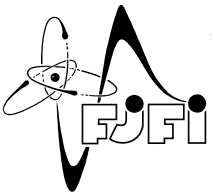
Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering



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.**

Language advisor: **Mgr.**

Academic year: 2020/2021

Acknowledgment:
I would like to thank ..... for (his/her expert guidance) and express my gratitude to ..... for (his/her language assistance).

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

Prague, June 1, 2021 Jméno Autora

Název práce:

Název práce

Autor: Jméno Autora

Obor: Celý název oboru (nikoliv zkratka)

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

Druh práce: Bakalářská práce

Vedoucí práce: prof. Ing. Jméno Školitele, DrSc., pracoviště školitele (název instituce, fakulty, katedry...)

Konzultant: doc. RNDr. Jméno Konzultanta, CSc., pracoviště konzultanta. Pouze pokud konzultant byl jmenován.

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. Max.

10 lines of English abstract text. Max. 10 lines of English abstract text. Max. 10 lines of English abstract text.

Key words: keywords in alphabetical order separated by commas

Contents

Introduction

Paragraphs of the Introduction...

Terminology

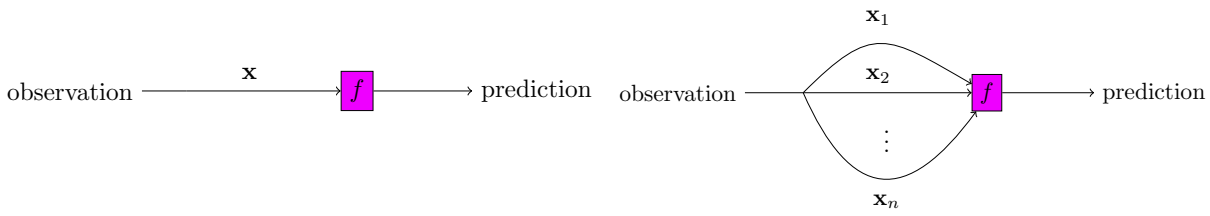
At the beginning of this work for the sake of avoiding confusion, it is appropriate to clarify the terminology.
We will use typical notation for random variables via uppercase letters, where, most of the time, input variable will be denoted by the symbol X and output variable will be denoted by the symbol Y. The realization of a random variable, observed value, or simply observation, will be denoted by correspodning lowercase letters x, y. Bold symbols $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{x}, \boldsymbol{y}, \ldots$ will be used to distinguish vectors from scalars.
Usual goal (learning task) is to make a good prediction of the output $\boldsymbol{Y}$, denoted by the symbol $\hat{\boldsymbol{Y}}$, with given input $\boldsymbol{X}$. This prediction is obtained through learning a model f_θ that minimizes a loss function $\mathcal{L}(f_\theta(x), y)$. To contruct this prediction we need data, hence it is supposed that we have available set of observations $\{(x_i, y_i)\}_{i=1}^N$, eventually $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$. Observations are considered to be independent and indentically distributed, often abbreviated as i.i.d.

Chapter 1

Multiple Instance Learning

1.1 Fundamentals

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



(a) Standard machine learning

(b) Multiple instance learning

Figure 1.1: The difference between standard ML and MIL [**?**]. Standard ML is special case of MIL with $|b| = 1$.

1.2 Cross-Validation

Probably the simplest and most widely used method for estimating prediction error is *cross-validation*. This method directly estimates the expected extra-sample error

$$\text{Err} = \mathbb{E}\left[\mathcal{L}\left(Y, \hat{f}\left(X\right)\right)\right], \tag{1.1}$$

the average generalization error when the method $\hat{f}\left(X\right)$ is applied to an independent test sample form the joint distribution of X and Y- As mentioned earlier, we might hope that cross-validation estimates the conditional error, with the training set $\mathcal{T}$ held fixed. But cross-validation typically estimates well only the expected prediction error.

