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

Název práce:

Hybridní diskriminativně-generativní modely pro množinová data

Autor: Bc. Jakub Bureš

Obor: Matematické inženýrství

Zaměření: Aplikované matematicko-stochastické metody

Druh práce: Výzkumný úkol

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

Abstrakt: Tento výzkumný úkol 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–instačnímu učení. Hlavním cílem je pak využít strukturu HMill s knihovnou Mill.jl a rozšířit 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 research project deals with hybrid discriminative and generative modeling and its possible utilisation in multiple–instance learning. At first, technicalities of this approach are introduced, consequently a simple experiment is performed and lastly multi–instance laerning itself is taken care of. The main aim of this work is to use the HMill framework with the Mill.jl library and involve the contrastive learning into it, where benefits of this approach are shown.

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

Contents

List of Symbols

Introduction

In the field of supervised learning [18] has been achieved a tremendous progress and success 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 Softmax function.

Contrastive learning [10, 11] is a machine learning method 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 aformentioned cross entropy loss. Consequently, this approach is applied on multiple instance learning problems , where is taken 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 is introduced the multiple instance learning with 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

1.1 Terminology

At the beginning of this work and for avoiding confusion, it is appropriate to clarify the terminology.

The notation for random variables via uppercase letters, for example X, Y, is very common and widely used. The realization of a random variable, also known as observed value, or simply observation, will be denoted by correspodning lowercase letters x, y. Most of the time, input variable will be denoted by the symbol x and output variable will be denoted by the symbol y. Bold symbols $\boldsymbol{x}, \boldsymbol{y}$ will be used to distinguish vectors from scalars.

For the probability density function (pdf) over x, will be used the symbol $p(x)$. In addition, this will be used for both disrete and continuous x. In this way can be achieved siginificant simplification and unification of all formulas and equations.

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

$$\mathbb{E}[g] = \int p(x)g(x)\mathrm{d}x. \tag{1.1}$$

In the case of a discrete variable, one has to keep in mind that an integration turns into a sum over all x.

Usual goal (learning task) is to make a good prediction of the output y, denoted by the symbol $\hat{y}$, with given input $\boldsymbol{x}$. Such tasks are called *supervised learning* problems [18]. This prediction is obtained through learning a model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = f(\boldsymbol{\theta}, \boldsymbol{x})$ that minimizes a loss function (also known as the error function) $\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$, where $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^s$ are the parameters of the model.

To contruct this prediction we need data, hence it is supposed that we have available set of observations, input–output paired samples denoted by $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, eventually, this may in fact be $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^D$ and $y_i \in \mathbb{R}$, $\forall i = 1, \ldots, N$, known as training data. Observations are considered to be independent and indentically distributed, often abbreviated as i.i.d.

1.2 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 $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^s$ denotes parameters of the model. The main idea behind Bayesian theory is the treatment of the uknown parameters $\boldsymbol{\theta}$ as a random variable. Bayes' rule is applied to infer model parameters $\boldsymbol{\theta}$, therefore

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\boldsymbol{\theta}, \mathcal{D})}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\Theta}} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}}. \tag{1.2}$$

Since $p(\mathcal{D})$ is just the normalization constant, Equation (1.2) is often simplified to

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}). \tag{1.3}$$

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.

Popular choices for an optimal value of the point estimate are:

1. Maximum A posteriori estimate (MAP)

$$\widehat{\boldsymbol{\theta}}_{\mathrm{MAP}} = \operatorname*{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}) \tag{1.4}$$

   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

$$\widehat{\boldsymbol{\theta}}_{\mathrm{B}} = \int_{\boldsymbol{\Theta}} \boldsymbol{\theta} \, p(\boldsymbol{\theta}|\mathcal{D})\mathrm{d}\boldsymbol{\theta} = \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|\mathcal{D})} \left[\boldsymbol{\theta}\right] \tag{1.5}$$

   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].

1.2.1 Choice of prior distribution

For the posterior computation, it is necessary to specify the prior distribution $p(\boldsymbol{\theta})$, unfortunatelly, this might not be easily determined. This can be achieved via knowledge of previous models, expert knowledge, their combination or even uncertainty about $\boldsymbol{\theta}$ can be viable option. There is also many practical aspects of priors:

- Regularization - supplementing the data, if there is insufficient data or poorly defined model.

- Imposing various restrictions on the parameters $\boldsymbol{\theta}$ reflecting physical constraints. The choice of prior distribution with bounded support will result to posterior distribution with bounded support as well.

- 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 so–called *Jeffreys priors* [5].

1.2.2 Prediction

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

$$p(y^\star|\boldsymbol{x}^\star,\mathcal{D}) = \int_{\Theta} p(y^\star|\boldsymbol{x}^\star,\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})\mathrm{d}\boldsymbol{\theta}. \tag{1.6}$$

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

$$p(y^\star|\boldsymbol{x}^\star,\mathcal{D}) = \int_{\Theta} p(y^\star|\boldsymbol{x}^\star,\boldsymbol{\theta})\delta(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}})\mathrm{d}\boldsymbol{\theta} = p(y^\star|\boldsymbol{x}^\star,\widehat{\boldsymbol{\theta}}), \tag{1.7}$$

causing an error. In typical MAP, this is known as *over–fitting*. Prediction error is then defined by a loss function measuring errors between y and $\widehat{y}$

$$\mathrm{Err} = \mathcal{L}\left(\widehat{y}, y\right), \tag{1.8}$$

where $\widehat{y} = \widehat{f_{\boldsymbol{\theta}}}(\boldsymbol{x}) = f(\widehat{\boldsymbol{\theta}}, \boldsymbol{x})$. As an example, we can mention couple of typically used loss functions

$$\mathcal{L}\left(y, \widehat{y}\right) = \begin{cases} (y - \widehat{y})^2 & \text{squared error} \\ |y - \widehat{y}| & \text{absolute error} \end{cases}. \tag{1.9}$$

We shall also discuss the problem of the model complexity. Consider a polynomial regression problem, where the model is defined by

$$f_{\boldsymbol{\theta}}(x) = \sum_{i=0}^{s-1} \theta_i x^i. \tag{1.10}$$

Here, over–fitting occurs very frequently. Model complexity 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 much higher order polynomial giving an excellent fit. Such polynomial passes exactly through each datapoint, however, oscilates wildly and gives poor prediction for new input variable $x^\star$.

To obtain some quantitative insight into dependance of the generalization performance on model complexity, consider separate test set of data (testing data) used to assess the performance of the model. In general, prediction error evaluated on the training data for increasing model complexity approaches zero. On the other hand, prediction error evaluated on the testing data for increasing model complexity is (from certain point) increasing as well. The typical scenario is illustrated in Figure 1.1.
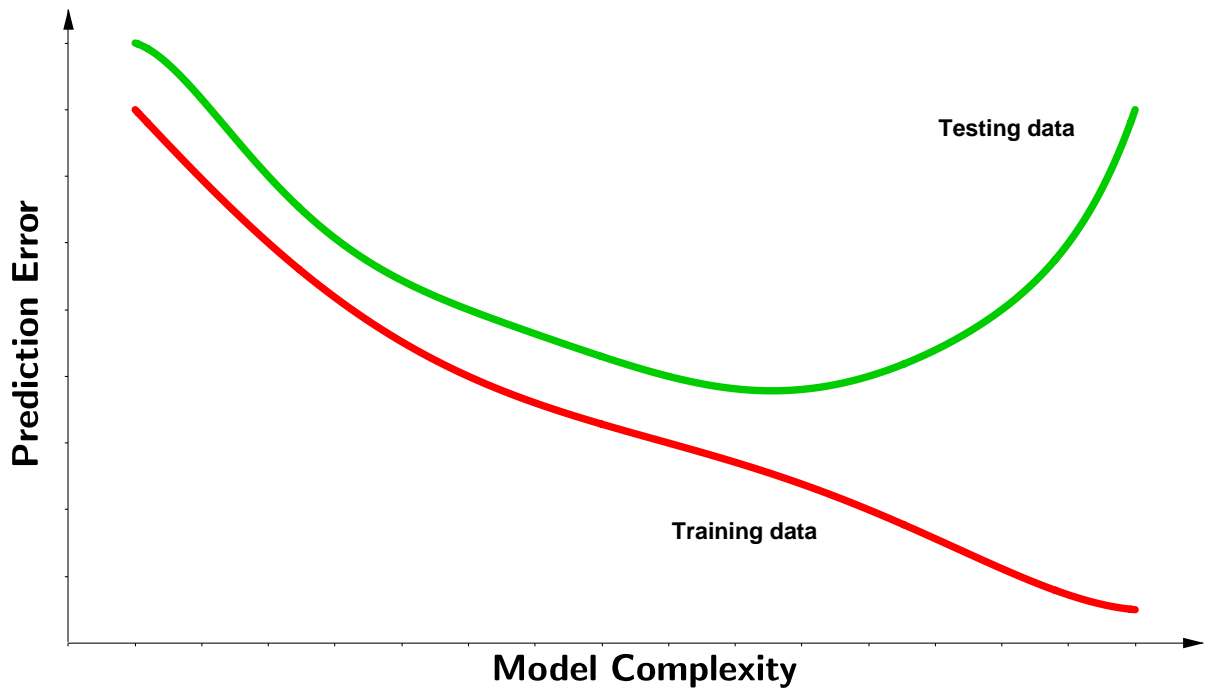
Figure 1.1: Evaluation of prediction error as a function of model complexity.

