Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering

Hybrid Discriminative-Generative Training for Set data

Hybridní diskriminativně-generativní modely pro množinová data

Master Thesis

Author: **Bc. Jakub Bureš**

Supervisor: **doc. Ing. Václav Šmídl, Ph.D.**

Academic year: 2021/2022

Acknowledgment:

I would like to thank doc. Ing. Václav Šmídl, Ph.D. for his expert guidance and patience during this academic year. In addition, I would like to thank my family for all the support that I have received on my academic journey.

Author's declaration:

I declare that this Master Thesis is entirely my own work and I have listed all the used sources in the bibliography.

Prague, May 2, 2022 Bc. Jakub Bureš

Název práce:

Hybridní diskriminativně-generativní modely pro množinová data

Autor: Bc. Jakub Bureš

Obor: Matematické inženýrství

Zaměření: Aplikované matematicko-stochastické metody

Druh práce: Diplomová práce

Vedoucí práce: doc. Ing. Václav Šmídl, Ph.D.

Abstrakt: Tato diplomová práce se zabývá hybridními diskriminativními a generativními modely a jejich možným využitím v multi–instačním učení, kde je jeden vzorek tvořen množinou vektorů. Nejprve se seznámíme s technikáliemi tohoto přístupu, po té provedeme jednoduchý experiment a nakonec se budeme věnovat samotnému multi–instančnímu učení. Hlavním cílem je pak využít strukturu HMill s knihovnou Mill.jl a rozšírit je o konstrastivní učení na množinová data, kde ukážeme výhody oproti diskriminativnímu učení.

Klíčová slova: hybridní diskriminativní a generativní modely, multi–instanční učení

Title:

Hybrid Discriminative-Generative Training for Set data

Author: Bc. Jakub Bureš

Abstract: This Master Thesis deals with hybrid discriminative and generative modeling and its possible utilization in multiple–instance learning. At first, technicalities of this approach are introduced; consequently, a simple experiment is performed, and finally, multi–instance learning itself is taken care of. The main aim of this work is to use the HMill framework with the Mill.jl library and to incorporate contrastive learning into it, where the benefits of this approach are shown.

Key words: hybrid discriminative and generative modeling, multiple instance learning

Contents

List of Figures

List of Tables

List of Symbols

The next list describes several symbols that will be later used within the body of the document

$\delta(.)$	Dirac delta function
$\mathbb{E}[.]$	expected value
$\gamma(.)$	indexing function
$\Pr(.)$	Probability of a argument
$\mathbb{N}_0$	the set of natural numbers and zero
$\mathcal{B}$	bag space
$\mathcal{C}$	finite set of labels
$\mathcal{D}$	set of data in supervised learning
$\mathcal{D}^\star$	set of data in multiple instance learning
$\mathcal{L}(.)$	loss function
$\mathcal{N}(.)$	Normal distribution
$\mathcal{X}$	instance space
$\mathbb{R}$	the set of real numbers
$p(.)$	probability distribution function
$Z(\boldsymbol{\theta})$	normalization constant (partition function)

List of Acronyms

PDF	probability density function
KL	Kullback-Leibler (divergence)
MLE	maximum likelihood estimation
ML	machine learning
CE	cross-entropy
CV	cross-validation
SL	supervised learning
UL	unsupervised learning
i.i.d.	independent and identically distributed
EBM	energy–based model
JEM	joint energy–based model
HDGM	hybrid discriminative generative energy-based model
MIL	multiple instance learning
NN	neural network
MCMC	markov chain monte carlo

Introduction

In the field of supervised learning [18] tremendous progress and success have been achieved in recent years. Examples of such successes include speech recognition [20] or anomaly detection [3]. A classification task is typically addressed minimizing a cross entropy loss, which is defined as an expected value of logarithm of the Softmax function.

Contrastive learning [10, 11] is a machine learning method that is often used in representation learning for image classification or video understanding. For training such models is, most of the time, minimized the contrastive loss, which reduces the 'distance' between representations of different augmented views of the same image and increases the distance between representations of augmented views of different images.

In this research project, these two objectives are brought together and utilized in the form of a hybrid combination [12], which is used instead of the mentioned cross entropy loss. Consequently, this approach is applied to multiple instance learning problems, taking advantage of the unified framework HMill and Mill.jl library [2] implemented in Julia programming language [19].

This work is arranged into 3 chapters in a logical sequence. In the first chapter is written theoretical introduction needed for a better understanding of the whole work. The second chapter consists of discriminative and generative modeling and its hybrid combination, where the simple polynomial regression experiment is performed. In the last, third, chapter, multiple instance learning is introduced with the following experiments. The primary goal of this work is to test out the hybrid approach on the real data and, eventually, show its benefits in comparison to discriminative learning.

Chapter 1

Theoretical Introduction

It is customary that vast academic paper, for better understanding, in its beginning outlines miscellaneous theoretical aspects, which are used during the other chapters. This thesis is no exception.

1.1 Mathematical Notation

It will be most appropriate to begin by introducing the basic notation that will be used throughout this thesis. This will ensure that any confusion will be avoided, even though the notation is quite standard.

Notation for random variables using upper case letters of the Latin alphabet is widely used. Typically, the letters used are from the end of the alphabet, i.e. X, Y or Z. The realization of a random variable, also known as an observed value or simply an observation, will be denoted by the appropriate lower case letters. Thus, the realization $x \in \mathbb{R}$ corresponds to the random variable X, which holds by analogy for other random variables. However, significant simplification can be achieved if one uses the same notation for random variables and realizations.

Bold symbols, for instance, $\boldsymbol{x} \in \mathbb{R}^D$ or $\boldsymbol{y} \in \mathbb{R}^D$, will be used to distinguish vectors and scalars. All vectors are assumed to be column vectors, so $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)^\top$ and hence $\boldsymbol{x}^\top$ is a row vector.

Any matrices will be denoted by blackboard bold Latin letters, for example, if one has N values of D-dimensional vector of observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, it can be simply combined into a $D \times N$ data matrix $\mathbb{X}$ in which the j^{th} row of $\mathbb{X}$ corresponds to the row vector $\boldsymbol{x}_j^\top$. The symbol $\mathbb{I}_N$ denotes the square $N \times N$ identity matrix, i.e., matrix with ones on the main diagonal and zeros elsewhere. The set of observations will be denoted by bold uppercase letter, for example $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$.

1.2 Probability Theory

Mathematical models are very well described by probability and for this reason, this section will look at some of the basic concepts of probability theory that we will need. The most important such concept is probability density (PDF). The symbol $p(x)$ will be used predominantly for the PDF, which is a function of x. In addition, this will be used for both discrete and continuous x. In this way can be achieved significant simplification and unification of all formulas and equations. Any PDF is a non-negative function and its integration over the entire space is equal to 1. This applies to multivariate case as well as it applies to univariate case, therefore integration of joint PDFs $p(\boldsymbol{x}) = p(x_1, x_2, \ldots, x_D)$ over the entire space is also equal to 1. In mathematical terms one can express it as follows

$$\int_{\mathbb{R}^D} p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1. \tag{1.1}$$

In other parts of this thesis we will use conditional PDFs such as $p_{\boldsymbol{\theta}}(\boldsymbol{x}) \equiv p(\boldsymbol{x}|\boldsymbol{\theta})$ that are conditioned by known parameters $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^s$, where Θ is called parameter space. The constraint (1.1) can be always fulfilled by redefining the PDF as

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{p^0_{\boldsymbol{\theta}}(\boldsymbol{x})}{Z(\boldsymbol{\theta})}, \qquad Z(\boldsymbol{\theta}) = \int_{\mathbb{R}^D} p^0_{\boldsymbol{\theta}}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{1.2}$$

where $p^0_{\boldsymbol{\theta}}(\boldsymbol{x})$ specifies the functional form of the $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and does not need to integrate to 1. The normalization constant $Z(\boldsymbol{\theta})$ is often called the partition function.

The average value of some function $g(x)$ under a probability distribution $p(x)$ is typically denoted by $\mathbb{E}\left[g(x)\right]$ and it is called expected value or mean [1]. For a continuous variable, expected value are expressed in terms of an integration with respect to the corresponding probability density

$$\mathbb{E}\left[g(x)\right] = \int_{\mathbb{R}} p(x) g(x) \mathrm{d}x. \tag{1.3}$$

In the case of a discrete variable, one has to keep in mind that an integration turns into a sum over all x. To specify over which PDF the expectation is calculated, the notation $\mathbb{E}_{p(x)}\left[g(x)\right]$ can be used.

1.3 Supervised Learning

Supervised learning (SL), in less academic terms called "learning with a teacher", is one of the machine learning tasks [18]. The goal of this approach is to make a good prediction of the output y (sometimes also called target variable), denoted by the symbol $\hat{y}$, with given input $\boldsymbol{x}$. This prediction is obtained through learning a model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) \equiv f(\boldsymbol{x}; \boldsymbol{\theta})$ that minimizes a loss function $\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$ (also known as the error function), where $\boldsymbol{\theta} \in \Theta$ are the parameters of the model.

To construct this prediction one needs data, hence it is supposed that we have available set of independent and identically distributed (i.i.d.) observations, input–output paired samples denoted by $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, eventually, this may in fact be

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N, \quad \boldsymbol{x}_i \in \mathbb{R}^D, \quad y_i \in \mathbb{R}, \quad \forall i = 1, \dots, N. \tag{1.4}$$

The index i will be omitted whenever it is clear that we are referring to terms associated with a single data point. Such setting is usually known as training data and its applications are regression problems. As an example, we can mention a couple of typically used loss functions for such problems. They are squared error and absolute error

$$\mathcal{L}(\widehat{y}, y) = \begin{cases} (y - \widehat{y})^2 \\ |y - \widehat{y}| \end{cases} \tag{1.5}$$

where $\widehat{y} = \widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = f(\boldsymbol{x}; \widehat{\boldsymbol{\theta}})$. As we are not quite interested in regression problems in this thesis, we will mainly deal with the second approach. That is classification problems, i.e. when $y \in \mathcal{C}$ is qualitative output and where $\mathcal{C}$ is a finite set. A typical example is binary classification, where $\mathcal{C} = \{0, 1\}$. However, classification will be object of interest later in Section 2.

Here it is clear why the term "learning with a teacher" is used. This metaphor means that the student presents output $\widehat{y}$ and the teacher provides either a correct answer and/or an error that corresponds to the student's answer.

1.3.1 Prediction

The generalization performance, i.e. the performance on out–of–sample data of the models learned by the algorithm relates to its prediction capability on independent test data $\mathcal{T}$. Assessment of this performance is essentially important in practice, since it conducts the choice of learning method or model, and gives a measure of the quality of the hereafter chosen model. There are in fact two seperate goals to achieve:

1. *Model selection* - estimating the performance of different models in order to choose the best one.

2. *Model assessment* - having chosen a final model, estimating its prediction error (generalization error) on new data.

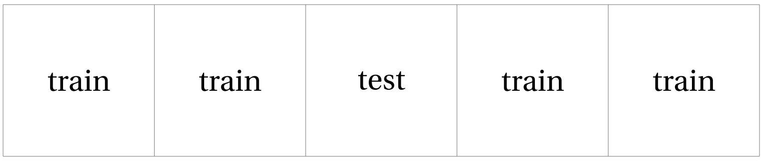
1.3.1.1 Cross-Validation

The simplest and most widely used method for estimating prediction error of the model $\hat{f}_{\boldsymbol{\theta}}$ is called *cross-validation* (CV) [17]. It is used for direct estimating of the expected extra-sample error

$$\text{err} = \mathbb{E}\left[\mathcal{L}(y, \widehat{y})\right], \tag{1.6}$$

the measure how accurately is the model able to predict output values for previously unseen data - independent test sample. In an ideal case, if sufficient number of data is available, a test set can be set aside and used to assess the performance of the employed prediction model. Since data are often scarce, this is usually not possible.

Very elegant solution to this problem is via *K-fold cross-validation* [17]. It uses part of the available data for fitting the model, and a different part for testing. We split the data into K roughly equal-sized parts, for example, when $K = 5$, the scenario is shown in Figure 1.1. For the j^{th} part (third in Figure 1.1), we train the model to the other $K - 1$ parts of the data, and calculate the prediction error of the fitted model when predicting the j^{th} part of the data. We repeat this process for $j \in \{1, 2, \ldots, K\}$ and combine the K estimates of prediction error. Let $\gamma : \{1, \ldots, N\} \to \{1, \ldots, K\}$ be an indexing function that indicates the partition to which observation j is allocated by the randomization. Symbol $\hat{f}_{\boldsymbol{\theta}}^{-j}(\boldsymbol{x})$ denotes the fitted model, computed with the j^{th} part of the data removed. Then the cross-validation estimate of prediction error is defined by



train	train	test	train	train

Figure 1.1: Splitting the data into $K = 5$ roughly equal-sized parts.

$$\text{CV}\left(\hat{f}_{\boldsymbol{\theta}}\right) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(y_i, \hat{f}_{\boldsymbol{\theta}}^{-\gamma(i)}(\boldsymbol{x}_i)\right). \tag{1.7}$$

Typical choices of K are 5 or 10 and even case $K = N$ that is known as *leave-one-out* CV. Generally, there is not an universal way of choosing K, since it strongly depends on the available number of data. The biggest problem of this method is a fact that it is computionally very expensive, because we usually train many models with different complexity and assess their performance. Let us now analyze the problem of the model complexity. Consider a polynomial regression problem, where the model is defined by

$$f_{\boldsymbol{\theta}}(x) = \sum_{i=0}^{s-1} \theta_i x^i. \tag{1.8}$$

Here, over–fitting occurs very frequently. The complexity of the model of this case is very intuitive, as it is just the order of the polynomial, $s - 1$. Smaller orders of the polynomial (may) give rather poor fits to the data in contrast to a much higher order polynomial giving an excellent fit. However, such a polynomial passes exactly through each data point, oscillates wildly, and gives a poor prediction for the new input variable $x_0 \in \mathbb{R}$.

To obtain some quantitative insight into the dependence of the generalization performance on model complexity, consider a separate test set of data (testing data) used to assess the performance of the model. In general, the prediction error evaluated on the training data for increasing the complexity of the model approaches zero. On the other hand, the prediction error evaluated on the testing data for increasing model complexity is (from a certain point) increasing as well. The typical scenario is illustrated in Figure 1.2. The goal is then to choose a model that performs best on testing data. For extremely complicated and complex models that are trained for hours or days, is cross–validation inconvenient approach of estimating the prediction error as we need to train numerous models of this complexity.

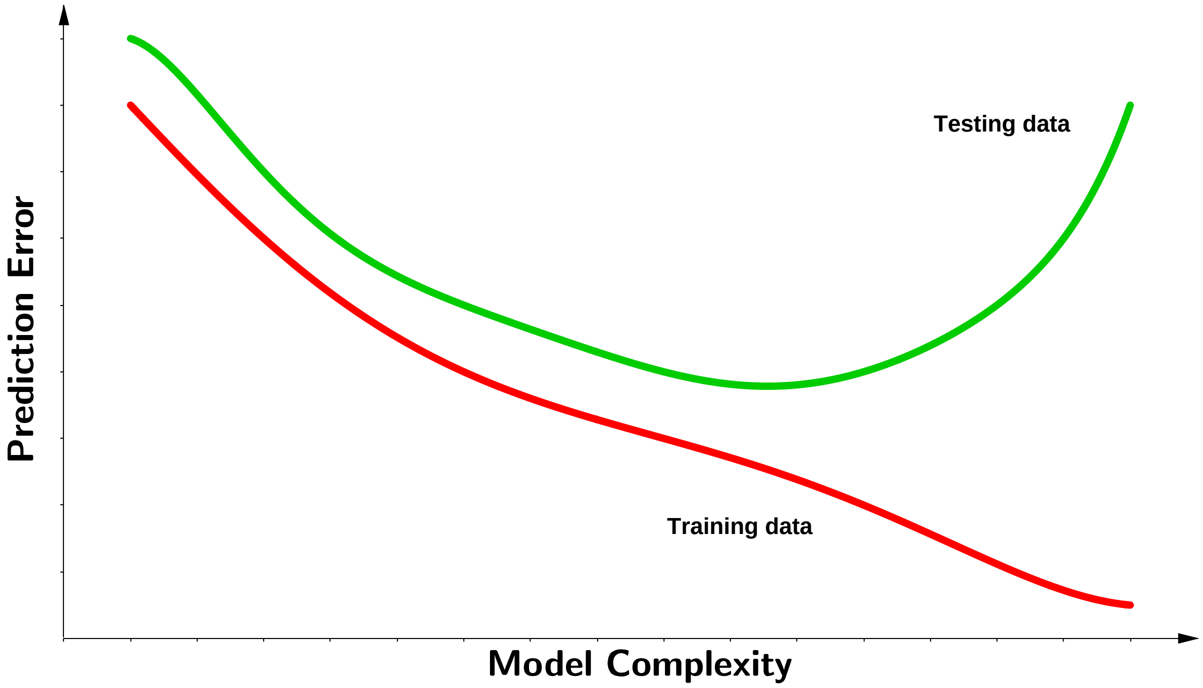


Figure 1.2: Evaluation of prediction error as a function of model complexity.