K-Fold Cross-Validation

Ideally, if we had enough data, we would set aside a validation set and use it to assess the performance of our prediction model. Since data are often scarce, this is usually not possible. To finesse the problem, K-fold cross-validation uses part of the available data to fit the model, and a different part to test it. We split the data into K roughly equal-sized parts; for example, when $K = 5$, the scenario could look like this

train	train	validation	train	train

For the kth part (third above), we fit the model to the other $K - 1$ parts of the data, and calculate the prediction error of the fitted model when predicting the kth part of the data. We do this for $k = \{1, 2, \ldots, K\}$ and combine the K estimates of prediction error.
Here are more details. Let $\kappa : \{1, \ldots, N\} \rightarrow \{1, \ldots, K\}$ be an indexing function that indicates the partition to which observation i is allocated by the randomization. Denote by $\hat{f}^{-k}\left(x\right)$ the fitted function, computed with the kth part of the data removed. Then the cross-validation estimate of prediction error is

$$\text{CV}\left(\hat{f}\right) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\left(y_i, \hat{f}^{-\kappa(i)}\left(x_i\right)\right). \tag{1.2}$$

Typical choices of K are 5 or 10 and even case $K = N$, which is known as *leave-one-out* cross-validation. Note that there is not an universal way of choosing K, since it strongly depends on the available number of data.

1.3 Experiment with mill datasets

Suppose we have two classes 0 and 1 (known as binary classification), which means that bags are labeled either as 0 or 1. What happens, if we have many more bags, for example, labeled as

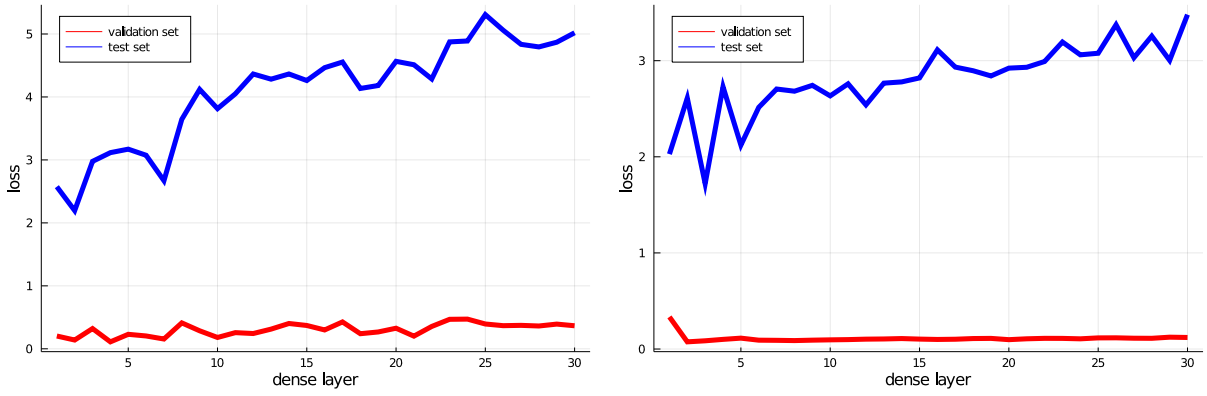
class 1? This situation is very common in anomaly detection, where known anomalies are quite rare.
Let's assume train set is composed of 80% bags labeled as 1 and 5% bags labeled as 0, all randomly chosen. Test set is composed of 20% bags labeled as 1 and 95% labeled as 0, in other words it is complement of train set. Validation set is very similar to train set in terms of ratios, it contains 20% bags labeled as 1 and 2% bags labeled as 0. Train set is used to train our model, after that we evaluate loss function of the model with help of validation a test set, where number of dense layers is our hyperparameter. The purpose of this simulation is to find number of dense layer in which the loss is minimal and compare these 2 values. This experiment was performed 5 times then results were averaged, totally on 6 different datasets.

As we can see in Figure 1.2, the loss evaluated on different sets varies therefore it is likely to make mistakes when choosing our hyperparameter if we don't have enough input data.

