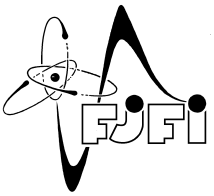
Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering



Generative and discriminative models for set data

Generativní a diskriminativní modely pro množinová data

Research Project

Author: **Bc. Jakub Bureš**

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

Academic year: 2020/2021

Acknowledgment:
I would like to thank doc. Ing. Václav Šmídl, Ph.D. for his expert guidance and patience during this academic year.

Author's declaration:
I declare that this Research Project is entirely my own work and I have listed all the used sources in the bibliography.

Prague, August 31, 2021 Bc. Jakub Bureš

Název práce:

Generativní a diskriminativní modely pro množinová data

Autor: Bc. Jakub Bureš

Obor: Celý název oboru (nikoliv zkratka)

Zaměření: Celý název zaměření (Pokud obor neobsahuje zaměření, tuto řádku odstranit.)

Druh práce: Výzkumný úkol

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

Abstrakt: Abstrakt max. na 10 řádků.

Klíčová slova: klíčová slova (nebo výrazy) seřazená podle abecedy a oddělená čárkou

Title:

Title of the Work

Author: Author's Name

Abstract: Max. 10 lines of English abstract text.

Key words: keywords in alphabetical order separated by commas

Contents

Introduction

Paragraphs of the Introduction...

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. 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. 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 $\boldsymbol{\theta}$ are the parameters of the model.
To contruct this prediction we need data, hence it is supposed that we have available set of observations $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, eventually, this may in fact be $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, where $\boldsymbol{x}_i \in \mathbb{R}^D$ and $\boldsymbol{y}_i \in \mathbb{R}^C \quad \forall i = 1, \ldots, N$, 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)

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

   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

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

   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 [9].

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 $\boldsymbol{y}^\star$ for the new input variable $\boldsymbol{x}^\star$. Note that the symbol $\mathcal{D}$ contains all of the previous given data $\boldsymbol{x}$ and $\boldsymbol{y}$. The posterior predictive distribution is then determined by distribution of the $\boldsymbol{y}^\star$, marginalized over the posterior

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

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

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

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

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

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

$$\mathcal{L}\left(\boldsymbol{y},\widehat{\boldsymbol{y}}\right) = \begin{cases} \left\|\boldsymbol{y}-\widehat{\boldsymbol{y}}\right\|_2^2 & \text{squared error} \\ \left\|\boldsymbol{y}-\widehat{\boldsymbol{y}}\right\| & \text{absolute error} \end{cases}. \tag{1.9}$$

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, model complexity 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.