1.3 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

$$\widehat{\text{Err}} = \mathbb{E}\left[\mathcal{L}\left(y, \hat{y}\right)\right], \tag{1.11}$$

the measure how accurately is the model able to predict output values for previously unseen data - independent test sample.

1.3.1 K-Fold Cross-Validation

In an ideal case, if we have sufficient number of data, we can set aside a test set and use it to assess the performance of our 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 1.2.

train	train	test	train	train

Figure 1.2: Splitting the data into $K = 5$ roughly equal-sized parts.

For the jth part (third in Figure 1.2), 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 jth part of the data. We repeat this process for $j = \{1, 2, \ldots, K\}$ and combine the K estimates of prediction error. Let $\gamma : \{1, \ldots, N\} \rightarrow \{1, \ldots, K\}$ be an indexing function that indicates the partition to which observation j is allocated by the randomization. Symbol $\hat{f}_{\theta}^{-j}(x)$ denotes the fitted model, computed with the j^{th}th part of the data removed. Then the cross-validation estimate of prediction error is defined by

$$\text{CV}\left(\hat{f}_{\theta}\right) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(y_i, \hat{f}_{\theta}^{-\gamma(i)}(x_i)\right). \tag{1.12}$$

Typical choices of K are 5 or 10 and even case $K = N$ that is known as *leave-one-out* cross–validation. 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. For extremely complicated and complex models that are trained for hours or days, is cross–validation inconvenient approach of estimating the prediction error.

Chapter 2

Discriminative modeling

Chapter 3

Generative modeling

Chapter 4

Hybrid Generative and Discriminative models

4.1 Discriminative modeling

In this chapter, we review basics of discriminative modeling that was proposed in [12]. Given a data distribution via probability density $p(\boldsymbol{x})$ and a label distribution with probability density $p(y|\boldsymbol{x})$ containing C categories. In other words, variable y is now cathegorical taking on C possible values and comes from a finite set C. A classification problem is typically solved using a parametric function $f_{\boldsymbol{\theta}} : \mathbb{R}^D \rightarrow C$, where $\boldsymbol{\theta}$ denotes parameters of the model. In practice, function $f_{\boldsymbol{\theta}}$ is often used in the form of $\mathbb{R}^D \rightarrow \mathbb{R}^C$. This function maps each data point $\boldsymbol{x} \in \mathbb{R}^D$ to C real-valued numbers known as logits. One has to keep in mind that $\mathbb{R}^C$ is allowed here due to the utilization of *one-hot encoding*, which will be explained in section 4.1.1. Logits are used to parametrize a categorical distribution via the function

$$q_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \frac{\exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]\right)}{\sum_{i=1}^{C} \exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y_i]\right)}, \tag{4.1}$$

which is known as the Softmax function. Note that the convention $f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]$ means the yth element of the $f_{\boldsymbol{\theta}}(\boldsymbol{x})$. For learning $f_{\boldsymbol{\theta}}$ is usually minimized cross-entropy loss

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)}\left[\log q_{\boldsymbol{\theta}}(y|\boldsymbol{x})\right]. \tag{4.2}$$

Rationale for this objective comes from minimizing the Kullback-Leibler divergence with a target distribution $p(y|\boldsymbol{x})$ [7]. In general, Kullback-Leibler divergence, for distribution functions Ψ and Π with corresponding probablity density functions ψ and π, is defined as

$$D_{\text{KL}}(\Psi||\Pi) = \mathbb{E}_{x \sim \psi}\left[\log \frac{\psi(x)}{\pi(x)}\right], \tag{4.3}$$

which can be further rewritten in the form

$$\mathbb{E}_{x \sim \psi}\left[\log \frac{\psi(x)}{\pi_{\theta}(x)}\right] = \mathbb{E}_{x \sim \psi}\left[\log \psi(x)\right] - \mathbb{E}_{x \sim \psi}\left[\log \pi_{\theta}(x)\right], \tag{4.4}$$

where subscript θ emphasizes that $\pi_\theta(x)$ is our approximative density we get to control. Note that the first term does not depend on θ and therefore minimazing either Cross-Entropy or Kullback-Leibler divergence is equivalent. Finally, by minimizing with respect to $\pi_\theta(x)$ we obtain

$$\min_{\theta} D_{\text{KL}}\left(\Psi||\Pi\right) = \min_{\theta} -\mathbb{E}_{x\sim\psi}\left[\log \pi_\theta\left(x\right)\right] \tag{4.5}$$

In practise it is dealt with discrete data, so $\mathbb{E}_{x\sim\psi}\left[\log \pi_\theta\left(x\right)\right]$ takes the form

$$\mathbb{E}_{x\sim\psi}\left[\log \pi_\theta\left(x\right)\right] = \sum_{i=1}^{N}\psi(x_i)\log \pi_\theta\left(x_i\right)) \tag{4.6}$$

4.1.1 One-hot encoding

Machine learning (ML) algorithms can misinterpret the numeric values of labels if there exists any hiearchy between them. One-hot encoding is a very common approach how to deal 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 up with 0 or 1 (0 for FALSE and 1 for TRUE). Transformation of a label encoding to the one-hot encoding is illustrated in following tables 4.1 and 4.2.

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

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

Table 4.1: Example of label encoding.

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

Table 4.2: Example of one–hot encoding.

4.2 Energy-Based Models

Energy-based models (EBM) was firstly introduced here [14], but following nomenclature was established here [15]. EBM assume that probability densities $p(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{R}^D$ can be

expressed in the form

$$p_\theta(\boldsymbol{x}) = \frac{\exp\left(-E_\theta(\boldsymbol{x})\right)}{Z(\boldsymbol{\theta})}, \tag{4.7}$$

where function $E_\theta : \mathbb{R}^D \to \mathbb{R}$ is called Energy and maps each datapoint $\boldsymbol{x}$ to a scalar. The denominator $Z(\boldsymbol{\theta})$ is a normalization constant (also known as partition function), thus

$$Z(\boldsymbol{\theta}) = \sum_{i=1}^{N} \exp\left(-E_\theta(\boldsymbol{x}_i)\right), \tag{4.8}$$

where the summation is over all datapoints $\boldsymbol{x}$ available. The sum turns into integral for a continuous $\boldsymbol{x}$. Very imporant observations made authors in [15], where they show, that classifiers in supervised learning are secretly energy-based models on $p(\boldsymbol{x}, y)$ and can be expressed as

$$p(\boldsymbol{x}, y) = \frac{\exp\left(f_\theta(\boldsymbol{x})[y]\right)}{Z(\boldsymbol{\theta})}. \tag{4.9}$$

The obbjective above is called joint energy model and it is obvious that $f_\theta(\boldsymbol{x})[y] = -E_\theta(\boldsymbol{x}, y)$. It may come in handy to have only a density model of datapoints $p(\boldsymbol{x})$ without labels. This could be achieved by marginalizing $p(\boldsymbol{x}, y)$ over y

$$p(\boldsymbol{x}) = \frac{\sum_{i=1}^{C} \exp\left(f_\theta(\boldsymbol{x})[y_i]\right)}{Z(\boldsymbol{\theta})}, \tag{4.10}$$

where energy is given by $E_\theta(\boldsymbol{x}) = -\log \sum_{i=1}^{C} \exp\left(f_\theta(\boldsymbol{x})[y_i]\right)$. Very useful property appears when computing $p(y|\boldsymbol{x})$. We can take advantage of the definition of conditional distribution $p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x},y)}{p(\boldsymbol{x})}$, yielding

$$p_\theta(y|\boldsymbol{x}) = \frac{\exp\left(f_\theta(\boldsymbol{x})[y]\right)}{\sum_{i=1}^{C} \exp\left(f_\theta(\boldsymbol{x})[y_i]\right)}. \tag{4.11}$$

Note that the normalization constant $Z(\boldsymbol{\theta})$ canceled out and we ended up with the same function, which was introduced in (4.1).