1.4 Unsupervised Learning

The previous section dealt with input–output paired samples D. The second approach is a logical modification of SL, based on data without labels. Such setting is called unsupervised learning (UL) or "learning without a teacher". Unlike SL, one has a set of N observations in the form of $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ and nothing more. In this case, the student learns without any feedback from a supervisor or teacher providing correct answers. The goal is to directly infer the properties of $p(\boldsymbol{x})$.

1.5 Bayesian Inference

The Bayesian methodology is a well established approach to statistical inference and became very important technique in statistics and data analysis. As its name suggests, Bayesian statistics is based on application of Bayes' rule. In this chapter, we briefly review basic concept of this approach, which was suggested here [9].

Let the measured data be denoted by $\mathcal{D}$, defined according to previous Section 1.1. A parametric probabilistic model of the data $\mathcal{D}$ is given by the probability density function $p(\mathcal{D}|\boldsymbol{\theta})$, where again $\boldsymbol{\theta} \in \Theta$ denotes parameters of the model. The main idea behind Bayesian theory is the treatment of the unknown parameters $\boldsymbol{\theta}$ as a random variable. Bayes' rule is applied to infer model parameters $\boldsymbol{\theta}$, therefore

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\boldsymbol{\theta},\mathcal{D})}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_\Theta p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}}. \tag{1.9}$$

Since $p(\mathcal{D})$ is just the normalization constant, Equation (1.9) is often simplified to

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}). \tag{1.10}$$

Symbol $\propto$ means equal up to the normalization constant. The term $p(\boldsymbol{\theta}|\mathcal{D})$ is known as the *posterior* distribution, $p(\mathcal{D}|\boldsymbol{\theta})$ as the *observation model*, and $p(\boldsymbol{\theta})$ is called the *prior* distribution of the $\boldsymbol{\theta}$. Note that evaluation of the normalization constant can be computionally expensive, in higher dimension even intractable.

There is of course many possible options how to obtain $\widehat{\boldsymbol{\theta}}$ from posterior. Popular choices for an optimal value of the point estimate are:

1. Maximum A posteriori estimate (MAP)

$$\widehat{\boldsymbol{\theta}}_{\mathrm{MAP}} = \operatorname*{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}) \tag{1.11}$$

   This method estimates $\boldsymbol{\theta}$ as the mode of the posterior distribution. It appears to be computionally attractive, as it is not necessary to evaluate the normalization constant.

2. Mean or expected value

$$\widehat{\boldsymbol{\theta}}_{\mathrm{B}} = \int_\Theta \boldsymbol{\theta}\ p(\boldsymbol{\theta}|D)\mathrm{d}\boldsymbol{\theta} = \mathbb{E}_{p(\boldsymbol{\theta}|D)}[\boldsymbol{\theta}] \tag{1.12}$$

   Mean value, unlike MAP estimate, may be very expensive to compute because of the required integration. This may lead to further approximations such as EM algorithm [13].

1.5.1 Choice of prior distribution

For the posterior computation, it is necessary to specify the prior distribution $p(\boldsymbol{\theta})$, unfortunately, this might not be easily determined. This can be achieved through knowledge of previous models, expert knowledge, their combination, or even uncertainty about $\boldsymbol{\theta}$ being a viable option.

There are also many practical aspects of priors:

- *Regularization* - supplementing the data if there are scarce, insufficient data, or poorly defined models.

- *Restrictive conditions* - imposing various restrictions on the parameters $\boldsymbol{\theta}$ reflecting physical constraints. The choice of a prior distribution with bounded support will also result in a posterior distribution with bounded support.

- *Non–informative prior* - if the data are informative enough to make a prediction, it is proposed to choose a prior with minimal impact on the posterior distribution, such as uniform distribution. However, typical choices of non–informative priors are the so–called *Jeffreys priors* [5].

1.5.2 Prediction

We are not usually interested in the value of $\widehat{\boldsymbol{\theta}}$ itself, but rather, once the model is estimated, we are interested in making a prediction of the output variable y_0 for the new input variable $\boldsymbol{x}_0$. Note that the symbol $\mathcal{D}$ contains all previously given data $\boldsymbol{x}$ and y. The posterior predictive distribution is then determined by the distribution of y_0, marginalized over the posterior

$$p(y_0|\boldsymbol{x}_0, \mathcal{D}) = \int_{\Theta} p(y_0|\boldsymbol{x}_0, \boldsymbol{\theta})\, p(\boldsymbol{\theta}|\mathcal{D})\mathrm{d}\boldsymbol{\theta}. \tag{1.13}$$

When the distribution $p(\boldsymbol{\theta}|\mathcal{D})$ is not available, we have to approximate leveraging the Dirac delta function $\delta(x)$ for which the property

$$\int_{\mathbb{R}} g(x)\,\delta(x - x_0)\mathrm{d}x = g(x_0) \tag{1.14}$$

holds. Once this property is applied to Equation (1.13) we get

$$p(y_0|\boldsymbol{x}_0, \mathcal{D}) = \int_{\Theta} p(y_0|\boldsymbol{x}_0, \boldsymbol{\theta})\delta(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}})\mathrm{d}\boldsymbol{\theta} = p(y_0|\boldsymbol{x}_0, \widehat{\boldsymbol{\theta}}), \tag{1.15}$$

causing an error. In typical MAP, this is known as *over–fitting*.

Chapter 2

Discriminative vs. Generative Models

2.1 Overview

Machine learning models can be classified into two main categories, discriminative and generative models. Simply put, a discriminative model makes predictions based on conditional probability $p\left(y|\boldsymbol{x}\right)$ and is used for classification or regression problems. In other words, discriminative models distinguishes the decision boundary between the classes. It corresponds to learning parameters that maximize the conditional probability distribution $p(y|\boldsymbol{x})$. On the contrary, a generative model revolves around the distribution of a data set to return a probability for a given example. Rather than looking at classes and trying to find something to separate them, it focuses only on the one class at the time and builds a model what that certain class looks like, then turns attention to the other class. To express it more formally, generative models learn parameters that maximize $p\left(\boldsymbol{x}|y\right)$ and $p\left(y\right)$. Since

$$p\left(\boldsymbol{x},y\right)=p\left(\boldsymbol{x}|y\right)\cdot p\left(y\right), \tag{2.1}$$

with joint PDF it is possible to generate new $\left\{\boldsymbol{x}',y'\right\}$ pairs. In some cases, the use of the second decomposition $p\left(\boldsymbol{x},y\right)=p\left(y|\boldsymbol{x}\right)\cdot p\left(\boldsymbol{x}\right)$ is also an option. Note that in an unsupervised setting, the task is reduced to inferring only $p\left(\boldsymbol{x}\right)$.

2.2 Discriminative Modeling

In this section, we review the basics of discriminative modeling proposed in [12]. Given data $\mathcal{D}$ according to (1.4), with the empirical distribution of $\boldsymbol{x}$ being referenced by $\tilde{p}(\boldsymbol{x})$ and the empirical label distribution $\tilde{p}(y|\boldsymbol{x})$ containing L categories. In this thesis, we focus on classification problems, where the variable y is now a qualitative variable called a class label, taking on L possible values, and comes from a finite set $\mathcal{C}$. A classification problem is typically solved using a parametric function $f_{\boldsymbol{\theta}}:\mathbb{R}^D\to\mathcal{C}$, where $\boldsymbol{\theta}$ denotes the parameters of the model. In practice, the function $f_{\boldsymbol{\theta}}$ is often used in the form of $\mathbb{R}^D\to\mathbb{R}^L$. This function maps each data point $\boldsymbol{x}\in\mathbb{R}^D$ to L real-valued numbers known as logits. It should be noted that $\mathbb{R}^L$

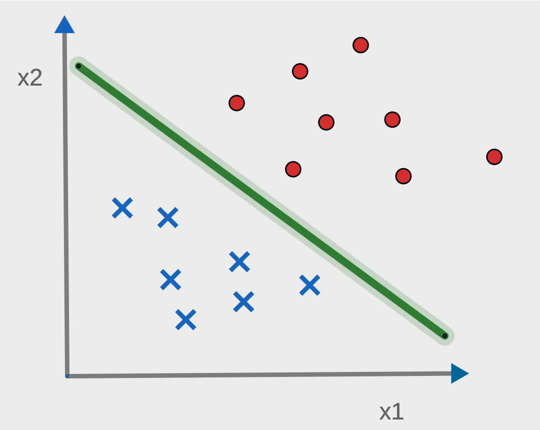


Figure 2.1: Discriminative approach.

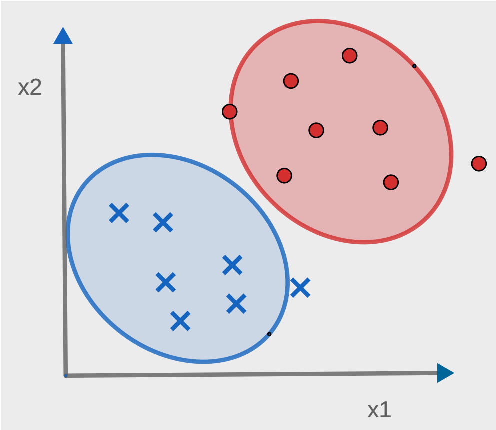


Figure 2.2: Generative approach.

is allowed here due to the utilization of *one-hot encoding*, which will be explained in Section 2.2.2. Logits are used to parameterize a categorical distribution through the transfer function

$$q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right) = \frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)\left[y\right]\right)}{\sum_{y\in\mathcal{C}}\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)\left[y\right]\right)}, \tag{2.2}$$

which is known as the Softmax. In other words, the true data distribution $\tilde{p}\left(y|\boldsymbol{x}\right)$ is modeled by a parameterized family of functions $\left\{q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)|\boldsymbol{\theta}\in\Theta\right\}$ and thus $\tilde{p}\left(y|\boldsymbol{x}\right)$ is assumed to belong to this family. Note that the convention $f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)\left[y\right]$ means the y^{th} element of $f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)$, that is, the logit corresponding to the y^{th} class label. For learning $f_{\boldsymbol{\theta}}$ is usually minimized cross-entropy (CE) loss

$$\text{CE}\left(\boldsymbol{\theta}\right) = -\mathbb{E}_{\tilde{p}(y,\boldsymbol{x})}\left[\log q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)\right] = -\sum_{i=1}^{N}\sum_{y\in\mathcal{C}}\log q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}_i\right), \tag{2.3}$$

as it is relatively easy to compute and has several other justifications. These justifications will be addressed in the following text.

2.2.1 Connection to Kullback–Leibler divergence

The rationale for objective (2.3) comes from minimizing the Kullback-Leibler (KL) divergence with a target distribution $\tilde{p}(y|\boldsymbol{x})$ [7]. In general, the KL divergence (or KL distance) from $\tilde{p}(y|\boldsymbol{x})$ to $q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)$ is defined as

$$D_{\text{KL}}\left(\tilde{p}(y|\boldsymbol{x})||q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)\right) = \int\tilde{p}(y|\boldsymbol{x})\log\frac{\tilde{p}(y|\boldsymbol{x})}{q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)}\mathrm{d}y = \mathbb{E}_{\tilde{p}(y|\boldsymbol{x})}\left[\log\frac{\tilde{p}(y|\boldsymbol{x})}{q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)}\right] \tag{2.4}$$

and has the following properties:

1. $D_{\text{KL}}\left(\tilde{p}(y|\boldsymbol{x})\|q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)\right) \geqslant 0$,

2. $D_{\mathrm{KL}}\left(\tilde{p}(y|\boldsymbol{x}) \| q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)\right) = 0$ iff $\tilde{p}(y|\boldsymbol{x}) = q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)$ almost everywhere,
3. $D_{\mathrm{KL}}\left(\tilde{p}(y|\boldsymbol{x}) \| q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)\right) \neq D_{\mathrm{KL}}\left(q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right) \| \tilde{p}(y|\boldsymbol{x})\right)$ and KL divergence does not obey the triangle inequality.

The third property indicates that care is needed in the syntax describing KL divergence. We say that (2.4) is from $\tilde{p}(y|\boldsymbol{x})$ to $q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)$. Using the logarithmic property, (2.4) can be further rewritten in the form

$$\mathbb{E}_{\tilde{p}(y|\boldsymbol{x})}\left[\log\frac{\tilde{p}(y|\boldsymbol{x})}{q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)}\right] = \mathbb{E}_{\tilde{p}(y|\boldsymbol{x})}\left[\log\tilde{p}(y|\boldsymbol{x})\right] - \mathbb{E}_{\tilde{p}(y|\boldsymbol{x})}\left[\log q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)\right], \tag{2.5}$$

where the first term is called entropy, often denoted by $H\left(\tilde{p}(y|\boldsymbol{x})\right)$ and the second term is called CE. The subscript $\boldsymbol{\theta}$ emphasizes that $q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)$ is the approximative density we get to control. This gives us KL distance for single data point $\boldsymbol{x}$, but for optimization we need to include all data points. Let

$$\mathcal{L}\left(\boldsymbol{\theta}\right) = \sum_{i=1}^{N} D_{\mathrm{KL}}^{(i)}\left(\tilde{p}(y|\boldsymbol{x}_i) \| q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}_i\right)\right) \tag{2.6}$$

be the sum of KL distances over all data points. Since the entropy of $\tilde{p}(y|\boldsymbol{x}_i)$ does not depend on $\boldsymbol{\theta}$ therefore by minimizing (2.6) with respect to $\boldsymbol{\theta}$ we obtain

$$\min_{\boldsymbol{\theta}}\mathcal{L}\left(\boldsymbol{\theta}\right) = \min_{\boldsymbol{\theta}} -\sum_{i=1}^{N}\mathbb{E}_{\tilde{p}(y|\boldsymbol{x}_i)}\left[\log q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}_i\right)\right] = \min_{\boldsymbol{\theta}} -\mathbb{E}_{\tilde{p}(y,\boldsymbol{x})}\left[\log q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)\right], \tag{2.7}$$

which corresponds to minimizing the objective $\mathrm{CE}\left(\boldsymbol{\theta}\right)$ defined in Equation (2.3). This part deserves further discussion for a few reasons:

- Maximum likelihood estimation (MLE) of $\boldsymbol{\theta}$ is equivalent to minimizing the KL distance.
- One may encounter the concepts of minimization or maximization of CE.

To address these reasons, it is necessary to briefly review the MLE. The MLE principle assumes that the most reasonable values for $\boldsymbol{\theta}$ are those for which the probability of the observed sample is highest. Since $q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)$ is the PDF model, we have to follow the log–likelihood function

$$\mathcal{L}_{\mathrm{ML}}\left(\boldsymbol{\theta}\right) = \sum_{i=1}^{N}\sum_{y\in\mathcal{C}}\log q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}_i\right), \tag{2.8}$$

Further optimization of $\mathcal{L}_{\mathrm{ML}}\left(\boldsymbol{\theta}\right)$ gives the point estimate

$$\widehat{\boldsymbol{\theta}}_{\mathrm{ML}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\sum_{i=1}^{N}\sum_{y\in\mathcal{C}}\log q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}_i\right) \tag{2.9}$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} -\sum_{i=1}^{N}\sum_{y\in\mathcal{C}}\log q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}_i\right). \tag{2.10}$$

It is much more common to minimize a function than to maximize it in practice, and therefore the log–likelihood function is inverted by adding a negative sign to the front of (2.9) yielding a negative log–likelihood or simply cross–entropy.

2.2.2 One–hot Encoding

Machine learning (ML) algorithms can misinterpret the numeric values of labels if there exists a hierarchy between them. One–hot encoding is a very common approach for dealing with this issue in order to improve the algorithm performance. Each unique category value is transformed into a new column, and these dummy variables are then filled with 0 or 1 (0 for FALSE and 1 for TRUE). For the sake of clarity, the transformation of a label encoding into a one–hot encoding is illustrated in the following table 2.1.

However, this method has its own downsides. For example, it creates new variables and if there exist many unique category values, the models have to deal with a large number of predictors, leading to the so-called *Big-p problem* [8]. Also, one–hot encoding causes multicolinearity between the individual variables, which may lead to reducing the model's accuracy.

Food Name	Categorical #	Calories
Pizza	1	266
Hamburger	2	295
Caviar	3	264

⇒

Pizza	Hamburger	Caviar	Calories
1	0	0	266
0	1	0	295
0	0	1	264

Table 2.1: Transformation of a label encoding (left) to the one–hot encoding (right).

2.3 Generative Modeling

2.3.1 Variational Autoencoder

The first generative modeling approach that will be discussed is the variational autoencoder (VAE). In this section, motivation will be addressed and individual mathematical aspects will be discussed in detail.

2.3.1.1 Problem Scenario

Assume that the data $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ are generated by some random process involving an unobserved continuous variable $\boldsymbol{z}$, which will be referenced as a latent variable or code. The objective is again to find the PDF of the given data in parametric form $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. One can choose an approximative distribution in the form of

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})\,\mathrm{d}\boldsymbol{z} = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\, p_{\boldsymbol{\theta}}(\boldsymbol{z})\,\mathrm{d}\boldsymbol{z}, \tag{2.11}$$

But such an approximation is usually very expensive to compute or can even be intractable. Intractability of the $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ makes posterior PDF $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ also intractable.