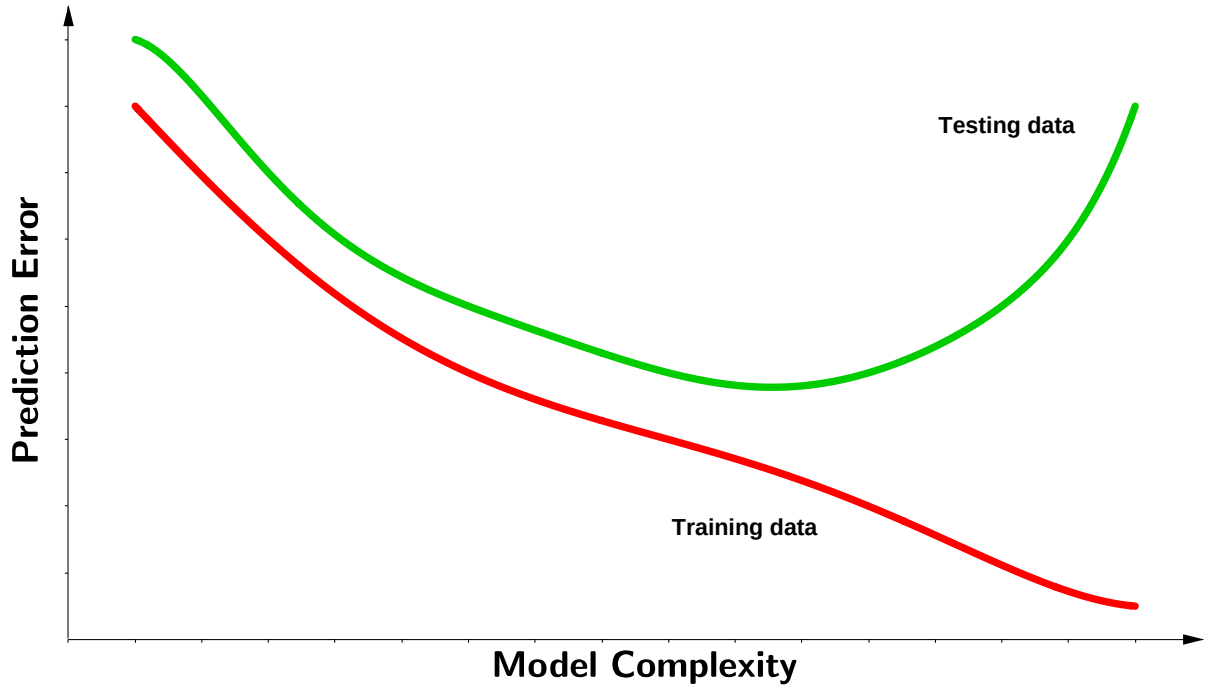


Figure 1.1: Evaluation of prediction error as a function of model complexity.

1.3 Cross-Validation

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

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

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. 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.

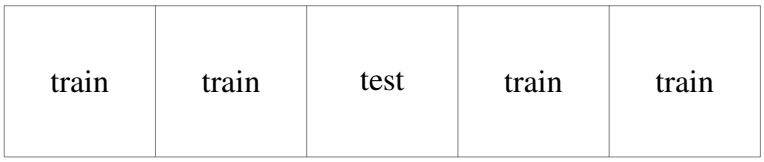


Figure 1.2: Splitting the data into $K = 5$ roughly equal-sized parts.

For the j^{th} 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 j^{th} 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 i is allocated by the randomization. Symbol $\hat{f}_{\theta}^{-j}(\boldsymbol{x})$ 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

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

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

Hybrid Generative and Discriminative models

2.1 Disriminative modeling

In this chapter, we review basics of discriminative modeling that was proposed in [12]. Given a data distribution in the form of 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 $f_{\boldsymbol{\theta}} : \mathbb{R}^D \to C$, where $\boldsymbol{\theta}$ denotes parameters of the model. In practice, function $f_{\boldsymbol{\theta}}$ is also used in the form of $\mathbb{R}^D \to \mathbb{R}^C$. 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 2.1.1. Logits are used to parametrize a categorical distribution via the function

$$q_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \frac{\exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y]\right)}{\sum_{i=1}^{C} \exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})[y_i]\right)}, \tag{2.1}$$

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

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{(x,y)\sim p_{\text{data}}}\left[\log q_{\boldsymbol{\theta}}(y|\boldsymbol{x})\right]. \tag{2.2}$$

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

$$D_{\text{KL}}(\Psi||\Pi) = \mathbb{E}_{x\sim\Psi}\left[\log\frac{\psi(x)}{\pi(x)}\right], \tag{2.3}$$

which can be further rewritten in the form

$$\mathbb{E}_{x\sim\Psi}\left[\log\frac{\psi(x)}{\pi_{\boldsymbol{\theta}}(x)}\right] = \mathbb{E}_{x\sim\Psi}\left[\log\psi(x)\right] - \mathbb{E}_{x\sim\Psi}\left[\log\pi_{\boldsymbol{\theta}}(x)\right], \tag{2.4}$$

where subscript $\boldsymbol{\theta}$ emphasizes that $\pi_\theta(x)$ is our approximative density we get to control. Finally, by minimizing with respect to $\pi_\theta(x)$ we obtain

$$\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}\left(\Psi||\Pi\right) = \min_{\boldsymbol{\theta}} -\mathbb{E}_{x\sim\Psi}\left[\log \pi_{\boldsymbol{\theta}}\left(x\right)\right] \tag{2.5}$$

2.1.1 One-hot encoding

Machine learning (ML) algorithms can misinteprete 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 2.1 and 2.2.
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 2.1: Example of label encoding.

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

Table 2.2: Example of one–hot encoding.

2.2 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

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right) = \frac{\exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x})\right)}{Z(\boldsymbol{\theta})}, \tag{2.6}$$

where function $E_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}$ 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

$$Z(\boldsymbol{\theta}) = \sum_{i=1}^{N} \exp\left(-E_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right), \tag{2.7}$$

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

$$p\left(\boldsymbol{x}, y\right) = \frac{\exp\left(f_\theta\left(\boldsymbol{x}\right)[y]\right)}{Z(\boldsymbol{\theta})}. \tag{2.8}$$

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

$$p\left(\boldsymbol{x}\right) = \frac{\sum_{i=1}^{C} \exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{Z(\boldsymbol{\theta})}, \tag{2.9}$$

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

$$p_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right) = \frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{C} \exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y_i]\right)}. \tag{2.10}$$

Note that the normalization constant $Z(\boldsymbol{\theta})$ canceled out and we ended up with the same function, which was introduced in (2.1).