4.3 Contrastive learning

Contrastive learning [10, 11] is a ML technique used to learn so-called general features of a dataset by teaching the model which datapoints are similar or different. This all happens without labels, therefore contrastive learning is often called *self–supervised* technique of ML. In contrastive learning problems, it is very usual to optimize an objective often called contrastive loss, which can be written in the form as follows

$$\min_{\theta} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\log \frac{\exp\left(h_\theta(\boldsymbol{x}) \cdot h_\theta(\boldsymbol{x}')\right)}{\sum_{i=1}^{N} \exp\left(h_\theta(\boldsymbol{x}) \cdot h_\theta(\boldsymbol{x}_i)\right)}\right]. \tag{4.12}$$

Function $h_\theta : \mathbb{R}^D \to \mathbb{R}^H$ maps each data point to a representation space with dimension H, while x and x' are two different augmented views of the same data point. If x is an image, then augmented view of x can be obtained for example by rotation or colorization of that image. Note that the inner product between two vectors can replaced with any distance metric, for example Euclidean distance.

What this objective does is that it tries to maximally distinguish an input x_i from an alternative input x'_i. In other words, (4.12) reduces the distance between representations of different augmented views of the same image x, x' (positive pairs) and increases the distance between representations of augmented views of from different images (negative pairs). This means that the model should be able to distinguish between different types of images without even knowing what these images really are.

4.3.1   Noise-contrastive estimation

Suppose we have to estimate a model that is specified by an non-normalized probability density function $q_\theta^0(x)$. Thus, we introduce the another parameter c among the estimated parameters θ. For clarity, the symbol $\theta^\star = \{\theta, c\}$ is introduced for the set of estimated parameters including c. Using this notation, the following equality then holds

$$\ln q_{\theta^\star}(x) = \ln q_\theta^0(x) + c \tag{4.13}$$

which means that the newly introduced parameter c is an estimate of the negative logarithm of the normalization constant $Z(\theta)$.

As the name of the subsection suggests, we use noise to estimate. By our convention, let $x_1, ..., x_N$ be the observations and $e_1, ..., e_N$ be the artificially generated noise data with known distribution $\psi(e)$. The $\widehat{\theta^\star}$ estimate is then defined as

$$\widehat{\theta^\star} = \underset{\theta^\star}{\operatorname{argmax}} \, \mathcal{L}^{\text{NC}}(\theta^\star) \tag{4.14}$$

$$= \underset{\theta^\star}{\operatorname{argmax}} \, \frac{1}{2N} \sum_{i=1}^{N} \ln \sigma_{\theta^\star}(x_i) + \ln(1 - \sigma_{\theta^\star}(e_i)) \tag{4.15}$$

where $\sigma_{\theta^\star}$ is logistic function

$$\sigma_{\theta^\star}(x) = \frac{1}{1 + \exp(-G_{\theta^\star}(x))} \tag{4.16}$$

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

$$G_{\theta^\star}(x) = \ln q_{\theta^\star}(x) - \ln \psi(x). \tag{4.17}$$

It may be noted that equation 4.15 also appears in supervised learning and is called binary cross-entropy loss. It is actually a special case of cross-entropy itself. Thus, it is used for classification between two classes. This gives an intuitive insight into how noise-contrastive estimation really works. By comparing data and noise, the model is learned, so this method can be called learning by comparison.

4.3.1.1 Experiment

To test this approach, we perform a simple experiment.There are a total of $N = 100$ i.i.d. one dimensional observations x from an unknown distribution that is assumed to be non-normalized and Gaussian. It is therefore of the form

$$q_{\theta^\star}(x) = \exp\left(-\frac{1}{2} \cdot \frac{(x-\mu)^2}{\sigma^2} + c\right) \tag{4.18}$$

where $\theta^\star = \{\mu, \sigma, c\}$. Next, we artificially generate $N = 100$ noise data e, which is again easiest to do using a Gaussian distribution. Which means that it can be chosen, for example

$$\psi(e) = \frac{1}{\sqrt{2\pi 10}} \exp\left(-\frac{1}{2} \cdot \frac{e^2}{10}\right). \tag{4.19}$$

At this point it is possible to construct a function $\mathcal{L}^{\text{NC}}(\theta^\star)$ that is minimized using stochastic gradient descent. The following figure shows the training process and the comparison between the estimated distribution and the true one. As can be seen in the figure 4.1, this approach works
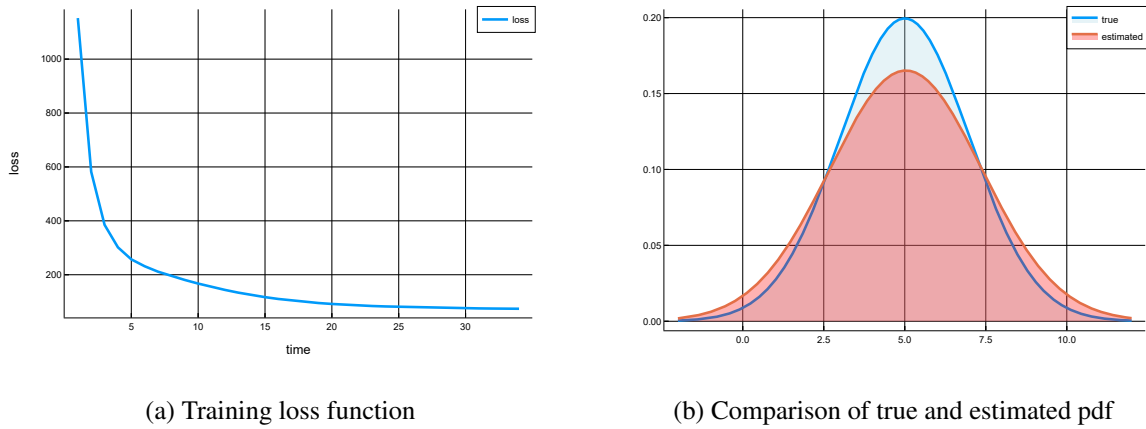

(a) Training loss function   (b) Comparison of true and estimated pdf

Figure 4.1: Results of the NCE experiment.

quite well and for more observations results would be even better.

