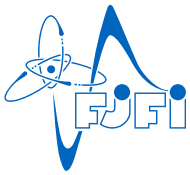
Czech Technical University in Prague
Faculty of Nuclear Sciences and
Physical Engineering

General Framework for Classicifcation at the Top

Dissertation



Author: Ing. Václav Mácha
Academic year: 2021/2022

Poděkování:

Thanks thanks

Čestné prohlášení:

Prohlašuji na tomto místě, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze dne 1. prosince 2021

.....
Ing. Václav Mácha

Název: **Title title title title title title**

Autor: Ing. Václav Mácha

Obor: Matematické inženýrství

Druh práce: Disertační práce

Školitel: doc. Ing Václav Šmídl, Ph.D.

Školitel specialista: Mgr. Lukáš Adam, Ph.D.

Abstrakt: Abstract abstract

Klíčová slova: Keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords

Title: **Title title title title title title**

Abstract: Abstract abstract

Keywords: Keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords

Contents

1 Introduction

The problem of data classification is very important mathematical problem. The goal of classification is to find a relation between a set of objects and a target variable based on some properties of the objects. The properties of the objects are usually called features. There are many problems in research as well as in the real world that can be formulated as classification tasks. We can find applications of data classification across all the fields:

- **Medical Diagnonsis:** In medicine, the classification is often used to improve disease diagnosis. In such a case, the features are medical records such as the patient's blood tests, temperature, or roentgen images. The target variable is if the patient has some disease. As an example, classification is used to process mammogram images and detect cancer [1, 2].

- **Internet Secutiry:** These days, the internet is a crucial part of our lives. With the increasing usage of the internet, the number of attacks increases as well. An essential part of the defense are intrusion detection systems [3, 4] that search for malicious activities (network attacks) in network traffic. Classification can be used to improve such systems [5, 6].

- **Marketing:** In marketing, the task can be to classify customers based on their buying interests. Such information can be used to build a personalized recommendation system for customers and therefore increase income [7, 8].

Many other classification problems can be found in almost all fields. Also, there is a vast number of classification algorithms that try to solve these classifications problems. Typically these algorithms consist of two phases:

- **Training Phase:** In the training phase, the algorithm uses training data to build a model. The classification algorithms fall into the category of supervised learning algorithms. It means, that these algorithms must have labeled training data to build the model, i.e. the algorithm must have the knowledge of the target classes. The training data typically consists of pairs (sample, label) and can be described as follows

$$\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n},$$

  where the sample $\boldsymbol{x}_i \in \mathbb{R}^d$ is a d-dimensional vector of features that describes the object of interes and the label $y_i \in \{1, 2, \ldots, k\}$ represents target class. Moreover $n \in \mathbb{N}$ is a number of training samples and $k \in \mathbb{N}$ is a number of target classes.

- **Testing Phase:** In the testing phase, the model is used to assign labels $\hat{y}_i \in \{1, 2, \ldots, k\}$ to the data from testing set which was not known during the training phase

$$\mathcal{D}_{\text{test}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m},$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \{1, 2, \ldots, k\}$ and $m \in \mathbb{N}$ is a number of testing samples. The ultimate goal of all classification algorithms is to classify testing samples with the highest accuracy possible.

The previous definitions of training and test set are general for classification problems with multiple classes. However, the main focus of this work is on a special subclass of classification problems with only two target classes: binary classification.

1.1 Binary Classification

The binary classification is a special case of classification in which the number of classes is $k = 2$. These two classes are usually referred to as negative and positive classes and the positive class is the one that we are more interested in. If we go back to the mammogram example, the positive class would represent cancer. The positive class is usually encoded using label 1. The encoding of the negative class differs for different algorithms. For example, SVM-like algorithms (from Support Vector Machines [9]) usually use the label -1 for the negative class. On the other hand, neural networks usually use the label 0 for the negative class. In this work, we will follow the notation used for neural networks. Consider the dataset in the following form