2.3 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

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\log \frac{\exp\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right) \cdot h_{\boldsymbol{\theta}}\left(\boldsymbol{x}'\right)\right)}{\sum_{i=1}^{N} \exp\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right) \cdot h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)}\right] \tag{2.11}$$

Function $h_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}^H$ maps each data point to a representation space with dimension H, while $\boldsymbol{x}$ and $\boldsymbol{x}'$ are two different augmented views of the same data point. If $\boldsymbol{x}$ is a image, then augmented view of $\boldsymbol{x}$ 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 $\boldsymbol{x}_i$ from alternative input $\boldsymbol{x}_i'$. This means that the model should be able to distinguish between different types of images without even knowing what these images really are.

2.4 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 [] 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 $\boldsymbol{x}$ to classes y. In addition, 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

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)} \left[\log q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right) + \log q_{\boldsymbol{\theta}}\left(\boldsymbol{x}|y\right)\right], \tag{2.12}$$

where the first term

$$q_{\boldsymbol{\theta}}\left(y|\boldsymbol{x}\right) = \frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{C} \exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y_i]\right)} \tag{2.13}$$

is a standard Softmax neural net classifier (as mentioned in Equation (2.1)) and the second term

$$q_{\boldsymbol{\theta}}\left(\boldsymbol{x}|y\right) = \frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{j=1}^{N} \exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_j\right)[y]\right)}, \tag{2.14}$$

is a contrastive loss attributed (2.11) label. This objective often struggles with the unknown partition function $\sum_{j=1}^{N} \exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_j\right)[y]\right)$, which is often intractable, specifically if the number of datapoints is very high. This obstacle is typically addressed using an approximation

$$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)} \left[\log q_{\boldsymbol{\theta}}\left(\boldsymbol{x}|y\right)\right] \tag{2.15}$$

$$= \mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)} \left[\log \frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{j=1}^{N} \exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}\right] \tag{2.16}$$

$$\approx \mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)} \left[\log \frac{\exp\left(f_{\theta}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{j=1}^{M} \exp\left(f_{\theta}\left(\boldsymbol{x}_j\right)[y]\right)}\right], \tag{2.17}$$

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 (2.15) to Equation (2.12) that gives a hybrid combination of supervised learning and constrastive learning in the form

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)} \left[\alpha \log q_\theta \left(y|\boldsymbol{x}\right) + (1-\alpha) \log q_\theta \left(\boldsymbol{x}|y\right)\right] \tag{2.18}$$

$$\approx \min_{\boldsymbol{\theta}} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)} \left[\alpha \log \frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{C} \exp\left(f_\theta\left(\boldsymbol{x}\right)[y_i]\right)} + (1-\alpha) \log \frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)[y]\right)}{\sum_{i=1}^{M} \exp\left(f_\theta\left(\boldsymbol{x}_i\right)[y]\right)}\right]. \tag{2.19}$$

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. Hybrid combination of supervised learning and contrastive learning is for us absolutely crucial.....

2.5 Toy problem - Polynomial Regression

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 (2.18) 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 2.2, we know that for joint distribution it holds

$$p(x,y) = \frac{\exp\left(f_\theta(x)[y]\right)}{Z(\boldsymbol{\theta})}, \tag{2.20}$$

where the model model is given by

$$f_{\boldsymbol{\theta}}(x)[y] = -E_{\boldsymbol{\theta}}(x,y). \tag{2.21}$$

At this point, we should transform our problem to the polynomial regression. We need to be aware of the discriminative term in the Equation (2.12), because we do not want to clasify, but we would like to find the best fit to the given data. For this reason, we replace that with the typical regression loss

$$S = S\left(\boldsymbol{\theta}\right) = \sum_{k=1}^{N} \left(y_k - \sum_{i=0}^{s-1} \theta_i x_k^i\right)^2. \tag{2.22}$$

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

$$p(x,y) = p(y,x) = p(y|x) \cdot p(x), \tag{2.23}$$

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

$$p(y|x,\boldsymbol{\theta}) = \mathcal{N}\left(\sum_{i=0}^{s-1} \theta_i x^i, \sigma^2\right), \tag{2.24}$$

where symbol $\mathcal{N}(\cdot)$ probability density fucntion 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

$$p(x|\tau) = \mathcal{N}\left(0, \tau^2\right), \tag{2.25}$$

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. Once the value of τ is high, data are spread very far from their expected value. Subtituting Equation (2.25) and (2.24) to the (2.23) results to

$$p(x,y) = \mathcal{N}\left(0,\tau^2\right) \cdot \mathcal{N}\left(\sum_{i=0}^{s-1} \theta_i x^i, \sigma^2\right) = \frac{1}{2\pi\sigma\tau} \exp\left(-\frac{\left(y - \sum_{i=0}^{s-1} \theta_i x^i\right)^2}{2\sigma^2} - \frac{x^2}{2\tau^2}\right), \tag{2.26}$$

whereas our desirable model is given by

$$f_\theta(x)[y] = -\frac{\left(y - \sum_{i=0}^{s-1} \theta_i x^i\right)^2}{2\sigma^2} - \frac{x^2}{2\tau^2}. \tag{2.27}$$