4.4 Hybrid Dicriminative and Generative Models, HDGM

In this section will be discussed an approach, how to combine both of the already mentioned models. Authors of the article [12] proposed a solution, however the rationale of this objective originates from [6], where authors show that hybrid models can out–perform purely generative or purely discriminative counterparts.
The primary goal is to train a model that can classify x to classes y. Secondarily, learned models

should be capable of out-of-distribution detection and serve as a generative model. To achieve these goals, a hybrid model consists of a discriminative conditional and a generative conditional by maximizing the sum of both conditional log-likelihoods

$$\min_{\theta} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)}\left[\log q_\theta\left(y|\boldsymbol{x}\right) + \log q_\theta\left(\boldsymbol{x}|y\right)\right], \tag{4.20}$$

where the first term

$$q_\theta\left(y|\boldsymbol{x}\right) = \frac{\exp\left(f_\theta\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{C}\exp\left(f_\theta\left(\boldsymbol{x}\right)[y_i]\right)} \tag{4.21}$$

is a standard Softmax neural net classifier (as mentioned in Equation (4.1)) and the second term

$$q_\theta\left(\boldsymbol{x}|y\right) = \frac{\exp\left(f_\theta\left(\boldsymbol{x}\right)[y]\right)}{\sum_{j=1}^{N}\exp\left(f_\theta\left(\boldsymbol{x}_j\right)[y]\right)}, \tag{4.22}$$

is a contrastive loss (4.12) attributed a label. This objective often struggles with the unknown partition function $\sum_{j=1}^{N}\exp\left(f_\theta\left(\boldsymbol{x}_j\right)[y]\right)$, which is often intractable, specifically if the number of datapoints is very high. This obstacle is typically addressed using an approximation

$$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)}\left[\log q_\theta\left(\boldsymbol{x}|y\right)\right] \tag{4.23}$$

$$= \mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)}\left[\log \frac{\exp\left(f_\theta\left(\boldsymbol{x}\right)[y]\right)}{\sum_{j=1}^{N}\exp\left(f_\theta\left(\boldsymbol{x}_j\right)[y]\right)}\right] \tag{4.24}$$

$$\approx \mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)}\left[\log \frac{\exp\left(f_\theta\left(\boldsymbol{x}\right)[y]\right)}{\sum_{j=1}^{M}\exp\left(f_\theta\left(\boldsymbol{x}_j\right)[y]\right)}\right], \tag{4.25}$$

where $M < N$ denotes the number of normalization samples. In order to have an sufficient approximation, M has to be sufficiently large - becoming exact in the limit $M \to N$. In practice, increasing M is not simple as it requires a larger memory. However, that does not apply to our experiments.

Now it is possible to substitute approximation (4.23) to Equation (4.20) that gives a hybrid combination of supervised learning and constrastive learning in the form

$$\min_{\theta} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)}\left[\alpha \log q_\theta\left(y|\boldsymbol{x}\right) + (1-\alpha) \log q_\theta\left(\boldsymbol{x}|y\right)\right] \tag{4.26}$$

$$\approx \min_{\theta} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)}\left[\alpha \log \frac{\exp\left(f_\theta\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{C}\exp\left(f_\theta\left(\boldsymbol{x}\right)[y_i]\right)} + (1-\alpha) \log \frac{\exp\left(f_\theta\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{M}\exp\left(f_\theta\left(\boldsymbol{x}_i\right)[y]\right)}\right]. \tag{4.27}$$

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 $\alpha = 0$, the objective is reduced to a case called an *end-to-end supervised version of contrastive learning*. The choice of parameter α is a decision of the experiment designer, however authors in [12] evaluated many possible variants in performed experiments and found out that the choice of $\alpha = 0.5$ yields to the highest performance on a classification accuracy. As far as we know, these experiments involved only image classification, unfortunately.

Hybrid combination of supervised learning and contrastive learning (4.27) is for us absolutely crucial as we extend this approach to multiple instance learning problem, but this is discussed more in further sections.

4.5   Toy problem - Polynomial Regression

At first, we would like to try out hybrid discriminative and generative approach on simple example before we head into more difficult cases. The goal is to train a model of the form (4.27) that was derived in previous section.
Assume data $\{x_i, y_i\}_{i=1}^N$, where $x_i, y_i \in \mathbb{R}$, therefore this is only 2-dimensional problem. According to energy–based models 4.2, we know that for joint distribution it holds

$$p(x,y) = \frac{\exp\left(f_{\boldsymbol{\theta}}(x)[y]\right)}{Z(\boldsymbol{\theta})}, \tag{4.28}$$

where the model model is given by

$$f_{\boldsymbol{\theta}}(x)[y] = -E_{\boldsymbol{\theta}}(x,y). \tag{4.29}$$

At this point, we should transform our problem to the polynomial regression. We need to be aware of the discriminative term in the Equation (4.20), 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

$$S = S\left(\boldsymbol{\theta}\right) = \sum_{k=1}^N \left(y_k - \sum_{i=0}^{s-1} \theta_i x_k^i\right)^2. \tag{4.30}$$

Any joint probability distribution can be break down into parts via chain-rule

$$p(x,y) = p(y,x) = p(y|x) \cdot p(x), \tag{4.31}$$

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

$$p(y|x, \boldsymbol{\theta}) = \mathcal{N}\left(\sum_{i=0}^{s-1} \theta_i x^i, \sigma^2\right), \tag{4.32}$$

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

$$p(x|\tau) = \mathcal{N}\left(0, \tau^2\right), \tag{4.33}$$

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, data are spread very far from

their expected value. Substituting Equation (4.33) and (4.32) to the (4.31) results to

$$p(x, y) = \mathcal{N}\left(0, \tau^2\right) \cdot \mathcal{N}\left(\sum_{i=0}^{s-1} \theta_i x^i, \sigma^2\right) = \frac{1}{2\pi\sigma\tau} \exp\left(-\frac{\left(y - \sum_{i=0}^{s-1} \theta_i x^i\right)^2}{2\sigma^2} - \frac{x^2}{2\tau^2}\right), \quad (4.34)$$

whereas our desirable model is given by

$$f_\theta(x)[y] = -\frac{\left(y - \sum_{i=0}^{s-1} \theta_i x^i\right)^2}{2\sigma^2} - \frac{x^2}{2\tau^2}. \quad (4.35)$$

We can now substite (4.35) and (4.30) into Equation (4.20), yielding

$$\min_{\theta} \left\{ \alpha\, S\left(\theta\right) - \mathbb{E}_{p_{\text{data}}(x,y)}\left[(1 - \alpha) \log q_\theta\left(x|y\right)\right] \right\} = \quad (4.36)$$

$$\min_{\theta} \left\{ \alpha\, S\left(\theta\right) - \mathbb{E}_{p_{\text{data}}(x,y)}\left[(1 - \alpha) \log \frac{\exp\left(f_\theta\left(x\right)[y]\right)}{\sum_{i=1}^{N} \exp\left(f_\theta\left(x_i\right)[y]\right)}\right] \right\} = \quad (4.37)$$

$$\min_{\theta} \left\{ \alpha\, S\left(\theta\right) - \mathbb{E}_{p_{\text{data}}(x,y)}\left[(1 - \alpha) \log \frac{\exp\left(-\frac{\left(y - \sum_{i=0}^{s-1} \theta_i x^i\right)^2}{2\sigma^2} - \frac{x^2}{2\tau^2}\right)}{\sum_{k=1}^{N} \exp\left(-\frac{\left(y - \sum_{i=0}^{s-1} \theta_i x_k^i\right)^2}{2\sigma^2} - \frac{x_k^2}{2\tau^2}\right)}\right] \right\}. \quad (4.38)$$

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

$$\log q_\theta\left(x|y\right) = \left(-\frac{\left(y - \sum_{i=0}^{s-1} \theta_i x^i\right)^2}{2\sigma^2} - \frac{x^2}{2\tau^2}\right) - \log \sum_{k=1}^{N} \exp\left(-\frac{\left(y - \sum_{i=0}^{s-1} \theta_i x_k^i\right)^2}{2\sigma^2} - \frac{x_k^2}{2\tau^2}\right). \quad (4.39)$$

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

4.5.1 Experiment setup and results

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

We generated synthetic data, two clusters consisting of 5 datapoints each, then the model was fitted for different weight $\alpha \in \{0.0, 0.1, 0.2, \ldots, 1.0\}$, giving 11 different models in total. Models estimated for $\alpha \in \{0.0, 0.5, 1.0\}$ are highlighted as they are more important to us then the other models.

At first, we trained the mentioned models for the fixed order of the polynomial, but for 6 different values of the parameter τ. Obtained results (Figure 4.2) for small τ barely vary from those for

high τ, that is exactly what we hoped for, since the prior distribution should be non–informative (4.33). This is very exciting discovery, because there is no need to know much information about the data distribution.

