf{Z}})$$

where

$$MMD(p_{\mathbf{Z}}, q_{\mathbf{Z}}) \doteq \sup_{\|\ell\|_{\mathcal{H}}\leq 1} \Big\{E_{\mathbf{z}\sim p_{\mathbf{Z}}}[\ell(\mathbf{z})] - E_{\mathbf{z}\sim q_{\mathbf{Z}}}[\ell(\mathbf{z})]\Big\}$$

is the maximum mean discrepancy between $p_{\mathbf{Z}}(\mathbf{z})$ and $q_{\mathbf{Z}}(\mathbf{z})$.

(c) Let $\mathcal{H}$ be defined as in (b), then $MMD(p_{\mathbf{Z}}, q_{\mathbf{Z}}) = 0$ if and only if $p_{\mathbf{Z}}(\mathbf{z}) = q_{\mathbf{Z}}(\mathbf{z})$.

The choice of the kernel function is of fundamental importance. In fact, authors in [126] have proved that gradient-based algorithms converge uniformly to the global solution of the second term in (4.1), only for few choices of kernel function, e.g. the Plummer kernel, which is defined as $k(\mathbf{z},\mathbf{z}') = 1/(\sqrt{\|\mathbf{z}-\mathbf{z}'\|^2+\epsilon})^{\beta}$, $\epsilon > 0$ is an arbitrary small parameter used to avoid the singularity of k at $\mathbf{z} = \mathbf{z}'$ and $\beta = h-2$. [2] The Gaussian kernel does not satisfy this property and therefore cannot be used to directly minimize the second term in (4.1). Practically speaking, no local minima are present in the function space when considering the Plummer kernel, whereas for other kernels the problem of local minima may arise. In reality, local minima are present when considering the parameter space of neural networks. Nevertheless, it is theoretically shown that for sufficiently powerful networks almost all local minima are global minima [127].

The result of [126] holds also for our objective, since the first addend in (4.1) is not affected by the kernel choice. We use the extended version of the Plummer kernel ($\beta = 1$), which has the same above-mentioned properties but leads to faster convergence [124].

It is important to mention that the integrals in (4.1) cannot be computed exactly since $p_{\mathbf{X}}(\mathbf{x})$ is unknown and $q_{\mathbf{Z}}(\mathbf{z})$ is not defined explicitly. As a consequence, we use the unbiased estimate of (4.1) as a surrogate for optimization, namely:

$$\hat{\mathcal{L}}(f,g)=\frac{1}{N}\sum_{\mathbf{x}_i\in\mathcal{D}_{\mathbf{x}}}\|\mathbf{x}_i-g(f(\mathbf{x}_i))\|^2+\lambda\Bigg\{\frac{1}{N(N-1)}\sum_{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}}\sum_{\substack{\mathbf{z}_j\in\mathcal{D}_{\mathbf{z}}\\ j\neq i}}k(\mathbf{z}_i,\mathbf{z}_j)$$

[2]See Sec. 3 in [126].

$$- \frac{2}{N(N-1)} \sum_{\mathbf{z}_i \in \mathcal{D}_\mathbf{z}} \sum_{\substack{\mathbf{z}_j \in \mathcal{D}^f_\mathbf{z} \\ j \neq i}} k(\mathbf{z}_i, \mathbf{z}_j) + \frac{1}{N(N-1)} \sum_{\mathbf{z}_i \in \mathcal{D}^f_\mathbf{z}} \sum_{\substack{\mathbf{z}_j \in \mathcal{D}^f_\mathbf{z} \\ j \neq i}} k(\mathbf{z}_i, \mathbf{z}_j) \Bigg\} \tag{4.2}$$

where $\mathcal{D}_\mathbf{x}=\{\mathbf{x}_i\}_{i=1}^N$, $\mathcal{D}_\mathbf{z}=\{\mathbf{z}_i\}_{i=1}^N$ and $\mathcal{D}^f_\mathbf{z}=\{f(\mathbf{x}_i)\}_{i=1}^N$ are two finite set of samples drawn from $p_\mathbf{X}(\mathbf{x})$, $p_\mathbf{Z}(\mathbf{z})$ and $q_\mathbf{Z}(\mathbf{z})$, respectively. Note that the first term in (4.2) corresponds to the reconstruction error on training data. Therefore, our model can be considered as an autoencoder. Based on this fact and on the chosen kernel, we refer to our model as Plummer autoencoder (PAE).

The following theorem provides a lower bound for the objective in (4.2) and a probabilistic bound on the estimation error between $\widehat{\mathcal{L}}(f,g)$ and $\mathcal{L}(f,g)$ (proof in the supplementary material).

Theorem 1. *Given the objective in (4.2), $\Omega_\mathbf{z}$ a compact set, $\Omega_\mathbf{x} = [-M, M]^d$, where $M \in \mathbb{R}^+$, and a symmetric, continuous and positive definite kernel $k : \Omega_\mathbf{z} \times \Omega_\mathbf{z} \to \mathbb{R}$, where $0 \leq k(\mathbf{z}, \mathbf{z}') \leq K$ for all $\mathbf{z}, \mathbf{z}' \in \Omega_\mathbf{z}$ and $K = k(\mathbf{z}, \mathbf{z})$.*

(a) There exists $\widetilde{k} : \Omega_\mathbf{z} \times \Omega_\mathbf{z} \to \mathbb{R}$ such that $k(\mathbf{z}, \mathbf{z}') = \int_{\Omega_\mathbf{z}} \widetilde{k}(\mathbf{z}, \mathbf{u})\widetilde{k}(\mathbf{z}', \mathbf{u})d\mathbf{u}$.

(b) The objective in (4.2) is equivalent to

$$\begin{aligned}\widetilde{\mathcal{L}}(f,g) =& \frac{1}{N} \sum_{\mathbf{x}_i \in \mathcal{D}_\mathbf{x}} \|\mathbf{x}_i - g(f(\mathbf{x}_i))\|^2 \\ &+ \lambda \Bigg\{ \frac{N}{N-1} \int_{\Omega_\mathbf{z}} \Big(p_{\mathbf{Z},\widetilde{k}}(\mathbf{u}) - q_{\mathbf{Z},\widetilde{k}}(\mathbf{u})\Big)^2 d\mathbf{u} - \frac{2K}{N-1} + \frac{2}{N(N-1)} \sum_{\substack{\mathbf{z}_i \in \mathcal{D}_\mathbf{z} \\ \mathbf{z}'_i \in \mathcal{D}^f_\mathbf{z}}} k(\mathbf{z}_i, \mathbf{z}'_i) \Bigg\}\end{aligned}$$

where $p_{\mathbf{Z},\widetilde{k}}(\mathbf{u}) = \frac{1}{N}\sum_{\mathbf{z}_i \in \mathcal{D}_\mathbf{z}} \widetilde{k}(\mathbf{z}_i, \mathbf{u})$ and $q_{\mathbf{Z},\widetilde{k}}(\mathbf{u}) = \frac{1}{N}\sum_{\mathbf{z}'_i \in \mathcal{D}^f_\mathbf{z}} \widetilde{k}(\mathbf{z}'_i, \mathbf{u})$ are the kernel density approximations of $p_\mathbf{Z}(\mathbf{u})$ and $q(\mathbf{u})$, respectively.

(c) Then, for any $s, t > 0$

$$Pr\Big\{|\widehat{\mathcal{L}} - \mathcal{L}| > t + \lambda s\Big\} \leq 2\exp\left\{-\frac{Nt^2}{8M^4d^2}\right\} + 2\exp\left\{-\frac{\lfloor N/2 \rfloor s^2}{8K^2}\right\}$$

Statement (b) of Theorem 1 provides an equivalent formulation of the objective in (4.2). This highlights the fact that[3] $\widetilde{\mathcal{L}}(f,g)$ is always greater than $-\frac{2\lambda K}{N-1}$ and therefore greater than zero when N is large.

Note that the most influential terms in $\widetilde{\mathcal{L}}(f,g)$ are the reconstruction error and the integral term, since the last addend decays faster to zero as the number of samples increases. Therefore, the model learns to reconstruct the traning data with low error and

[3]When the second and the last addend in $\widetilde{\mathcal{L}}(f,g)$ are zero.

Table 4.1: Test log-likelihood (LL) for different models on grid dataset.

Method	PAE	CouGAN	WGAN	BiGAN	VEEGAN	VAE	AAE	AVB-AC
LL	-4.8±0.4	-4.5±0.2	-4.1±0.1	-103.4±98.9	-26.9±6.8	-5.5±0.2	-7.5±0.4	-7.5±0.8

learns to match $q_{\mathbf{Z},\widetilde{k}}(\mathbf{z})$ with $p_{\mathbf{Z},\widetilde{k}}(\mathbf{z})$. Consequently, we can state that for sufficiently large N our model generalizes.

Statement (c) provides a probabilistic bound on the estimation error between $\widehat{\mathcal{L}}(f, g)$ and $\mathcal{L}(f, g)$. The bound consists of two terms which both vanish when N is large. The first one, due to the reconstruction error term, depends also on the number of features d and on the side length M of the hypercube $\Omega_{\mathbf{x}}$, while the second one, due to the estimation of the second integral in (4.1), depends on the maximum value of the kernel function K.

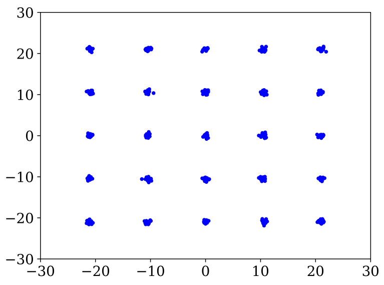
In Algorithm 2, we provide the complete procedure of PAE. Note that this requires to minimize only a single objective function.

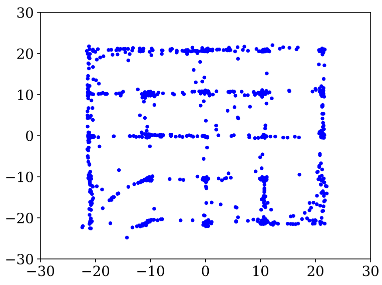
Algorithm 2 PAE, our proposed algorithm. All experiments in the paper used the default values $\epsilon = 0.0001$, $\eta = 0.0001$, $\beta_1 = 0.5$, $\beta_2 = 0.9$

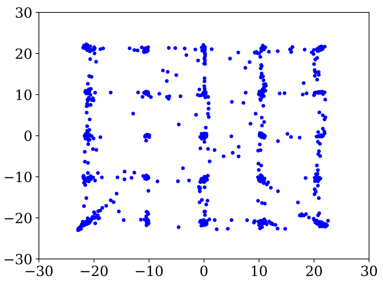
```
Input: $N$ mini-batch size, $\eta$ learning rate.
Input: $\boldsymbol{\gamma}_0$ inital parameter vector for $f$, $\boldsymbol{\theta}_0$ initial parameter vector for $g$.
repeat
    Sample $\{\mathbf{x}_i\}_{i=1}^N \sim p_{\mathbf{X}}(\mathbf{x})$ a batch from training data.
    Sample $\{\mathbf{z}_i\}_{i=1}^N \sim p_{\mathbf{Z}}(\mathbf{z})$ a batch from prior samples.
    $\boldsymbol{g}_{\boldsymbol{\gamma}} \leftarrow \nabla_{\boldsymbol{\gamma}}\widehat{\mathcal{L}}(f,g)$.
    $\boldsymbol{g}_{\boldsymbol{\theta}} \leftarrow \nabla_{\boldsymbol{\theta}}\widehat{\mathcal{L}}(f,g)$.
    $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} - \eta \cdot Adam(\boldsymbol{\gamma}, \boldsymbol{g}_{\boldsymbol{\gamma}}, \beta_1, \beta_2)$.
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot Adam(\boldsymbol{\theta}, \boldsymbol{g}_{\boldsymbol{\theta}}, \beta_1, \beta_2)$.
until $\boldsymbol{\gamma}, \boldsymbol{\theta}$ have not converged
```

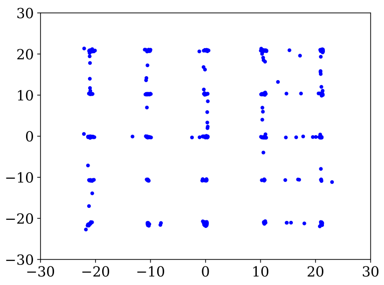
4.4 Experiments

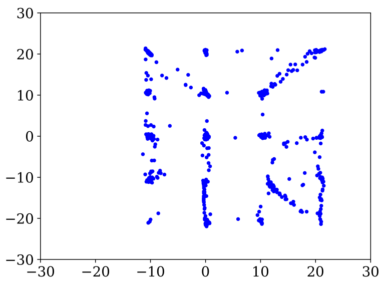
Our proposed solution is compared against 7 methods, namely Coulomb GAN (CouGAN) [124], which is the most closely related method, the improved version of Wasserstein GAN (WGAN) [111], Bidirectional GAN (BiGAN) [120], Variational Encoder Enhancement to GANs (VEEGAN) [122], Variational Autoencoders (VAE) [99], Adversarial Autoencoders (AAE) [123] and Adversarial Variational Bayes with Adaptive Contrast (AVB-AC) [125]. Note that the first four competitors belong to the family of generative adversarial networks, while the last three are generative models based on autoencoders. We make a huge

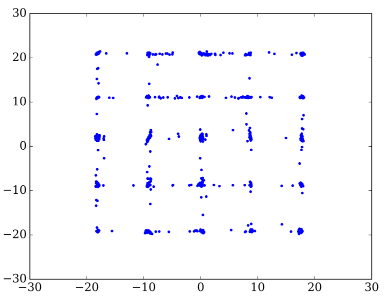

(a) True Data

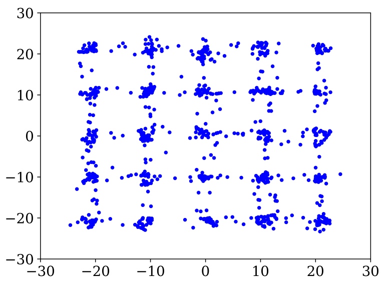

(b) PAE

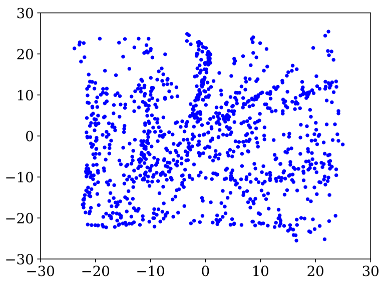

(c) CouGAN


(d) WGAN


(e) BiGAN


(f) VEEGAN


(g) VAE


(h) AAE

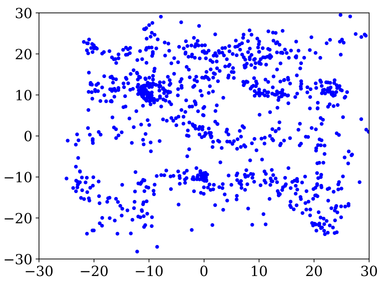

(i) AVB-AC

Figure 4.2: Visualization of generated data from different models on grid dataset.

effort in implementing and comparing all approaches on different synthetic and real-world datasets. The code to replicate all experiments, including the ones for the competitors are available at `https://github.com/TRENTO-AI/PAE`. Note that in all experiments $p_{\boldsymbol{Z}}(\boldsymbol{z}) = \mathcal{U}([-1,1]^h)$.