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n},$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \{0, 1\}$ and $n \in \mathbb{N}$ is a number of samples in the dataset. To simplify future notation, we denote set of all indices of training samples as $\mathcal{I} = \mathcal{I}^- \cup \mathcal{I}^+$, where

$$\begin{aligned} \mathcal{I}^- &= \{i \mid i \in \{1, 2, \ldots, n\} \ \wedge \ y_i = 0\}, \\ \mathcal{I}^+ &= \{i \mid i \in \{1, 2, \ldots, n\} \ \wedge \ y_i = 1\}. \end{aligned} \tag{1.1}$$

We also denote the number of negative samples in the training set as $n^- = |\mathcal{I}^-|$ and the number of positive samples in the training set as $n^+ = |\mathcal{I}^+|$, i.e. total number of training samples is $n = n^- + n^+$. Generally, the problem of binary classification can be written as follows

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \lambda_1 \sum_{i \in \mathcal{I}^-} \mathbb{1}_{[s_i \geq t]} + \lambda_2 \sum_{i \in \mathcal{I}^+} \mathbb{1}_{[s_i < t]} \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i = 1, 2, \ldots, n, \\ & t \text{ is fixed}, \end{aligned} \tag{1.2}$$

where $\lambda_1, \lambda_2 \in \mathbb{R}$, the function $f \colon \mathbb{R}^d \to \mathbb{R}$ represents the model and maps samples $\boldsymbol{x}_i$ to classification scores s_i and $\mathbb{1}_{[\cdot]}$ represents Iverson function that is used to counts misclassified samples and is defined as

$$\mathbb{1}_{[x]} = \begin{cases} 0 & \text{if } x \text{ is false}, \\ 1 & \text{if } x \text{ is true}. \end{cases}$$

Moreover, the vector $\boldsymbol{w} \in \mathbb{R}^d$ represents trainable parameters (weights) of the model f and $t \in R$ is a decision threshold. The parameters $\boldsymbol{w}$ are determined from training data

during the training phase of classification algorithm. Although the decision threshold t can also be determined from the training data, in many cases it is fixed. For example, for many algorithms the classification score s_i given by the model f represents the probability that the sample $\boldsymbol{x}_i$ belongs to the positive class. Therefore, it makes sense to set the decision threshold to $t = 0.5$ and classify the sample as positive if its classification score is larger than this threshold.

1.2 Performance Criteria

In the previous section we defined general binary classification problem 1.2. However, we did not discuss yet how to measure the accuracy of the resulting classifier. With fixed classifier f (classifier with fixed weights $\boldsymbol{w}$) and fixed decision threshold t, the following formula can be used to obtain predictions $\hat{y}_i$ for all $i = 1, 2, \ldots, n$

$$\hat{y}_i = \begin{cases} 1\ldots & \text{if } f(\boldsymbol{x}_i; \boldsymbol{w}) \geq t, \\ 0\ldots & \text{otherwise.} \end{cases}$$

Based on the prediction $\hat{y}_i$ and an actual label y_i of the sample $\boldsymbol{x}_i$, each sample can be assigned to one of the following categories

- **True negative:** $\boldsymbol{x}_i$ is negative and is classified as negative, i.e. $y_i = 0 \ \wedge \ \hat{y}_i = 0$.
- **False positive:** $\boldsymbol{x}_i$ is negative and is classified as positive, i.e. $y_i = 0 \ \wedge \ \hat{y}_i = 1$.
- **False negative:** $\boldsymbol{x}_i$ is positive and is classified as negative, i.e. $y_i = 1 \ \wedge \ \hat{y}_i = 0$.
- **True positive:** $\boldsymbol{x}_i$ is positive and is classified as positive, i.e. $y_i = 1 \ \wedge \ \hat{y}_i = 1$.