1.4 Experiment with Point Cloud Mnist

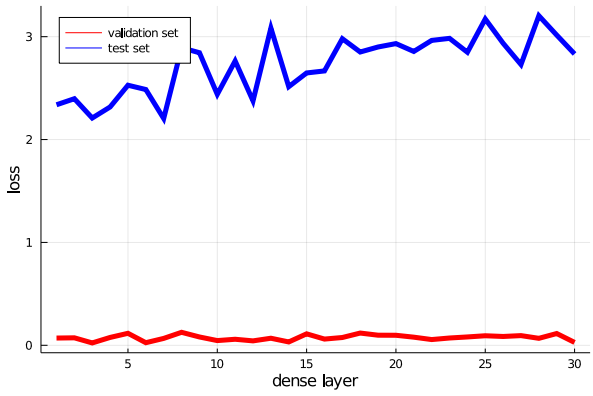
We applied previous method on Point Cloud Mnist 2D (PCM) dataset. However, we needed to make a little adjustment to this dataset, because we perform the binary classification. Originally, PCM has tottaly 10 labels (labels are numbers 0-9), so we separated arbitrarily 8 of them to obtain just 2 classes. In addition, we randomly split data into bags.
The advantage of PCM is that we can split the data into bags in many ways, e.g. 0&1 , 2&3, 5&7,... we can even take 0&1 as first class and, for example, 2&3 as a second class.

labels	difference
0&1	0.81
2&3	2.07
4&5	2.05
6&7	1.99
8&9	1.91

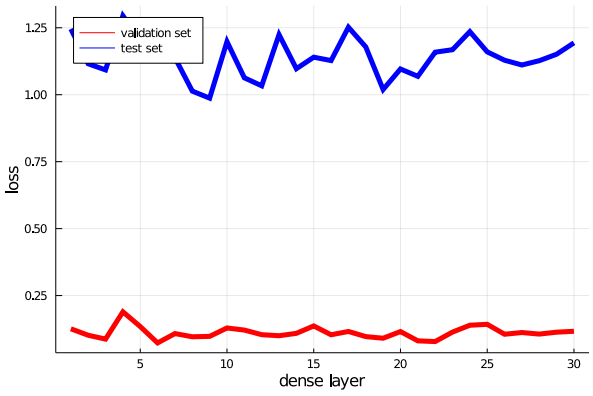


(a)

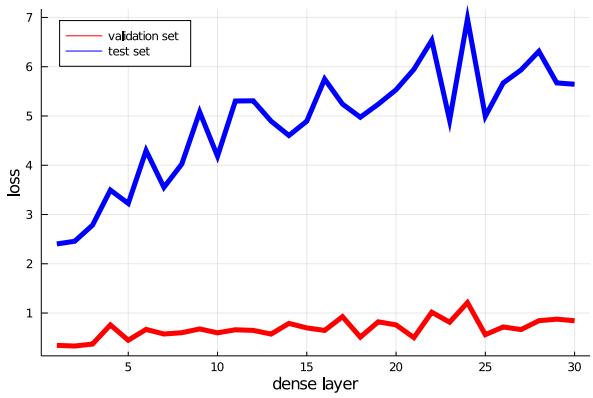
(b)



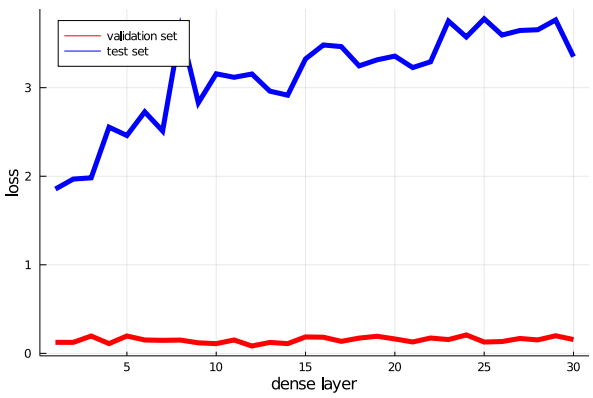
(c)



(d)



(e)



(f)

Figure 1.2: Evaluation of loss function with the use of validation and test set on different MILL datasets.