4.4.1 Grid dataset

We start by comparing the approaches on a two-dimensional dataset consisting of 25 isotropic Gaussians placed according to a grid (see Figure 4.2(a)), and call it the grid dataset [128]. The training dataset contains 500 samples generated from the true density.

Following the methodology of other works (see for example [128, 124]), we choose

Table 4.2: Test log-likelihood (LL) for different models on low dimensional embedding dataset.

Method	PAE	CouGAN	WGAN	BiGAN	VEEGAN	VAE	AAE	AVB-AC
LL	1224.4±76.3	Failure	848.9±534.4	-3940.3±3519.4	-7457806.0±4036262.3	-320.1±1081.6	1025.9±177.1	-965.4±1196.9

fully connected MLPs with two hidden layers (128 neurons each) in encoder, decoder and discriminator networks and set $h = 2$. All models are trained for two million iterations using Adam optimizer.

In our model, λ is chosen from the range $\{0.1, 1, 10\}$ to obtain the lowest value of objective at the end of training.[4] For other models, we strictly follow the details of the original papers. It is important to mention that we observe poor performance in VEEGAN and better results are obtained by applying batch normalization to all hidden layers (results without batch normalization and details of simulations are available in the supplementary material).

Models are evaluated qualitatively by visual inspecting generated samples and quantitatively by computing the log-likelihood on test data. To compute log-likelihood, we first apply kernel density estimation using a Gaussian kernel on 10^5 generated samples[5] and then evaluate the log-likelihood on 10^4 test samples from the true distribution. Results are averaged over 10 repetitions.

Figure 4.2 shows samples generated by all models, while Table 4.1 provides quantitative results. It is immediate to see that BiGAN is affected by mode collapse, while AAE and AVB-AC generate very noisy samples. VEEGAN achieves very low performance in terms of log-likelihood since it assigns high prior to only few modes. Our model, VAE and CouGAN compare favourably with WGAN, which in turn obtain the best performance.

4.4.2 Low dimensional embedding dataset

The second dataset consists of ten $10D$ isotropic Gaussians embedded in a 1000 dimensional vector space and call it the low dimensional embedding dataset. We generate 500 samples from the true density to train all models.

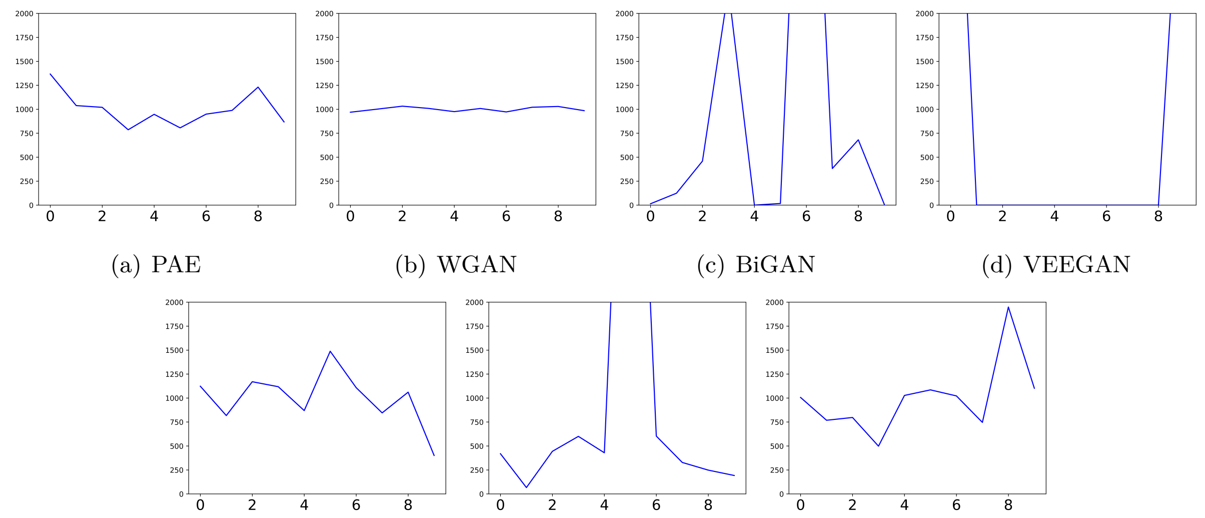
The methodology is similar to the one used with previous dataset.[6] The main difference is in the evaluation. Due to the difficulty of visualizing samples in high dimensions, we propose to use a classifier[7] to count the number of samples generated by the models for

[4]In this set of experiments, $\lambda = 10$.

[5]Bandwidth is selected from a subset of values obtained from the interval $[10^{-3}, 10^{1.5}]$ by sampling 10 values equally spaced in logarithmic scale.

[6]In this set of experiments, $\lambda = 1$.

[7]Consisting of a MLP with two hidden layers of 256 and 128 neurons each and trained on an infinitely large dataset sampled from the true density for 1000 iterations and using Adam with learning rate equal to 0.001.



(a) PAE (b) WGAN (c) BiGAN (d) VEEGAN

(e) VAE (f) AAE (g) AVB-AC

Figure 4.3: Detection of mode collapse through classification of generated samples on the low dimensional embedding dataset. The x-axis of each plot represents modes (classes), while the y-axis counts the number of generated samples for a given mode.

each mode. This evaluation procedure allows to only detect the presence of mode collapse. It is important to mention that this procedure can be fooled by specific pathological cases, like memorization of training samples. Therefore, we use log-likelihood on test data to assess the quality of the learnt distribution. The other differences are in the use of batch normalization for BiGAN and AAE, which lead to better performance. We experience high instability when training CouGAN and also convergence failures.

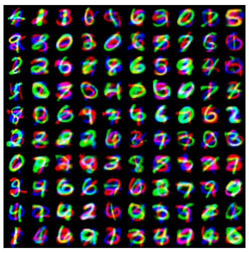
Figure 4.3 shows the histograms of generated samples obtained by all models. Note that BiGAN and VEEGAN are affected by mode collapse. Table 4.2 provides log-likelihood scores. Our model achieves the best performance, meaning that it is able to better estimate the underlying true density.

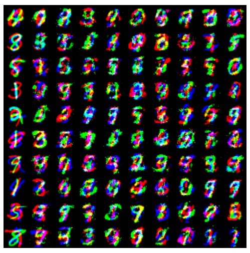
4.4.3 Stacked-MNIST

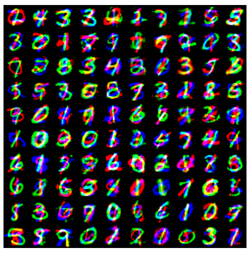
The third dataset is created by stacking random digits from the MNIST dataset on top of each other to produce colored images [122]. This creates a ground truth density with 1000 classes.

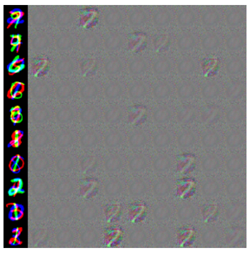
The methodology is similar to the one of previous dataset.[8] We use the MLP network from [129] as encoder, decoder and discriminator networks (see supplementary material for further details and simulations). We train all models up to 1000 epochs with batch

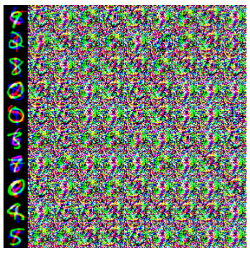
[8]In this set of experiments, $\lambda = 0.1$.

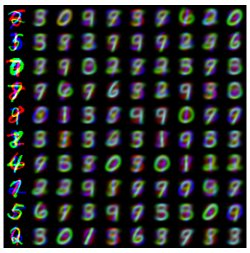

(a) PAE

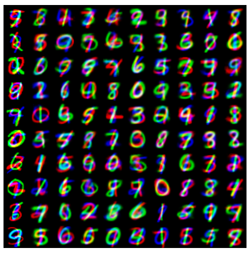

(b) CouGAN


(c) WGAN


(d) BiGAN


(e) VEEGAN


(f) VAE


(g) AAE

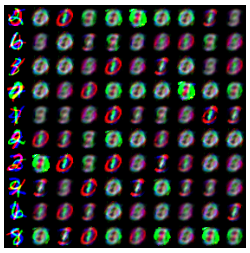

(h) AVB-AC

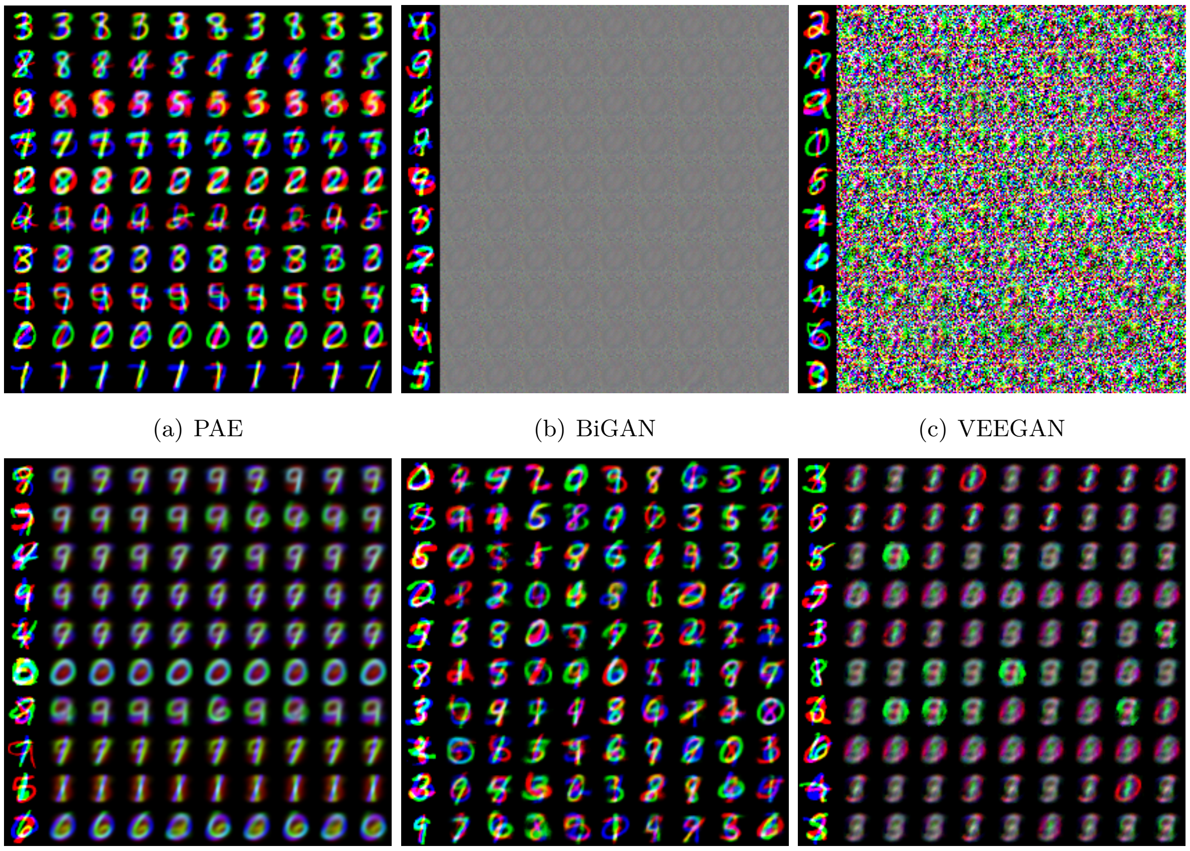
Figure 4.4: Visualization of generated samples from different models on Stacked-MNIST. The first column of each plot contains true samples.

Table 4.3: Average number of training iterations per second for different models on Stacked-MNIST.

Method	PAE	CouGAN	WGAN	BiGAN	VEEGAN	VAE	AAE	AVB-AC
iters/secs	40.4	29.0	7.1	14.7	15.3	48.4	10.4	25.8

size equal to 128. We visualize both generated samples and nearest neighbors in the latent space, to understand the quality of the learnt representation and see whether semantic consistency is preserved in the neighborhood of samples. Similarly to the previous case, we use batch normalization for BiGAN, VEEGAN and AAE.