Secondly, 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 (4.36) affects the polynomial regression loss for different values of $s - 1$. As can be seen in Figure 4.3, for small $s - 1$, such as $s - 1 = 2$, term $\log q_\theta(x|y)$ does not affect polynomial regression too much. However, we get considerable difference for higher orders of the polynomial. Furthermore, it seems that the curve $\alpha = 0$ prefers not to oscillate. This also could be very interesting, because combining of discriminative and generative models could result in better model predictions.
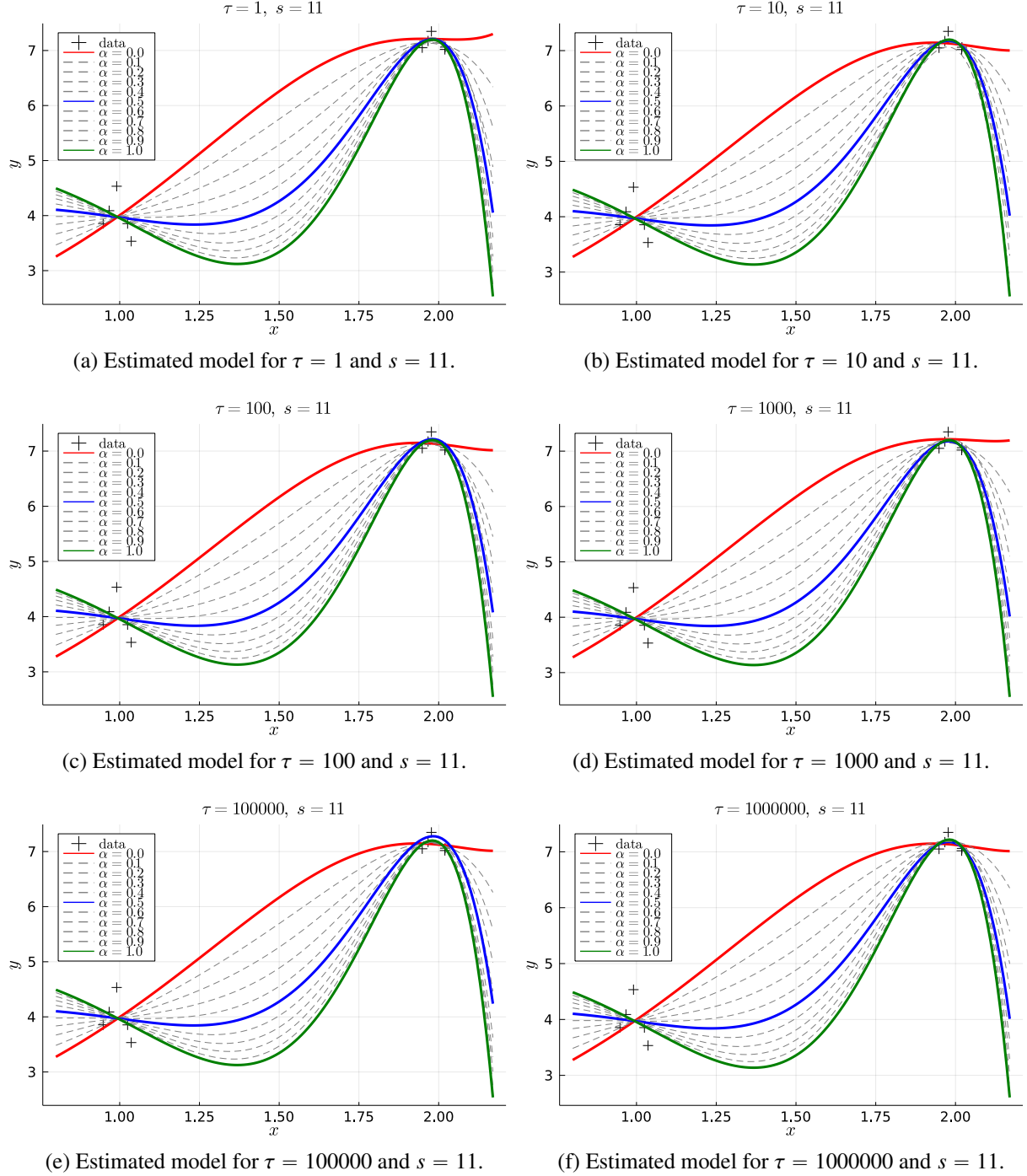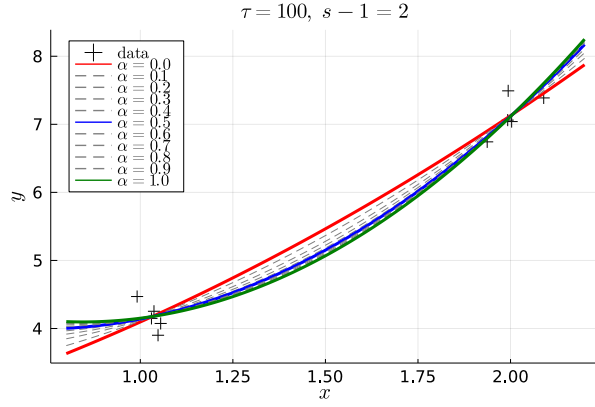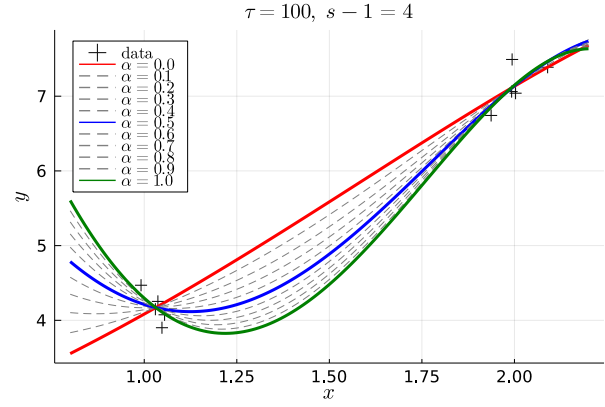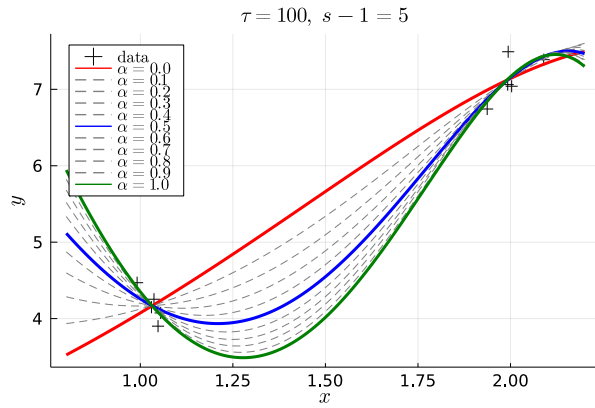
(a) Estimated model for $\tau = 1$ and $s = 11$.

(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 $s = 11$.

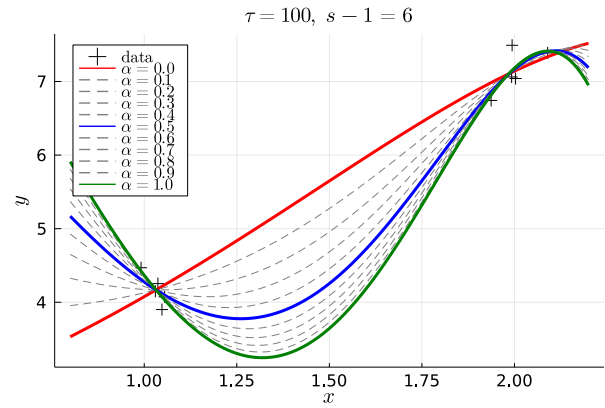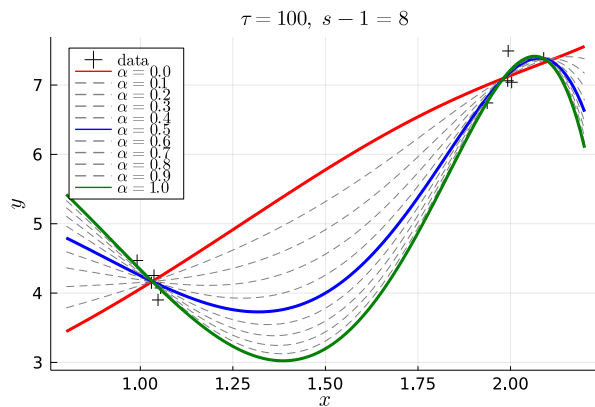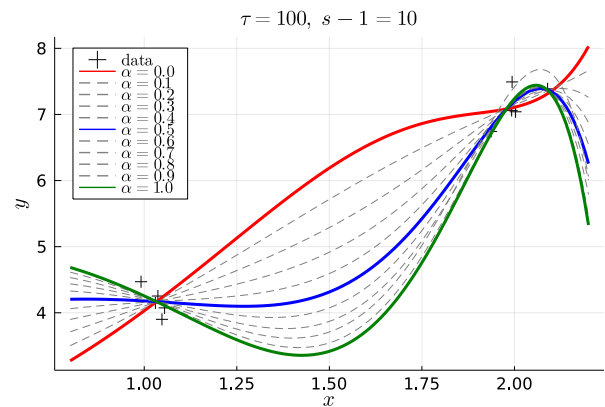(f) Estimated model for $\tau = 1000000$ and $s = 11$.