2.3.1.2 Naive Approach

One of the simplest ways to solve this problem may seem to be to build a model depending on the latent variable $f_{\boldsymbol{\theta}}(\boldsymbol{z})$ and try to train its parameters. For simplicity, let $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbb{I}_P)$, where P denotes the dimension of the latent space $\boldsymbol{z}$, and also let

$$\boldsymbol{x} = f_{\boldsymbol{\theta}}(\boldsymbol{z}) + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \cdot \mathbb{I}_D\right) \tag{2.12}$$

which actually gives

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{x}; f_{\boldsymbol{\theta}}(\boldsymbol{z}), \sigma^2 \cdot \mathbb{I}_D\right). \tag{2.13}$$

It may be noted that the subscript D represents the dimension of the data point $\boldsymbol{x}$. The true PDF of the given data can be cleverly written using the empirical PDF, i.e., in the form of $\tilde{p}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x} - \boldsymbol{x}_i)$, which can be exploited by finding the parameters $\boldsymbol{\theta}$ by minimizing $D_{\mathrm{KL}}\left(\tilde{p}(\boldsymbol{x}) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{x})\right)$. Since minimizing the KL distance is equivalent to ML estimation and using the approximative form (2.11), the following holds

$$\begin{aligned}
\widehat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} - \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) && (2.14)\\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} - \sum_{i=1}^{N} \log \int \mathcal{N}\left(\boldsymbol{x}_i; f_{\boldsymbol{\theta}}(\boldsymbol{z}), \sigma^2 \cdot \mathbb{I}_D\right) \cdot \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \mathbb{I}_P)\, \mathrm{d}\boldsymbol{z} && (2.15)\\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} - \sum_{i=1}^{N} \log \sum_{j=1}^{P} \exp\left(-\frac{1}{2\sigma^2}\left(\boldsymbol{x}_i - f_{\boldsymbol{\theta}}(\boldsymbol{z}_j)\right)^{\top}\left(\boldsymbol{x}_i - f_{\boldsymbol{\theta}}(\boldsymbol{z}_j)\right)\right). && (2.16)
\end{aligned}$$

Integration over $\boldsymbol{z}$ is represented by sampling. In iterations, for an incorrect value of $\boldsymbol{\theta}$, all the generated samples may be away from the samples of $\boldsymbol{x}$, and the gradient is poor.

2.3.1.3 Variational Bayes Approach

To solve this problem, it is necessary to introduce a further approximative posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \approx p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ with parameters $\boldsymbol{\phi}$, preferably Gaussian. The standard terminology refers to the model $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ as a probabilistic *encoder* and $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is called a probabilistic

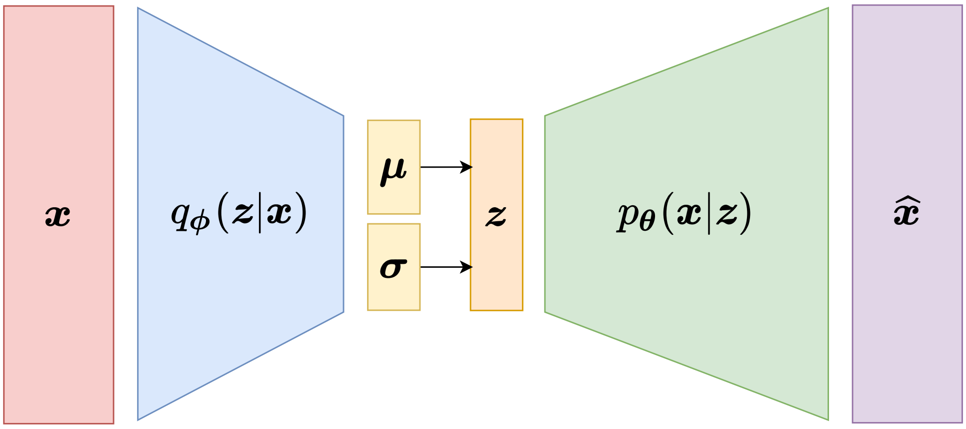

$\boldsymbol{x}$
$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$
$\boldsymbol{\mu}$
$\boldsymbol{\sigma}$
$\boldsymbol{z}$
$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$
$\widehat{\boldsymbol{x}}$

Figure 2.3: VAE diagram.

decoder. For VAE, the idea is to use the KL distance from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ to $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$, which produces

$$\begin{aligned} D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\right) &= \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})} \mathrm{d}\boldsymbol{z} \\ &= \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\, p_{\boldsymbol{\theta}}(\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\, p_{\boldsymbol{\theta}}(\boldsymbol{z})} \mathrm{d}\boldsymbol{z} \\ &= \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) + \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\, p_{\boldsymbol{\theta}}(\boldsymbol{z})} \mathrm{d}\boldsymbol{z} \\ &= \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{z})} - \log p(\mathbf{x}|\boldsymbol{z})\right] \\ &= \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) + D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z})\right) - \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right]. \end{aligned} \tag{2.17}$$

Using the last equality of (2.22), it is possible to rewrite the equation in its typical form

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\right) = \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] - D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z})\right)}_{= L(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x})}, \tag{2.18}$$

where the right-hand side is called *variational lower bound* for a single data point. There is no uniformity in terminology, and thus one can also encounter the name evidence lower bound (ELBO). The first term on the right-hand side is known as reconstruction loss, and the second term is often called a regularization term. As a KL distance is always non-negative, it holds

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geqslant L(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x}). \tag{2.19}$$

The objective is to maximize the log-likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ which is equivalent to minimizing the negative log-likelihood and that is what will be used here. At this point, we have a lower bound for one data point $\boldsymbol{x}$, but we need to include all observations in the lower bound. The joint log-likelihood can be rewritten as a sum over the marginal log-likelihoods of individual observations $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ that completes all the building blocks needed to determine the optimization equation. This formulation provides one major advantage, which is that it is now possible to jointly optimize both the generative parameters $\boldsymbol{\theta}$ and the variational parameters $\boldsymbol{\phi}$ as follows

$$\begin{aligned}
\widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\phi}} &= \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\operatorname{argmin}} - \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) && (2.20)\\
&= \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\operatorname{argmin}} - \sum_{i=1}^{N} L(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_i) && (2.21)\\
&= \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\operatorname{argmin}} - \sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|\boldsymbol{z})\right] - D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z})\right). && (2.22)
\end{aligned}$$

For a better understanding of the problem, a VAE diagram is shown in Figure 2.3. Note that the latent space is usually much smaller than the input space, and for this reason, it is also sometimes called the bottleneck.

2.3.1.4 Reparameterization Trick

The key success of VAE lies in the fact that (2.22) can be efficiently computed using *reparameterization trick*. We express $\boldsymbol{z}$ as a deterministic variable

$$\boldsymbol{z} = g_{\boldsymbol{\phi}}(\boldsymbol{\varepsilon}, \boldsymbol{x}), \qquad (2.23)$$

where $\boldsymbol{\varepsilon}$ stands for an auxiliary variable with independent marginal $p(\boldsymbol{\varepsilon})$ and $g_{\boldsymbol{\phi}}(.)$ is a function parameterized by $\boldsymbol{\phi}$.

A common explanation for this trick is that during the optimization the gradient cannot back–propagate through a random node. So, in the case of VAE, the reparameterization trick shifts the source of randomness to another variable different from $\boldsymbol{z}$ and allows differentiation with respect to $\boldsymbol{z}$. However, this explanation may not be sufficient, and for this reason we will state a more formal justification. Consider taking the gradient with respect to $\boldsymbol{\theta}$ of $\mathbb{E}_{p(\boldsymbol{z})}\left[f_{\boldsymbol{\theta}}(\boldsymbol{z})\right]$. It can

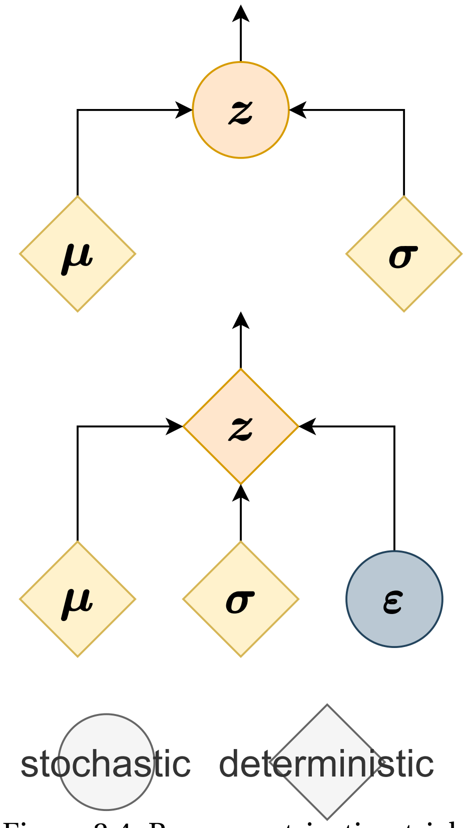


Figure 2.4: Reparametrization trick.

be easily computed as

$$\nabla_{\boldsymbol{\theta}}\mathbb{E}_{p(\boldsymbol{z})}\left[f_{\boldsymbol{\theta}}\left(\boldsymbol{z}\right)\right] = \nabla_{\boldsymbol{\theta}}\int p(\boldsymbol{z})f_{\boldsymbol{\theta}}\left(\boldsymbol{z}\right)\mathrm{d}\boldsymbol{z} \tag{2.24}$$

$$= \int p(\boldsymbol{z})\nabla_{\boldsymbol{\theta}}f_{\boldsymbol{\theta}}\left(\boldsymbol{z}\right)\mathrm{d}\boldsymbol{z} \tag{2.25}$$

$$= \mathbb{E}_{p(\boldsymbol{z})}\left[\nabla_{\boldsymbol{\theta}}f_{\boldsymbol{\theta}}\left(\boldsymbol{z}\right)\right]. \tag{2.26}$$

The result is obvious; the gradient of the expectation is equal to the expectation of the gradient. However, the gradient of the expectation becomes much more interesting if the PDF $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ is also parameterized by $\boldsymbol{\theta}$, resulting in

$$\nabla_{\boldsymbol{\theta}}\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z})}\left[f_{\boldsymbol{\theta}}\left(\boldsymbol{z}\right)\right] = \nabla_{\boldsymbol{\theta}}\int p_{\boldsymbol{\theta}}(\boldsymbol{z})f_{\boldsymbol{\theta}}\left(\boldsymbol{z}\right)\mathrm{d}\boldsymbol{z} \tag{2.27}$$

$$= \int p_{\boldsymbol{\theta}}(\boldsymbol{z})\nabla_{\boldsymbol{\theta}}f_{\boldsymbol{\theta}}\left(\boldsymbol{z}\right)\mathrm{d}\boldsymbol{z} + \int f_{\boldsymbol{\theta}}\left(\boldsymbol{z}\right)\nabla_{\boldsymbol{\theta}}p_{\boldsymbol{\theta}}(\boldsymbol{z})\mathrm{d}\boldsymbol{z} \tag{2.28}$$

$$= \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z})}\left[\nabla_{\boldsymbol{\theta}}f_{\boldsymbol{\theta}}\left(\boldsymbol{z}\right)\right] + \int f_{\boldsymbol{\theta}}\left(\boldsymbol{z}\right)\nabla_{\boldsymbol{\theta}}p_{\boldsymbol{\theta}}(\boldsymbol{z})\mathrm{d}\boldsymbol{z}. \tag{2.29}$$

The second term of (2.29) is not guaranteed to be an expectation and this very fact indicates that backpropagation would not compute an estimate of $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z})}\left[f_{\boldsymbol{\theta}}\left(\boldsymbol{z}\right)\right]$. That being the case, if we apply the reparameterization trick $\boldsymbol{z} = g_{\boldsymbol{\theta}}\left(\boldsymbol{\varepsilon}, \boldsymbol{x}\right)$ to this simple example, we get

$$\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z})}\left[f_{\boldsymbol{\theta}}\left(\boldsymbol{z}\right)\right] = \mathbb{E}_{p(\boldsymbol{\varepsilon})}\left[f\left(g_{\boldsymbol{\theta}}(\boldsymbol{\varepsilon}, \boldsymbol{x})\right)\right]. \tag{2.30}$$

At this point, it is possible to take the gradient $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{p(\boldsymbol{\varepsilon})}\left[f\left(g_{\boldsymbol{\theta}}(\boldsymbol{\varepsilon}, \boldsymbol{x})\right)\right]$ analogously to that in (2.26). To be perfectly clear, the authors of [] proposed an easy exercise. Take the univariate Gaussian case $p(z|x) = \mathcal{N}\left(z; \mu, \sigma^2\right)$. In such a case, proper reparametrization takes the shape of

$$z = \mu + \sigma\varepsilon, \tag{2.31}$$

where $\varepsilon \sim \mathcal{N}\left(0, 1\right)$ and, therefore, the expectation

$$\mathbb{E}_{\mathcal{N}(z;\mu,\sigma^2)}\left[f(z)\right] = \mathbb{E}_{\mathcal{N}(\varepsilon;0,1)}\left[f(\mu + \sigma\varepsilon)\right] \approx \frac{1}{M}\sum_{j=1}^{M} f\left(\mu + \sigma\varepsilon_j\right). \tag{2.32}$$

Note that this is nothing more than a transformation of a random variable. (prepsat: And this is exactly the problem with optimizing ELBO (2.22)). We need to rewrite the expectation $\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)}$ so that the Monte Carlo estimate of the expected value is differentiable with respect to $\boldsymbol{\phi}$.

2.3.1.5 Variational autoencoder

So far we have only dealt with VAE in general. In this section, we put everything together and specify the individual parts of the ELBO (2.22). Let the probabilistic encoder be a multivariate Gaussian with a diagonal covariance matrix

$$q_{\boldsymbol{\phi}}\left(\boldsymbol{z}|\boldsymbol{x}\right) = \mathcal{N}\left(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}, \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}\mathbb{I}_P\right) \tag{2.33}$$

and let the probabilistic decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ takes the form depending on the type of given data and model. This is typically either multivariate Gaussian or Bernoulli. Finally, let the prior $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ be the centered izotropic multivariate Gaussian, i.e.

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}) = p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \mathbb{I}_P), \tag{2.34}$$

where the generative parameters $\boldsymbol{\theta}$ are omitted, since the chosen prior distribution lacks parameters. When using (2.33), $\boldsymbol{\mu}_{\boldsymbol{\phi}}$ and $\boldsymbol{\sigma}_{\boldsymbol{\phi}}$ are non-linear functions of the data point $\boldsymbol{x}$ and the variational parameters $\boldsymbol{\phi}$. For further simplification of the notation, the index $\boldsymbol{\phi}$ will be omitted. This setting actually allows us to take the reparameterization trick in a form similar to that of Equation (2.31), which means that

$$\boldsymbol{z}_{i,j} = \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \odot \boldsymbol{\varepsilon}_j, \tag{2.35}$$

where the symbol $\odot$ denotes the Hadamard product, i.e. the element product and the auxiliary variable $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \mathbb{I}_P)$. Another major fact is that the KL distance from a Gaussian distribution to a Gaussian distribution has an analytical solution (for a full derivation, see appendix A.1), so $D_{\text{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{z})\right)$ can be expressed in closed form:

$$\begin{aligned} D_{\text{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{z})\right) &= D_{\text{KL}}\left(\mathcal{N}\left(\boldsymbol{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mathbb{I}_P\right) \| \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \mathbb{I}_P)\right) \\ &= \frac{1}{2} \sum_{j=1}^{P} \left(-1 - \log \sigma_j^2 + \mu_j^2 + \sigma_j^2\right). \end{aligned} \tag{2.36}$$

Now all that is left is to plug everything into equation (2.22), which leads to the final form for optimization

$$\widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\phi}} = \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\operatorname{argmin}} - \sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|\boldsymbol{z})\right] - D_{\text{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{z})\right) \tag{2.37}$$

$$= \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\operatorname{argmin}} - \sum_{i=1}^{N} \left(\frac{1}{P} \sum_{j=1}^{P} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|\boldsymbol{z}_{i,j}) + \frac{1}{2} \sum_{j=1}^{P} \left(1 + \log \sigma_{i,j}^2 - \mu_{i,j}^2 - \sigma_{i,j}^2\right) \right). \tag{2.38}$$

2.3.1.6 Toy problem

The goal of this example is to verify that VAE can be utilized to generate new data points. Assume that we have a data set of 2D i.i.d. observations $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ generated from the unknown distribution and that we would like to sample new observations from this distribution. We should see similar patterns of the true and estimated samples. Let the probabilistic decoder be in Gaussian form 2.13, but with the identity matrix as a covariance matrix; therefore,

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{x}; f_{\boldsymbol{\theta}}(\boldsymbol{z}), \mathbb{I}_D\right). \tag{2.39}$$

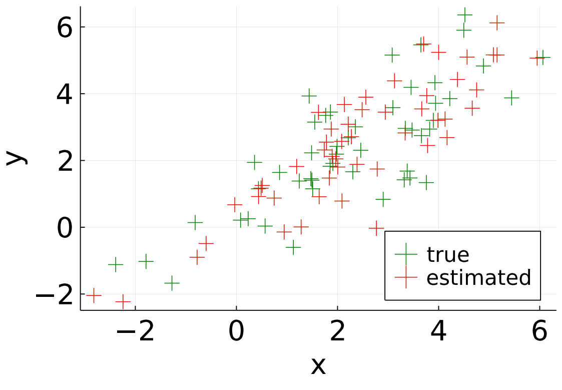


Figure 2.5: True and estimated samples using VAE.