We can now substite (2.27) and (2.22) into Equation (2.12), yielding

$$\min_{\boldsymbol{\theta}} \left\{ \alpha\, S(\boldsymbol{\theta}) - \mathbb{E}_{p_{\text{data}}(x,y)} \left[(1-\alpha) \log q_\theta(x|y) \right] \right\} = \tag{2.28}$$

$$\min_{\boldsymbol{\theta}} \left\{ \alpha\, S(\boldsymbol{\theta}) - \mathbb{E}_{p_{\text{data}}(x,y)} \left[(1-\alpha) \log \frac{\exp\left(f_\theta(x)[y]\right)}{\sum_{i=1}^{N} \exp\left(f_\theta(x_i)[y]\right)} \right] \right\} = \tag{2.29}$$

$$\min_{\boldsymbol{\theta}} \left\{ \alpha\, S(\boldsymbol{\theta}) - \mathbb{E}_{p_{\text{data}}(x,y)} \left[(1-\alpha) \log \frac{\exp\left(-\frac{\left(y-\sum_{i=0}^{s-1} \theta_i x^i\right)^2}{2\sigma^2} - \frac{x^2}{2\tau^2}\right)}{\sum_{k=1}^{N} \exp\left(-\frac{\left(y-\sum_{i=0}^{s-1} \theta_i x_k^i\right)^2}{2\sigma^2} - \frac{x_k^2}{2\tau^2}\right)} \right] \right\}. \tag{2.30}$$

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

$$\log q_\theta(x|y) = \left(-\frac{\left(y - \sum_{i=0}^{s-1} \theta_i x^i\right)^2}{2\sigma^2} - \frac{x^2}{2\tau^2}\right) - \log \sum_{k=1}^{N} \exp\left(-\frac{\left(y - \sum_{i=0}^{s-1} \theta_i x_k^i\right)^2}{2\sigma^2} - \frac{x_k^2}{2\tau^2}\right). \tag{2.31}$$

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 perform the experiment.

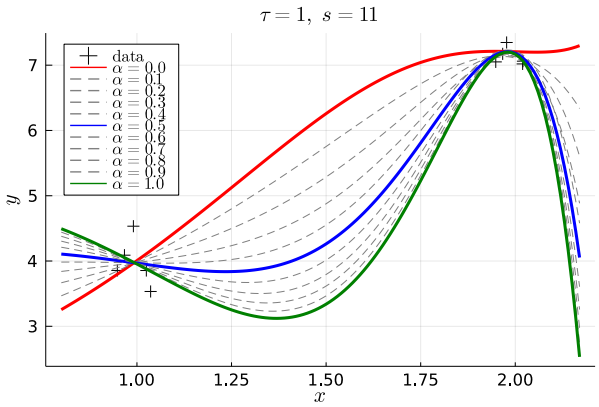
2.5.1 Results

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.

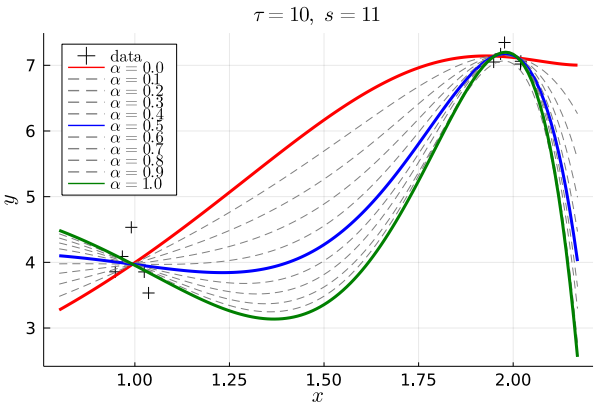
We generated synthetic data, two clusters consisting of 5 datapoints each, then the model was fitted for different weight $\alpha \in \{0.0, 0.1, 0.2, \ldots, 1.0\}$, giving 11 different models in total. Models estimated for $\alpha \in \{0.0, 0.5, 1.0\}$ are highlighted as they are more important to 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 2.1) 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 (2.25). This is very exciting discovery, because there is no need to know much information about the data distribution.