Using these four categories, we can construct a so-called confusion matrix (sometimes also called contingency table) [10] that represents the results of predictionS for all samples from the given dataset $\mathcal{D}$. An illustration of the confusion matrix is shown in Figure 1.1. If we denote vector classification scores given by classifier f as $\boldsymbol{s} \in \mathbb{R}^n$, where $s_i = f(\boldsymbol{x}_i; \boldsymbol{w})$ for $i = 1, 2, \ldots, n$, we can compute all fields of the confusion matrix as follows

$$\begin{aligned} \mathrm{tp}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}^+} \mathbb{1}_{[s_i \geq t]}, & \mathrm{fn}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}^+} \mathbb{1}_{[s_i < t]}, \\ \mathrm{tn}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}^-} \mathbb{1}_{[s_i < t]}, & \mathrm{fp}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}^-} \mathbb{1}_{[s_i \geq t]}. \end{aligned} \tag{1.3}$$

In the following text, we will sometimes use simplified notation $\mathrm{tp} = \mathrm{tp}(\boldsymbol{s}, t)$ (similarly fo the other counts) for example to define classification metrics. In such cases, the vector of classification scores and decision threshold is fixed and is known from the context. Using the simplified notation we can simply define true-positive, false-positive, true-negative and false-negative rates as follows

$$\mathrm{tpr} = \frac{\mathrm{tp}}{n^+}, \qquad \mathrm{fnr} = \frac{\mathrm{fn}}{n^+}, \qquad \mathrm{tnr} = \frac{\mathrm{tn}}{n^-}, \qquad \mathrm{fpr} = \frac{\mathrm{fp}}{n^-}. \tag{1.4}$$

Figure 1.2 show the relation between these for classification rates and the classification scores and the decision threshold. The blue and red curves represent theoretical distribution of the scores of negative and positive samples samples respectively. The position

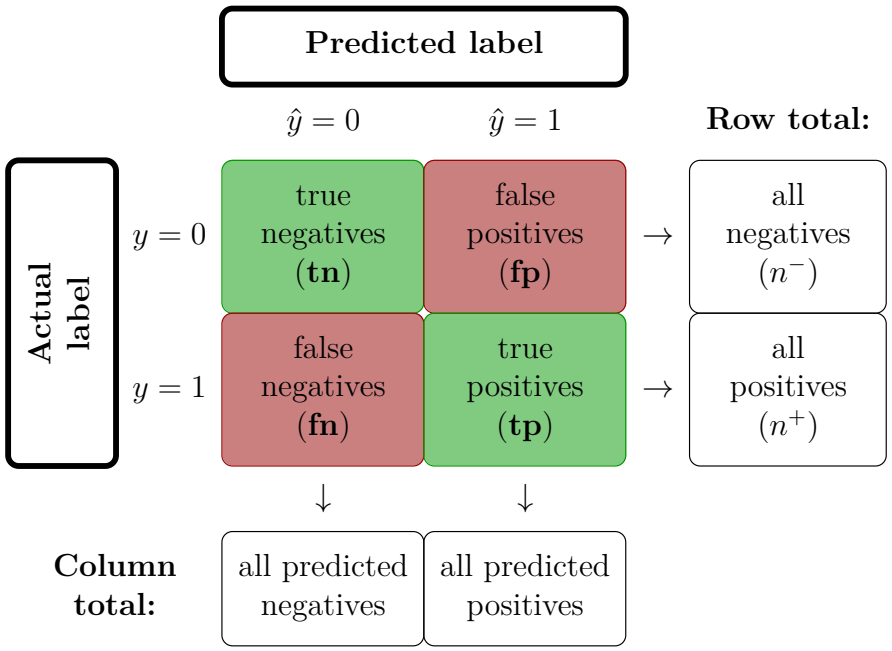


Figure 1.1: Representation of the confusion matrix for the binary classification problem, where the negative class has label 0 and the positive class has label 1. The true (target) label is denoted as y and predicted label is denoted as $\hat{y}$.

of the decision threshold determines the values of the classification rates. The higher the value of the decision threshold, the smaller the false-positive rate, but at the same time the higher the false-negative rate. Similarly, the smaller the value of the decision threshold, the higher the false-positive rate and the smaller the false-negative rate. Ideally, classification without errors is the goal, but it is not usually possible and therefore we have to try to find some trade-off between false positive and a false negative rate. There is no universal truth, which error is worse. For example, we may want to detect cancer from some medical data. In this case, it is probably better to classify a healthy patient as sick than the other way around. On the other hand, in the computer security we do not want to an antivirus program that makes a lot of false-positive alerts since it will be disruptive for the user. If we get look at the general definition of the binary classification problem (1.2), we can see, that the objective function is in fact just the weighted sum of false positive and false negative samples, i.e. we can use the notation (1.3) and rewrite the problem (1.2) to the following form

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \lambda_1 \cdot \mathrm{fp}(\boldsymbol{s}, t) + \lambda_2 \cdot \mathrm{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i = 1, 2, \ldots, n, \\ & t \text{ is fixed.} \end{aligned} \tag{1.5}$$

The parameters $\lambda_1 \lambda_2 \in \mathbb{R}$ are used to specify which error is more serious for the particular classification task.