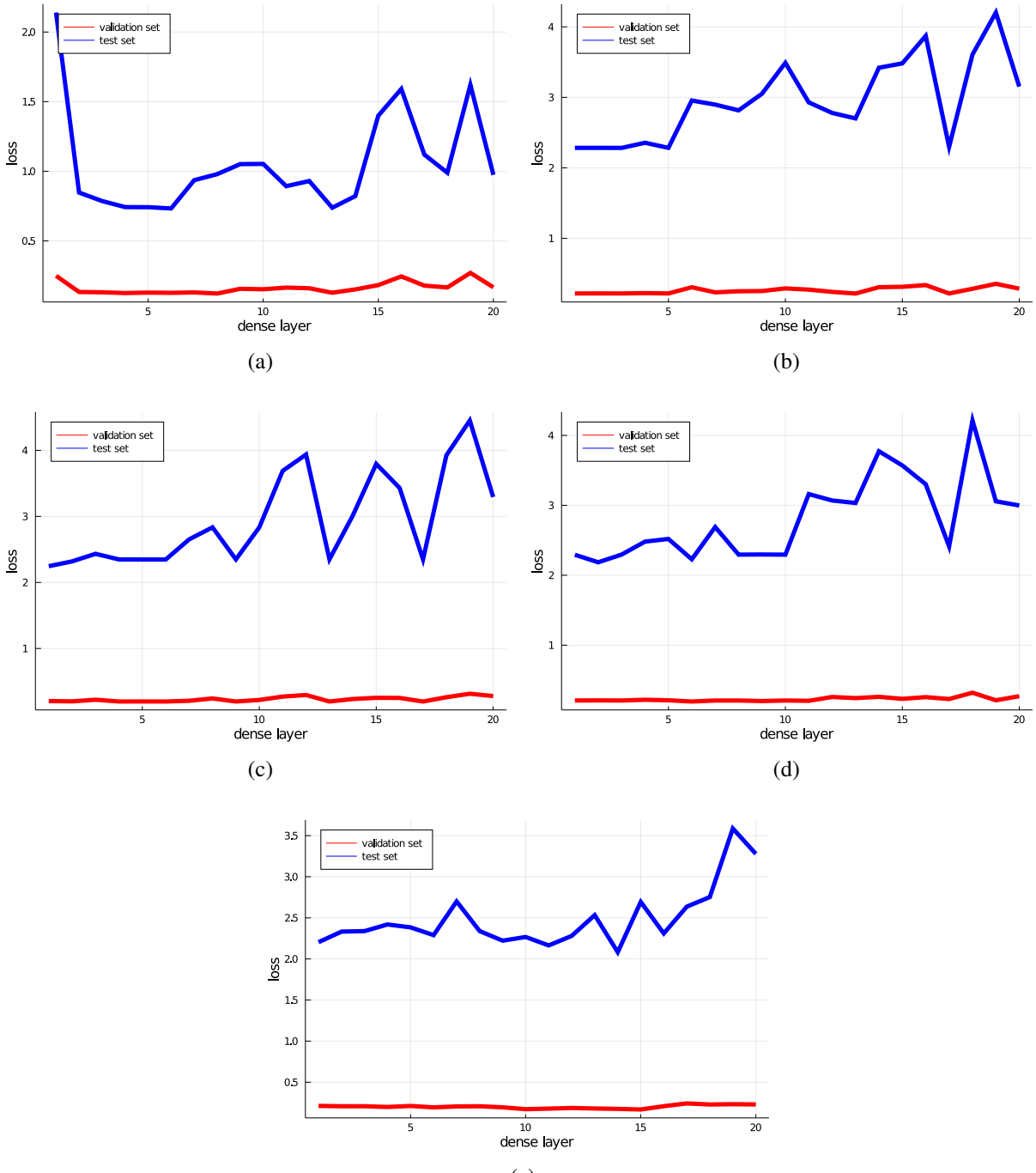


Figure 1.3: Evaluation of loss function with the use of validation and test set on Point Cloud Mnist