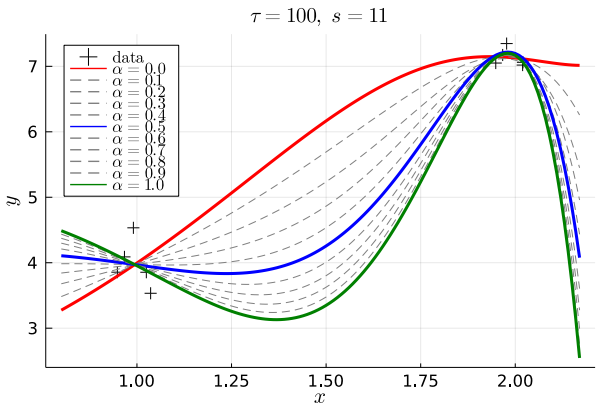
Secondly, we trained our polynomial models for the fixed value of τ with 6 different values of the order of the polynomial. The goal of this part of the experiment is to observe how the contrastive part of Equation (2.28) affects the polynomial regression loss for different values of $s-1$. As can be seen in Figure 2.2, 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 oscilate. This also could be very interesting, because combining of discriminative and generative models could result in better model predictions.



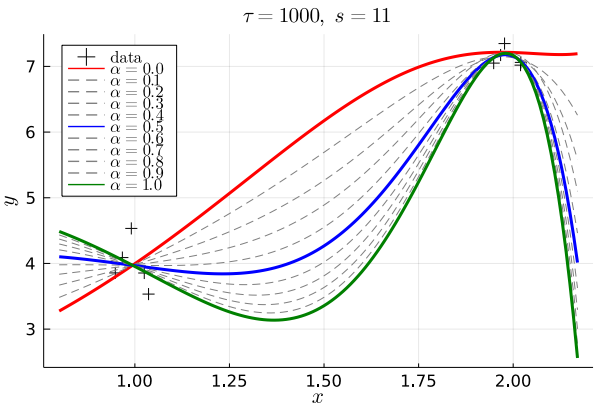
(a) Estimated model for $\tau = 1$ and $s = 11$.



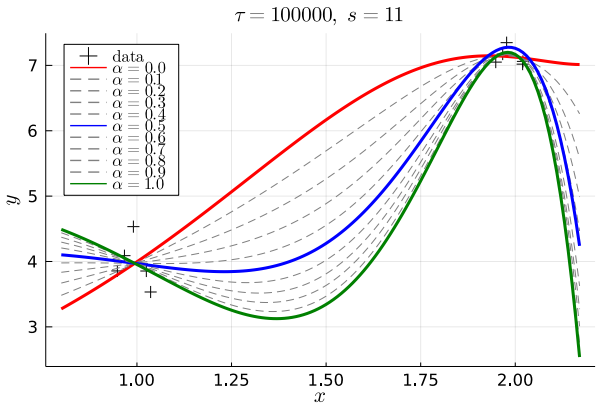
(b) Estimated model for $\tau = 10$ and $s = 11$.



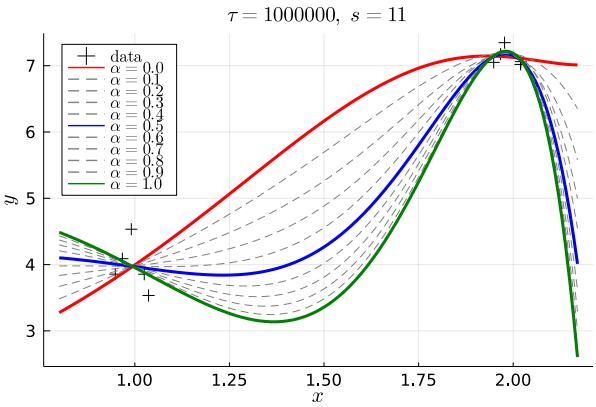
(c) Estimated model for $\tau = 100$ and $s = 11$.



(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 2.1: Sensitivity of the polynomial model to parameter τ with parameter s held fixed.