Many performance metrics used for binary classification such as accuracy, precision, or recall can be easily derived from the confusion matrix:

$$\begin{aligned} \text{accuracy} &= \frac{\mathrm{tp} + \mathrm{tn}}{n}, & \text{balanced accuracy} &= \frac{1}{2}(\mathrm{tpr} + \mathrm{tnr}), \\ \text{precision} &= \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fp}}, & \text{recall} &= \mathrm{tpr}, \end{aligned} \tag{1.6}$$

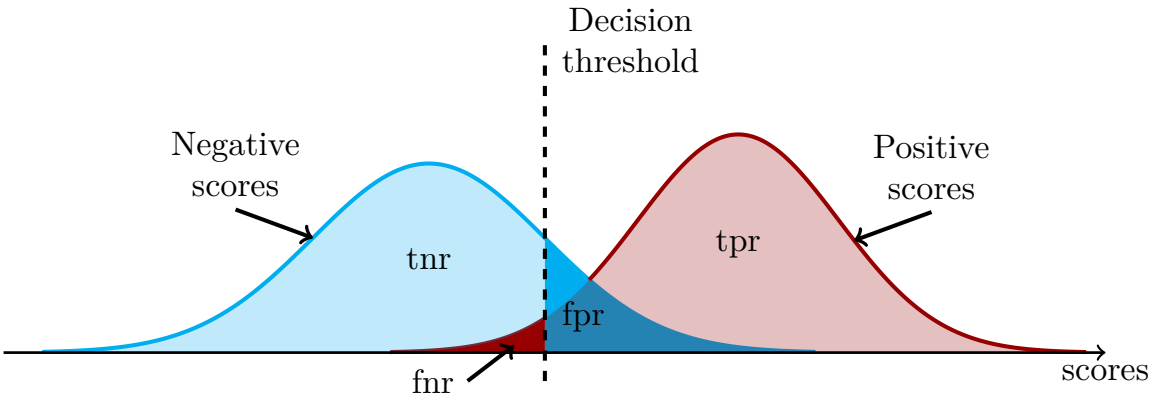


Figure 1.2: The relation between classification scores and rates. The blue curve represents theoretical distribution of the scores of negative samples and the red curve the same for the score of positive samples. Filled areas with light blue or red color represent true-negative and true-positive rates respectively. Similarly the filled areas with dark blue or red color represent false-positive and false-negaives rates.

In other words, accuracy represents the ratio of correctly classified samples from all samples. In the case, where the dataset is unbalanced (the number of samples in one class is significantly higher then the number of samples in the other class) the accuracy is not a good metric, since both classes have the same weights. In such a case, balanced accuracy is a better choice. The balanced accuracy is defined as an average of true-positive and true-negative rate. Precision represents the ratio of correctly classified positive samples from all samples classified as positives. Finally, recall is just an alias for true-positive rate.

2

Linear Classification at the Top

Many binary classification problems focus on separating the dataset by a linear hyperplane $\boldsymbol{w}^\top \boldsymbol{x} - t$. A sample $\boldsymbol{x}$ is deemed to be positive or relevant (depending on the application) if its score $\boldsymbol{w}^\top \boldsymbol{x}$ is above a threshold t. Multiple problem categories belong to this framework:

- *Ranking problems* select the most relevant samples and rank them. To each sample, a numerical score is assigned, and the ranking is performed based on this score. Often, only scores above a threshold are considered.