This allows us to once again rewrite Equation (2.38) in concrete optimizable form

$$\widehat{\boldsymbol{\theta}},\widehat{\boldsymbol{\phi}} = \operatorname*{argmin}_{\boldsymbol{\theta},\boldsymbol{\phi}} -\sum_{i=1}^{N}\left(\frac{1}{P}\sum_{j=1}^{P}\log\mathcal{N}\left(\boldsymbol{x}_i; f_{\boldsymbol{\theta}}\left(\boldsymbol{z}_{i,j}\right),\mathbb{I}_D\right)+\frac{1}{2}\sum_{j=1}^{P}\left(1+\log\sigma_{i,j}^2-\mu_{i,j}^2-\sigma_{i,j}^2\right)\right) \tag{2.40}$$

$$= \operatorname*{argmin}_{\boldsymbol{\theta},\boldsymbol{\phi}} -\sum_{i=1}^{N}\left(\frac{1}{P}\sum_{j=1}^{P}\left(\boldsymbol{x}_i - f\left(\boldsymbol{z}_{i,j}\right)\right)^{\top}\left(\boldsymbol{x}_i - f\left(\boldsymbol{z}_{i,j}\right)\right)+\frac{1}{2}\sum_{j=1}^{P}\left(1+\log\sigma_{i,j}^2-\mu_{i,j}^2-\sigma_{i,j}^2\right)\right), \tag{2.41}$$

where $f_{\boldsymbol{\theta}}, \boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are represented via the neural network (NN). These are in fact dense layers, i.e., the NN layer, where neurons are connected to every neuron of its preceding layer. Clearly, new data points $\widehat{\boldsymbol{x}}$ (reconstructed input) are then sampled using $\widehat{\boldsymbol{x}} = \widehat{f_{\boldsymbol{\theta}}}\left(\boldsymbol{z}\right)$. For the training of parameters $\widehat{\boldsymbol{\theta}}$ and $\widehat{\boldsymbol{\phi}}$, the ADAM optimization algorithm [] is used. It is sufficient to initialize the optimization with standard default values, the learning rate $\alpha = 0.001$, the decay rates $\beta_1 = 0.9$, $\beta_2 = 0.999$, and finally $\epsilon = 10^{-8}$. The results are shown in Figure 2.5, where the true data and the estimated data are depicted. The estimated distribution is very close to the true distribution, as the pattern of the samples is indistinguishable. This experiment revealed that VAE is a good way of successfully generating new data points.

2.3.2 Semi-Supervised Variational Autoencoder

Semi-Supervised Variational Autoencoder (SSVAE) copes with input–output pair samples $\mathcal{D}$ as was defined in (1.4). Each pair sample $\left(\boldsymbol{x}_i, y_i\right)$ has its corresponding latent variable $\boldsymbol{z}_i$. The authors of [] propose a probabilistic model that describes the data as generated by a latent class variable y in addition to a continuous latent variable $\boldsymbol{z}$. However, only a subset of observations $\boldsymbol{x}$ have the corresponding class labels. Observe that these latent variables are marginally independent. The empirical distributions of the labeled and unlabeled subsets are denoted by $\tilde{p}_l\left(\boldsymbol{x}, y\right)$ and $\tilde{p}_u\left(\boldsymbol{x}\right)$, respectively. As with standard VAE, the data is generated by some random process which can be described as follows

$$p\left(y\right) = \operatorname{Cat}\left(y;\boldsymbol{\pi}\right), \qquad p\left(\boldsymbol{z}\right) = \mathcal{N}\left(\boldsymbol{z};\boldsymbol{0},\mathbb{I}_P\right), \qquad p_{\boldsymbol{\theta}}\left(\boldsymbol{x}|y,\boldsymbol{z}\right) = h_{\boldsymbol{\theta}}\left(\boldsymbol{x};y,\boldsymbol{z}\right). \tag{2.42}$$

The symbol $\operatorname{Cat}\left(y;\boldsymbol{\pi}\right)$ denotes the multinomial distribution with probability vector $\boldsymbol{\pi}$, and $h_{\boldsymbol{\theta}}$ is a suitable likelihood function depending on the type of given data parameterized by a non–linear transformation of the latent variables. The predictions of missing labels are obtained from the inferred posterior distribution $p_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)$. The standard VAE employs an inference model $q_{\boldsymbol{\phi}}\left(\boldsymbol{z}|\boldsymbol{x}\right)$, however, in the case of a semi–supervised learning, this model should also contain class labels. Ideally, for a given observation $\boldsymbol{x}$, the model should predict the class label y and, in addition, be able to construct the latent space $\boldsymbol{z}$ for given $\boldsymbol{x}$ and y. Under these circumstances, the model is introduced in the factorized form $q_{\boldsymbol{\phi}}\left(\boldsymbol{z}, y|\boldsymbol{x}\right) = q_{\boldsymbol{\phi}}\left(\boldsymbol{z}|y,\boldsymbol{x}\right) q_{\boldsymbol{\phi}}\left(y|\boldsymbol{x}\right)$. This factorization is further specified as

$$q_{\boldsymbol{\phi}}\left(\boldsymbol{z}|y,\boldsymbol{x}\right) = \mathcal{N}\left(\boldsymbol{z};\boldsymbol{\mu}_{\boldsymbol{\phi}}\left(y,\boldsymbol{x}\right),\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}\left(\boldsymbol{x}\right)\right), \qquad q_{\boldsymbol{\phi}}\left(y|\boldsymbol{x}\right) = \operatorname{Cat}\left(y;\boldsymbol{\pi}_{\boldsymbol{\phi}}\left(\boldsymbol{x}\right)\right), \tag{2.43}$$

where $\boldsymbol{\mu}_{\boldsymbol{\phi}}, \boldsymbol{\sigma}_{\boldsymbol{\phi}}$ and $\boldsymbol{\pi}_{\boldsymbol{\phi}}$ are represented as NNs. For VAE, we derived the variational lower bound

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geqslant \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] - D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{z})\right) = L\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}\right), \tag{2.44}$$

from which we now derive a bound for SSVAE. This derivation consists of two steps. First, consider observation $\boldsymbol{x}$ that has its class label y. The variational lower bound is then easily extended as

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, y) &\geqslant \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},y)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}, y) + \log p_{\boldsymbol{\theta}}(y)\right] - D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, y) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{z})\right) && (2.45)\\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},y)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}, y) + \log p_{\boldsymbol{\theta}}(y) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, y) + \log p_{\boldsymbol{\theta}}(\boldsymbol{z})\right] && (2.46)\\ &= J\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}, y\right). && (2.47)\end{aligned}$$

In the case of observation $\boldsymbol{x}$ lacking its class label y, it is treated as another latent variable over which posterior inference is performed. We get

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &\geqslant \mathbb{E}_{q_{\boldsymbol{\phi}}(y,\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}, y) + \log p_{\boldsymbol{\theta}}(y)\right] - D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(y, \boldsymbol{z}|\boldsymbol{x}) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{z})\right) && (2.48)\\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(y,\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}, y) + \log p_{\boldsymbol{\theta}}(y) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, y) + p_{\boldsymbol{\theta}}(\boldsymbol{z}) + \log q_{\boldsymbol{\phi}}(y|\boldsymbol{x})\right] && (2.49)\\ &= \sum_{y} q_{\boldsymbol{\phi}}(y|\boldsymbol{x})\, J\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}, y\right) + H\left(q_{\boldsymbol{\phi}}(y|\boldsymbol{x})\right) = U\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}\right), && (2.50)\end{aligned}$$

where $H\left(q_{\boldsymbol{\phi}}(y|\boldsymbol{x})\right)$ indicates the entropy of $q_{\boldsymbol{\phi}}(y|\boldsymbol{x})$. To include the whole data set in the bound, we can write

$$\tilde{J}\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) = \sum_{(\boldsymbol{x},y)\sim\tilde{p}_l(\boldsymbol{x},y)} J\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}, y\right) + \sum_{\boldsymbol{x}\sim\tilde{p}_u(\boldsymbol{x})} U\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}\right). \tag{2.51}$$

This objective lacks the predictive distribution of the label $q_{\boldsymbol{\phi}}(y|\boldsymbol{x})$ in its first expression. Since the purpose of this distribution is to use it as a classifier, we need to ensure that its parameters are learned in all cases. Currently, the objective (2.51) would completely lack $q_{\boldsymbol{\phi}}(y|\boldsymbol{x})$ if all data were labeled. To fix this problem, it is suggested to add the classification loss to (2.51), which yields

$$\tilde{J}^{\beta}\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) = \tilde{J}\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right) + \beta \cdot \mathbb{E}_{\tilde{p}_l(\boldsymbol{x},y)}\left[-\log q_{\boldsymbol{\phi}}(y|\boldsymbol{x})\right]. \tag{2.52}$$

This gives the hybrid combination of generative and purely discriminative modeling. The hyper–parameter β weighs the discriminative counterpart with a common value of $\beta = 0.1\tilde{N}$, where $\tilde{N}$ denotes the size of the supervised data set. Following Equation (2.38), optimization can be performed jointly

$$\widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\phi}} = \operatorname*{argmin}_{\boldsymbol{\theta},\boldsymbol{\phi}} \tilde{J}^{\beta}\left(\boldsymbol{\theta}, \boldsymbol{\phi}\right). \tag{2.53}$$

2.3.2.1 Toy problem

In this example, the ability to generate new samples will be tested in a similar way to the standard VAE, but also the ability to predict class labels. In addition, the current example is completely supervised, so all data points have the corresponding class label. Such precondition means that the bound $U\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}\right)$ does not participate in the objective (2.51) at all.

2.3.3 Noise–Contrastive Estimation

Suppose one has to estimate a model that is specified by an non-normalized probability density function $q^0_{\boldsymbol{\theta}}(\boldsymbol{x})$. In such a case, one can utilize noise–contrastive estimation (NCE). The first step is to introduce another parameter c among the estimated parameters $\boldsymbol{\theta}$. For clarity, the symbol $\boldsymbol{\theta}^\star = \left\{\boldsymbol{\theta}, c\right\}$ is introduced for the set of estimated parameters, including c. Using this notation, we can write the following equality

$$\log q_{\boldsymbol{\theta}^\star}(\boldsymbol{x}) = \log q^0_{\boldsymbol{\theta}^\star}(\boldsymbol{x}) + c, \tag{2.54}$$

which means that the newly introduced parameter c is an estimate of the negative logarithm of the normalization constant $Z(\boldsymbol{\theta})$ (1.2). As the name suggests, we use noise to estimate. By our convention, let $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ be the observations and $\Xi = \{\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \ldots, \boldsymbol{\varepsilon}_N\}$ be the artificially generated noise data with known distribution $\psi(\boldsymbol{\varepsilon})$. The estimate $\widehat{\boldsymbol{\theta}^\star}$ is then defined as

$$\widehat{\boldsymbol{\theta}^\star} = \operatorname*{argmax}_{\boldsymbol{\theta}^\star} \mathcal{L}^{\mathrm{NC}}\left(\boldsymbol{\theta}^\star\right) \tag{2.55}$$

$$= \operatorname*{argmax}_{\boldsymbol{\theta}^\star} \frac{1}{2N} \sum_{i=1}^{N} \log S_{\boldsymbol{\theta}^\star}(\boldsymbol{x}_i) + \log\left(1 - S_{\boldsymbol{\theta}^\star}(\boldsymbol{\varepsilon}_i)\right) \tag{2.56}$$

$$= \operatorname*{argmin}_{\boldsymbol{\theta}^\star} -\frac{1}{2N} \sum_{i=1}^{N} \log S_{\boldsymbol{\theta}^\star}(\boldsymbol{x}_i) + \log\left(1 - S_{\boldsymbol{\theta}^\star}(\boldsymbol{\varepsilon}_i)\right) \tag{2.57}$$

where $S_{\boldsymbol{\theta}^\star}$ stands for a logistic function,

$$S_{\boldsymbol{\theta}^\star}(\boldsymbol{x}) = \frac{1}{1 + \exp\left(-G_{\boldsymbol{\theta}^\star}(\boldsymbol{x})\right)} \tag{2.58}$$

and finally, the function $G_{\boldsymbol{\theta}^\star}$ represents the difference of the log-likelihoods of $q_{\boldsymbol{\theta}^\star}$ and ψ, hence

$$G_{\boldsymbol{\theta}^\star}(\boldsymbol{x}) = \log q_{\boldsymbol{\theta}^\star}(\boldsymbol{x}) - \log \psi(\boldsymbol{x}). \tag{2.59}$$

It may be noted that equation (2.56) also appears in SL tasks and is called binary CE loss. It is actually a special case of CE itself. Thus, it is used for the classification of two classes. This gives an intuitive insight into how noise–contrastive estimation really works. When data and noise are compared, the model is learned, so this method can be called learning by comparison. To make the connection with SL more explicit, denote $U = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{2N}\}$ the union of two sets $\boldsymbol{X}$ and Ξ. Then each data point $\boldsymbol{u}_i$ is assigned a binary class label y_i, where $y_i = 1$ if $\boldsymbol{u}_i \in \boldsymbol{X}$ and $y_i = 0$ if $\boldsymbol{u}_i \in \Xi$. The aim is to estimate the posterior probabilities of the classes given the data $\boldsymbol{u}_i$. To do this, one needs the class–conditional PDFs that are given by

$$p\left(\boldsymbol{u} | y = 1\right) = q_{\boldsymbol{\theta}^\star}(\boldsymbol{u}) \qquad p\left(\boldsymbol{u} | y = 0\right) = \psi(\boldsymbol{u}). \tag{2.60}$$

Class labels are equally likely, so that $\Pr\left(y=1\right) = \Pr\left(y=0\right) = \frac{1}{2}$ and the posteriors are determined as follows

$$\Pr\left(y=1|\boldsymbol{u}\right) = \frac{q_{\boldsymbol{\theta}^{\star}}\left(\boldsymbol{u}\right)}{q_{\boldsymbol{\theta}^{\star}}\left(\boldsymbol{u}\right)+\psi\left(\boldsymbol{u}\right)} = S_{\boldsymbol{\theta}^{\star}}\left(\boldsymbol{u}\right), \tag{2.61}$$

$$\Pr\left(y=0|\boldsymbol{u}\right) = 1 - S_{\boldsymbol{\theta}^{\star}}\left(\boldsymbol{u}\right). \tag{2.62}$$

The class labels y_i are Bernoulli–distributed so that for the log–likelihood of Bernoulli we get

$$\mathcal{L}^{\mathrm{NC}}\left(\boldsymbol{\theta}\right) = \sum_{i=1}^{2N} y_i \log \Pr\left(y=1|\boldsymbol{u}_i\right) + \left(1-y_i\right)\log\Pr\left(y=0|\boldsymbol{u}_i\right) \tag{2.63}$$

$$= \sum_{i=1}^{N} \log S_{\boldsymbol{\theta}^{\star}}\left(\boldsymbol{x}_i\right) + \log\left(1 - S_{\boldsymbol{\theta}^{\star}}\left(\boldsymbol{\epsilon}_i\right)\right), \tag{2.64}$$

which is the equation (up to extrinsic factor $\frac{1}{2N}$) that is optimized in (2.56) or (2.57).

2.3.3.1 Choice of the contrastive noise PDF

The noise distribution $\psi\left(\boldsymbol{\varepsilon}\right)$ can be considered as a design parameter. But this choice is not completely arbitrary, because in practice the noise distribution should meet certain conditions. These are:

1. It is easy to sample from, because NCE approach relies on artificially generated noise data $\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \ldots, \boldsymbol{\varepsilon}_N$.

2. In order to smoothly evaluate (2.59), closed form for $\log\psi\left(.\right)$ is requisite.

3. It leads to a small mean squared error $\mathbb{E}\left[\left(\widehat{\boldsymbol{\theta}^{\star}} - \boldsymbol{\theta}^{\star}\right)^2\right]$.

The authors of [] suggest using a Gaussian or uniform distribution, eventually a Gaussian mixture.

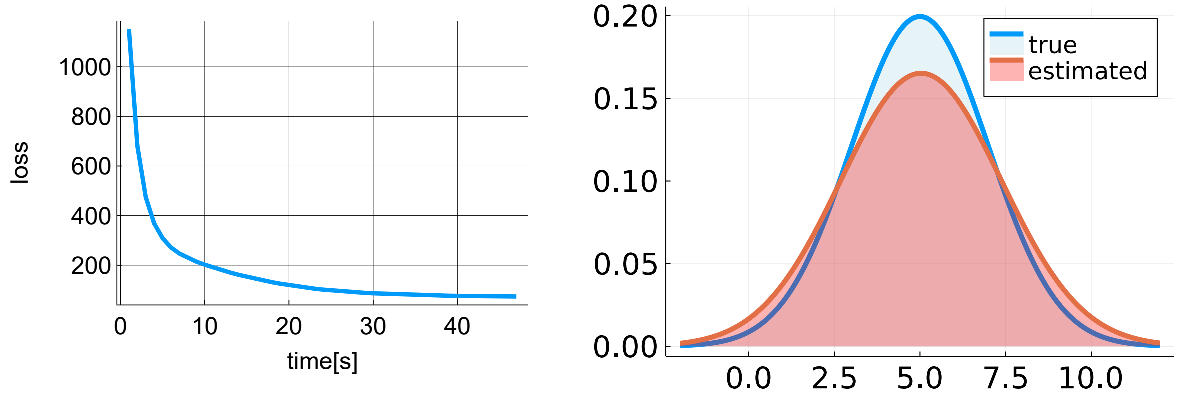
Example 2.1 (One–dimensional Gaussian distribution).