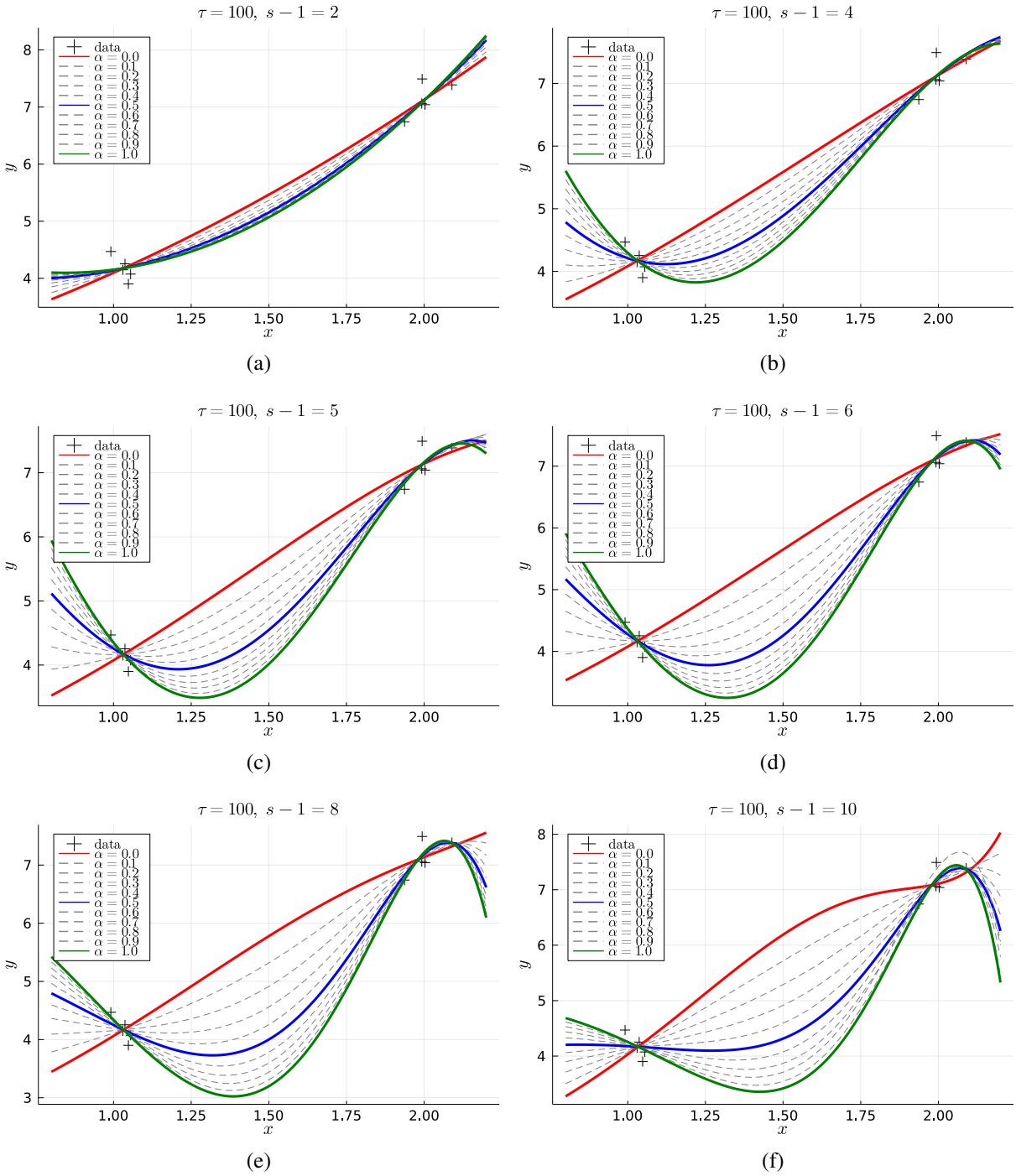


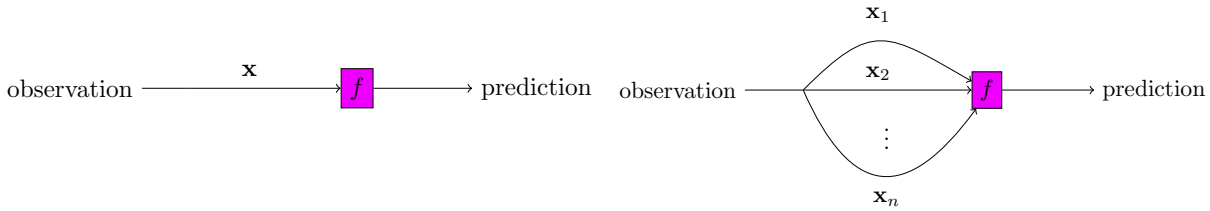
Figure 2.2: Sensitivity of the polynomial model to the order of polynomial $s - 1$.

Chapter 3

Multiple Instance Learning

3.1 Fundamentals

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



(a) Standard machine learning

(b) Multiple instance learning

Figure 3.1: The difference between standard ML and MIL [2]. Standard ML is special case of MIL with $|b| = 1$.