Chapter 2

Hybrid Generative and Discriminative models

2.1 Background

Given a data distribution in the form of probability density $p(\boldsymbol{x})$ and a label distribution with probability density $p(y|x)$ containing C categories. A classification problem is typically solved using a parametric function $f_\theta : \mathbb{R}^D \to \mathbb{R}^C$. This function maps each data point $x \in \mathbb{R}^D$ to C real-valued numbers known as logits. They are used to parametrize a categorical distribution using the function

$$q_\theta(y|x) = \frac{\exp\left(f_\theta(x)[y]\right)}{\sum_{y'} \exp\left(f_\theta(x)[y']\right)}, \tag{2.1}$$

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

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

Rationale for this objective comes from minimizing the Kullback-Leibler divergence with a target distribution $p(y|x)$ [**?**]. Since Kullback-Leibler divergence for two distributions P and Q with corresponding probablity density functions p and q is defined as

$$D_{\text{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\log \frac{p(X)}{q(X)}\right],$$

which can be further rewritten in the form

$$\mathbb{E}_{x\sim P}\left[\log \frac{p(X)}{q_\theta(X)}\right] = \mathbb{E}_{x\sim P}\left[\log p(X)\right] - \mathbb{E}_{x\sim P}\left[\log q_\theta(X)\right]$$

where subscript θ emphasizes that $q_\theta(X)$ is our approximative distribution we get to control. Finally, by minimizing with respect to $q_\theta(X)$ we obtain

$$\min_\theta D_{\text{KL}}(P||Q) = \min_\theta -\mathbb{E}_{x\sim P_{\text{data}}}\left[\log q_\theta(X)\right]$$

2.2 Energy-Based Models

Energy-based models assume that probability densities $p(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{R}^D$ can be expressed in the form

$$p_\theta(\boldsymbol{x}) = \frac{\exp(-E_\theta(\boldsymbol{x}))}{Z(\theta)}, \tag{2.3}$$

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

$$Z(\theta) = \sum_{\boldsymbol{x}} \exp(-E_\theta(\boldsymbol{x})), \tag{2.4}$$

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 [**?**] (also see for further details), where they show that classifiers in supervised learning are secretly energy-based models on $p(\boldsymbol{x}, y)$ and can be expressed as

$$p(\boldsymbol{x}, y) = \frac{\exp(f_\theta(\boldsymbol{x})[y])}{Z(\theta)}. \tag{2.5}$$

It is obvious that $f_\theta(\boldsymbol{x})[y] = -E_\theta(\boldsymbol{x}, y)$. It could 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

$$p(\boldsymbol{x}) = \frac{\sum_y \exp(f_\theta(\boldsymbol{x})[y])}{Z(\theta)}, \tag{2.6}$$

where energy is given by $E_\theta(\boldsymbol{x}) = -\log \sum_y \exp(f_\theta(\boldsymbol{x})[y])$.

2.3 Contrastive learning

To briefly introduce contrastive learning [**?**] let's just mention that it is very usual to optimize an objective, which can be written in the form as follows

$$\min_\theta -\mathbb{E}_{p_{\text{data}}(x)} \left[\log \frac{\exp(h_\theta(\boldsymbol{x}) \cdot h_\theta(\boldsymbol{x}'))}{\sum_{i=1}^K \exp(h_\theta(\boldsymbol{x}) \cdot h_\theta(\boldsymbol{x}_i))} \right]. \tag{2.7}$$

Function $h_\theta : \mathbb{R}^D \to \mathbb{R}^H$ 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. Note that the inner product between two vectors can replaced with any distance metric, for example Euclidean distance.
What this objective does is basically that it tries to maximally distinguish an input $\boldsymbol{x}_i$ from alterantive inputs $\boldsymbol{x}'_i$. The denominator in the expression (2.7) is for $K \to \infty$ equal to the partition function $Z(\theta)$ and