To test this approach, we performed a simple experiment. There are a total of $N = 100$ i.i.d. and one-dimensional observations $x_1, x_2, \ldots, x_N$ from an unknown distribution that is assumed to be non–normalized and Gaussian. Therefore, it is of the form

$$q_{\boldsymbol{\theta}^{\star}}\left(x\right) = \exp\left(-\frac{1}{2}\cdot\frac{\left(x-\mu\right)^2}{\sigma^2} + c\right), \tag{2.65}$$

where $\boldsymbol{\theta}^{\star} = \left\{\mu, \sigma^2, c\right\}$. Next, we artificially generate noise data $e_1, e_2, \ldots, e_N$, which is again easier to do using a Gaussian distribution. This means that it can be chosen, for example,

$$\psi\left(e\right) = \frac{1}{\sqrt{2\pi 10}}\exp\left(-\frac{1}{2}\cdot\frac{e^2}{10}\right). \tag{2.66}$$

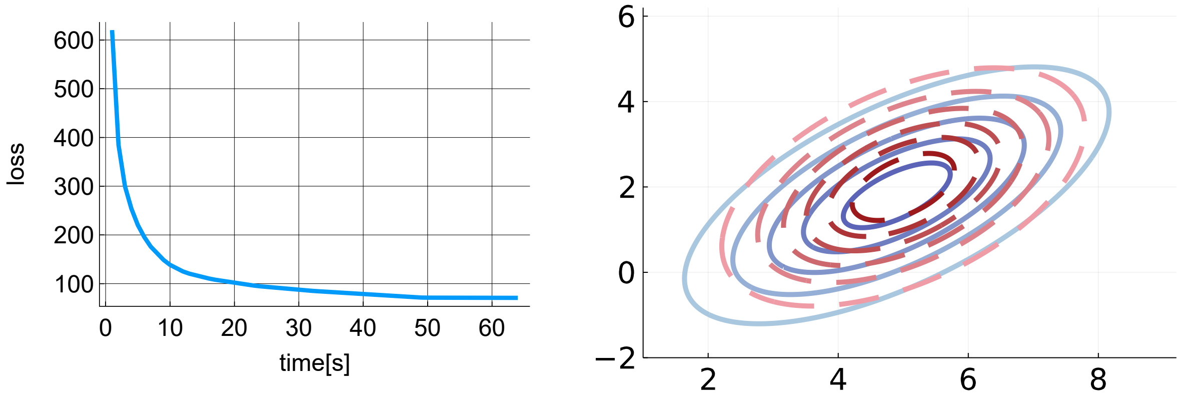


(a) Loss function minimizing.

(b) Comparison of true (blue) and estimated (red) PDF.

Figure 2.6: Results of the NCE experiment for one–dimensional Gaussian case.

We choose the noise PDF intentionally so widely spread from its mean value because these two PDFs, i.e. (2.65) and (2.66), should at least partially overlap. At this point, we have all the components available and it is possible to construct a function $-\mathcal{L}^{\mathrm{NC}}\left(\boldsymbol{\theta}^{\star}\right)$ that is minimized by using the ADAM optimization algorithm []. The following figure shows the training process and the comparison between the estimated distribution and the true one. As can be seen in Figure 2.6, this approach works quite well and for more observations, the results would be even better. In addition, the minimization of $-\mathcal{L}^{\mathrm{NC}}\left(\boldsymbol{\theta}^{\star}\right)$ is very fast.



(a) Loss function minimizing.

(b) Comparison of true (blue) and estimated (red) PDF.

Figure 2.7: Results of the NCE experiment for two–dimensional Gaussian case.

Example 2.2 (Two–dimensional Gaussian distribution). The one–dimensional case may seem too simple, and therefore an example with a two-dimensional Gaussian distribution was performed. The experimental setup remains nearly the same; only the dimensionality of the problem differs.

Recall that the non–normalized multivariate Gaussian distribution in $\mathbb{R}^2$ can be written as

$$q_{\boldsymbol{\theta}^\star}(\boldsymbol{x}) = \exp\left(-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)^\top \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)+c\right), \tag{2.67}$$

where $\boldsymbol{\mu} \in \mathbb{R}^2$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{2\times 2}$ is a symmetric and positive semidefinite covariance matrix. As the noise PDF is chosen $\psi(\boldsymbol{e}) = \mathcal{N}\left(\boldsymbol{e};\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1\right)$, where $\boldsymbol{\mu}_1 = (2,2)^\top$ and $\boldsymbol{\Sigma}_1 = 10\cdot\mathbb{I}_2$. Figure 2.7 shows the results in a similar vein to the previous case.

Chapter 3

Hybrid Generative and Discriminative models

In the previous chapter, we have introduced the basics of discriminative and generative modeling. An interesting and key approach is SSVAE, which combines both types of modeling. This chapter attempts to do the same, but from a slightly different perspective.

3.1 Energy-Based Models

In the first part of this chapter, we review the theory of energy-based models (EBM). It is motivated by statistical physics and aims at PDF estimation. Given a large data set, we want to estimate the PDF over the entire data space. Assuming that we are modeling images from, for example, the CIFAR10 data set [], the goal would be to estimate a PDF over all possible images of size 32 × 32 × 3, where those images have a high likelihood of looking realistic and are one of the CIFAR classes. Nowadays, images are extremely high-dimensional and that is why simple methods such as interpolation between images fail [].

The fundamental idea of EBMs is to transform any function that predicts values larger than zero into a PDF by dividing by its volume (normalization constant). This implies that the probability densities $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{R}^D$ in EBM are assumed to be expressed in the form

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{\exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right)}{Z(\boldsymbol{\theta})}, \tag{3.1}$$

where $Z(\boldsymbol{\theta})$ is a normalization constant and $E_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}$ is called the energy function with parameters $\boldsymbol{\theta}$, which maps each data point $\boldsymbol{x}$ to a scalar. In front of the objective $E_{\boldsymbol{\theta}}(\boldsymbol{x})$ there is a negative sign because we want data points with high likelihood to have low energy, while data points with low likelihood should have high energy. The significant advantage of the density formulation of EBM is that there are a relatively large number of ways to choose energy $E_{\boldsymbol{\theta}}$. The exponential function captures major variations in probability, and log–likelihood is a natural scale to work with. Unfortunately, we cannot compute $Z(\boldsymbol{\theta})$ for the vast majority of them (not even numerically, as computing time scales exponentially in the number of dimensions of $\boldsymbol{x}$), and we have to rely on the Monte Carlo estimate. The idea here is to take the gradient with respect to $\boldsymbol{\theta}$ from the log–likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$, which decomposes as the sum of two terms

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\left(\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}) + \nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{\theta})\right). \tag{3.2}$$

The second term can be rewritten as expectation $\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x})}\left[-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x})\right]$, which yields the following (for the full derivation, see Appendix A.2)

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x})}\left[-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x})\right] - \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}). \tag{3.3}$$

Drawing samples from $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is far from trivial, thus we exploit a well-established Monte Carlo method called stochastic gradient Langevin dynamics (SGLD). For any continuous $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ we can compute the score function

$$\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} E_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{3.4}$$

and generate samples using the following stochastic process

$$\boldsymbol{x}_0 \sim \psi(\boldsymbol{x}), \quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta \nabla_{\boldsymbol{x}} E_{\boldsymbol{\theta}}(\boldsymbol{x}) + \sqrt{2\eta}\,\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbb{I}_D), \quad t \in \{0, 1, \ldots, T-1\} \tag{3.5}$$

where $\psi(\boldsymbol{x})$ is the prior distribution (that is easy to sample from) used to generate the initial sample $\boldsymbol{x}_0$ and $\eta \in \mathbb{R}$ is a step size. This algorithm guarantees that for $\eta \to 0$ and $T \to \infty$, $\boldsymbol{x}_T$ is distributed as $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ [].

Note that for two different data points $\boldsymbol{x}$ and $\boldsymbol{x}^{\star}$, computing $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}^{\star})$ requires $Z(\boldsymbol{\theta})$, however, the ratio $\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x}^{\star})}$ does not involve $Z(\boldsymbol{\theta})$ and one can easily check which data point is more likely.

3.1.1 Joint Energy Models

Recall, that in Section 2.2 we described the Softmax function, which is used to model the true data distribution. It is defined as

$$q_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \frac{\exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]\right)}{\sum_{y \in \mathcal{C}} \exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]\right)}. \tag{3.6}$$

Crucial observation is made by the authors in [15], where they show that supervised learning classifiers are secretly EBMs on $p_{\boldsymbol{\theta}}(\boldsymbol{x}, y)$, i.e., the logits $f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]$ in (2.2) can be seen as defining an EBM and it can be expressed as

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}, y) = \frac{\exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]\right)}{Z(\boldsymbol{\theta})}. \tag{3.7}$$

This objective is called a joint energy model (JEM), and it is obvious that $f_{\boldsymbol{\theta}}(\boldsymbol{x})[y] = -E_{\boldsymbol{\theta}}(\boldsymbol{x}, y)$. The desirable model $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ can be obtained by marginalizing $p(\boldsymbol{x}, y)$ over y, resulting in the following density

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{\sum_{y \in \mathcal{C}} \exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]\right)}{Z(\boldsymbol{\theta})}, \tag{3.8}$$

where the energy is given by $E_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\log \sum_{y \in \mathcal{C}} \exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]\right)$. A very useful property appears when computing $p_{\boldsymbol{\theta}}(y|\boldsymbol{x})$, because we can take advantage of the definition of a conditional distribution $p_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x},y)}{p_{\boldsymbol{\theta}}(\boldsymbol{x})}$, resulting in

$$p_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right) = q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right) = \frac{\exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]\right)}{\sum_{y \in \mathcal{C}} \exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]\right)}. \tag{3.9}$$

Note that the normalization constant $Z(\boldsymbol{\theta})$ canceled out and we ended up with the same function, which was introduced in (2.2).

3.2 Contrastive learning

Contrastive learning [10, 11] is an ML technique used to learn the so-called general features of a data set by teaching the model which data points are similar or different. All this happens without labels; therefore, contrastive learning is often called the *self–supervised* technique of ML. Given $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, in contrastive learning problems, it is very common to optimize an objective often called contrastive loss, which can be written in the form as follows

$$\mathrm{CL}(\boldsymbol{\theta}) = -\mathbb{E}_{\tilde{p}(\boldsymbol{x})}\left[\log \frac{\exp\left(m_{\boldsymbol{\theta}}(\boldsymbol{x}) \cdot m_{\boldsymbol{\theta}}(\boldsymbol{x}')\right)}{\sum_{i=1}^{M} \exp\left(m_{\boldsymbol{\theta}}(\boldsymbol{x}) \cdot m_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)}\right], \tag{3.10}$$

where $M < N$ denotes the number of normalization samples. The function $m_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}^H$ maps each data point to a representation space of dimension H, while $\boldsymbol{x}$ and $\boldsymbol{x}'$ are two different augmented views of the same data point. If $\boldsymbol{x}$ is an image, then an augmented view of $\boldsymbol{x}$ can be obtained, for example, by rotating or colorizing that image. Note that the inner product between two vectors can be replaced with any distance metric, for instance, the Euclidean distance.

This objective tries to maximally distinguish an input $\boldsymbol{x}_i$ from an alternative input $\boldsymbol{x}'_i$. In other words, (3.10) reduces the distance between the representations of different augmented views of the same image $\boldsymbol{x}, \boldsymbol{x}'$ (positive pairs) and increases the distance between the representations of augmented views of different images (negative pairs). This means that the model should be able to distinguish between different types of image without even knowing what these images really are.

3.3 Hybrid Dicriminative and Generative Models

In this section, we will put everything together and present an approach to combine both types of models. The authors of article [12] proposed a solution, however, the rationale for this objective originates from [6], where the authors show that hybrid models can outperform their purely generative or purely discriminative counterparts.

To achieve this goal, a hybrid model consists of a discriminative conditional and a generative conditional, and it is trained by minimizing the negative sum of both conditional log-likelihoods, concretly

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{\tilde{p}(\boldsymbol{x},y)}\left[\log q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)+\log q_{\boldsymbol{\theta}}\left(\boldsymbol{x}|y\right)\right], \tag{3.11}$$

where the first term is a standard Softmax NN classifier (as mentioned in Equation (2.2) or (3.6)), while the second term differs from Softmax in its denominator, so that

$$q_{\boldsymbol{\theta}}\left(\boldsymbol{x}|y\right)=\frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{N}\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)[y]\right)}. \tag{3.12}$$

The objective $q_{\boldsymbol{\theta}}\left(\boldsymbol{x}|y\right)$ can cause serious problems with the unknown normalization constant $\sum_{i=1}^{N}\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)[y]\right)$, which is often intractable, as we have already mentioned in previous considerations. The authors of [] recommend resolving this obstacle using an approximation via contrastive loss

$$\mathbb{E}_{\tilde{p}(\boldsymbol{x},y)}\left[\log q_{\boldsymbol{\theta}}\left(\boldsymbol{x}|y\right)\right]=\mathbb{E}_{\tilde{p}(\boldsymbol{x},y)}\left[\log\frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{N}\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)[y]\right)}\right] \tag{3.13}$$

$$\approx\mathbb{E}_{\tilde{p}(\boldsymbol{x},y)}\left[\log\frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{M}\exp\left(f_{\theta}\left(\boldsymbol{x}_i\right)[y]\right)}\right], \tag{3.14}$$

where $M<N$ denotes the number of normalization samples. This loss is related to (3.10), which was mentioned in the previous section, but there are few distinctions to discuss. We use labels in this formulation as we assume a supervised setting and, more importantly, we do not use any mapping $m_{\boldsymbol{\theta}}$ or augmented views. The main contribution of (3.10) is the proposed approximation in the denominator. To have an adequate approximation, M must be sufficiently large, becoming exact in the limit $M\to N$. In practice, increasing M is not straightforward as it requires a larger memory. However, this does not apply to our experiments.

Now it is possible to substitute the approximation (3.13) in Equation (3.11), which results in a hybrid combination of supervised learning and constrastive learning in the form of

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{\tilde{p}(\boldsymbol{x},y)}\left[\alpha\log q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right)+(1-\alpha)\log q_{\boldsymbol{\theta}}\left(\boldsymbol{x}|y\right)\right] \tag{3.15}$$

$$\approx\min_{\boldsymbol{\theta}} -\mathbb{E}_{\tilde{p}(\boldsymbol{x},y)}\left[\alpha\log\frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{y\in\mathcal{C}}\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}+(1-\alpha)\log\frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{M}\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)[y]\right)}\right], \tag{3.16}$$

called the hybrid discriminative generative energy-based model (HDGM). The hyper–parameter α is a weight between $[0,1]$. It is obvious that in the case of $\alpha=1$, the objective reduces to the standard cross-entropy loss, while in $\alpha=0$, the objective is reduced to a case called *end-to-end supervised version of contrastive learning*. The choice of parameter α is a decision of the experiment designer; however, the authors of [12] evaluated many possible variants in their experiments and found that the choice of $\alpha=0.5$ produces the highest classification accuracy performance. Unfortunately, these experiments involved only image classification.

The HDGM (3.16) is absolutely crucial for us as we extend this approach to the multi-instance learning problem, but this is discussed in further sections.

3.4 Toy problem - Polynomial Regression

At first, we would like to try the hybrid discriminative and generative approach on a simple example before moving on to more difficult cases. The goal is to train a model of the form (3.16) that was derived in the previous section.