3.2 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) and we refer to [2] for more details about the model definition.
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

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{(\boldsymbol{x},y)\sim p_{\text{data}}} \left[\log \frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)\left[y\right]\right)}{\sum_{i=1}^{C} \exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)\left[y_i\right]\right)} \right], \tag{3.1}$$

already mentioned in section 2.1. This is very imporant for us and we will take advantage of that in our experiment.

3.3 K-fold CV 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.
Model Complexity As was mentioned in section 3.2, 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 was selected as model complexity metric, 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 3.2, however this is not the exact NN used in HMill.

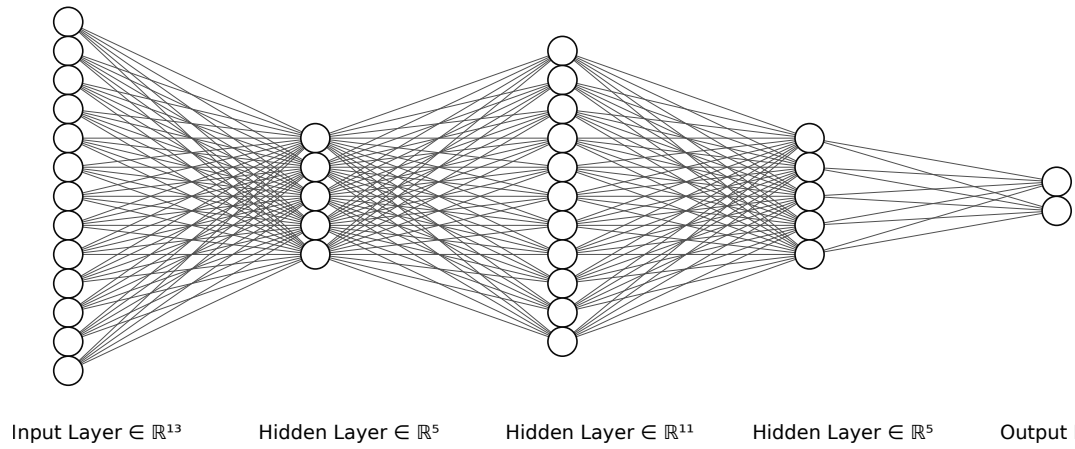


Figure 3.2: NN example for $z = 5$, where input and output layer are only illustrative.

Prediction Error For prediction error metrics will be simply used standard cross–entropy loss 2.2 as there is no need for any trickier objective.

3.3.1 Results

As can be seen in Figure 3.3, the prediction error evaluated on the training data for higher model complexity approaches zero. However, testing data give an oscilating curve and therefore the model selection is neccessary. In addition, following table 3.1 summarizes the results evaluated on the training data for number of folds $K = 5$.

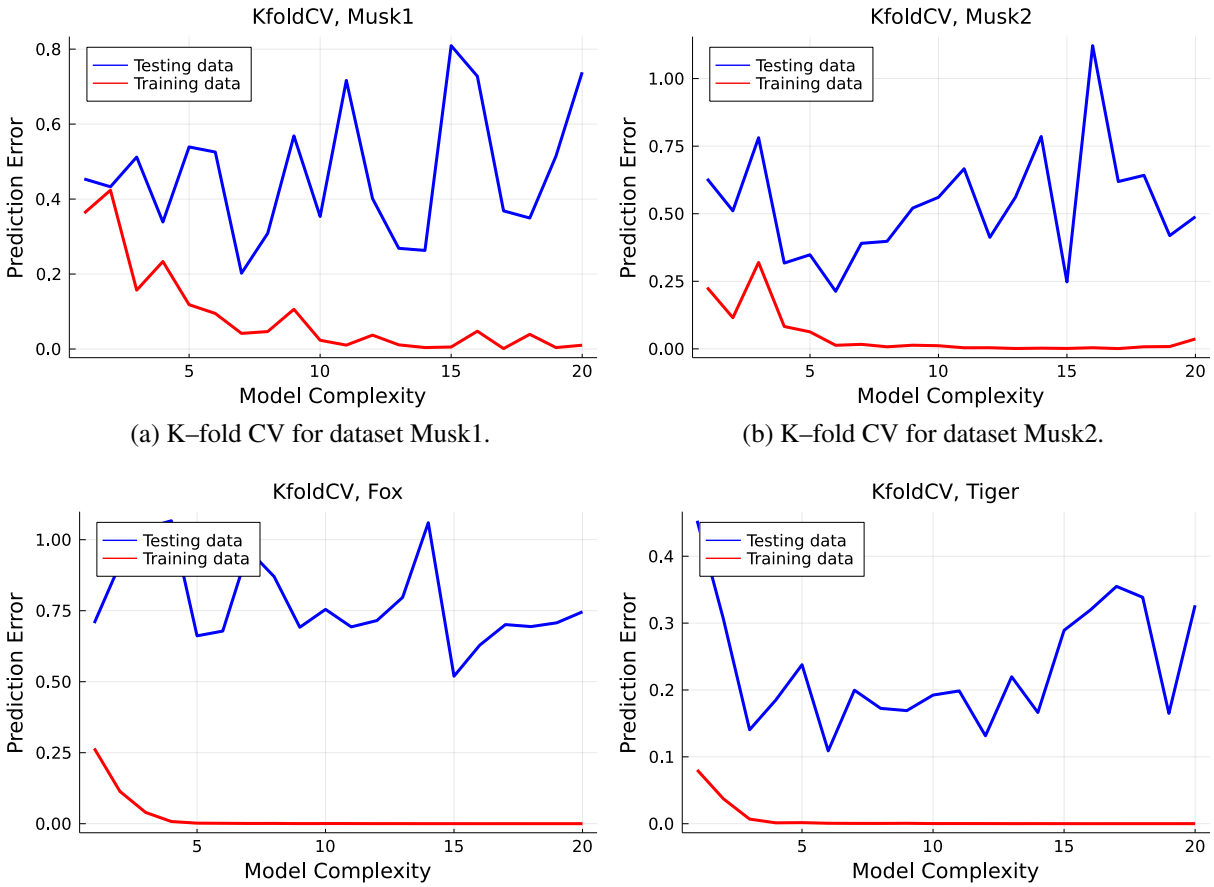
Dataset	Accuracy(%)	argmin L	min L
Musk1	88.8	7	0.20
Musk2	84.0	6	0.23
Fox	90.0	15	0.51
Tiger	95.5	6	0.11