2.4 Hybrid Dicriminative and Generative Models

In this chapter will be discussed a way, how to combine both of the already mentioned models. Authors of the arcticle [**?**] proposed a solution, however the rationale of this objective originates from [**?**], where authors show that hybrid models can outperform purely generative or purely discriminative counterparts.
The primary goal is to train a model that can classify x to y. In addition, learned models should be capable of out-of-distribution detection and could generate. 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

$$\min_{\theta} -\mathbb{E}_{p_{\text{data}}(x,y)} \left[\log q_\theta\left(y|x\right) + \log q_\theta\left(x|y\right)\right], \tag{2.8}$$

where the first term $q_\theta\left(y|x\right) = \frac{\exp(f_\theta(x)[y])}{\sum_{y'} \exp(f_\theta(x)[y'])}$ is a standard Softmax neural net classifier (as mentioned in Equation (2.1)) and the second term

$$q_\theta\left(x|y\right) = \frac{\exp\left(f_\theta\left(x\right)[y]\right)}{\sum_x \exp\left(f_\theta\left(x\right)[y]\right)}. \tag{2.9}$$

This objective has issues with the unknown partition function $\sum_x \exp\left(f_\theta\left(x\right)[y]\right)$, which is often intractable. To overcome this obstacle, it is proposed a approximation with a contrastive loss (2.7)

$$\mathbb{E}_{p_{\text{data}}(x,y)} \left[\log q_\theta\left(x|y\right)\right] \tag{2.10}$$

$$= \mathbb{E}_{p_{\text{data}}(x,y)} \left[\log \frac{\exp\left(f_\theta\left(x\right)[y]\right)}{\sum_x \exp\left(f_\theta\left(x\right)[y]\right)}\right] \tag{2.11}$$

$$\approx \mathbb{E}_{p_{\text{data}}(x,y)} \left[\log \frac{\exp\left(f_\theta\left(x\right)[y]\right)}{\sum_{i=1}^{K} \exp\left(f_\theta\left(x_i\right)[y]\right)}\right], \tag{2.12}$$

where K denotes the number of normalization samples and it has to be sufficiently large - becoming exact in the limit of summing over all $x \in \mathcal{X}$. Now it is possible to insert this approximation to Equation (2.8), which gives a hybrid combination of supervised learning and constrastive learning in the form

$$\min_{\theta} -\mathbb{E}_{p_{\text{data}}(x,y)} \left[\alpha \log q_\theta\left(y|x\right) + (1-\alpha) \log q_\theta\left(x|y\right)\right] \tag{2.13}$$

$$= \min_{\theta} -\mathbb{E}_{p_{\text{data}}(x,y)} \left[\alpha \log \frac{\exp\left(f_\theta\left(x\right)[y]\right)}{\sum_{y'} \exp\left(f_\theta\left(x\right)[y']\right)} + (1-\alpha) \log \frac{\exp\left(f_\theta\left(x\right)[y]\right)}{\sum_{i=1}^{K} \exp\left(f_\theta\left(x_i\right)[y]\right)}\right]. \tag{2.14}$$

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 (2.2), while $\alpha = 0$, obejctive 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 user, however authors in [**?**] evaluated many possible variants in experiments and found out that the choice of $\alpha = 0.5$ yields to the highest performance on a classification accuracy, robustness, calibration and out-of-distribution detection.

2.5 Toy problem - simple linear regression

We would like to test hybrid discriminative and generative approach on simple example before we head into more difficult cases. The goal is to train model of the form 2.13 that was derived in previous section

$$\min_{\theta} -\mathbb{E}_{p_{\text{data}}(x,y)} \left[\alpha \log \frac{\exp\left(f_\theta\left(x\right)\left[y\right]\right)}{\sum_{y'} \exp\left(f_\theta\left(x\right)\left[y'\right]\right)} + (1-\alpha) \log \frac{\exp\left(f_\theta\left(x\right)\left[y\right]\right)}{\sum_{i=1}^{K} \exp\left(f_\theta\left(x_i\right)\left[y\right]\right)} \right]. \tag{2.15}$$