Assume that data $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$, where $x_i, y_i \in \mathbb{R}$, therefore, this is only a two-dimensional problem. According to the energy–based models 3.1, we know that for the joint distribution, it holds

$$p(x, y) = \frac{\exp\left(f_{\boldsymbol{\theta}}(x)[y]\right)}{Z(\boldsymbol{\theta})}, \tag{3.17}$$

where the model is given by

$$f_{\boldsymbol{\theta}}(x)[y] = -E_{\boldsymbol{\theta}}(x, y). \tag{3.18}$$

At this point, we should transform our problem into a polynomial regression. We must be aware of the discriminative term in Equation (3.11), because we do not want to classify, but we would like to find the best fit to the given data. For this reason, we replace that with the typical regression loss

$$S = S(\boldsymbol{\theta}) = \sum_{k=1}^{N} \left(y_k - \sum_{i=0}^{s-1} \theta_i x_k^i \right)^2. \tag{3.19}$$

Any joint probability distribution can be broken down into parts by the chain rule.

$$p(x, y) = p(y, x) = p(y|x) \cdot p(x), \tag{3.20}$$

Therefore, we need to find $p(y|x)$ and $p(x)$. From polynomial regression, we can obtain the conditional probability density

$$p(y|x, \boldsymbol{\theta}) = \mathcal{N}\left(y; \sum_{i=0}^{s-1} \theta_i x^i, \sigma^2\right), \tag{3.21}$$

where the symbol $\mathcal{N}(\cdot)$ denotes the probability density function of the Normal distribution and σ^2 is the known variance. In this case, we also need to determine the prior distribution of x. To keep this example simple, let the PDF takes the form

$$p(x|\tau) = \mathcal{N}\left(x; 0, \tau^2\right), \tag{3.22}$$

where the choice of parameter τ is based on the fact that we would like to have a non–informative prior, thus τ should be adequately high. If the value of τ is high, the data are spread very far from their expected value. Substituting equations (3.22) and (3.21) into (3.20) results

$$p(x, y) = \mathcal{N}\left(x; 0, \tau^2\right) \cdot \mathcal{N}\left(y; \sum_{i=0}^{s-1} \theta_i x^i, \sigma^2\right) = \frac{1}{2\pi\sigma\tau} \exp\left(-\frac{\left(y - \sum_{i=0}^{s-1} \theta_i x^i\right)^2}{2\sigma^2} - \frac{x^2}{2\tau^2} \right), \tag{3.23}$$

whereas our desirable model is given by

$$f_{\boldsymbol{\theta}}(x)[y] = -\frac{\left(y-\sum_{i=0}^{s-1}\theta_i x^i\right)^2}{2\sigma^2} - \frac{x^2}{2\tau^2}. \tag{3.24}$$

We can now substitute (3.24) and (3.19) in Equation (3.11), resulting in

$$\min_{\boldsymbol{\theta}}\left\{\alpha\, S(\boldsymbol{\theta}) - \mathbb{E}_{p_{\text{data}}(x,y)}\left[(1-\alpha)\log q_{\boldsymbol{\theta}}\left(x|y\right)\right]\right\} = \tag{3.25}$$

$$\min_{\boldsymbol{\theta}}\left\{\alpha\, S(\boldsymbol{\theta}) - \mathbb{E}_{p_{\text{data}}(x,y)}\left[(1-\alpha)\log\frac{\exp\left(f_{\boldsymbol{\theta}}(x)[y]\right)}{\sum_{i=1}^{N}\exp\left(f_{\boldsymbol{\theta}}(x_i)[y]\right)}\right]\right\} = \tag{3.26}$$

$$\min_{\boldsymbol{\theta}}\left\{\alpha\, S(\boldsymbol{\theta}) - \mathbb{E}_{p_{\text{data}}(x,y)}\left[(1-\alpha)\log\frac{\exp\left(-\frac{\left(y-\sum_{i=0}^{s-1}\theta_i x^i\right)^2}{2\sigma^2} - \frac{x^2}{2\tau^2}\right)}{\sum_{k=1}^{N}\exp\left(-\frac{\left(y-\sum_{i=0}^{s-1}\theta_i x_k^i\right)^2}{2\sigma^2} - \frac{x_k^2}{2\tau^2}\right)}\right]\right\}. \tag{3.27}$$

Finally, we simplify the generative term $\log q_{\boldsymbol{\theta}}\left(x|y\right)$ into

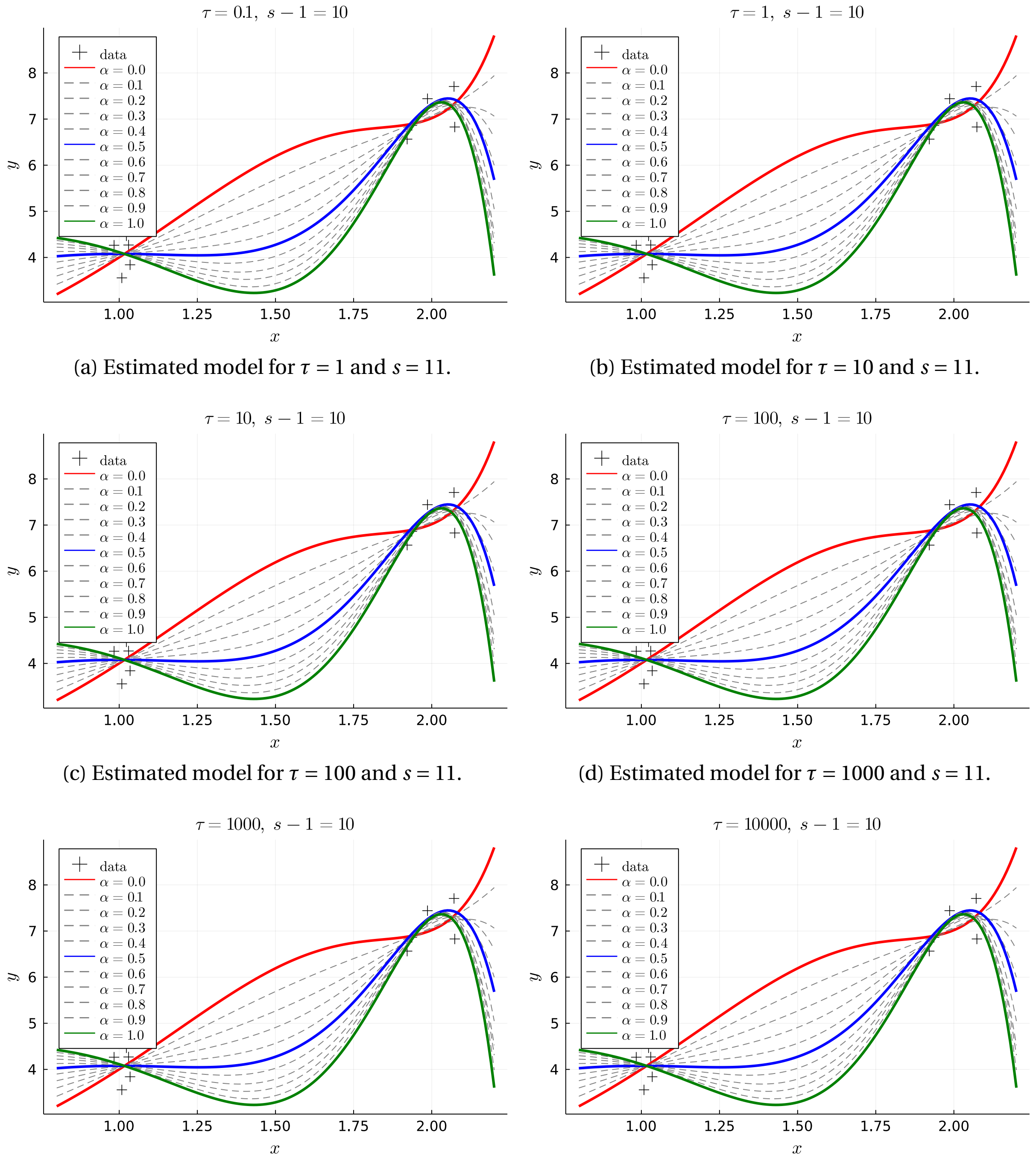
$$\log q_{\boldsymbol{\theta}}\left(x|y\right) = \left(-\frac{\left(y-\sum_{i=0}^{s-1}\theta_i x^i\right)^2}{2\sigma^2} - \frac{x^2}{2\tau^2}\right) - \log\sum_{k=1}^{N}\exp\left(-\frac{\left(y-\sum_{i=0}^{s-1}\theta_i x_k^i\right)^2}{2\sigma^2} - \frac{x_k^2}{2\tau^2}\right). \tag{3.28}$$

Note that for $\alpha = 1$ we get purely polynomial regression, and for $\alpha = 0$, the term $S(\boldsymbol{\theta})$ is not involved at all. Now, we have everything we need to carry out the experiment.

3.4.1 Experiment setup and results

We would like to test the sensitivity of this approach to the unknown parameter τ and the order of the polynomial $s-1$. In addition, we would like to observe how the estimated model behaves in relation to α.

We generate synthetic data, two clusters consisting of 5 data points each, then the model was fitted for different weights $\alpha \in \{0.0, 0.1, 0.2, \ldots, 1.0\}$, giving 11 different models in total. The models estimated for $\alpha \in \{0.0, 0.5, 1.0\}$ are highlighted because they are more important to us than the other models. At first, we trained the models mentioned for the fixed order of the polynomial, but for 6 different values of the parameter τ. The results obtained (Figure 3.1) for small τ barely vary from those for high τ, which is exactly what we hoped for, as the previous distribution should be non–informative (3.22). This is a very exciting discovery because there is no need to know much about the data distribution. Second, we trained our polynomial models for the fixed value of τ with 6 different values of the order of the polynomial. The goal of this part of the experiment is to observe how the contrastive part of Equation (3.25) affects the loss of polynomial regression for different values of $s-1$. As can be seen in Figure 3.2, for small $s-1$, such as $s-1=2$, the term $\log q_\theta\left(x|y\right)$ does not affect the polynomial regression too much. However, we get a considerable difference for higher orders of the polynomial. Furthermore, it seems that the curve $\alpha = 0$ prefers not to oscillate. This could also be very interesting because combining discriminative and generative models could result in better model predictions.



(a) Estimated model for $\tau = 1$ and $s = 11$.

(b) Estimated model for $\tau = 10$ and $s = 11$.

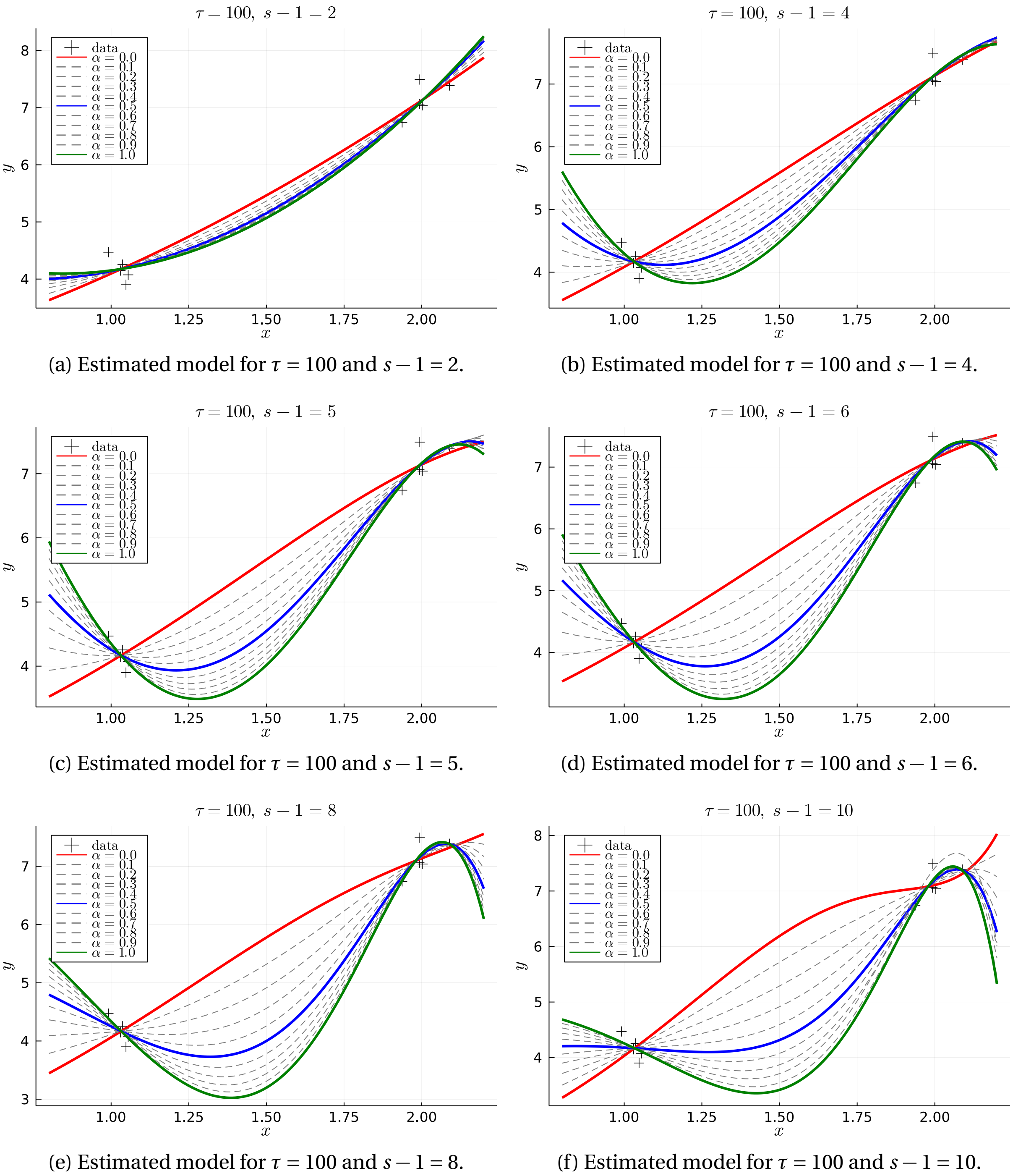
(c) Estimated model for $\tau = 100$ and $s = 11$.

(d) Estimated model for $\tau = 1000$ and $s = 11$.

(e) Estimated model for $\tau = 100000$ and $s = 11$.

(f) Estimated model for $\tau = 1000000$ and $s = 11$.

Figure 3.1: Sensitivity of the polynomial model to parameter τ with parameter $s-1$ held fixed for six different cases.



(a) Estimated model for $\tau = 100$ and $s-1=2$.

(b) Estimated model for $\tau = 100$ and $s-1=4$.

(c) Estimated model for $\tau = 100$ and $s-1=5$.

(d) Estimated model for $\tau = 100$ and $s-1=6$.

(e) Estimated model for $\tau = 100$ and $s-1=8$.

(f) Estimated model for $\tau = 100$ and $s-1=10$.

Figure 3.2: Sensitivity of the polynomial model to the order of polynomial $s-1$ for six different cases.

Chapter 4

Multiple Instance Learning

4.1 Fundamentals

The term multiple instance learning originates from [16] and in [2], the authors proposed the following nomenclature for MIL, which will be reviewed and will be used gladly in our work.

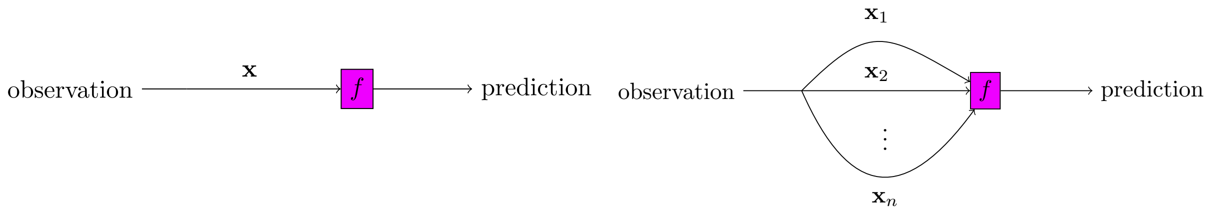
In standard machine learning problems, each sample is represented by a fixed vector $\boldsymbol{x}$ of observations, however, in multiple instance learning (MIL) it is dealt with samples which are represented by a set of vectors. These vectors are called *instances* and come from an instance space $\mathcal{X}$, for example $\mathbb{R}^D$. The sets of these instances are called *bags* and come from the bag space $\mathcal{B} = \mathcal{P}_F(\mathcal{X})$, where $\mathcal{P}_F(\mathcal{X})$ denotes all finite subsets of $\mathcal{X}$. With this in mind, we can easily write any bag as $b = \{\boldsymbol{x} \in \mathcal{X}\}_{\boldsymbol{x} \in b}$. Each bag b can be arbitrarily large or empty, thus the size of the bag is defined in the form $|b| \in \mathbb{N}_0$. There may exist intrinsic labeling of instances, but we are only interested in labeling at the bag levels. Bag labels come from a finite set $\mathcal{C}$, and what we want in MIL is to learn a predictor in the form $f_{\boldsymbol{\theta}} : \mathcal{B}(\mathcal{X}) \rightarrow \mathcal{C}$ that can also be rewritten in the form $f_{\boldsymbol{\theta}}(\{\boldsymbol{x}\}_{\boldsymbol{x} \in b})$. Unlike ML, where a predictor is learned in the form $f_{\boldsymbol{\theta}} : \mathbb{R}^D \rightarrow \mathcal{C}$. We consider the supervised setting in which each sample of the data set is assigned a label. We can denote the available data by the notation

$$\mathcal{D}^\star = \left\{ \left(b_i, y_i\right) \in \mathcal{B} \times \mathcal{C} \mid i \in \left\{1, 2, \ldots, |\mathcal{D}^\star|\right\} \right\}, \tag{4.1}$$

where $|\mathcal{D}^\star|$ apparently denotes the size of $\mathcal{D}^\star$. The difference between standard ML and MIL is visualized graphically in Figure 4.1.

4.2 Embedded-space paradigm

Very elegant solution to deal with samples on the level of bags provides an embedded–space paradigm[2]. This paradigm defines a vector space for the representation of bags and specifies a mapping from each bag $b \in \mathcal{B}$ to this space. Assume that the target vector space is $\mathbb{R}^D$, then the partial mapping $\phi_i \colon \mathcal{B} \rightarrow \mathbb{R}, \;\; i \in \{1, 2 \ldots, D\}$ is defined and overall embedding



(a) Standard machine learning

(b) Multiple instance learning

Figure 4.1: The difference between standard ML and MIL [2]. Standard ML is special case of MIL with $|b| = 1$.