Figure 4.4 shows the generated samples of each model. Note that BiGAN and VEEGAN fail to learn the true distribution, while VAE and AVB-AC suffer from mode collapse. CouGAN seems to generate more diverse samples, but with very strong artifacts. Only PAE, AAE and WGAN are able to learn a good approximation of the true density. Figure 4.5 shows nearest neighbors for true data. It is worth to mention that AAE does not preserve semantic consistency in the representation. In fact, for a small perturbation of the latent representation the output image can completely change its semantic content. Instead our model is capable to fulfill this property. Finally, Table 4.3 shows an indicative measure of the training rates achieved by all methods. It is quite evident that AAE and



(a) PAE (b) BiGAN (c) VEEGAN

(d) VAE (e) AAE (f) AVB-AC

Figure 4.5: Conditional generation of samples. The first column of each plot contains true samples, while the other columns are obtained by generating samples from the latent codes associated to true data and perturbing the latent representation with additive isotropic Gaussian noise (0.01 element variance).

our method outperform the other approaches in terms of training speed.

Chapter 5

Conclusions and Future Directions

This dissertation focuses on the exploitation of unlabeled data in machine learning. We have shown how to learn a semi-supervised model in the context of limited data, we have improved the scalability of existing frameworks for positive unlabeled learning and finally we have provided a new formulation for the problem of density estimation in unsupervised learning, improving the stability and the efficiency of training in deep generative models. The code of all proposed algorithms is available online to guarantee the reproducibility of the experiments.

We believe that the research introduced in this dissertation represents an initial step towards achieving general artificial intelligence. We think that understanding how to exploit unlabeled data for learning, more specifically solving the problem of unsupervised learning, represents a milestone for this long-term goal: firstly because we believe that supervised information (provided in the form of class labels) is also originated from an unsupervised process, and secondly because the amount of unlabeled data is largely available. Future research will continue to investigate this topic. In particular, we will answer to the two following questions:

1. Is it possible to learn motor skills in robots in a completely unsupervised way (without reward functions)?

2. Is it possible to develop a natural language between agents with unsupervised learning?

Bibliography

[1] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning.* The MIT Press, 2010.

[2] K. P. Bennett and A. Demiriz. Semi-Supervised Support Vector Machines. In *Advances in Neural Information Processing Systems*, pages 368–374, 1999.

[3] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[4] I. Färber, S. Günnemann, H-P Kriegel, P. Kröger, E. Müller, E. Schubert, T. Seidl, and A. Zimek. On Using Class-Labels in Evaluation of Clusterings. In *MULTICLUST: International Workshop on Discovering, Summarizing and Using Multiple Clusterings*, page 1, 2010.

[5] V. Moskvina, N. Craddock, P. Holmans, et al. Gene-Wide Analyses of Genome-Wide Association Data Sets: Evidence for Multiple Common Risk Alleles for Schizophrenia and Bipolar Disorder and for Overlap in Genetic Risk. *Molecular Psychiatry*, 14:252–260, 2009.

[6] D. J. Miller and J. Browning. A Mixture Model and EM-based Algorithm for Class Discovery, Robust Classification, and Outlier Rejection in Mixed Labeled/Unlabeled Data Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1468–1483, 2003.

[7] D. J. Miller, J. Raghuram, G. Kesidis, and C. M. Collins. Improved Generative Semisupervised Learning Based on Finely Grained Component-Conditional Class Labeling. *Neural Computation*, 24:1926–1966, 2012.

[8] M. H. C. Law, A. P. Topchy, and A. K. Jain. Model-based Clustering With Probabilistic Constraints. In *SIAM International Conference on Data Mining*, pages 641–645, 2005.

[9] Z. Lu. Semi-Supervised Clustering with Pairwise Constraints: A Discriminative Approach. In *International Conference on Artificial Intelligence and Statistics*, pages 299–306, 2007.

[10] Z. Li, J. Liu, and X. Tang. Pairwise Constraint Propagation by Semidefinite Programming for Semi-Supervised Classification. In *International Conference on Machine Learning*, pages 576–583, 2008.

[11] M. S. Baghshah and S. B. Shouraki. Semi-Supervised Metric Learning Using Pairwise Constraints. In *International Joint Conference on Artificial Intelligence*, pages 1217–1222, 2009.

[12] J. Raghuram, D. J. Miller, and G. Kesidis. Instance-Level Constraint-Based Semisupervised Learning With Imposed Space-Partitioning. *IEEE Transactions on Neural Networks and Learning Systems*, 25:1520–1537, 2014.

[13] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-Supervised Learning with Deep Generative Models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[14] Z. Ghahramani and M. J. Beal. Variational Inference for Bayesian Mixtures of Factor Analysers. In *Advances in Neural Information Processing Systems*, pages 449–455, 1999.

[15] C. M. Bishop. Variational Principal Components. In *International Conference on Artificial Neural Networks*, pages 509–514, 1999.

[16] H. Attias. Inferring Parameters and Structure of Latent Variable Models by Variational Bayes. In *Uncertainty in Artificial Intelligence*, pages 21–30, 1999.

[17] M. J. Beal. Variational Algorithms for Approximate Bayesian Inference. *Ph. D. Thesis*, 2003.

[18] Y. Grandvalet and Y. Bengio. Semi-Supervised Learning by Entropy Minimization. In *Advances in Neural Information Processing Systems*, pages 529–536, 2005.

[19] M. Diem, S. Fiel, A. Garz, M. Keglevic, F. Kleber, and R. Sablatnig. ICDAR 2013 Competition on Handwritten Digit Recognition (HDRC 2013). In *International Conference on Document Analysis and Recognition*, pages 1422–1427, 2013.

[20] L. Van der Maaten and G. Hinton. Visualizing Data Using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[21] S. Anand, S. Mittal, O. Tuzel, and P. Meer. Semi-Supervised Kernel Mean Shift Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:1201–1215, 2014.

[22] N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Computing Gaussian Mixture Models with EM Using Equivalence Constraints. In *Advances in Neural Information Processing Systems*, pages 465–472, 2004.

[23] S. Yi, L. Zhang, R. Jin, Q. Qian, and A. Jain. Semi-Supervised Clustering by Input Pattern Assisted Pairwise Similarity Matrix Completion. In *International Conference on Machine Learning*, pages 1400–1408, 2013.

[24] M. Bilenko, S. Basu, and R. Mooney. Integrating Constraints and Metric Learning in Semi-Supervised Clustering. In *International Conference on Machine Learning*, page 11, 2004.

[25] S. Basu, A. Banerjee, and R. Mooney. Semi-Supervised Clustering by Seeding. In *International Conference on Machine Learning*, pages 27–34, 2002.

[26] O. Chapelle and A. Zien. Semi-Supervised Classification by Low Density Separation. In *International Conference on Artificial Intelligence and Statistics*, pages 57–64, 2005.

[27] C. Curtis, S. Shah, S-F Chin, et al. The Genomic and Transcriptomic Architecture of 2,000 Breast Tumours Reveals Novel Subgroups. *Nature*, 486:346–352, 2012.

[28] R. Shen, A. Olshen, and M. Ladanyi. Integrative Clustering of Multiple Genomic Data Types Using a Joint Latent Variable Model with Application to Breast and Lung Cancer Subtype Analysis. *Bioinformatics*, 25:2906–2912, 2009.

[29] A. Kapp and R. Tibshirani. Are Clusters Found in One Dataset Present in Another Dataset? *Biostatistics*, 8:9–31, 2007.

[30] C. Perou, T. Sørlie, M. Eisen, et al. Molecular Portraits of Human Breast Tumours. *Nature*, 406:747–752, 2000.

[31] J. Parker, M. Mullins, M. Cheang, et al. Supervised Risk Predictor of Breast Cancer Based on Intrinsic Subtypes. *Journal of Clinical Oncology*, 27:1160–1167, 2009.

[32] C. Elkan and K. Noto. Learning Classifiers from Only Positive and Unlabeled Data. In *International Conference on Knowledge Discovery and Data Mining*, pages 213–220, 2008.

[33] T. Onoda, H. Murata, and S. Yamada. One Class Support Vector Machine Based Non-Relevance Feedback Document Retrieval. In *IEEE International Joint Conference on Neural Networks*, pages 552–557, 2005.

[34] S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama, and T. Kanamori. Inlier-Based Outlier Detection via Direct Density Ratio Estimation. In *IEEE International Conference on Data Mining*, pages 223–232, 2008.

[35] W. Li, Q. Guo, and C. Elkan. A Positive and Unlabeled Learning Algorithm for One-Class Classification of Remote-Sensing Data. *IEEE Transactions on Geoscience and Remote Sensing*, 49:717–725, 2011.

[36] W. J. Scheirer, A. De R. Rocha, A. Sapkota, and T. E. Boult. Toward Open Set Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1757–1772, 2013.

[37] M. C. Du Plessis, G. Niu, and M. Sugiyama. Analysis of Learning from Positive and Unlabeled Data. In *Advances in Neural Information Processing Systems*, pages 703–711, 2014.

[38] M. C. Du Plessis, G. Niu, and M. Sugiyama. Convex Formulation for Learning from Positive and Unlabeled Data. In *International Conference on Machine Learning*, pages 1386–1394, 2015.

[39] V. N. Vapnik. An Overview of Statistical Learning Theory. *IEEE Transactions on Neural Networks*, pages 988–999, 1999.

[40] C. J. Hsieh, N. Natarajan, and I. S. Dhillon. PU Learning for Matrix Completion. In *International Conference on Machine Learning*, pages 2445–2453, 2015.

[41] J. T. Zhou, S. J. Pan, Q. Mao, and I. W. Tsang. Multi-View Positive and Unlabeled Learning. In *Asian Conference on Machine Learning*, pages 555–570, 2012.

[42] T. Sakai, M. C. Du Plessis, G. Niu, and M. Sugiyama. Beyond the Low-Density Separation Principle: A Novel Approach to Semi-Supervised Learning. 2016.

[43] X. Li, S. Y. Philip, B. Liu, and S. K. Ng. Positive Unlabeled Learning for Data Stream Classification. In *SIAM International Conference on Data Mining*, pages 257–268, 2009.

[44] M. N. Nguyen, X. L. Li, and S. K. Ng. Positive Unlabeled Leaning for Time Series Classification. In *International Joint Conference on Artificial Intelligence*, pages 1421–1426, 2011.

[45] J. Silva and R. Willett. Hypergraph-Based Anomaly Detection of High-Dimensional Co-Occurrences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:563–569, 2009.

[46] B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially Supervised Classification of Text Documents. In *International Conference on Machine Learning*, pages 387–394, 2002.

[47] H. Yu, J. Han, and K. C. C. Chang. PEBL: Positive Example Based Learning for Web Page Classification Using SVM. In *International Conference on Knowledge Discovery and Data Mining*, pages 239–248, 2002.

[48] X. Li and B. Liu. Learning to Classify Texts Using Positive and Unlabeled Data. In *International Joint Conference on Artificial Intelligence*, pages 587–592, 2003.

[49] H. Yu. Single-Class Classification with Mapping Convergence. *Machine Learning*, 61:49–69, 2005.

[50] B. Liu, Y Dai, X. Li, W. S. Lee, and P. S. Yu. Building Text Classifiers Using Positive and Unlabeled Examples. In *IEEE International Conference on Data Mining*, pages 179–186, 2003.

[51] A. Skabar. Single-Class Classifier Learning Using Neural Networks: An Application to the Prediction of Mineral Deposits. In *International Conference on Machine Learning and Cybernetics*, pages 2127–2132, 2003.

[52] G. Blanchard, G. Lee, and C. Scott. Semi-Supervised Novelty Detection. *Journal of Machine Learning Research*, 11:2973–3009, 2010.

[53] J. Kittler, W. Christmas, T. De Campos, D. Windridge, F. Yan, J. Illingworth, and M. Osman. Domain Anomaly Detection in Machine Perception: A System Architecture and Taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:845–859, 2014.

[54] M. Markou and S. Singh. Novelty Detection: A ReviewâĂŤPart 1: Statistical Approaches. *Signal processing*, 83:2481–2497, 2003.

[55] M. Markou and S. Singh. Novelty Detection: A ReviewâĂŤPart 2: Neural Network Based Approaches. *Signal processing*, 83:2499–2521, 2003.

[56] M. Koppel and J. Schler. Authorship Verification as a One-Class Classification Problem. In *International Conference on Machine Learning*, page 62, 2004.

[57] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-Class Collaborative Filtering. In *IEEE International Conference on Data Mining*, pages 502–511, 2008.

[58] B. Schölkopf, R. Williamson, A. J. Smola, and J. Shawe-Taylor. SV Estimation of a DistributionâĂŹs Support. In *Advances in Neural Information Processing Systems*, 1999.

[59] D. M. J. Tax and R. P. W. Duin. Support Vector Domain Description. *Pattern Recognition Letters*, 20:1191–1199, 1999.

[60] M. M. Moya and D. R. Hush. Network Constraints and Multi-Objective Optimization for One-Class Classification. *Neural Networks*, 9:463–474, 1996.

[61] B. Schölkopf, J. C. Platt, and A. J. Smola. Kernel Method for Percentile Feature Extraction. Technical report, Microsoft Research, 2000.

[62] D. M. J. Tax and R. P. W. Duin. Uniform Object Generation for Optimizing One-Class Classifiers. *Journal of Machine Learning Research*, 2:155–173, 2001.

[63] C. Campbell and K. P. Bennett. A Linear Programming Approach to Novelty Detection. In *Advances in Neural Information Processing Systems*, pages 395–401, 2001.

[64] E. Pekalska, D. Tax, and R. P. W. Duin. One-Class LP Classifiers for Dissimilarity Representations. In *Advances in Neural Information Processing Systems*, pages 761–768, 2002.

[65] L. M. Manevitz and M. Yousef. One-Class SVMs for Document Classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

[66] D. M. J. Tax. *One-Class Classification*. PhD thesis, Delft University of Technology, 2001.