According to energy-based model, we know that

$$p(x,y) = \frac{\exp\left(f_\theta(x)[y]\right)}{Z(\theta)} \tag{2.16}$$

where

$$f_\theta(x)[y] = -E_\theta(x,y) \tag{2.17}$$

and for partition function $Z(\theta)$, it holds

$$Z(\theta) = \sum_x \sum_y \exp\left(-E_\theta(x,y)\right). \tag{2.18}$$

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

$$p(x,y) = p(y,x) = p(y|x) \cdot p(x), \tag{2.19}$$

from simple linear regression we can get probability density

$$p(y|x) = \mathcal{N}\left(\theta_0 + \theta_1 x, \sigma^2\right) \tag{2.20}$$

and aprior density $p(x)$ is considered to be known. For sake of simplicity it is set to be

$$p(x) = \mathcal{N}\left(0, \sigma^2\tau^2\right), \tag{2.21}$$

where σ is known paramater and parameter τ is chosen due to the need for σ adjustment. It results to

$$p(x,y) = \mathcal{N}\left(0, \sigma^2\tau^2\right) \cdot \mathcal{N}\left(\theta_0 + \theta_1 x, \sigma^2\right) = \frac{1}{2\pi\sigma\tau} \exp\left(-\frac{(y-\theta_0-\theta_1 x)^2}{2\sigma^2} - \frac{x^2}{2\sigma^2\tau^2} \right) \tag{2.22}$$

whereas our desirable model is given by

$$f_\theta(x)[y] = -\frac{(y-\theta_0-\theta_1 x)^2}{2\sigma^2} - \frac{x^2}{2\sigma^2\tau^2}. \tag{2.23}$$

At this point we insert equation (2.23) to equation (2.13)

$$\min_{\theta} -\mathbb{E}_{p_{\text{data}}(x,y)} \left[\alpha \log \frac{\exp\left(-\frac{(y-\theta_0-\theta_1 x)^2}{2\sigma^2} - \frac{x^2}{2\sigma^2\tau^2} \right)}{\sum_y \exp\left(-\frac{(y-\theta_0-\theta_1 x)^2}{2\sigma^2} - \frac{x^2}{2\sigma^2\tau^2} \right)} + (1-\alpha) \log \frac{\exp\left(-\frac{(y-\theta_0-\theta_1 x)^2}{2\sigma^2} - \frac{x^2}{2\sigma^2\tau^2} \right)}{\sum_x \exp\left(-\frac{(y-\theta_0-\theta_1 x)^2}{2\sigma^2} - \frac{x^2}{2\sigma^2\tau^2} \right)} \right]. \tag{2.24}$$

and after using some of logarithmic identities first expression of (2.24) can be simplified to the form

$$\alpha \log q_\theta(y|x) = \alpha\left(-\frac{(y-\theta_0-\theta_1 x)^2}{2\sigma^2} - \frac{x^2}{2\sigma^2\tau^2}\right) - \alpha \log \sum_y \exp\left(-\frac{(y-\theta_0-\theta_1 x)^2}{2\sigma^2} - \frac{x^2}{2\sigma^2\tau^2}\right) \tag{2.25}$$

and for the second expression it holds

$$(1-\alpha)\log q_\theta(x|y) = (1-\alpha)\left(-\frac{(y-\theta_0-\theta_1 x)^2}{2\sigma^2} - \frac{x^2}{2\sigma^2\tau^2}\right) - (1-\alpha)\log \sum_x \exp\left(-\frac{(y-\theta_0-\theta_1 x)^2}{2\sigma^2} - \frac{x^2}{2\sigma^2\tau^2}\right. \tag{2.26}$$

One way of extension is that we can simply add more parameters $\theta_2, \theta_3, \ldots$ (with corresponding adjustment of equations) if there is for example cubic relation.

Chapter 3

MIL extension to contrastive learning

Conclusion

Text of the conclusion...

Bibliography

[1] S. Allen, J. W. Cahn: *A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening*. Acta Metall., 27:1084-1095, 1979.

[2] G. Ballabio et al.: *High Performance Systems User Guide*. High Performance Systems Department, CINECA, Bologna, 2005. `www.cineca.it`

[3] J. Becker, T. Preusser, M. Rumpf: *PDE methods in flow simulation post processing*. Computing and Visualization in Science, 3(3):159-167, 2000.