$\boldsymbol{\phi} : \mathcal{B} \to \mathbb{R}^D$ is given by

$$\begin{aligned}\boldsymbol{\phi}(b) &= \left(\phi_1(b), \phi_2(b), \ldots, \phi_D(b)\right) && (4.2)\\ &= \left(\phi_1\left(\{\boldsymbol{x}\}_{\boldsymbol{x}\in b}\right), \phi_2\left(\{\boldsymbol{x}\}_{\boldsymbol{x}\in b}\right), \ldots, \phi_D\left(\{\boldsymbol{x}\}_{\boldsymbol{x}\in b}\right)\right). && (4.3)\end{aligned}$$

Mappings ϕ_i are instrumental in obtaining and aggregating the appropriate information on the level of instances. They can be defined by some instance transformation $k : \mathcal{X} \to \mathbb{R}^D$ and an aggregation function $g : \mathcal{P}_F\left(\mathbb{R}^D\right) \to \mathbb{R}^D$ of the form

$$\phi_i(b) = g\left(k\,\{\boldsymbol{x}\}_{\boldsymbol{x}\in b}\right). \tag{4.4}$$

On the resulting embedded representation of bag samples, any standard machine learning algorithm can be applied, that is, training a bag-level classifier $f^B_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathcal{C}$ using an adjusted dataset $\mathcal{D}^\star_{\text{ES}} = \left\{\left(\boldsymbol{\phi}(b_i), y_i\right) \in \mathbb{R}^D \times \mathcal{C} \;\mid\; i \in \left\{1, 2, \ldots, |\mathcal{D}^\star|\right\}\right\}$. The most widely used aggregation functions are minimum, maximum, or mean value

$$g\left(\left\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|b|}\right\}\right) = \begin{cases}\min\left\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|b|}\right\}\\ \max\left\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|b|}\right\}\\ \frac{1}{|b|}\sum_{\boldsymbol{x}\in b}\boldsymbol{x}\end{cases} \tag{4.5}$$

4.3 Training

Authors of [2] proposed a versatile unified framework called HMill (Hierarchical multi–instance learning library) for the definition and training of the models and even implemented this functionality in the *Julia* programming language. Furthermore, the framework was published as an open-source project entitled *Mill.jl* under the MIT license. The aim of this work is not to rigorously derive the MIL model, since it is fairly complicated and requires a considerable amount of work. Taking into account that we settled for the MIL model being a neural network (NN) that utilizes aggregation functions on the level of instances and refer to [2] for more details on the definition of the model and its composition. Learning of the MIL model

$f_{\boldsymbol{\theta}}$ is also supervised, a specific case called binary classification, and therefore achieved by minimizing standard cross–entropy

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)} \left[\log \frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)\left[y\right]\right)}{\sum_{i=1}^{C} \exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)\left[y_i\right]\right)} \right], \tag{4.6}$$

already mentioned in section 2. This is very important to us and we will take advantage of that in our experiments.

4.4 Cross–validation on MIL datasets

For the MIL testing, we have four datasets available, namely Musk1, Musk2, Tiger and Fox. These datasets are from the UCI database and specially modified for modeling with set data. All of them will be used to assess the performance of the MIL model.

4.4.1 Setup

In this experiment, data sets are 100 times randomly split into 2 sets in advance, train and test sets, with 80% of observations being in train set and 20% of observations belonging to the test set. For future simplification, let a number of random splits be denoted by r, and thus $r = 100$. We fit the model on train set, then we evaluate the prediction error on the train data by cross–entropy 2.3. The objective here is to plot the dependence of the prediction error on the complexity of the model. A smaller number of random splits r were tested, but the results obtained were too noisy. For this reason, such a high r was selected, although the experiment became noticeably expensive to compute.

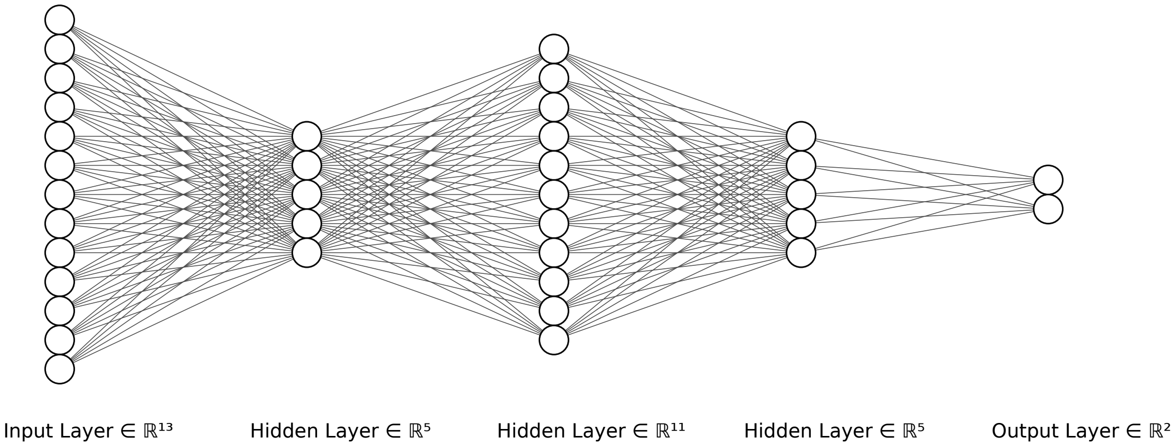


Figure 4.2: NN example for $z = 5$, where input and output layer are only illustrative.

Model Complexity As was mentioned in section 4.3, defining such model for the MIL problem is very complex task, therefore choosing a right model complexity metric is not trivial.

Consider a neural network consisting of input layer, 3 hidden layers $h_1 \in \mathbb{R}^z, h_2 \in \mathbb{R}^{2z+1}, h_3 \in \mathbb{R}^z$ and output layer. Then $z \in \{1,2,3\ldots,20\}$ was selected as model complexity metrics, because this is one of the easiest ways, how to control complexity of the defined MIL model. To gain a better insight, example of such NN is illustrated in Figure 4.2, however this is not the exact NN used in HMill.

Prediction Error For prediction error metrics, we simply used the standard cross–entropy loss as outlined at the beginning of this section. There is no need for any trickier objective. Let the total cross–entropy loss evaluated in the k^{th} random split for fixed z be denoted by $\mathcal{L}_k(z)$, then the estimated prediction error is given by

$$\widehat{\text{Err}}(z) = \frac{1}{r}\sum_{k=1}^{r}\mathcal{L}_k(z). \tag{4.7}$$

To summarize, we fit 100 models for selected z on train data and evaluate the prediction error for all of them on test data, then we take the mean value. This process is repeated for each $z \in \{1,2,3\ldots,20\}$, giving 2000 models in total.

4.4.2 Results

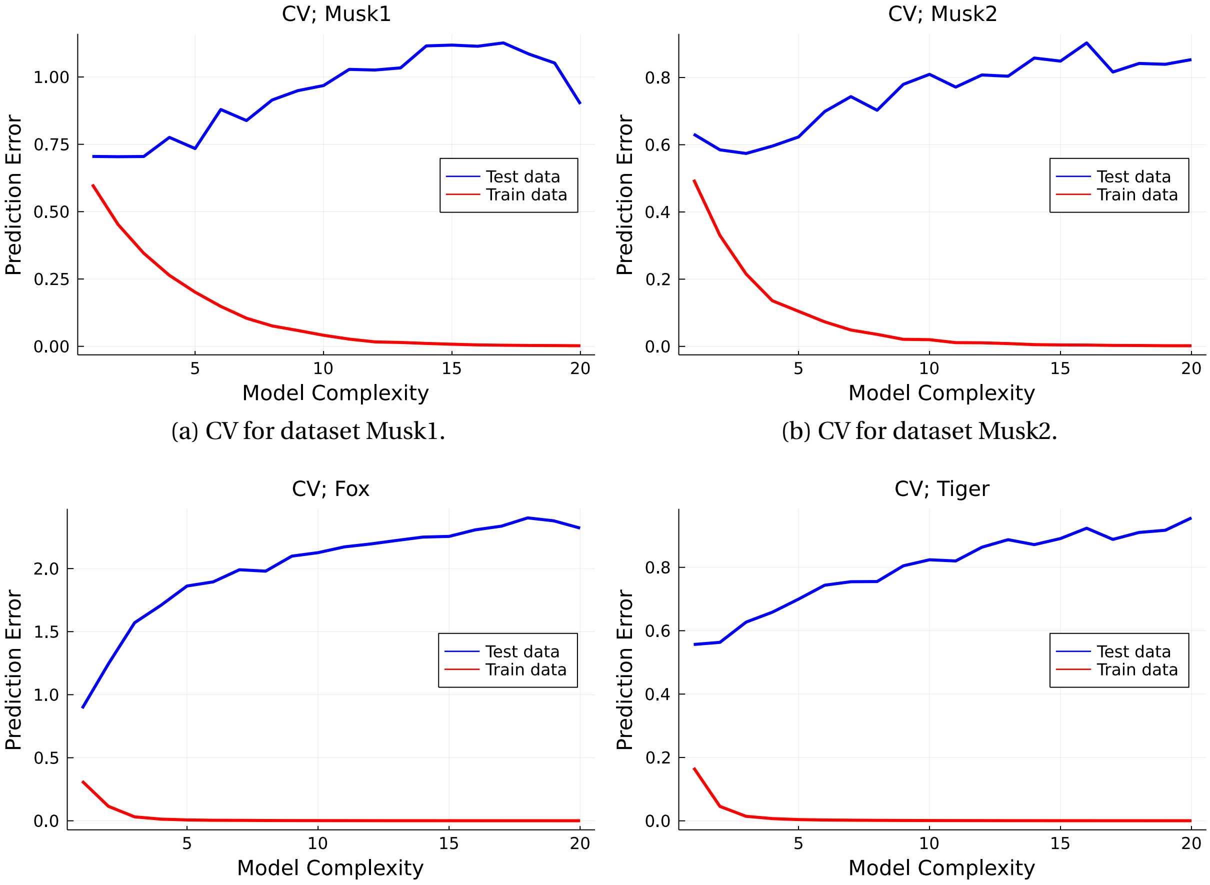
As can be seen in Figure 4.3, the results obtained are totally expected. The prediction error evaluated on the training data for the higher complexity of the model approaches zero. However, testing data give oscillating curves with an increasing trend (with a little exception of Musk1), therefore, model selection is necessary. This applies to each data set. Furthermore, Table 4.1 numerically summarizes the results evaluated in the test data.

Dataset	$\arg\min \widehat{\text{Err}}(z)$	$\min \widehat{\text{Err}}(z)$	$\widehat{\text{Err}}(z=10)$
Musk1	2	0.70	0.97
Musk2	3	0.57	0.81
Fox	1	0.89	2.13
Tiger	1	0.56	0.82

Table 4.1: Results of CV evaluated on the testing data.

4.5 MIL to HDGM problem

In the previous part, the CV experiment was performed with the expected results. However, the estimated prediction error seems to be rather high. Logically, the question of whether the prediction error can be reduced has been raised, and also whether it is possible to make r smaller has been raised.



(a) CV for dataset Musk1.

(b) CV for dataset Musk2.

(c) CV for dataset Fox.

(d) CV for dataset Tiger.

Figure 4.3: Evaluation of prediction error with the use of training data and testing data on MIL datasets Musk1, Musk2, Fox and Tiger.

4.5.1 Setup

On the initiative to reduce prediction error, it is proposed to train a MIL model that is obtained by minimizing the hybrid loss function

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)}\left[\alpha\log\frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{C}\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y_i]\right)}+(1-\alpha)\log\frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)[y]\right)}{\sum_{j=1}^{N}\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_j\right)[y]\right)}\right]. \tag{4.8}$$

Since the discriminative part is already used in HMill framework, the only task is to add the generative part into it. At this point are available all normalization samples, thus $M = N$. This modification should lead to a reduced prediction error evaluated on training data.
In the first part of this experiment, we would like to train models in relation to parameter α. For this setup, we need to choose fixed z. Since authors of [2] usually use $z = 10$, we use this value as well. For the prediction error evaluation is used standard cross–entropy as in

the previous experiment, again with $r = 100$ and $\alpha \in \{0.0, 0.1, 0.2, \dots, 1.0\}$. Therefore estimated prediction error can be written in the form

$$\widehat{\mathrm{Err}}(\alpha) = \frac{1}{r}\sum_{k=1}^{r} \mathcal{L}_k(z = 10, \alpha). \tag{4.9}$$

We hope to see a curve in the shape of a bowl that has its global minimum at a point $\alpha = 0.5$ or somewhere near.

In the second part, evaluating the prediction error is approached in the other way. The fixed $\alpha = 0.5$ is selected and the dependency on $z \in \{1, 2, 3 \dots, 20\}$ is evaluated as in Section 4.4. Finally, the estimated prediction error in this case is defined by

$$\widehat{\mathrm{Err}}(z) = \frac{1}{r}\sum_{k=1}^{r} \mathcal{L}_k(z, \alpha = 0.5). \tag{4.10}$$

In other words, the setup is the same as in 4.4, therefore, the results obtained will be added to the figure 4.3 and the table 4.1 for a convenient comparison. Note that curves for train data from this experiment will be omitted because they are not important at this point. We are only interested in predictions for test data.

4.5.2 Results

The results of the first part are represented in Figure 4.4 and Table 4.2, where it can be seen that adding the generative term to the MIL loss function decreased the prediction error evaluated on all datasets. This improvement is considerable on datasets Musk1 and Musk2, where a nice bowl can be seen. On Fox and Tiger datasets such an improvement does not occur. This means that HDGM approach works, thus regularization in the form of a very simple generative term may bring improvement in predictions. Unfortunately, the choice of $\alpha = 0.5$ was not confirmed as the best in our experiment; see Table 4.2.

In the second part of the experiment, the results are shown in Figure 4.5 and Table 4.3. Here is a situation very similar to the previous part of this experiment. Improvement of the prediction error is quite noticeable on the first two datasets Musk1 and Musk2, while Fox and Tiger do not look so convincingly. Furthermore, the number of random splits $r = 100$ is still needed to remove the noise from the prediction error.

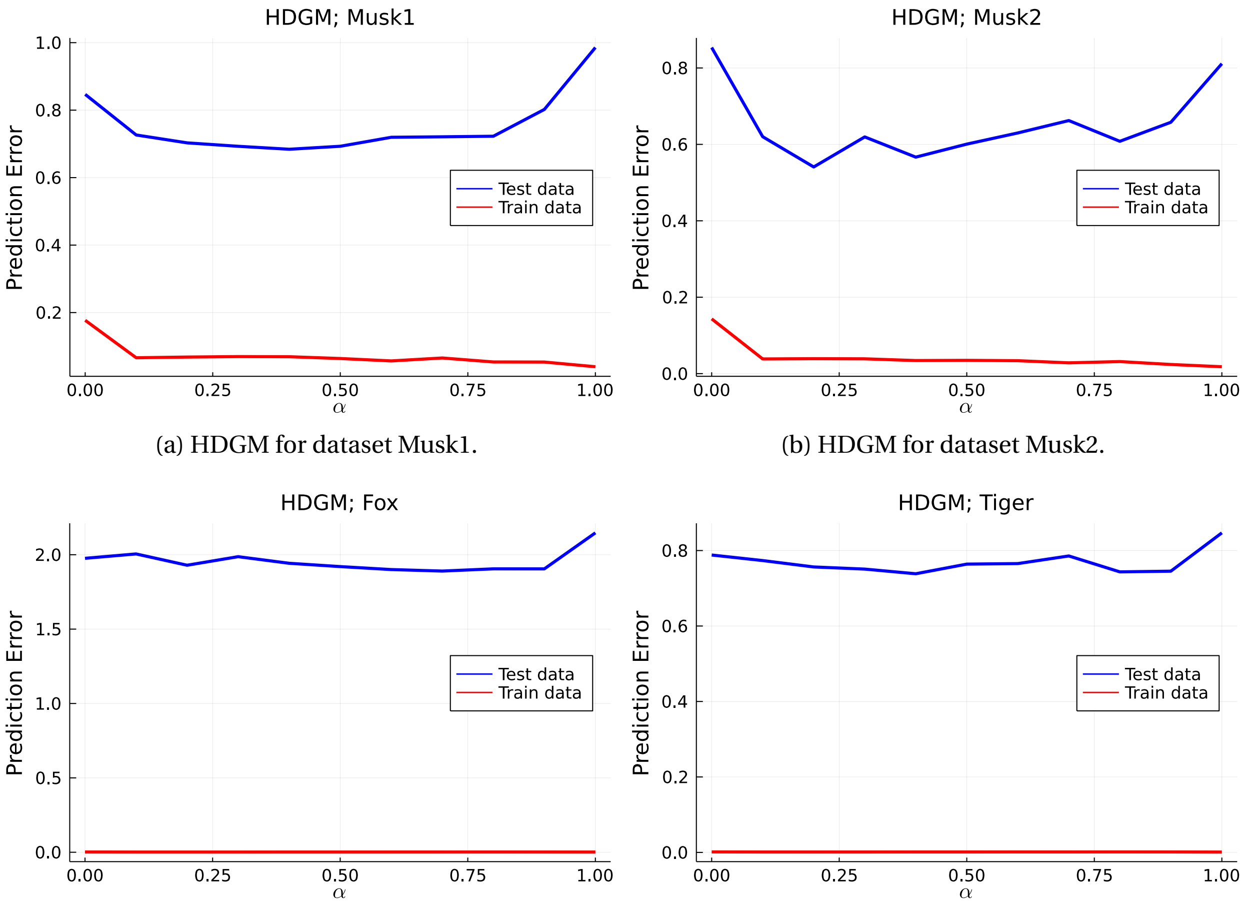
In general, it can be said that the HDGM approach leads to a small decrease in the prediction error.