[67] D. M. J. Tax and R. P. W. Duin. Combining One-Class Classifiers. In *International Workshop on Multiple Classifier Systems*, pages 299–308, 2001.

[68] A. D. Shieh and D. F. Kamm. Ensembles of One Class Support Vector Machines. In *International Workshop on Multiple Classifier Systems*, pages 181–190, 2009.

[69] G. Ratsch, S. Mika, B. Schölkopf, and K. R. Müller. Constructing Boosting Algorithms from SVMs: An Application to One-Class Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:1184–1199, 2002.

[70] V. Jumutc and J. A. K. Suykens. Multi-Class Supervised Novelty Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:2510–2523, 2014.

[71] G. Niu, M. C. Du Plessis, T. Sakai, Y. Ma, and M. Sugiyama. Theoretical Comparisons of Positive-Unlabeled Learning Against Positive-Negative Learning. In *Advances in Neural Information Processing Systems*, pages 1199–1207, 2016.

[72] S. S. Khan and M. G. Madden. One-Class Classification: Taxonomy of Study and Review of Techniques. *The Knowledge Engineering Review*, 29:345–374, 2014.

[73] B. M. Shahshahani and D. A. Landgrebe. The Effect of Unlabeled Samples in Reducing the Small Sample Size Problem and Mitigating the Hughes Phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, 32:1087–1095, 1994.

[74] D. J. Miller and H. S. Uyar. A Mixture of Experts Classifier with Learning Based on Both Labelled and Unlabelled Data. In *Advances in Neural Information Processing Systems*, pages 571–577, 1997.

[75] T. Zhang and F. Oles. The Value of Unlabeled Data for Classification Problems. In *International Conference on Machine Learning*, pages 1191–1198, 2000.

[76] E. Sansone, A. Passerini, and F. G. B. De Natale. Classtering: Joint Classification and Clustering with Mixture of Factor Analysers. In *European Conference on Aritificial Intelligence*, pages 1089–1095, 2016.

[77] Z. H. Zhou and M. Li. Semi-Supervised Learning by Disagreement. *Knowledge and Information Systems*, 24:415–439, 2010.

[78] Y. F. Li, I. W. Tsang, J. T. Kwok, and Z. H. Zhou. Convex and Scalable Weakly Labeled SVMs. *Journal of Machine Learning Research*, 14:2151–2188, 2013.

[79] C. C. Chang and C. J. Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27, 2011.

[80] P. H. Chen, R. E. Fan, and C. J. Lin. A Study on SMO-Type Decomposition Methods for Support Vector Machines. *IEEE Transactions on Neural Networks*, 17:893–908, 2006.

[81] R. Fergus, Y. Weiss, and A. Torralba. Semi-Supervised Learning in Gigantic Image Collections. In *Advances in Neural Information Processing Systems*, pages 522–530, 2009.

[82] A. Talwalkar, S. Kumar, and H. Rowley. Large-Scale Manifold Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[83] Q. Da, Y. Yu, and Z. H. Zhou. Learning with Augmented Class by Exploiting Unlabeled Data. In *AAAI Conference on Artificial Intelligence*, pages 1760–1766, 2014.

[84] M. Kearns. Efficient Noise-Tolerant Learning from Statistical Queries. *Journal of the ACM*, 45:983–1006, 1998.

[85] F. De Comité, F. Denis, R. Gilleron, and F. Letouzey. Positive and Unlabeled Examples Help Learning. In *International Conference on Algorithmic Learning Theory*, pages 219–230, 1999.

[86] F. Letouzey, F. Denis, and R. Gilleron. Learning from Positive and Unlabeled Examples. In *International Conference on Algorithmic Learning Theory*, pages 71–85, 2000.

[87] J. He, Y. Zhang, X. Li, and Y. Wang. Naive Bayes Classifier for Positive Unlabeled Learning with Uncertainty. In *SIAM International Conference on Data Mining*, pages 361–372, 2010.

[88] A. Jordan. On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In *Advances in Neural Information Processing Systems*, pages 841–848, 2002.

[89] N. Aronszajn. Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, pages 337–404, 1950.

[90] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[91] M. A. Hanson. Invexity and the Kuhn–Tucker Theorem. *Journal of Mathematical Analysis and Applications*, 236:594–604, 1999.

[92] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, pages 1443–1471, 2001.

[93] S. Mehrotra. On the Implementation of a Primal-Dual Interior Point Method. *SIAM Journal on Optimization*, 2:575–601, 1992.

[94] A. L. Yuille and A. Rangarajan. The Concave-Convex Procedure (CCCP). In *Advances in Neural Information Processing Systems*, pages 1033–1040, 2002.

[95] W. S. Lee and B. Liu. Learning with Positive and Unlabeled Examples Using Weighted Logistic Regression. In *International Conference on Machine Learning*, pages 448–455, 2003.

[96] M. C. Du Plessis, G. Niu, and M. Sugiyama. Class-Prior Estimation for Learning from Positive and Unlabeled Data. In *Asian Conference on Machine Learning*, 2015.

[97] H. Larochelle and I. Murray. The Neural Autoregressive Distribution Estimator. In *International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2011.

[98] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[99] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2013.

[100] A. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel Recurrent Neural Networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016.

[101] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.

[102] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017.

[103] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled Generative Adversarial Networks. In *International Conference on Learning Representations*, 2017.

[104] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[105] I. Durugkar, I. Gemp, and S. Mahadevan. Generative Multi-Adversarial Networks. In *International Conference on Learning Representations*, 2017.

[106] L. Mescheder, S. Nowozin, and A. Geiger. The Numerics of GANs. In *Advances in Neural Information Processing Systems*, pages 1823–1833, 2017.

[107] J. Zhao, M. Mathieu, and Y. A. LeCun. Energy-Based Generative Adversarial Network. In *International Conference on Learning Representations*, 2017.

[108] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations*, 2018.

[109] S. Nowozin, B. Cseke, and R. Tomioka. f-GAN: Training Generative Neural Samplers Using Variational Divergence Minimization. In *Advances in Neural Information Processing System*, pages 271–279, 2016.

[110] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein Generative Adversarial Networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[111] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[112] A. Müller. Integral Probability Metrics and Their Generating Classes of Functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

[113] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A Kernel Method for the Two Sample Problem. Technical report, Max Planck Institute for Biological Cybernetics, 2008.

[114] Y. Li, K. Swersky, and R. Zemel. Generative Moment Matching Networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015.

[115] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. Training Generative Neural Networks via Maximum Mean Discrepancy Optimization. In *Uncertainty in Artificial Intelligence*, pages 258–267, 2015.

[116] A. Ramdas, S. J. Reddi, B. Poczos, A. Singh, and L. Wasserman. On the Decreasing Power of Kernel and Distance Based Nonparametric Hypothesis Tests in High Dimensions. In *AAAI Conference on Artificial Intelligence*, pages 3571–3577, 2015.

[117] C. L. Li, W. C. Chang, Y. Cheng, Y. Yang, and B. Póczos. MMD GAN: Towards Deeper Understanding of Moment Matching Network. In *Advances in Neural Information Processing Systems*, pages 2200–2210, 2017.

[118] Y. Mroueh, T. Sercu, and V. Goel. McGan: Mean and Covariance Feature Matching GAN. In *International Conference on Machine Learning*, pages 2527–2535, 2017.

[119] X. Chen, X.Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[120] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial Feature Learning. In *International Conference on Learning Representations*, 2017.

[121] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially Learned Inference. In *International Conference on Learning Representations*, 2017.

[122] A. Srivastava, L. Valkoz, C. Russell, M. U. Gutmann, and C. Sutton. VEEGAN: Reducing Mode Collapse in GANs Using Implicit Variational Learning. In *Advances in Neural Information Processing Systems*, pages 3310–3320, 2017.

[123] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial Autoencoders. In *International Conference on Learning Representations*, 2013.

[124] T. Unterthiner, B. Nessler, G. Klambauer, M. Heusel, H. Ramsauer, and S. Hochreiter. Coulomb GANs: Provably Optimal Nash Equilibria via Potential Fields. In *International Conference on Learning Representations*, 2018.

[125] L. Mescheder, S. Nowozin, and A. Geiger. Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks. In *International Conference on Machine Learning*, pages 2391–2400, 2017.

[126] S. Hochreiter and K. Obermayer. Optimal Kernels for Unsupervised Learning. In *IEEE International Joint Conference on Neural Networks*, pages 1895–1899, 2005.

[127] Q. Nguyen and M. Hein. The Loss Surface of Deep and Wide Neural Networks. In *International Conference on Machine Learning*, pages 2603–2612, 2016.

[128] J. H. Lim and J. C. Ye. Geometric Gan. 2017.

[129] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. *Neural Computation*, 22:3207–3220, 2010.

[130] B. Schölkopf, R. Herbrich, and A. J. Smola. A Generalized Representer Theorem. In *Computational Learning Theory*, pages 416–426, 2001.

[131] W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Appendix A

Supplementary Material for Chapter 3

A.1 Proof of the Representer Theorem

Representer Theorem 2. *Given the training set $D = D_p \cup D_n$ and the Mercer kernel k associated with the RKHS $\mathcal{H}_k$, any minimizer $f^* \in \mathcal{H}_k$ of (3.4) admits the following representation*

$$f^*(\mathbf{x}) = \sum_{\mathbf{x}_i \in D} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$$

where $\alpha_i \in \mathbb{R}$ for all i.

Proof. Similarly to [130], define Φ as the set, whose elements are the representers of the training dataset $D = D_p \cup D_n$, namely $\Psi = \Big\{\varphi_{\mathbf{x}_i} \in \mathcal{H}_k | i : \boldsymbol{x}_i \in D\Big\}$. Be $\mathcal{H}_\Psi$ the linear subspace of $\mathcal{H}_k$ spanned by the elements in Ψ and $\bar{\mathcal{H}}_\Psi$ its orthogonal complement, such that $\mathcal{H}_k = \mathcal{H}_\Psi \oplus \bar{\mathcal{H}}_\Psi$:

$$\begin{aligned}
\mathcal{H}_\Psi &= \Bigg\{ g \in \mathcal{H}_k | g = \sum_{\boldsymbol{x}_i \in D} \alpha_i \varphi_{\mathbf{x}_i}, \alpha_i \in \mathbb{R} \Bigg\} \\
\bar{\mathcal{H}}_\Psi &= \Big\{ h \in \mathcal{H}_k | \langle h, g \rangle_{\mathcal{H}_k} = 0, \forall g \in \mathcal{H}_\Psi \Big\}
\end{aligned}$$

Therefore, any function $f \in \mathcal{H}_k$ can be decomposed in two orthogonal components, namely $f = f^* + f^\perp$ where $f^* \in \mathcal{H}_\Psi$ and $f^\perp \in \bar{\mathcal{H}}_\Psi$. Evaluating the function f at the training point $\mathbf{x}_j$ is performed by exploiting the previous properties, viz.

$$\begin{aligned}
f(\mathbf{x}_j) &= \langle \varphi_{\mathbf{x}_j}, f^* + f^\perp \rangle_{\mathcal{H}_k} \\
&= \langle \varphi_{\mathbf{x}_j}, f^* \rangle_{\mathcal{H}_k} \\
&= \langle \varphi_{\mathbf{x}_j}, \sum_{\boldsymbol{x}_i \in D} \alpha_i \varphi_{\mathbf{x}_i} \rangle_{\mathcal{H}_k}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{\boldsymbol{x}_i \in D} \alpha_i \langle \varphi_{\mathbf{x}_j}, \varphi_{\mathbf{x}_i} \rangle_{\mathcal{H}_k} \\
&= \sum_{\boldsymbol{x}_i \in D} \alpha_i k(\mathbf{x}_j, \mathbf{x}_i) \\
&= \sum_{\boldsymbol{x}_i \in D} \alpha_i k(\mathbf{x}_i, \mathbf{x}_j)
\end{aligned}
$$

where the first and second equalities are due to the reproducing property of RKHS and orthogonality, respectively. The third and the fourth equalities are simple application of the inner product properties. The fifth equality holds by the definition of reproducing kernel, while the last one is valid thanks to the simmetry of any Mercer kernel. This relation highlights the fact that the evaluation of any function f at any training point $\mathbf{x}_j$ is independent of $f^{\perp}$. Consequently, since $R_{emp}(f)$ is a functional of f evaluated at all samples of the training dataset, we have that $R_{emp}(f)$ is also independent of $f^{\perp}$. In other words, $R_{emp}(f) = R_{emp}(f^*)$. Furthermore, thanks to the orthogonality property, one can express the regularization term in (3.4) in the following way:

$$
\|f\|^2_{\mathcal{H}_k} = \|f^* + f^{\perp}\|^2_{\mathcal{H}_k} = \|f^*\|^2_{\mathcal{H}_k} + \|f^{\perp}\|^2_{\mathcal{H}_k}
$$

The objective function in (3.4) can be therefore lower bounded in the following way:

$$
\begin{aligned}
\mathcal{R}_{emp}(f) + \lambda\|f\|^2_{\mathcal{H}_k} &= \mathcal{R}_{emp}(f^*) + \lambda\|f^*\|^2_{\mathcal{H}_k} + \lambda\|f^{\perp}\|^2_{\mathcal{H}_k} \\
&\geq \mathcal{R}_{emp}(f^*) + \lambda\|f^*\|^2_{\mathcal{H}_k}
\end{aligned}
$$

which is valid for any $f \in \mathcal{H}_k$. f^* is therefore the minimizer of (3.4) and it assumes the following form:

$$
f^*(\mathbf{x}) = \sum_{\boldsymbol{x}_i \in D} \alpha_i \langle \varphi_{\mathbf{x}_i}, \varphi_{\mathbf{x}} \rangle_{\mathcal{H}_k} = \sum_{\boldsymbol{x}_i \in D} \alpha_i k(\mathbf{x}_i, \mathbf{x})
$$