Table 3.1: Results of 5-fold CV evaluated on the training data.

3.4 MIL to HDGE experiment

Assesing model performance and subsequent model selection via K-fold cross–validation may be computionally expensive. For this reason, we propose to train only 1 model that is minimizing the hybrid loss function

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},y)} \left[\alpha \log \frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)[y]\right)}{\sum_{i=1}^{C} \exp\left(f_{\theta}\left(\boldsymbol{x}\right)[y_i]\right)} + (1-\alpha) \log \frac{\exp\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)[y]\right)}{\sum_{i=1}^{N} \exp\left(f_{\theta}\left(\boldsymbol{x}_i\right)[y]\right)} \right] \tag{3.2}$$



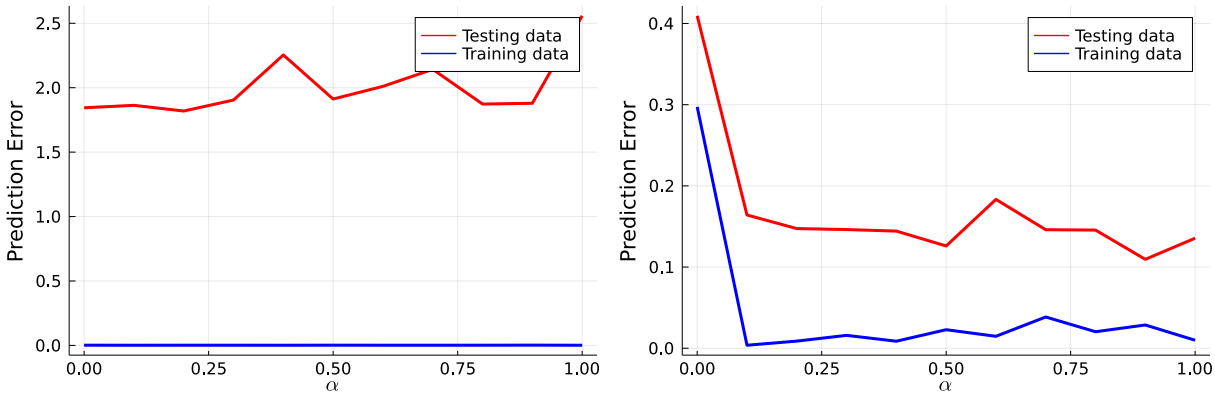
(a) K–fold CV for dataset Musk1.

(b) K–fold CV for dataset Musk2.

(c) K–fold CV for dataset Fox.

(d) K–fold CV for dataset Tiger.

Figure 3.3: Evaluation of loss function with the use of training data and testing data on MIL datasets Musk1,Musk2, Fox and Tiger.



(a) HDGM for dataset Musk1.

(b) HDGM for dataset Musk2.

Figure 3.4: Evaluation of hybrid loss function with the use of training data and testing data on MIL datasets Musk1 and Musk2.

Conclusion

Text of the conclusion...

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.