Figure 4.2: Sensitivity of the polynomial model to parameter τ with parameter s 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 4.3: Sensitivity of the polynomial model to the order of polynomial $s - 1$ for six different cases.

Chapter 5

Multiple Instance Learning

5.1 Fundamentals

The term multiple instance learning originates from [16] and in [2], authors proposed following nomenclature for MIL, which will be reviewed and gladly used 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 $\mathbb{R}^n$. Sets of these instances are called *bags* and come from 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 down 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 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 C and what we want in MIL is learning a predictor in the form $f_{\boldsymbol{\theta}} : \mathcal{B}(\mathcal{X}) \rightarrow C$ which can also be rewritten in the form $f_{\boldsymbol{\theta}}\left(\{\boldsymbol{x}\}_{\boldsymbol{x} \in b}\right)$. In contrast to ML, where a predictor is learned in the form $f_{\boldsymbol{\theta}} : \mathbb{R}^n \rightarrow C$. We consider supervised setting, in which each sample of the dataset is attributed a label. We can denote available data by notation $\mathcal{D} = \left\{(b_i, y_i) \in \mathcal{B} \times C \mid i \in \{1, 2, \ldots, |\mathcal{D}|\}\right\}$, where $|\mathcal{D}|$ apparently denotes the size of $\mathcal{D}$.

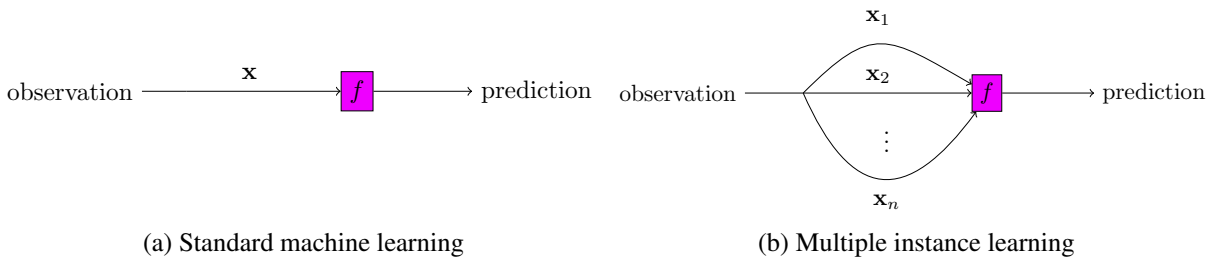(a) Standard machine learning   (b) Multiple instance learning

Figure 5.1: The difference between standard ML and MIL [2]. Standard ML is special case of MIL with $|b| = 1$.

5.2 Embedded-space paradigm

Embedded space paradigm [2] defines a vector space for bag representation and specify a mapping from each bag $b \in \mathcal{B}$ to this space. Assume that target vector space is $\mathbb{R}^d$, then partial mapping $\phi_i \colon \mathcal{B} \to \mathbb{R}, \quad \forall i = 1, 2 \ldots d$ is defined and overall embedding $\boldsymbol{\phi} : \mathcal{B} \to \mathbb{R}^d$ is given by

$$\boldsymbol{\phi}(b) = (\phi_1(b), \phi_2(b), \ldots, \phi_m(b)) . \tag{5.1}$$

Mappings ϕ_i are instrumentals towards extraction and suitable aggregation of information on the level of instances. They can be defined via some instance transformation $k : \mathcal{X} \to \mathbb{R}^d$ and aggregation function $g : \mathcal{P}_F\left(\mathbb{R}^d\right) \to \mathbb{R}^s$ in the form

$$\phi_i(b) = g\left(k\left\{\boldsymbol{x}\right\}_{\boldsymbol{x} \in b}\right) . \tag{5.2}$$

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

5.3 Training

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

Learning of the MIL model $f_{\boldsymbol{\theta}}$ is also supervised, a specific case called binary classification, and therefore accomplished by minimizing standard cross–entropy

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x}, y)} \left[\log \frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{C} \exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y_i]\right)} \right], \tag{5.3}$$

already mentioned in section 4.1. This is very imporant for us and we will take advantage of that in our experiments.

5.4 Cross–validation on MIL datasets

For testing MIL, we have avaliable four datasets, namely Musk1, Musk2, Tiger and Fox. All of them will be used to assess performance of the MIL model.

5.4.1 Setup

In this experiment, datasets 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 test set. For a future simplification, let a number of random splits be denoted by r, thus $r = 100$. We fit the model on train set, then we evaluate the prediction error on the train data via cross–entropy 4.2. The objective here is to plot the prediction error dependancy on the model complexity. Smaller number of random splits r were tested, but obtained results were too noisy. For this reason, such high r was selected, although the experiment became noticeably expensive to compute.
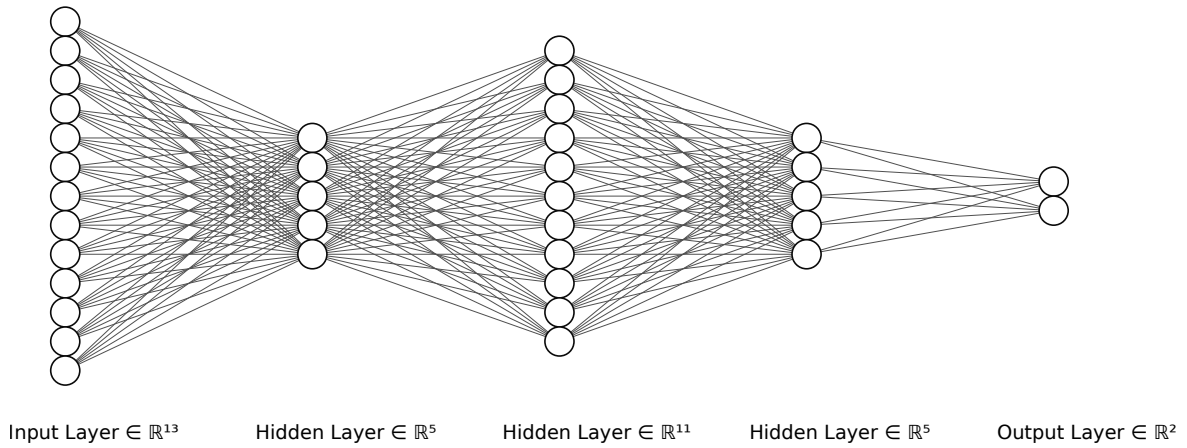

Input Layer $\in \mathbb{R}^{13}$ Hidden Layer $\in \mathbb{R}^{5}$ Hidden Layer $\in \mathbb{R}^{11}$ Hidden Layer $\in \mathbb{R}^{5}$ Output Layer $\in \mathbb{R}^{2}$

Figure 5.2: NN example for $z = 5$, where input and output layer are only illustrative.