This concludes the proof. □

A.2 PU Learning Formulation

A.2.1 Derivation of the Primal Problem

By taking into account the definition of the double Hinge loss function and its composite loss, namely $\ell(x,y) = \max\{-xy, \max\{0, \frac{1}{2} - \frac{xy}{2}\}\}$ and $\tilde{\ell}(x,y) = -xy$, we can express the empirical risk functional (3.3) in the following way:

$$
-\frac{\pi}{p} \sum_{\boldsymbol{x}_i \in D_p} f(\mathbf{x}_i) + \frac{1}{n} \sum_{\boldsymbol{x}_i \in D_n} \max\left\{ f(\mathbf{x}_i), \max\left\{0, \frac{1}{2} + \frac{f(\mathbf{x}_i)}{2}\right\}\right\} \tag{A.1}
$$

and by exploiting the result stated by the representer theorem, the optimization problem (3.4) becomes:

$$\begin{aligned}
&\min_{\boldsymbol{\alpha},\boldsymbol{\xi},\beta}\left\{-c_1\sum_{\boldsymbol{x}_i\in D_p}\left(\sum_{\boldsymbol{x}_j\in D}\alpha_j k(\mathbf{x}_i,\mathbf{x}_j)+\beta\right)+c_2\sum_{\boldsymbol{x}_i\in D_n}\xi_i+\frac{1}{2}\sum_{\boldsymbol{x}_i\in D}\sum_{\boldsymbol{x}_j\in D}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j)\right\}\\
&\quad\text{s.t. } \xi_i\geq 0,\\
&\quad\quad\ \ \xi_i\geq\sum_{\boldsymbol{x}_j\in D}\alpha_j k(\mathbf{x}_i,\mathbf{x}_j)+\beta,\\
&\quad\quad\ \ \xi_i\geq\frac{1}{2}+\frac{1}{2}\left(\sum_{\boldsymbol{x}_j\in D}\alpha_j k(\mathbf{x}_i,\mathbf{x}_j)+\beta\right)
\end{aligned}\tag{A.2}$$

where $c_1=\frac{\pi}{2\lambda p}$, $c_2=\frac{1}{2\lambda n}$ and ξ_i is the slack variable associated with sample $\boldsymbol{x}_i\in D_n$. Notice that slack variables are used to make the objective function differentiable.

Finally, by using vector notation, (A.2) can be rewritten in a more compact form:

$$\begin{aligned}
&\min_{\boldsymbol{\alpha},\boldsymbol{\xi},\beta}\left\{-c_1\tilde{\mathbf{1}}^T\mathbf{K}\boldsymbol{\alpha}-c_1\tilde{\mathbf{1}}^T\mathbf{1}\beta+c_2\mathbf{1}_n^T\boldsymbol{\xi}+\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}\right\}\\
&\quad\text{s.t. } \boldsymbol{\xi}\succeq\mathbf{0}_n,\\
&\quad\quad\ \ \boldsymbol{\xi}\succeq\mathbf{U}\mathbf{K}\boldsymbol{\alpha}+\beta\mathbf{1}_n,\\
&\quad\quad\ \ \boldsymbol{\xi}\succeq\frac{1}{2}\mathbf{1}_n+\frac{1}{2}\mathbf{U}\mathbf{K}\boldsymbol{\alpha}+\frac{\beta}{2}\mathbf{1}_n
\end{aligned}$$

A.2.2 Derivation of the Dual Problem

The Lagrangian function for problem (3.5) is defined as follows:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\alpha},\boldsymbol{\xi},\beta,\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\delta})=&\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}-c_1\tilde{\mathbf{1}}^T\mathbf{K}\boldsymbol{\alpha}-c_1\tilde{\mathbf{1}}^T\mathbf{1}\beta+c_2\mathbf{1}_n^T\boldsymbol{\xi}\\
&-\boldsymbol{\eta}^T\boldsymbol{\xi}+\boldsymbol{\gamma}^T\Big(\mathbf{U}\mathbf{K}\boldsymbol{\alpha}+\beta\mathbf{1}_n-\boldsymbol{\xi}\Big)\\
&+\boldsymbol{\delta}^T\left(\frac{1}{2}\mathbf{1}_n+\frac{1}{2}\mathbf{U}\mathbf{K}\boldsymbol{\alpha}+\frac{\beta}{2}\mathbf{1}_n-\boldsymbol{\xi}\right)
\end{aligned}$$

where $\boldsymbol{\eta}\succeq\mathbf{0}_n,\boldsymbol{\gamma}\succeq\mathbf{0}_n$ and $\boldsymbol{\delta}\succeq\mathbf{0}_n$ are vectors of size u containing the Lagrange multipliers associated to the constraints of the primal problem. By taking the derivatives of $\mathcal{L}$ with respect to $\boldsymbol{\alpha},\boldsymbol{\xi},\beta$ and equating them to zero, we obtain the following relations:

$$\begin{aligned}
\tfrac{\partial\mathcal{L}}{\partial\boldsymbol{\alpha}}=\mathbf{0}_n\quad&\Rightarrow\boldsymbol{\alpha}=c_1\tilde{\mathbf{1}}-\mathbf{U}^T\boldsymbol{\gamma}-\tfrac{1}{2}\mathbf{U}^T\boldsymbol{\delta},\\
\tfrac{\partial\mathcal{L}}{\partial\beta}=0\quad&\Rightarrow c_1\tilde{\mathbf{1}}^T\mathbf{1}-\boldsymbol{\gamma}^T\mathbf{1}_n-\tfrac{1}{2}\boldsymbol{\delta}^T\mathbf{1}_n=0\\
&\Rightarrow\mathbf{1}^T\left(c_1\tilde{\mathbf{1}}-\mathbf{U}^T\boldsymbol{\gamma}-\tfrac{1}{2}\mathbf{U}^T\boldsymbol{\delta}\right)=0,\\
\tfrac{\partial\mathcal{L}}{\partial\boldsymbol{\xi}}=\mathbf{0}_n\quad&\Rightarrow c_2\mathbf{1}_n-\boldsymbol{\eta}-\boldsymbol{\gamma}-\boldsymbol{\delta}=\mathbf{0}_n\ \wedge\ \boldsymbol{\eta},\boldsymbol{\gamma},\boldsymbol{\delta}\succeq\mathbf{0}_n\\
&\Rightarrow\boldsymbol{\gamma}+\boldsymbol{\delta}\preceq c_2\mathbf{1}_n\ \wedge\ \mathbf{0}_n\preceq\boldsymbol{\gamma},\boldsymbol{\delta}\preceq c_2\mathbf{1}_n
\end{aligned}$$

which are then used to build the Lagrange dual function and consequently derive the following Lagrange dual problem:

$$\begin{aligned}
&\max_{\boldsymbol{\gamma},\boldsymbol{\delta}} \left\{ -\frac{1}{2}\left(\boldsymbol{\gamma}+\frac{1}{2}\boldsymbol{\delta}\right)^T \mathbf{U}\mathbf{K}\mathbf{U}^T\left(\boldsymbol{\gamma}+\frac{1}{2}\boldsymbol{\delta}\right) + c_1\tilde{\mathbf{1}}^T\mathbf{K}\mathbf{U}^T\left(\boldsymbol{\gamma}+\frac{1}{2}\boldsymbol{\delta}\right) + \frac{1}{2}\mathbf{1}_n^T\boldsymbol{\delta}\right\} \\
&\text{s.t. } \mathbf{1}^T\left[c_1\tilde{\mathbf{1}} - \mathbf{U}^T\left(\boldsymbol{\gamma}+\frac{1}{2}\boldsymbol{\delta}\right)\right] = 0, \\
&\qquad \boldsymbol{\gamma}+\boldsymbol{\delta} \preceq c_2\mathbf{1}_n, \\
&\qquad \mathbf{0}_n \preceq \boldsymbol{\gamma},\boldsymbol{\delta} \preceq c_2\mathbf{1}_n
\end{aligned}$$

(3.6) can be finally derived by defining $\boldsymbol{\sigma} = \boldsymbol{\gamma} + \frac{1}{2}\boldsymbol{\delta}$ and rewriting it as a minimization problem.

A.3 Proof of Lemma 1

Lemma 6. *Given $S = \{i, j\}$, any optimal solution $\boldsymbol{\sigma}_S^* = [\sigma_i^*\, \sigma_j^*]^T$, $\boldsymbol{\delta}_S^* = [\delta_i^*\, \delta_j^*]^T$ of the QP subproblem (3.8) has to satisfy the following condition: $\forall u : \boldsymbol{x}_u \in S \wedge 0 \leq \delta_u^* \leq c_2$ either $\sigma_u^* = c_2 - \frac{\delta_u^*}{2}$ or $\sigma_u^* = \frac{\delta_u^*}{2}$.*

Proof. By introducing the following notation, namely

$$\boldsymbol{\sigma}_S^k = \begin{bmatrix}\sigma_i^k \\ \sigma_j^k\end{bmatrix}, \boldsymbol{\delta}_S^k = \begin{bmatrix}\delta_i^k \\ \delta_j^k\end{bmatrix}, \boldsymbol{K}_{SS} = \begin{bmatrix} k(\boldsymbol{x}_i,\boldsymbol{x}_i) & k(\boldsymbol{x}_i,\boldsymbol{x}_j) \\ k(\boldsymbol{x}_j,\boldsymbol{x}_i) & k(\boldsymbol{x}_j,\boldsymbol{x}_j)\end{bmatrix}, \boldsymbol{e} = \begin{bmatrix}\boldsymbol{e}_1 \\ \boldsymbol{e}_2\end{bmatrix}$$

the objective function of the QP subproblem (3.8) evaluated at $(\boldsymbol{\sigma}_S^k, \boldsymbol{\delta}_S^k)$ can be rewritten as:

$$F(\sigma_i^k, \sigma_j^k) - \frac{\delta_i^k}{2} - \frac{\delta_j^k}{2} \tag{A.3}$$

where

$$F(\sigma_i^k, \sigma_j^k) = \frac{1}{2}\begin{bmatrix}\sigma_i^k & \sigma_j^k\end{bmatrix}\begin{bmatrix} k(\boldsymbol{x}_i,\boldsymbol{x}_i) & k(\boldsymbol{x}_i,\boldsymbol{x}_j) \\ k(\boldsymbol{x}_j,\boldsymbol{x}_i) & k(\boldsymbol{x}_j,\boldsymbol{x}_j)\end{bmatrix}\begin{bmatrix}\sigma_i^k \\ \sigma_j^k\end{bmatrix} + \begin{bmatrix}\boldsymbol{e}_1 & \boldsymbol{e}_2\end{bmatrix}\begin{bmatrix}\sigma_i^k \\ \sigma_j^k\end{bmatrix}$$

and the constraints of (3.8) at $(\boldsymbol{\sigma}_S^k, \boldsymbol{\delta}_S^k)$ are therefore:

$$\begin{aligned}
&\sigma_i^k + \sigma_j^k = a^k, \\
&\sigma_i^k + \frac{\delta_i^k}{2} \leq c_2 \ \wedge\ \sigma_j^k + \frac{\delta_j^k}{2} \leq c_2, \\
&\sigma_i^k - \frac{\delta_i^k}{2} \geq 0 \ \wedge\ \sigma_j^k - \frac{\delta_j^k}{2} \geq 0, \\
&0 \leq \delta_i^k, \delta_j^k \leq c_2
\end{aligned} \tag{A.4}$$

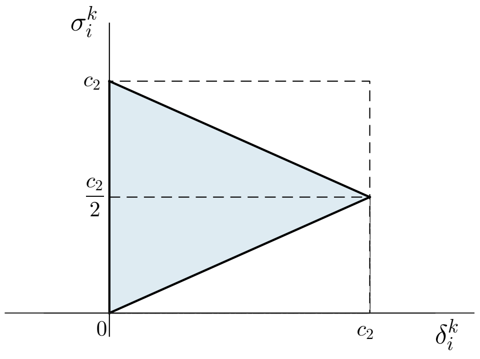


Figure A.1: Inequality constraints of the QP subproblem (3.8) for a given sample $\boldsymbol{x}_i$. The coloured area corresponds to the feasible region.

where $a^k = c_1 p - \mathbf{1}^T \boldsymbol{\sigma}_{\bar{S}}^k$ is a constant scalar for iteration k.