- *Accuracy at the Top* is similar to ranking problems. However, instead of ranking the most relevant samples, it only maximizes the accuracy (equivalently minimizes the misclassification) in these top samples. The prime examples of both categories include search engines or problems where identified samples undergo expensive post-processing such as human evaluation.

- *Hypothesis testing* states a null and an alternative hypothesis. The Neyman-Pearson problem minimizes the Type II error (the null hypothesis is false but it fails to be rejected) while keeping the Type I error (the null hypothesis is true but is rejected) small. If the null hypothesis states that a sample has the positive label, then Type II error happens when a positive sample is below the threshold and thus minimizing the Type II error amounts to minimizing the positives below the threshold.

Examples of this type can be found in search engines, where the user is interested only in the first few queries. These queries need to be of high quality. Other examples include cybersecurity [3], where a low false-negative rate is crucial as a high number of false alarms would result in the software being uninstalled, or drug development, where potentially useful drugs need to be preselected and manually investigated. All these three applications may be written (possibly after a reformulation) in a similar form as a minimization of the false-negatives (misclassified positives) above a threshold. They only differ in the way they define the threshold. Despite this striking similarity, they are usually considered separately in the literature. The main goal of this paper is to provide a unified framework for these three applications and perform its theoretical and numerical analysis.

The goal of the ranking problems is to rank the relevant samples higher than the non-relevant ones. A prototypical example is the RankBoost [11] maximizing the area under the ROC curve, the Infinite Push [12] or the p-norm push [13] which concentrate on the high-ranked negatives and push them down. Since all these papers include pairwise comparisons of all samples, they can be used only for small datasets. This was alleviated in [14], where the authors performed the limit $p \to \infty$ in p-norm push and obtained the

linear complexity in the number of samples. Moreover, since the l_∞-norm is equal to the maximum, this method falls into our framework with the threshold equal to the largest score computed from negative samples.

Accuracy at the Top (τ-quantile) was formally defined in [15] and maximizes the number of relevant samples in the top τ-fraction of ranked samples. When the threshold equals the top τ-quantile of all scores, this problem falls into our framework. The early approaches aim at solving approximations, for example, [16] optimizes a convex upper bound on the number of errors among the top samples. Due to the presence of exponentially many constraints, the method is computationally expensive. [15] presented an SVM-like formulation which fixes the index of the quantile and solves n problems. While this removes the necessity to handle the (difficult) quantile constraint, the algorithm is computationally infeasible for a large number of samples. [17] derived upper approximations, their error bounds and solved these approximations. [3] proposed the projected gradient descent method where after each gradient step, the quantile is recomputed. [18] suggested new formulations for various criteria and argued that they keep desired properties such as convexity. [19] showed that accuracy at the top is maximized by thresholding the posterior probability of the relevant class. The closest approach to our framework is [20, 21], where the authors considered multi-class classification problems, and their goal was to optimize the performance on the top few classes and [22], where the authors implicitly removed some variables and derived an efficient algorithm.

2.1 Framework for Minimizing Missclassification Above a Threshold

Many important binary classification problems minimize the number of misclassified samples below (or above) certain threshold. Since these problems are usually considered separately, in this section, we provide a unified framework for their handling and present several classification problems falling into this framework.

For samples $\boldsymbol{x}$, we consider the linear classifier $f(\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x} - t$, where $\boldsymbol{w}$ is the normal vector to the separating hyperplane and t is a threshold. The most well-known example is the support vector machines, where t is an optimization variable. In many cases the threshold t is computed from the scores $s = \boldsymbol{w}^\top \boldsymbol{x}$. For example, *TopPush* from [14] sets the threshold t to the largest score s^- corresponding to negative samples and [3] sets it to the quantile of all scores.