Model Complexity   As was mentioned in section 5.3, 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 5.2, however this is not the exact NN used in HMill.
Prediction Error   For prediction error metrics will be simply used 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 on k^{th} random split for fixed z be denoted by $\mathcal{L}_k(z)$, then the estimated prediction error is given by

$$\widehat{\text{Err}}(z) = \frac{1}{r} \sum_{k=1}^{r} \mathcal{L}_k(z). \tag{5.4}$$

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

5.4.2 Results

As can be seen in Figure 5.3, obtained results are totally expected. The prediction error evaluated on the training data for higher model complexity approaches zero. However, testing data give oscillating curves with an increasing trend (with a little exception of Musk1), therefore the model selection is necessary. This applies for each dataset. In addition, following table 5.1 numerically summarizes the results evaluated on testing data.
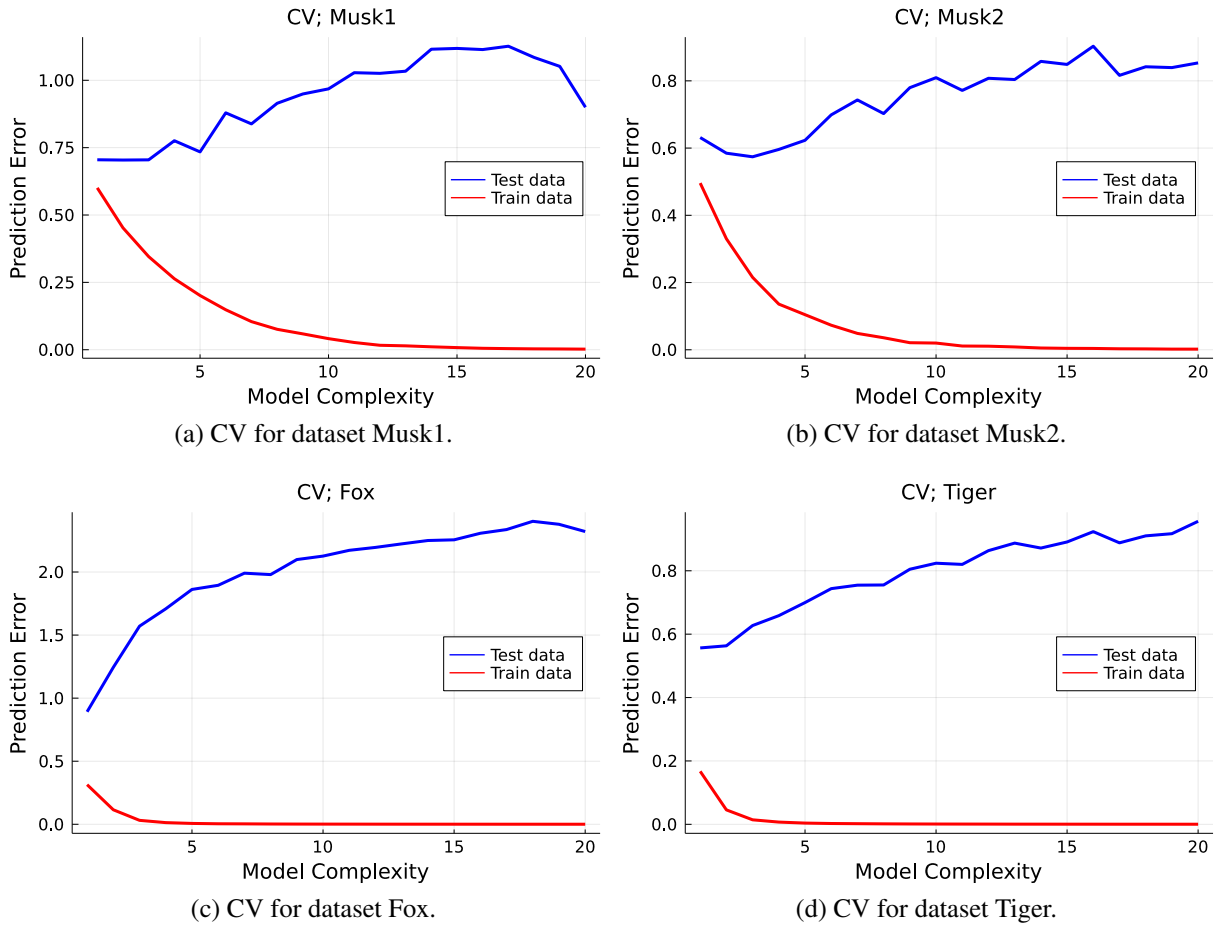

(a) CV for dataset Musk1.

(b) CV for dataset Musk2.

(c) CV for dataset Fox.

(d) CV for dataset Tiger.

Figure 5.3: Evaluation of prediction error with the use of training data and testing data on MIL datasets Musk1, Musk2, Fox and Tiger.

5.5 MIL to HDGM problem

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

Dataset	argmin $\widehat{\text{Err}}(z)$	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 5.1: Results of CV evaluated on the testing data.

5.5.1   Setup

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

$$\min_{\theta} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)} \left[\alpha \log \frac{\exp\left(f_{\theta}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{C} \exp\left(f_{\theta}\left(\boldsymbol{x}\right)[y_i]\right)} + (1 - \alpha) \log \frac{\exp\left(f_{\theta}\left(\boldsymbol{x}_i\right)[y]\right)}{\sum_{j=1}^{N} \exp\left(f_{\theta}\left(\boldsymbol{x}_j\right)[y]\right)} \right]. \quad (5.5)$$

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 parametr 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 $\alpha \in \{0.0, 0.1, 0.2, \ldots, 1.0\}$. Therefore estimated prediction error can be written in the form

$$\widehat{\text{Err}}(\alpha) = \frac{1}{r} \sum_{k=1}^{r} \mathcal{L}_k(z = 10, \alpha). \quad (5.6)$$

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

In the second part, evaluating of the prediction error is approached the other way. Fixed $\alpha = 0.5$ is selected and the dependancy on $z \in \{1, 2, 3 \ldots, 20\}$ is evaluated as in the section 5.4. Finally, estimated prediction error in this case is defined by

$$\widehat{\text{Err}}(z) = \frac{1}{r} \sum_{k=1}^{r} \mathcal{L}_k(z, \alpha = 0.5). \quad (5.7)$$

In other words, the setup is the same as in the 5.4, therefore obtained results will be added to the Figure 5.3 and table 5.1 for a convenient comparison. Note that curves for train data from this experiment will be omitted, since they are not important at this point. We are only interested in predictions for test data.

5.5.2 Results

Results of the first part are represented in Figure 5.4 and in Table 5.2, where can be seen that adding the generative term into the MIL loss function decreased the prediction error evaluated on all datasets. This improvement is considerable on datasets Musk1 and Musk2, where a can be seen a nice bowl. On datasets Fox and Tiger such improvement does not occur. This means that HDGM approach works, thus a regularization in the form of 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 5.2.

In the second part of the experiment results are shown in Figure 5.5 and in Table 5.3. Here is the situation very similar to the previous part of this experiment. Improvement of the prediction error is quite noticeable on first two datasets Musk1 and Musk2, while Fox and Tiger does not look so convincingly. In addition, number of random splits $r = 100$ is still needed to get rid of the noise on the prediction error.

Overall, it can be said that HDGM approach leads to the decreased prediction error in a small way.

Dataset	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 5.2: Prediction error statistics for HDGM in case of $z = 10$.

Dataset	**Discriminative part only**			**HDGM; $\alpha = 0.5$**		
	argmin $\widehat{\mathrm{Err}}(z)$	min $\widehat{\mathrm{Err}}(z)$	$\widehat{\mathrm{Err}}(z = 10)$	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 5.3: Comparison of prediction error statistics for HDGM $\alpha = 0.5$ and discriminative part only. Pay attention especially to the last column in each approach, $\widehat{\mathrm{Err}}(z = 10)$.