Since $(\boldsymbol{\sigma}_S^*, \boldsymbol{\delta}_S^*)$ is an optimal solution of (3.8), it has to satisfy the Karush-Kuhn-Tucker (KKT) conditions. In particular, the stationarity conditions can be expressed in the following way:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \sigma_i^k} &= \frac{\partial F(\sigma_i^*, \sigma_j^*)}{\sigma_i^k} + \beta + \lambda_i - \mu_i = 0,\\ \frac{\partial \mathcal{L}}{\partial \sigma_j^k} &= \frac{\partial F(\sigma_i^*, \sigma_j^*)}{\sigma_j^k} + \beta + \lambda_j - \mu_j = 0,\\ \frac{\partial \mathcal{L}}{\partial \delta_i^k} &= -\frac{1}{2} + \frac{\lambda_i}{2} + \frac{\mu_i}{2} + \xi_i - \eta_i = 0,\\ \frac{\partial \mathcal{L}}{\partial \delta_j^k} &= -\frac{1}{2} + \frac{\lambda_j}{2} + \frac{\mu_j}{2} + \xi_j - \eta_j = 0 \end{aligned} \tag{A.5}$$

where $\mathcal{L}$ is the Lagrange dual function obtained from the QP subproblem and β,λ_i,λ_j,μ_i,μ_j,ξ_i, ξ_j,η_i,η_j are its Lagrange multipliers, namely:

$$\begin{aligned}\mathcal{L} =& F(\sigma_i^*, \sigma_j^*) - \frac{\delta_i^*}{2} - \frac{\delta_j^*}{2} + \beta(\sigma_i^* + \sigma_j^* - a^k)\\ &+ \lambda_i(\sigma_i^* + \frac{\delta_i^*}{2} - c_2) + \lambda_j(\sigma_j^* + \frac{\delta_j^*}{2} - c_2)\\ &- \mu_i(\sigma_i^* - \frac{\delta_i^*}{2}) - \mu_j(\sigma_j^* - \frac{\delta_j^*}{2})\\ &+ \xi_i(\delta_i^* - c_2) + \xi_j(\delta_j^* - c_2) - \eta_i\delta_i^* - \eta_j\delta_j^*\end{aligned}$$

Now, focus on terms associated with sample $\boldsymbol{x}_i$ and specifically on its inequality constraints in (A.4). Based on them, it is possible to distinguish the following four cases

(Figure A.1 helps to understand this):

$$0 \leq \delta_i^k < c_2 \ \wedge \ \frac{\delta_i^k}{2} < \sigma_i^k < c_2 - \frac{\delta_i^k}{2}, \tag{A.6}$$

$$0 \leq \delta_i^k < c_2 \ \wedge \ \sigma_i^k = c_2 - \frac{\delta_i^k}{2}, \tag{A.7}$$

$$0 \leq \delta_i^k < c_2 \ \wedge \ \sigma_i^k = \frac{\delta_i^k}{2}, \tag{A.8}$$

$$\delta_i^k = c_2 \ \wedge \ \sigma_i^k = \frac{\delta_i^k}{2} \tag{A.9}$$

By considering the KKT complementary slackness conditions together with (A.5), we can derive the following statements:

$$\begin{aligned}
\text{Case (A.6)} &\Rightarrow \lambda_i = 0, \mu_i = 0, \xi_i = 0, \eta_i \geq 0 \\
&\Rightarrow \frac{\partial F(\sigma_i^*, \sigma_j^*)}{\sigma_i^k} + \beta = 0 \ \wedge \ \eta_i = -\frac{1}{2}, \\
\text{Case (A.7)} &\Rightarrow \lambda_i \geq 0, \mu_i = 0, \xi_i = 0, \eta_i \geq 0 \\
&\Rightarrow \frac{\partial F(\sigma_i^*, \sigma_j^*)}{\sigma_i^k} + \beta \leq -1 \ \wedge \ \lambda_i \geq 1 \ \wedge \ \eta_i \geq 0, \\
\text{Case (A.8)} &\Rightarrow \lambda_i = 0, \mu_i \geq 0, \xi_i = 0, \eta_i \geq 0 \\
&\Rightarrow \frac{\partial F(\sigma_i^*, \sigma_j^*)}{\sigma_i^k} + \beta \geq 1 \ \wedge \ \mu_i \geq 1 \ \wedge \ \eta_i \geq 0, \\
\text{Case (A.9)} &\Rightarrow \lambda_i \geq 0, \mu_i \geq 0, \xi_i \geq 0, \eta_i = 0, \\
&\Rightarrow -1 \leq \frac{\partial F(\sigma_i^*, \sigma_j^*)}{\sigma_i^k} + \beta \leq 1 \ \wedge \ -1 \leq \lambda_i, \mu_i \leq 1
\end{aligned} \tag{A.10}$$

The first statement in (A.10) is clearly a contradiction, implying that condition (A.6) is not valid for KKT. In other words, any optimal solution (σ_i^*, δ_i^*) does not satisfy condition (A.6), but only conditions (A.7)-(A.9). This fact is valid $\forall \boldsymbol{x}_u \in S$ due to the symmetry of the QP subproblem (3.8), which concludes the proof. □

Appendix B

Supplementary Material for Chapter 4

B.1 Proof of Lemma 5

Lemma 7. *(**Lemma 5 restated**) Given $k : \Omega_{\mathbf{z}} \times \Omega_{\mathbf{z}} \rightarrow \mathbb{R}$ a symmetric positive definite kernel, then:*

(a) there exists a unique Hilbert space $\mathcal{H}$ of real-valued functions over $\Omega_{\mathbf{z}}$, for which k is a reproducing kernel. $\mathcal{H}$ is therefore a Reproducing kernel Hilbert Space (RKHS).

(b) For all $h \in \mathcal{H}$

$$\int_{\Omega_{\mathbf{z}}} \int_{\Omega_{\mathbf{z}}} \phi(\mathbf{z})\phi(\mathbf{z}')k(\mathbf{z},\mathbf{z}')d\mathbf{z}d\mathbf{z}' = MMD^2(p_{\mathbf{Z}}, q_{\mathbf{Z}})$$

where

$$MMD(p_{\mathbf{Z}}, q_{\mathbf{Z}}) \doteq \sup_{\|h\|_{\mathcal{H}} \leq 1} \Big\{ E_{\mathbf{z}\sim p_{\mathbf{Z}}}[h(\mathbf{z})] - E_{\mathbf{z}\sim q_{\mathbf{Z}}}[h(\mathbf{z})] \Big\}$$

is the maximum mean discrepancy between $p_{\mathbf{Z}}(\mathbf{z})$ and $q_{\mathbf{Z}}(\mathbf{z})$.

(c) Let $\mathcal{H}$ be defined as in (b), then $MMD(p_{\mathbf{Z}}, q_{\mathbf{Z}}) = 0$ if and only if $p_{\mathbf{Z}}(\mathbf{z}) = q_{\mathbf{Z}}(\mathbf{z})$.

Proof. (a) follows directly from the Moore-Aronszajn theorem [89].

Now we prove statement (b). For the sake of notation compactness, define $J \doteq \int_{\Omega_{\mathbf{z}}} \int_{\Omega_{\mathbf{z}}} \phi(\mathbf{z})\phi(\mathbf{z}')k(\mathbf{z},\mathbf{z}')d\mathbf{z}d\mathbf{z}'$. Therefore,

$$\begin{aligned}
J = &\int_{\Omega_{\mathbf{z}}} \int_{\Omega_{\mathbf{z}}} p_{\mathbf{Z}}(\mathbf{z})p_{\mathbf{Z}}(\mathbf{z}')k(\mathbf{z},\mathbf{z}')d\mathbf{z}d\mathbf{z}' - \int_{\Omega_{\mathbf{z}}} \int_{\Omega_{\mathbf{z}}} p_{\mathbf{Z}}(\mathbf{z})q(\mathbf{z}')k(\mathbf{z},\mathbf{z}')d\mathbf{z}d\mathbf{z}' \\
&- \int_{\Omega_{\mathbf{z}}} \int_{\Omega_{\mathbf{z}}} p_{\mathbf{Z}}(\mathbf{z}')q_{\mathbf{Z}}(\mathbf{z})k(\mathbf{z},\mathbf{z}')d\mathbf{z}d\mathbf{z}' + \int_{\Omega_{\mathbf{z}}} \int_{\Omega_{\mathbf{z}}} q(\mathbf{z}')q_{\mathbf{Z}}(\mathbf{z})k(\mathbf{z},\mathbf{z}')d\mathbf{z}d\mathbf{z}' \\
= &\int_{\Omega_{\mathbf{z}}} \int_{\Omega_{\mathbf{z}}} p_{\mathbf{Z}}(\mathbf{z})p_{\mathbf{Z}}(\mathbf{z}')\langle r(\mathbf{z}), r(\mathbf{z}')\rangle_{\mathcal{H}} d\mathbf{z}d\mathbf{z}' - \int_{\Omega_{\mathbf{z}}} \int_{\Omega_{\mathbf{z}}} p_{\mathbf{Z}}(\mathbf{z})q(\mathbf{z}')\langle r(\mathbf{z}), r(\mathbf{z}')\rangle_{\mathcal{H}} d\mathbf{z}d\mathbf{z}'
\end{aligned}$$

$$
\begin{aligned}
&- \int_{\Omega_{\mathbf{z}}}\int_{\Omega_{\mathbf{z}}} p_{\mathbf{Z}}(\mathbf{z}')q_{\mathbf{Z}}(\mathbf{z})\langle r(\mathbf{z}), r(\mathbf{z}')\rangle_{\mathcal{H}} d\mathbf{z}d\mathbf{z}' + \int_{\Omega_{\mathbf{z}}}\int_{\Omega_{\mathbf{z}}} q(\mathbf{z}')q_{\mathbf{Z}}(\mathbf{z})\langle r(\mathbf{z}), r(\mathbf{z}')\rangle_{\mathcal{H}} d\mathbf{z}d\mathbf{z}' \\
&= \langle E_{\mathbf{z}\sim p_{\mathbf{Z}}}[r(\mathbf{z})], E_{\mathbf{z}'\sim p_{\mathbf{Z}}}[r(\mathbf{z}')]\rangle_{\mathcal{H}} - \langle E_{\mathbf{z}\sim p_{\mathbf{Z}}}[r(\mathbf{z})], E_{\mathbf{z}'\sim q}[r(\mathbf{z}')]\rangle_{\mathcal{H}} \\
&\quad - \langle E_{\mathbf{z}'\sim p_{\mathbf{Z}}}[r(\mathbf{z}')], E_{\mathbf{z}\sim q}[r(\mathbf{z})]\rangle_{\mathcal{H}} + \langle E_{\mathbf{z}'\sim q}[r(\mathbf{z}')], E_{\mathbf{z}\sim q}[r(\mathbf{z})]\rangle_{\mathcal{H}}
\end{aligned} \tag{B.1}
$$

Note that the second equality in (B.1) follows from the fact that $k(\mathbf{z},\mathbf{z}') = \langle r(\mathbf{z}), r(\mathbf{z}')\rangle_{\mathcal{H}}$ for a unique $r \in \mathcal{H}$,[1] where $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ is the inner product of $\mathcal{H}$. If we define $\mu_{p_{\mathbf{Z}}} \doteq E_{\mathbf{z}\sim p_{\mathbf{Z}}}[r(\mathbf{z})]$ and $\mu_q \doteq E_{\mathbf{z}\sim q}[r(\mathbf{z})]$,[2] then (B.1) can be rewritten in the following way:

$$
\begin{aligned}
J &= \langle \mu_{p_{\mathbf{Z}}},\mu_{p_{\mathbf{Z}}}\rangle_{\mathcal{H}} - \langle \mu_{p_{\mathbf{Z}}},\mu_q\rangle_{\mathcal{H}} - \langle \mu_{p_{\mathbf{Z}}},\mu_q\rangle_{\mathcal{H}} + \langle \mu_q,\mu_q\rangle_{\mathcal{H}} \\
&= \langle \mu_{p_{\mathbf{Z}}} - \mu_q,\mu_{p_{\mathbf{Z}}} - \mu_q\rangle_{\mathcal{H}} \\
&= \|\mu_{p_{\mathbf{Z}}} - \mu_q\|^2_{\mathcal{H}}
\end{aligned} \tag{B.2}
$$

Notice that

$$
\begin{aligned}
\|\mu_{p_{\mathbf{Z}}} - \mu_q\|_{\mathcal{H}} &= \left\langle \mu_{p_{\mathbf{Z}}} - \mu_q, \frac{\mu_{p_{\mathbf{Z}}} - \mu_q}{\|\mu_{p_{\mathbf{Z}}} - \mu_q\|_{\mathcal{H}}}\right\rangle_{\mathcal{H}} \\
&= \sup_{\|h\|_{\mathcal{H}}\leq 1} \left\{\langle \mu_{p_{\mathbf{Z}}} - \mu_q, h\rangle_{\mathcal{H}}\right\} \\
&= \sup_{\|h\|_{\mathcal{H}}\leq 1} \left\{E_{\mathbf{z}\sim p_{\mathbf{Z}}}[\langle r(\mathbf{z}), h\rangle_{\mathcal{H}}] - E_{\mathbf{z}\sim q}[\langle r(\mathbf{z}), h\rangle_{\mathcal{H}}]\right\} \\
&= \sup_{\|h\|_{\mathcal{H}}\leq 1} \left\{E_{\mathbf{z}\sim p_{\mathbf{Z}}}[h(\mathbf{z})] - E_{\mathbf{z}\sim q}[h(\mathbf{z})]\right\} \\
&= \mathrm{MMD}(p_{\mathbf{Z}}, q_{\mathbf{Z}})
\end{aligned}
$$

Substituting this result into (B.2) concludes the proof of the statement.

Statement (c) is equivalent to Theorem 3 in [113]. □

B.2 Proof of Theorem 1

Theorem 2. ***(Theorem 1 restated)*** *Given the objective in (4.2), $\Omega_{\mathbf{z}}$ a compact set, $\Omega_{\mathbf{x}} = [-M, M]^d$, where $M \in \mathbb{R}^+$, and a symmetric, continuous and positive definite kernel $k : \Omega_{\mathbf{z}} \times \Omega_{\mathbf{z}} \to \mathbb{R}$, where $0 \leq k(\mathbf{z},\mathbf{z}') \leq K$ for all $\mathbf{z},\mathbf{z}' \in \Omega_{\mathbf{z}}$ and $K = k(\mathbf{z},\mathbf{z})$.*

(a) There exists $\widetilde{k} : \Omega_{\mathbf{z}} \times \Omega_{\mathbf{z}} \to \mathbb{R}$ such that $k(\mathbf{z},\mathbf{z}') = \int_{\Omega_{\mathbf{z}}} \widetilde{k}(\mathbf{z},\mathbf{u})\widetilde{k}(\mathbf{z}',\mathbf{u})d\mathbf{u}$.

(b) The objective in (4.2) is equivalent to

$$
\widetilde{\mathcal{L}}(f, g) = \frac{1}{N}\sum_{\mathbf{x}_i\in\mathcal{D}_{\mathbf{x}}} \|\mathbf{x}_i - g(f(\mathbf{x}_i))\|^2 +
$$

[1] This is a classical result due to the Riesz representation theorem.