Dataset	$\operatorname{argmin} \widehat{\mathrm{Err}}(\alpha)$	$\min \widehat{\mathrm{Err}}(\alpha)$
Musk1	0.4	0.68
Musk2	0.2	0.54
Fox	0.7	1.89
Tiger	0.4	0.74

Table 4.2: Prediction error statistics for HDGM in case of $z = 10$.

	Discriminative part only			**HDGM**; $\alpha = 0.5$		
Dataset	$\operatorname{argmin} \widehat{\mathrm{Err}}(z)$	$\min \widehat{\mathrm{Err}}(z)$	$\widehat{\mathrm{Err}}(z = 10)$	$\operatorname{argmin} \widehat{\mathrm{Err}}(z)$	$\min \widehat{\mathrm{Err}}(z)$	$\widehat{\mathrm{Err}}(z = 10)$
Musk1	2	0.70	0.97	6	0.64	**0.69**
Musk2	3	0.57	0.81	6	0.55	**0.66**
Fox	1	0.89	2.13	1	0.95	**1.96**
Tiger	1	0.56	0.82	2	0.58	**0.80**

Table 4.3: Comparison of prediction error statistics for HDGM $\alpha = 0.5$ and discriminative part only. Pay attention especially to the last column in each approach, $\widehat{\mathrm{Err}}(z = 10)$.



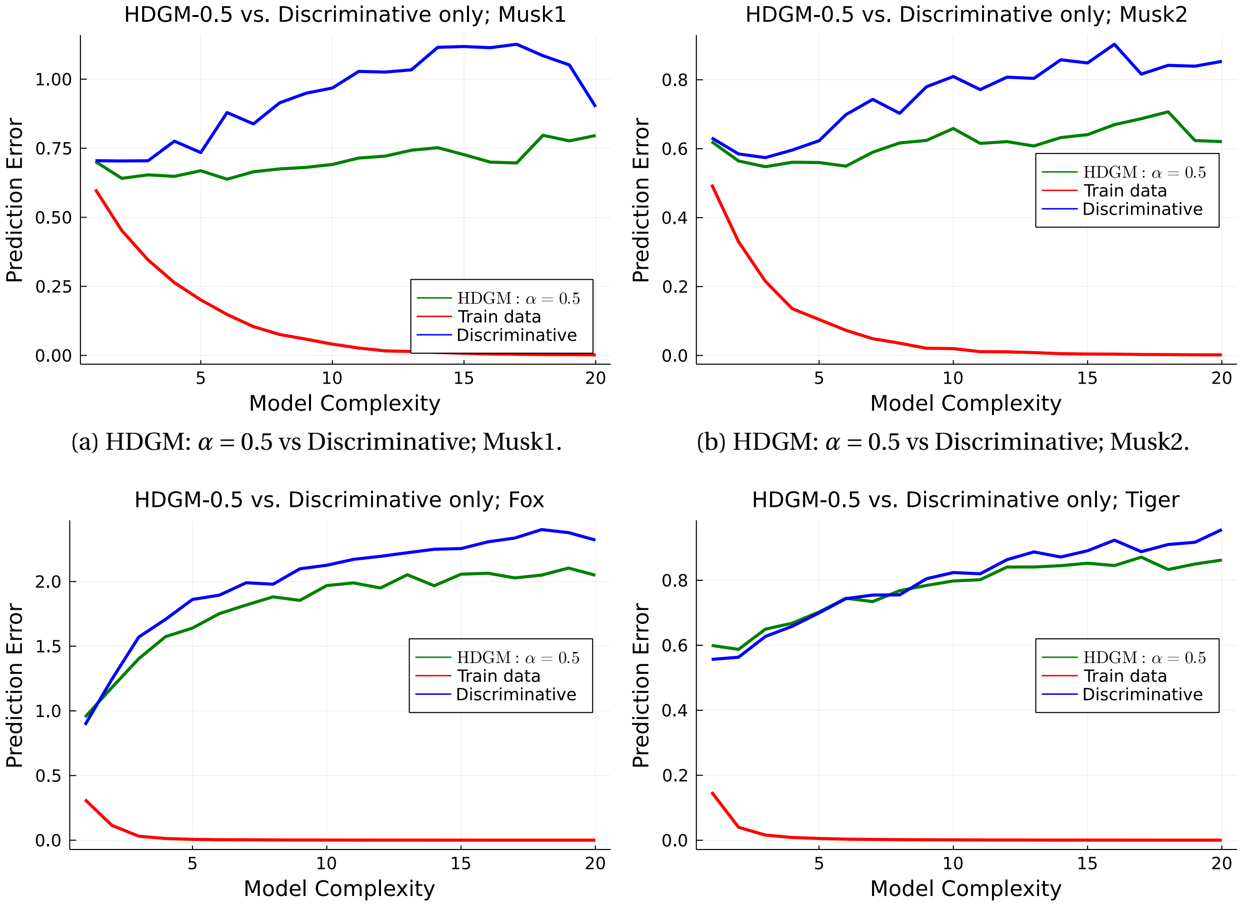
(a) HDGM for dataset Musk1.

(b) HDGM for dataset Musk2.

(c) HDGM for dataset Fox.

(d) HDGM for dataset Tiger.

Figure 4.4: Evaluation of the prediction error $\widehat{\text{Err}}(\alpha)$ with the use of training data and testing data on MIL datasets.



(a) HDGM: $\alpha = 0.5$ vs Discriminative; Musk1.

(b) HDGM: $\alpha = 0.5$ vs Discriminative; Musk2.

(c) HDGM: $\alpha = 0.5$ vs Discriminative; Fox.

(d) HDGM: $\alpha = 0.5$ vs Discriminative; Tiger.

Figure 4.5: Comparison of the prediction error $\widehat{\text{Err}}(z)$ for HDGM $\alpha = 0.5$ and only the discriminative part.

Conclusion

At the beginning of this work, supervised learning and energy-based models were introduced. The following passage consists of contrastive learning, which was briefly reviewed, and thereafter supervised learning and contrastive learning was merged into hybrid discriminative and generative models. Subsequently, this approach was applied and tested on a simple example consisting of polynomial regression. After this example, we briefly introduce multiple instance learning with a short description of the embedded space and its way of training. As a first MIL experiment, cross–validation was performed on four MIL datasets, where enough space was given to a proper definition of the model complexity metrics. The results obtained were completely expected; they included decreasing prediction errors for train data and increasing for test data. In the next step, a solution was searched on how to decrease the prediction error evaluated on the test data. In this initiative, HDGM was trained instead of a standard discriminative model. In the consecutive result, HDGM was found to lead to a decrease in prediction error, however, nothing significant. In addition, a large number of random splits of the data set were still necessary to eliminate the noise from the prediction error. In conclusion, this approach has proven to be functional, although a proposed generative regularization is very simple and can be replaced by much more complicated models. From this point of view, this approach has great potential that has not yet been fully exploited.

Bibliography

[1] Christopher M.Bishop: *Pattern recognition and machine learning.* [New York]: Springer, 2006. Information science and statistics. ISBN 0-387-31073-8.

[2] S.Mandlik.: *Mapping the Internet: Modelling Entity Interactions in Complex Heterogeneous Networks.* arXiv preprint arXiv:2104.09650.

[3] T. Pevny and P. Somol: *Discriminative models for multi-instance problems with tree structure.* In Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, 2016, 83-91.

[4] J. Wu, S. Pan, X. Zhu, C. Zhang, X. Wu: *Multi-instance learning with discriminative bag mapping.* IEEE Transactions on Knowledge and Data Engineering, 30(6), 2018, 1065-1080.

[5] H. Jeffrey: *An invariant form for the prior probability in estimation problems.* Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences. 1946, 1–9.

[6] M. I. Jordan and A. Y. Ng. :*On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes.* Advances in neural information processing systems. 2002.

[7] D. Commenges: *Information Theory and Statistics: an overview.* ArXiv preprint arXiv:1511.00860, 2015, 1–22.

[8] J. Brownlee: *How to Handle Big-p, Little-n (p » n) in Machine Learning.* (2020). [on-line]. Available from: https://machinelearningmastery.com/how-to-handle-big-p-little-n-p-n-in-machine-learning/.

[9] V. Smidl: *The Variational Bayes Approach in Signal procesing.* PhD Thesis. Trinity College Dublin. 2004.

[10] M. Zhuang and M. Collins: *Ma, Zhuang, and Michael Collins. "Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency.* arXiv preprint arXiv:1809.01812, 2018.

[11] M. Gutmann and A. Hyvärinen: *Noise-contrastive estimation: A new estimation principle for unnormalized statistical models.* Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2010.

[12] H. Liu and P. Abbeel: *Hybrid discriminative-generative training via contrastive learning.* arXiv preprint arXiv:2007.09070, 2020.

[13] S.K. Ng, T. Krishnan and G.J.McLachlan: *Handbook of computional statistics.* The EM algorithm. Springer, Berlin, Heidelberg. 2012, 139-172.

[14] W. Gratwohl, K.C. Wang and J.H. Jacobsen: *Your classifier is secretly an energy based model and you should treat it like one.* GRATHWOHL, Will, et al. Your classifier is secretly an energy based model and you should treat it like one. arXiv preprint arXiv:1912.03263, 2019.

[15] Y. LeCun, S. Chopra, R.Hadsell, M. Ranzato and F. Huang: *A tutorial on energy-based learning.* Predicting structured data, 2006, 1.0.

[16] T.G. Dietterich, R.H. Lathrop and T. Lozano-Pérez: *Solving the multiple instance problem with axis-parallel rectangles.* Artificial intelligence, 1997, 89.1-2: 31-71.

[17] J. Friedman, T. Hastie and R. Tibshirani: *The elements of statistical learning.* [New York]: Springer, 2001. Series in statistics.

[18] M. Talabis, R. McPherson, I. Miyamoto, J. Martin and D.Kaye: *Information Security Analytics.* Syngress, 2015.

[19] J.Bezanson, S. Karpinski, V.B. Skah:*A fast dynamic language for technical computing.* arXiv preprint arXiv:1209.5145, 2012.

[20] D. Yu and L. Deng: *Automatic Speech Recognition.* Springer london limited, 2016.

Appendix A

Computional formulas

A.1 Solution of $D_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{z};\boldsymbol{\mu},\sigma^2\mathbb{I}_P\right)\|\mathcal{N}\left(\boldsymbol{z};\mathbf{0},\mathbb{I}_P\right)\right)$

In the VAE section, the ELBO (2.22) is derived and subsequently optimized. One of the ELBO expressions is the KL distance mentioned above. For two multivariate Gaussian distributions, we have a KL distance analytical solution. The complete calculation is given here. First, recall that the PDF for a multivariate Gaussian distribution in $\mathbb{R}^P$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is defined as

$$\mathcal{N}\left(\boldsymbol{z};\boldsymbol{\mu},\boldsymbol{\Sigma}\right)=\frac{1}{\sqrt{(2\pi)^P\det\boldsymbol{\Sigma}}}\exp\left(-\frac{1}{2}\left(\boldsymbol{z}-\boldsymbol{\mu}\right)^\top\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{z}-\boldsymbol{\mu}\right)\right). \tag{A.1}$$

Then, the KL distance for two different multivariate Gaussian distributions can be computed as

$$\mathrm{KL}=D_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{z};\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1\right)\|\mathcal{N}\left(\boldsymbol{z};\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2\right)\right)=\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)}\left[\log\mathcal{N}\left(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1\right)-\log\mathcal{N}\left(\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2\right)\right] \tag{A.2}$$

$$=\frac{1}{2}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)}\left[-\log\det\boldsymbol{\Sigma}_1-\left(\boldsymbol{z}-\boldsymbol{\mu}_1\right)^\top\boldsymbol{\Sigma}_1^{-1}\left(\boldsymbol{z}-\boldsymbol{\mu}_1\right)+\log\det\boldsymbol{\Sigma}_2+\left(\boldsymbol{z}-\boldsymbol{\mu}_1\right)^\top\boldsymbol{\Sigma}_1^{-1}\left(\boldsymbol{z}-\boldsymbol{\mu}_1\right)\right] \tag{A.3}$$

$$=\frac{1}{2}\log\frac{\det\boldsymbol{\Sigma}_2}{\det\boldsymbol{\Sigma}_1}+\frac{1}{2}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)}\left[-\left(\boldsymbol{z}-\boldsymbol{\mu}_1\right)^\top\boldsymbol{\Sigma}_1^{-1}\left(\boldsymbol{z}-\boldsymbol{\mu}_1\right)+\left(\boldsymbol{z}-\boldsymbol{\mu}_2\right)^\top\boldsymbol{\Sigma}_2^{-1}\left(\boldsymbol{z}-\boldsymbol{\mu}_2\right)\right] \tag{A.4}$$

$$=\frac{1}{2}\log\frac{\det\boldsymbol{\Sigma}_2}{\det\boldsymbol{\Sigma}_1}+\frac{1}{2}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)}\left[-\mathrm{Tr}\left(\boldsymbol{\Sigma}_1^{-1}\left(\boldsymbol{z}-\boldsymbol{\mu}_1\right)\left(\boldsymbol{z}-\boldsymbol{\mu}_1\right)^\top\right)+\mathrm{Tr}\left(\boldsymbol{\Sigma}_2^{-1}\left(\boldsymbol{z}-\boldsymbol{\mu}_2\right)\left(\boldsymbol{z}-\boldsymbol{\mu}_2\right)^\top\right)\right] \tag{A.5}$$

$$=\frac{1}{2}\log\frac{\det\boldsymbol{\Sigma}_2}{\det\boldsymbol{\Sigma}_1}+\frac{1}{2}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)}\left[-\mathrm{Tr}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_1\right)+\mathrm{Tr}\left(\boldsymbol{\Sigma}_2^{-1}\left(\boldsymbol{z}\boldsymbol{z}^\top-2\boldsymbol{z}\boldsymbol{\mu}_2^\top+\boldsymbol{\mu}_2\boldsymbol{\mu}_2^\top\right)\right)\right] \tag{A.6}$$

$$=\frac{1}{2}\log\frac{\det\boldsymbol{\Sigma}_2}{\det\boldsymbol{\Sigma}_1}-\frac{1}{2}P+\frac{1}{2}\mathrm{Tr}\left(\boldsymbol{\Sigma}_2^{-1}\left(\boldsymbol{\Sigma}_1+\boldsymbol{\mu}_1\boldsymbol{\mu}_1^\top-2\boldsymbol{\mu}_2\boldsymbol{\mu}_1^\top+\boldsymbol{\mu}_2\boldsymbol{\mu}_2^\top\right)\right) \tag{A.7}$$

$$=\frac{1}{2}\left(\log\frac{\det\boldsymbol{\Sigma}_2}{\det\boldsymbol{\Sigma}_1}-P+\mathrm{Tr}\left(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right)+\mathrm{Tr}\left(\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_1-2\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2+\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\right)\right) \tag{A.8}$$

$$=\frac{1}{2}\left(\log\frac{\det\boldsymbol{\Sigma}_2}{\det\boldsymbol{\Sigma}_1}-P+\mathrm{Tr}\left(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right)+\left(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1\right)^\top\boldsymbol{\Sigma}_2^{-1}\left(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1\right)\right) \tag{A.9}$$

Finally, substituting $\boldsymbol{\mu}_1 = \boldsymbol{\mu}$, $\boldsymbol{\Sigma}_1 = \sigma^2 \mathbb{I}_P$, $\boldsymbol{\mu}_2 = \mathbf{0}$ and $\boldsymbol{\Sigma}_2 = \mathbb{I}_P$ gives

$$\mathrm{KL} = \frac{1}{2}\left(\log \frac{\det \mathbb{I}_P}{\det \sigma^2 \mathbb{I}_P} - P + \mathrm{Tr}\left(\sigma^2 \mathbb{I}_P \cdot \mathbb{I}_P\right) + \boldsymbol{\mu}^\top \boldsymbol{\mu}\right) \tag{A.10}$$

$$= \frac{1}{2}\left(-P \log \sigma^2 - P + \sum_{j=1}^{P} \sigma^2 + \sum_{j=1}^{P} \mu_j^2\right) \tag{A.11}$$

$$= \frac{1}{2} \sum_{j=1}^{P} \left(-1 - \log \sigma^2 + \sigma^2 + \mu_j^2\right). \tag{A.12}$$

A.2 Derivation of $\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x})}\left[-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x})\right]$

In the third chapter, we introduced EBM and its way of training. The necessary step is to compute the following gradient

$$\nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \log \int \exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} \tag{A.13}$$

$$= \left(\int \exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}\right)^{-1} \nabla_{\boldsymbol{\theta}} \int \exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} \tag{A.14}$$

$$= \left(\int \exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}\right)^{-1} \int \nabla_{\boldsymbol{\theta}} \exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} \tag{A.15}$$

$$= \left(\int \exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}\right)^{-1} \int \exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \left(-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} \tag{A.16}$$

$$= \int \left(\int \exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}\right)^{-1} \exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \left(-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} \tag{A.17}$$

$$= \int \frac{\exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right)}{Z(\boldsymbol{\theta})} \left(-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} \tag{A.18}$$

$$= \int p_{\boldsymbol{\theta}}(\boldsymbol{x}) \left(-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} \tag{A.19}$$

$$= \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x})}\left[-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x})\right]. \tag{A.20}$$