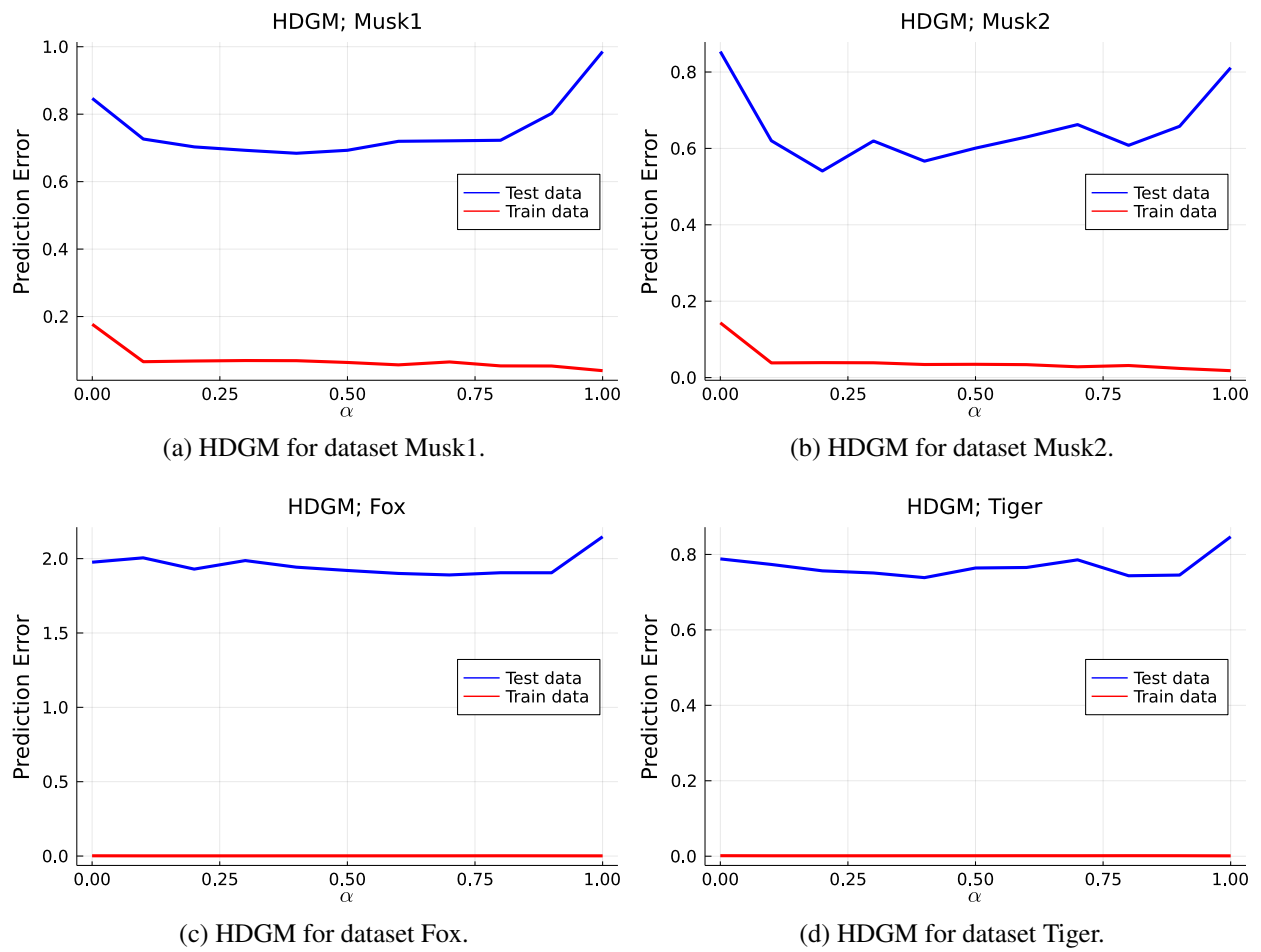(a) HDGM for dataset Musk1.

(b) HDGM for dataset Musk2.

(c) HDGM for dataset Fox.

(d) HDGM for dataset Tiger.

Figure 5.4: Evaluation of the prediction error $\widehat{\mathrm{Err}}(\alpha)$ 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.
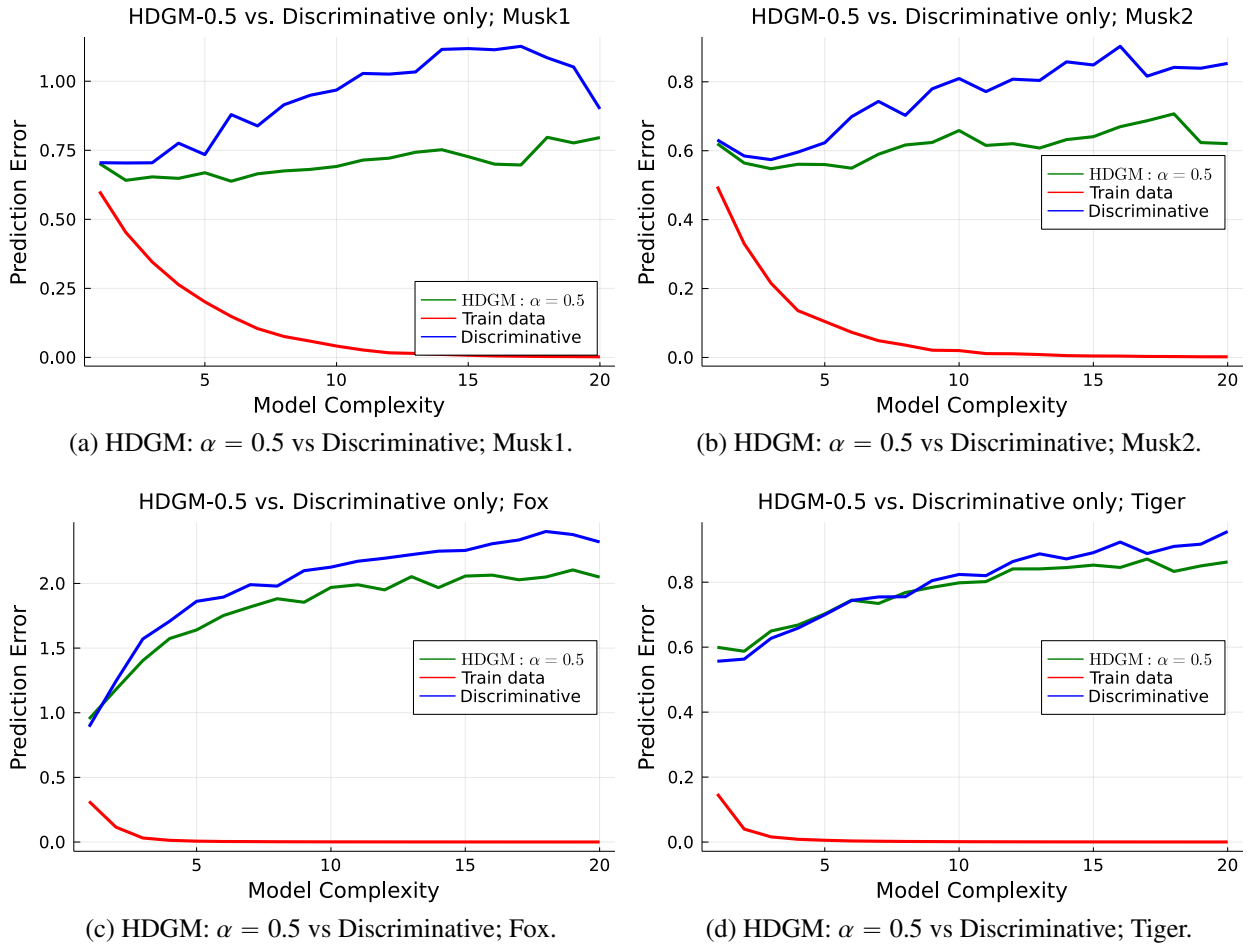
(d) HDGM: $\alpha = 0.5$ vs Discriminative; Tiger.

Figure 5.5: Comaparison of the prediction error $\widehat{\text{Err}}(z)$ for HDGM $\alpha = 0.5$ and discriminative part only.

Conclusion

At the begining of this work, supervised learning and energy-based models were introduced. Following passage consists of contrastive learning, which was briefly reviewed and thereafter supervised learning and contrastive learning was merged into a hybrid discriminative and generative models. Subsequently, this approach was applied and tested on a simple example consisting of polynomial regression.

After this example was briefly introduced multiple instance learning with a short description of embedded space and its way of training. As a first MIL experiment, cross–validation on 4 MIL datasets was performed, where was given enough space to a proper definition of the model complexity metrics. Obtained results were totally expected, it included decreasing prediction error for train data and increasing for test data. In the next step was searched for a solution, how to decrease the prediction error evaluated on test data. On this iniciative, HDGM was trained instead of a standard discriminative model. In consecutive result assessment was found out that HDGM leads to a decrease in prediction error, however nothing significant. In addition, high number of random splits of dataset was still necessary to get rid of the noise on the prediction error.

In conclusion, this approach was proven to be functional, although a proposed generative regularization is very simple and can be replaced by much complicated models. From this point of view, this approach has a great potential that has not yet been properly 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.