[2] Their existence can be guaranteed assuming that $\|\mu_{p_{\mathbf{Z}}}\|^2_{\mathcal{H}} < \infty$ and $\|\mu_q\|^2_{\mathcal{H}} < \infty$. In other words, $E_{\mathbf{z},\mathbf{z}'\sim p_{\mathbf{Z}}}[k(\mathbf{z},\mathbf{z}')] < \infty$ and $E_{\mathbf{z},\mathbf{z}'\sim q}[k(\mathbf{z},\mathbf{z}')] < \infty$.

$$+\lambda\left\{\frac{N}{N-1}\int_{\Omega_{\mathbf{z}}}\left(p_{\mathbf{Z},\widetilde{k}}(\mathbf{u})-q_{\mathbf{Z},\widetilde{k}}(\mathbf{u})\right)^2 d\mathbf{u}-\frac{2K}{N-1}+\frac{2}{N(N-1)}\sum_{\substack{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}\\ \mathbf{z}'_i\in\mathcal{D}^f_{\mathbf{z}}}}k(\mathbf{z}_i,\mathbf{z}'_i)\right\}$$

where $p_{\mathbf{Z},\widetilde{k}}(\mathbf{u})=\frac{1}{N}\sum_{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}}\widetilde{k}(\mathbf{z}_i,\mathbf{u})$ *and* $q_{\mathbf{Z},\widetilde{k}}(\mathbf{u})=\frac{1}{N}\sum_{\mathbf{z}'_i\in\mathcal{D}^f_{\mathbf{z}}}\widetilde{k}(\mathbf{z}'_i,\mathbf{u})$ *are the kernel density approximations of* $p_{\mathbf{Z}}(\mathbf{u})$ *and* $q(\mathbf{u})$*, respectively.*

(c) Then, for any $s,t>0$

$$Pr\left\{|\widehat{\mathcal{L}}-\mathcal{L}|>t+\lambda s\right\}\leq 2\exp\left\{-\frac{Nt^2}{8M^4d^2}\right\}+2\exp\left\{-\frac{\lfloor N/2\rfloor s^2}{8K^2}\right\}$$

Proof. Property (a) can be proved using the Mercer's theorem (see Theorem 1 in [126]). Now we prove property (b):

$$\begin{aligned}
\widehat{\mathcal{L}}(f,g)&=\frac{1}{N}\sum_{\mathbf{x}_i\in\mathcal{D}_{\mathbf{x}}}\|\mathbf{x}_i-g(f(\mathbf{x}_i))\|^2++\lambda\left\{\frac{1}{N(N-1)}\sum_{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}}\sum_{\substack{\mathbf{z}_j\in\mathcal{D}_{\mathbf{z}}\\ j\neq i}}k(\mathbf{z}_i,\mathbf{z}_j)\right.\\
&\quad\left.-\frac{2}{N(N-1)}\sum_{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}}\sum_{\substack{\mathbf{z}_j\in\mathcal{D}^f_{\mathbf{z}}\\ j\neq i}}k(\mathbf{z}_i,\mathbf{z}_j)+\frac{1}{N(N-1)}\sum_{\mathbf{z}_i\in\mathcal{D}^f_{\mathbf{z}}}\sum_{\substack{\mathbf{z}_j\in\mathcal{D}^f_{\mathbf{z}}\\ j\neq i}}k(\mathbf{z}_i,\mathbf{z}_j)\right\}\\
&=\frac{1}{N}\sum_{\mathbf{x}_i\in\mathcal{D}_{\mathbf{x}}}\|\mathbf{x}_i-g(f(\mathbf{x}_i))\|^2+\lambda\frac{N}{N-1}\left\{\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}}\sum_{\substack{\mathbf{z}_j\in\mathcal{D}_{\mathbf{z}}\\ j\neq i}}k(\mathbf{z}_i,\mathbf{z}_j)\right.\\
&\quad\left.-\frac{2}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}}\sum_{\substack{\mathbf{z}_j\in\mathcal{D}^f_{\mathbf{z}}\\ j\neq i}}k(\mathbf{z}_i,\mathbf{z}_j)+\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}^f_{\mathbf{z}}}\sum_{\substack{\mathbf{z}_j\in\mathcal{D}^f_{\mathbf{z}}\\ j\neq i}}k(\mathbf{z}_i,\mathbf{z}_j)\right\}\\
&=\frac{1}{N}\sum_{\mathbf{x}_i\in\mathcal{D}_{\mathbf{x}}}\|\mathbf{x}_i-g(f(\mathbf{x}_i))\|^2+\lambda\frac{N}{N-1}\left\{\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}}\sum_{\mathbf{z}_j\in\mathcal{D}_{\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}_j)\right.\\
&\quad-\frac{2}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}}\sum_{\mathbf{z}_j\in\mathcal{D}^f_{\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}_j)+\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}^f_{\mathbf{z}}}\sum_{\mathbf{z}_j\in\mathcal{D}^f_{\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}_j)\\
&\quad\left.-\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}_i)+\frac{2}{N^2}\sum_{\substack{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}\\ \mathbf{z}'_i\in\mathcal{D}^f_{\mathbf{z}}}}k(\mathbf{z}_i,\mathbf{z}'_i)-\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}^f_{\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}_i)\right\}\\
&=\frac{1}{N}\sum_{\mathbf{x}_i\in\mathcal{D}_{\mathbf{x}}}\|\mathbf{x}_i-g(f(\mathbf{x}_i))\|^2+\lambda\frac{N}{N-1}\left\{\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}}\sum_{\mathbf{z}_j\in\mathcal{D}_{\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}_j)\right.\\
&\quad-\frac{2}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_{\mathbf{z}}}\sum_{\mathbf{z}_j\in\mathcal{D}^f_{\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}_j)+\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}^f_{\mathbf{z}}}\sum_{\mathbf{z}_j\in\mathcal{D}^f_{\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}_j)
\end{aligned}$$

$$-\frac{2K}{N}+\frac{2}{N^2}\sum_{\substack{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}\\ \mathbf{z}'_i\in\mathcal{D}^f_\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}'_i)\Bigg\}$$

$$=\frac{1}{N}\sum_{\mathbf{x}_i\in\mathcal{D}_\mathbf{x}}\|\mathbf{x}_i-g(f(\mathbf{x}_i))\|^2+\lambda\frac{N}{N-1}\Bigg\{\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}}\sum_{\mathbf{z}_j\in\mathcal{D}_\mathbf{z}}\int_{\Omega_\mathbf{z}}\widetilde{k}(\mathbf{z}_i,\mathbf{u})\widetilde{k}(\mathbf{z}_j,\mathbf{u})d\mathbf{u}$$
$$-\frac{2}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}}\sum_{\mathbf{z}_j\in\mathcal{D}^f_\mathbf{z}}\int_{\Omega_\mathbf{z}}\widetilde{k}(\mathbf{z}_i,\mathbf{u})\widetilde{k}(\mathbf{z}_j,\mathbf{u})d\mathbf{u}+\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}^f_\mathbf{z}}\sum_{\mathbf{z}_j\in\mathcal{D}^f_\mathbf{z}}\int_{\Omega_\mathbf{z}}\widetilde{k}(\mathbf{z}_i,\mathbf{u})\widetilde{k}(\mathbf{z}_j,\mathbf{u})d\mathbf{u}$$
$$-\frac{2K}{N}+\frac{2}{N^2}\sum_{\substack{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}\\ \mathbf{z}'_i\in\mathcal{D}^f_\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}'_i)\Bigg\}$$

$$=\frac{1}{N}\sum_{\mathbf{x}_i\in\mathcal{D}_\mathbf{x}}\|\mathbf{x}_i-g(f(\mathbf{x}_i))\|^2+\lambda\frac{N}{N-1}\Bigg\{\int_{\Omega_\mathbf{z}}\Bigg\{\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}}\sum_{\mathbf{z}_j\in\mathcal{D}_\mathbf{z}}\widetilde{k}(\mathbf{z}_i,\mathbf{u})\widetilde{k}(\mathbf{z}_j,\mathbf{u})$$
$$-\frac{2}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}}\sum_{\mathbf{z}_j\in\mathcal{D}^f_\mathbf{z}}\widetilde{k}(\mathbf{z}_i,\mathbf{u})\widetilde{k}(\mathbf{z}_j,\mathbf{u})+\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}^f_\mathbf{z}}\sum_{\mathbf{z}_j\in\mathcal{D}^f_\mathbf{z}}\widetilde{k}(\mathbf{z}_i,\mathbf{u})\widetilde{k}(\mathbf{z}_j,\mathbf{u})\Bigg\}d\mathbf{u}$$
$$-\frac{2K}{N}+\frac{2}{N^2}\sum_{\substack{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}\\ \mathbf{z}'_i\in\mathcal{D}^f_\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}'_i)\Bigg\}$$

$$=\frac{1}{N}\sum_{\mathbf{x}_i\in\mathcal{D}_\mathbf{x}}\|\mathbf{x}_i-g(f(\mathbf{x}_i))\|^2+\lambda\frac{N}{N-1}\Bigg\{\int_{\Omega_\mathbf{z}}\Bigg\{\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}}\widetilde{k}(\mathbf{z}_i,\mathbf{u})\sum_{\mathbf{z}_j\in\mathcal{D}_\mathbf{z}}\widetilde{k}(\mathbf{z}_j,\mathbf{u})$$
$$-\frac{2}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}}\widetilde{k}(\mathbf{z}_i,\mathbf{u})\sum_{\mathbf{z}_j\in\mathcal{D}^f_\mathbf{z}}\widetilde{k}(\mathbf{z}_j,\mathbf{u})+\frac{1}{N^2}\sum_{\mathbf{z}_i\in\mathcal{D}^f_\mathbf{z}}\widetilde{k}(\mathbf{z}_i,\mathbf{u})\sum_{\mathbf{z}_j\in\mathcal{D}^f_\mathbf{z}}\widetilde{k}(\mathbf{z}_j,\mathbf{u})\Bigg\}d\mathbf{u}$$
$$-\frac{2K}{N}+\frac{2}{N^2}\sum_{\substack{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}\\ \mathbf{z}'_i\in\mathcal{D}^f_\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}'_i)\Bigg\}$$

$$=\frac{1}{N}\sum_{\mathbf{x}_i\in\mathcal{D}_\mathbf{x}}\|\mathbf{x}_i-g(f(\mathbf{x}_i))\|^2+\lambda\frac{N}{N-1}\Bigg\{\int_{\Omega_\mathbf{z}}\Bigg\{\frac{1}{N}\sum_{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}}\widetilde{k}(\mathbf{z}_i,\mathbf{u})$$
$$-\frac{1}{N}\sum_{\mathbf{z}_j\in\mathcal{D}^f_\mathbf{z}}\widetilde{k}(\mathbf{z}_j,\mathbf{u})\Bigg\}^2d\mathbf{u}-\frac{2K}{N}+\frac{2}{N^2}\sum_{\substack{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}\\ \mathbf{z}'_i\in\mathcal{D}^f_\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}'_i)\Bigg\}$$

$$=\frac{1}{N}\sum_{\mathbf{x}_i\in\mathcal{D}_\mathbf{x}}\|\mathbf{x}_i-g(f(\mathbf{x}_i))\|^2+\lambda\Bigg\{\frac{N}{N-1}\int_{\Omega_\mathbf{z}}\Bigg\{\frac{1}{N}\sum_{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}}\widetilde{k}(\mathbf{z}_i,\mathbf{u})$$
$$-\frac{1}{N}\sum_{\mathbf{z}_j\in\mathcal{D}^f_\mathbf{z}}\widetilde{k}(\mathbf{z}_j,\mathbf{u})\Bigg\}^2d\mathbf{u}-\frac{2K}{N-1}+\frac{2}{N(N-1)}\sum_{\substack{\mathbf{z}_i\in\mathcal{D}_\mathbf{z}\\ \mathbf{z}'_i\in\mathcal{D}^f_\mathbf{z}}}k(\mathbf{z}_i,\mathbf{z}'_i)\Bigg\}$$

$$= \widetilde{\mathcal{L}}(f, g)$$

To prove property (c), we first derive the statistical bounds for the two addends in (4.2), and then combine these results in the final bound.

Consider the reconstruction error term in (4.2) and define $\xi_{\mathbf{x}} \doteq \|\mathbf{x} - g(f(\mathbf{x}))\|^2$. Note that $\Omega_{\mathbf{x}} = [-M, M]^d$ and therefore $\xi_{\mathbf{x}}$ is bounded in the interval $[0, 4M^2 d]$. By considering $\xi_{\mathbf{x}}$ a random variable, we can apply the Hoeffding's inequality [see Theorem 2 in [131]] to obtain the following statistical bound:

$$\begin{aligned} \Pr\left\{\left|\frac{1}{N}\sum_{\mathbf{x}\in\mathcal{D}_{\mathbf{x}}}\xi_{\mathbf{x}} - \int_{\Omega_{\mathbf{x}}}\xi_{\mathbf{x}} p_{\mathbf{X}}(\mathbf{x})d\mathbf{x}\right| \geq t\right\} &\leq 2\exp\left\{\frac{-2N^2t^2}{(4M^2d)^2N}\right\} \\ &= 2\exp\left\{-\frac{Nt^2}{8M^4d^2}\right\} \end{aligned} \tag{B.3}$$

where t is an arbitrary small positive constant.

Note that the second addend in (4.2) is equivalent to the empirical unbiased estimate of $\mathrm{MMD}^2(p_{\mathbf{Z}}, q_{\mathbf{Z}})$ [113]. We refer to it as $\widehat{\mathrm{MMD}}^2(p_{\mathbf{Z}}, q_{\mathbf{Z}})$. Based on this observation, we can reuse the statistical bound provided in Theorem 14 of [113], which is valid for any bounded positive definite kernel, namely:

$$\Pr\left\{\left|\widehat{\mathrm{MMD}}^2(p_{\mathbf{Z}}, q_{\mathbf{Z}}) - \mathrm{MMD}^2(p_{\mathbf{Z}}, q_{\mathbf{Z}})\right| > s\right\} \leq 2\exp\left\{-\frac{\lfloor N/2\rfloor s^2}{8K^2}\right\} \tag{B.4}$$

By combining (B.3) and (B.4), we obtain the following aggregated bound:

$$\begin{aligned} &\Pr\left\{\left|\frac{1}{N}\sum_{\mathbf{x}\in\mathcal{D}_{\mathbf{x}}}\xi_{\mathbf{x}} - \int_{\Omega_{\mathbf{x}}}\xi_{\mathbf{x}} p_{\mathbf{X}}(\mathbf{x})d\mathbf{x}\right| \geq t\right\} + \\ &+\Pr\left\{\left|\widehat{\mathrm{MMD}}^2(p_{\mathbf{Z}}, q_{\mathbf{Z}}) - \mathrm{MMD}^2(p_{\mathbf{Z}}, q_{\mathbf{Z}})\right| > s\right\} \leq \\ &\leq 2\exp\left\{-\frac{Nt^2}{8M^4d^2}\right\} + 2\exp\left\{-\frac{\lfloor N/2\rfloor s^2}{8K^2}\right\} \end{aligned} \tag{B.5}$$

Note that the sum of the two probalities in (B.5) can be lower-bounded in the following way:

$$\begin{aligned} &\Pr\left\{\left|\frac{1}{N}\sum_{\mathbf{x}\in\mathcal{D}_{\mathbf{x}}}\xi_{\mathbf{x}} - \int_{\Omega_{\mathbf{x}}}\xi_{\mathbf{x}} p_{\mathbf{X}}(\mathbf{x})d\mathbf{x}\right| \geq t\right\} + \\ &+\Pr\left\{\left|\widehat{\mathrm{MMD}}^2(p_{\mathbf{Z}}, q_{\mathbf{Z}}) - \mathrm{MMD}^2(p_{\mathbf{Z}}, q_{\mathbf{Z}})\right| > s\right\} \geq \\ &\geq \Pr\left\{\left|\frac{1}{N}\sum_{\mathbf{x}\in\mathcal{D}_{\mathbf{x}}}\xi_{\mathbf{x}} - \int_{\Omega_{\mathbf{x}}}\xi_{\mathbf{x}} p_{\mathbf{X}}(\mathbf{x})d\mathbf{x}\right| \geq t \cup \right. \end{aligned}$$

$$\begin{aligned}
&\cup\, \lambda\left|\widehat{\mathrm{MMD}}^2(p_{\mathbf{Z}},q_{\mathbf{Z}}) - \mathrm{MMD}^2(p_{\mathbf{Z}},q_{\mathbf{Z}})\right| > \lambda s\Big]\Big\} \geq \\
\geq \Pr\Big\{&\left|\frac{1}{N}\sum_{\mathbf{x}\in\mathcal{D}_{\mathbf{x}}}\xi_{\mathbf{x}} - \int_{\Omega_{\mathbf{x}}}\xi_{\mathbf{x}}p_{\mathbf{X}}(\mathbf{x})d\mathbf{x}\right| + \\
&+ \lambda\left|\widehat{\mathrm{MMD}}^2(p_{\mathbf{Z}},q_{\mathbf{Z}}) - \mathrm{MMD}^2(p_{\mathbf{Z}},q_{\mathbf{Z}})\right| > \lambda s + t\Big\} \geq \\
\geq \Pr\Big\{&\Big|\frac{1}{N}\sum_{\mathbf{x}\in\mathcal{D}_{\mathbf{x}}}\xi_{\mathbf{x}} - \int_{\Omega_{\mathbf{x}}}\xi_{\mathbf{x}}p_{\mathbf{X}}(\mathbf{x})d\mathbf{x} + \\
&+ \lambda\widehat{\mathrm{MMD}}^2(p_{\mathbf{Z}},q_{\mathbf{Z}}) - \lambda\mathrm{MMD}^2(p_{\mathbf{Z}},q_{\mathbf{Z}})\Big| > \lambda s + t\Big\} = \\
= \Pr\Big\{&|\widehat{\mathcal{L}} - \mathcal{L}| > \lambda s + t\Big\} \qquad \text{(B.6)}
\end{aligned}$$

where the first inequality in (B.6) is obtained by applying the union bound and by introducing λ; the second inequality is obtained by replacing the union operator with the addition; the third inequality is obtained by exploiting the triangle inequality; the last equality follows from Lemma 5.

By combining (B.6) with (B.5), we get statement (c). □

B.3 Experiments on Grid Dataset

Table B.1 provides the details of the network architectures. Hyperbolic tangent activations are used as output activations of encoders, while the outputs of decoders/generators are linear. For all methods, we use ReLU hidden activations. CouGAN uses ELU [124], but we provide also results with ReLU. Furthermore, we show results of VEEGAN with and without batch normalization (BN). See Figure B.1.

B.4 Experiments on Low Dimensional Embedding Dataset

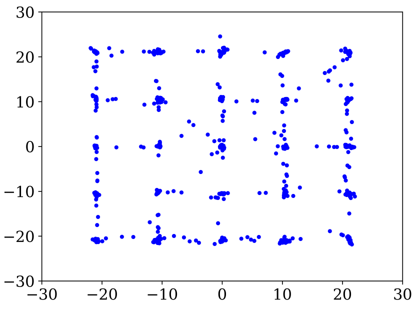
Table B.2 provides the details of the network architectures. Hyperbolic tangent activations are used as output activations of encoders, while the outputs of decoders/generators are linear. For all methods, we use ReLU hidden activations. CouGAN uses ELU [124], but we provide also results with ReLU. Furthermore, we report results of BiGAN, VEEGAN and AAE with and without batch normalization (BN). See Table B.3.

Table B.1: Network architectures.

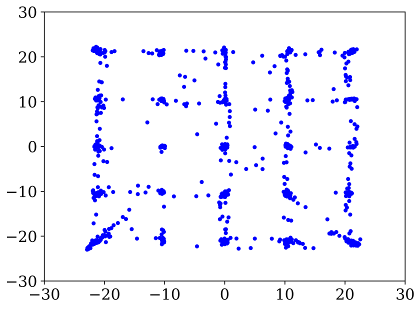
Method	Network	Neurons per layer
PAE	Encoder	2,128,128,2
	Decoder	2,128,128,2
	Discriminator	-
CouGAN	Encoder	-
	Decoder	2,128,128,2
	Discriminator	2,128,128,1
WGAN	Encoder	-
	Decoder	2,128,128,2
	Discriminator	2,128,128,1
BiGAN	Encoder	2,128,128,2
	Decoder	2,128,128,2
	Discriminator	4,128,128,1
VEEGAN	Encoder	2,128,128,254
	Decoder	254+1,128,128,2
	Discriminator	256,128,128,1
VAE	Encoder	2,128,128,4
	Decoder	2,128,128,2
	Discriminator	-
AAE	Encoder	2,128,128,2
	Decoder	2,128,128,2
	Discriminator	2,128,128,1
AVB-AC	Encoder	2+64,128,128,2
	Decoder	2,128,128,2
	Discriminator	4,128,128,1

B.5 Experiments on Stacked-MNIST

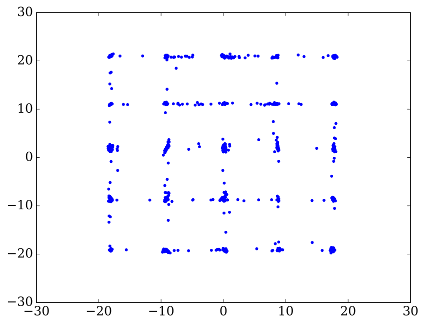
Table B.4 provides the details of the network architectures. Hyperbolic tangent activations are used as output activations of encoders, while the outputs of decoders/generators are sigmoid (standard logistic activations). For all methods, we use ReLU hidden activations. CouGAN uses ELU [124], but we provide also results with ReLU. Furthermore, we report results of BiGAN, VEEGAN and AAE with and without batch normalization (BN). See Figure B.2.



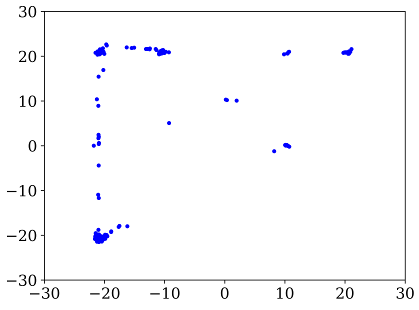
(a) CouGAN (ReLU)



(b) CouGAN (ELU)

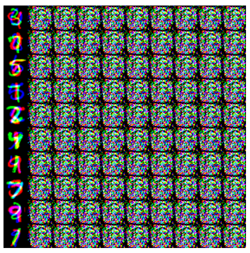


(c) VEEGAN w BN

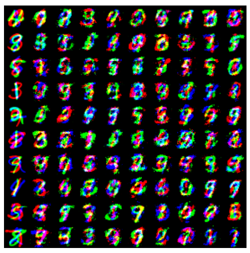


(d) VEEGAN w/o BN

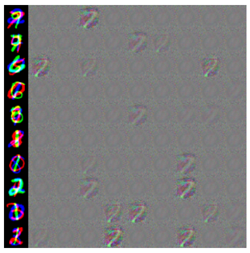
Figure B.1: Grid dataset.



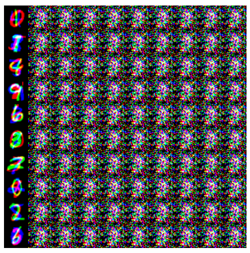
(a) CouGAN (ReLU)



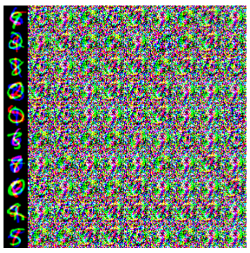
(b) CouGAN (ELU)



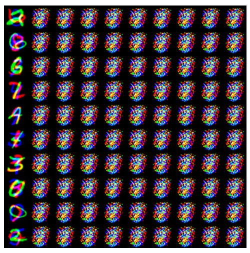
(c) BiGAN w BN



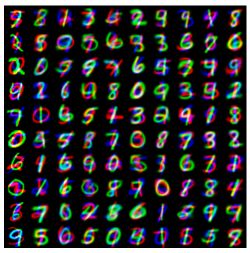
(d) BiGAN w/o BN



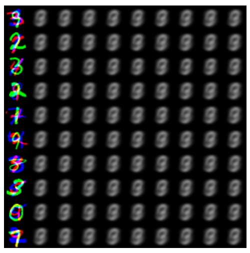
(e) VEEGAN w BN



(f) VEEGAN w/o BN



(g) AAE w BN



(h) AAE w/o BN

Figure B.2: Stacked-MNIST dataset.

Table B.2: Network architectures.

Method	Network	Neurons per layer
PAE	Encoder	1000,128,128,10
	Decoder	10,128,128,1000
	Discriminator	-
CouGAN	Encoder	-
	Decoder	10,128,128,1000
	Discriminator	1000,128,128,1
WGAN	Encoder	-
	Decoder	10,128,128,1000
	Discriminator	1000,128,128,1
BiGAN	Encoder	1000,128,128,10
	Decoder	1000,128,128,10
	Discriminator	1010,128,128,1
VEEGAN	Encoder	1000,128,128,10
	Decoder	10+1,128,128,1000
	Discriminator	1010,128,128,1
VAE	Encoder	1000,128,128,20
	Decoder	10,128,128,1000
	Discriminator	-
AAE	Encoder	1000,128,128,10
	Decoder	10,128,128,1000
	Discriminator	10,128,128,1
AVB-AC	Encoder	1000+64,128,128,10
	Decoder	10,128,128,1000
	Discriminator	1010,128,128,1

Table B.3: Test log-likelihood (LL) for different models on low dimensional embedding dataset.

Method	LL
CouGAN (ELU)	Failure
CouGAN (ReLU)	Failure
BiGAN w BN	-3940.3±3519.4
BiGAN w/o BN	-55528.9±45685.8
VEEGAN w BN	-157286274.0±1036296.3
VEEGAN w/o BN	-7457806.0±4036262.3
AAE w BN	1025.9±177.1
AAE w/o BN	-44627.7±39387.2

Table B.4: Network architectures.

Method	Network	Neurons per layer
PAE	Encoder	2352,2500,2000,1500,1000,500,10
	Decoder	10,500,1000,1500,2000,2500,2352
	Discriminator	-
CouGAN	Encoder	-
	Decoder	10,500,1000,1500,2000,2500,2352
	Discriminator	2352,2500,2000,1500,1000,500,1
WGAN	Encoder	-
	Decoder	10,500,1000,1500,2000,2500,2352
	Discriminator	2352,2500,2000,1500,1000,500,1
BiGAN	Encoder	2352,2500,2000,1500,1000,500,10
	Decoder	10,500,1000,1500,2000,2500,2352
	Discriminator	2362,2500,2000,1500,1000,500,1
VEEGAN	Encoder	2352+1,2500,2000,1500,1000,500,10
	Decoder	10,500,1000,1500,2000,2500,2352
	Discriminator	2362,2500,2000,1500,1000,500,1
VAE	Encoder	2352,2500,2000,1500,1000,500,20
	Decoder	10,500,1000,1500,2000,2500,2352
	Discriminator	-
AAE	Encoder	2352,2500,2000,1500,1000,500,10
	Decoder	10,500,1000,1500,2000,2500,2352
	Discriminator	10,2500,2000,1500,1000,500,1
AVB-AC	Encoder	2352+64,2500,2000,1500,1000,500,10
	Decoder	10,500,1000,1500,2000,2500,2352
	Discriminator	2362,2500,2000,1500,1000,500,1