To be able to determine the missclassification above and below the threshold t, we define the true-positive, false-negative, true-negative and false-positive counts by

$$\begin{aligned} \mathrm{tp}(\boldsymbol{w}, t) &= \sum_{\boldsymbol{x} \in \mathcal{X}^+} \left[\boldsymbol{w}^\top \boldsymbol{x} - t \geq 0\right], \quad \mathrm{fn}(\boldsymbol{w}, t) = \sum_{\boldsymbol{x} \in \mathcal{X}^+} \left[\boldsymbol{w}^\top \boldsymbol{x} - t < 0\right], \\ \mathrm{tn}(\boldsymbol{w}, t) &= \sum_{\boldsymbol{x} \in \mathcal{X}^-} \left[\boldsymbol{w}^\top \boldsymbol{x} - t < 0\right], \quad \mathrm{fp}(\boldsymbol{w}, t) = \sum_{\boldsymbol{x} \in \mathcal{X}^-} \left[\boldsymbol{w}^\top \boldsymbol{x} - t \geq 0\right]. \end{aligned} \tag{2.1}$$

Here $[\cdot]$ is the 0-1 loss (Iverson bracket, characteristic function) which is equal to 1 if the argument is true and to 0 otherwise. Moreover, $\mathcal{X}/\mathcal{X}^+/\mathcal{X}^-$ denotes the sets of all/positive/negative samples and by $n/n^+/n^-$ their respective sizes.

Since the misclassified samples below the threshold are the false-negatives, we arrive

at the following problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^+}\,\mathrm{fn}(\boldsymbol{w}, t) \\ \text{subject to} \quad & \text{threshold } t \text{ is a function of } \{\boldsymbol{w}^\top \boldsymbol{x}_i\}_{i=1}^n. \end{aligned} \tag{2.2}$$

As the 0-1 loss in (2.1) is discontinuous, problem (2.2) is difficult to handle. The usual approach is to employ a surrogate function such as the hinge loss function defined by

$$l_{\text{hinge}}(s) = \max\{0, 1 + s\}. \tag{2.3}$$

In the text below, the symbol l denotes any convex non-negative non-decreasing function with $l(0) = 1$. Using the surrogate function, the counts (2.1) may be approximated by their surrogate counterparts

$$\begin{aligned} \overline{\mathrm{tp}}(\boldsymbol{w}, t) &= \sum_{\boldsymbol{x} \in \mathcal{X}^+} l(\boldsymbol{w}^\top \boldsymbol{x} - t), \quad \overline{\mathrm{fn}}(\boldsymbol{w}, t) = \sum_{\boldsymbol{x} \in \mathcal{X}^+} l(t - \boldsymbol{w}^\top \boldsymbol{x}), \\ \overline{\mathrm{tn}}(\boldsymbol{w}, t) &= \sum_{\boldsymbol{x} \in \mathcal{X}^-} l(t - \boldsymbol{w}^\top \boldsymbol{x}), \quad \overline{\mathrm{fp}}(\boldsymbol{w}, t) = \sum_{\boldsymbol{x} \in \mathcal{X}^-} l(\boldsymbol{w}^\top \boldsymbol{x} - t). \end{aligned} \tag{2.4}$$

Since $l(\cdot) \geq [\cdot]$, the surrogate counts (2.4) provide upper approximations of the true counts (2.1). Replacing the counts in (2.2) by their surrogate counterparts and adding a regularization results in

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^+}\,\overline{\mathrm{fn}}(\boldsymbol{w}, t) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & \text{threshold } t \text{ is a function of } \{\boldsymbol{w}^\top \boldsymbol{x}_i\}_{i=1}^n. \end{aligned} \tag{2.5}$$

In the rest of this section, we list formulations which fall into the framework of (2.2) and (2.5).

2.1.1 Methods based on pushing positives to the top

The first category of formulations falling into our framework (2.2) and (2.5) are ranking methods which attempt to put as many positives (relevant samples) to the top as possible. Specifically, for each sample $\boldsymbol{x}$, they compute the score $s = \boldsymbol{w}^\top \boldsymbol{x}$ and then sort the vector $\boldsymbol{s}$ into $\boldsymbol{s}_{[\cdot]}$ with decreasing components $s_{[1]} \geq s_{[2]} \geq \cdots \geq s_{[n]}$. The number of positives on top equals to the number of positives above the highest negative. This amounts to maximizing true-positives or, equivalently, minimizing false-negatives, which may be written as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^+}\,\mathrm{fn}(\boldsymbol{w}, t) \\ \text{subject to} \quad & t = s^-_{[1]}, \\ & \text{components of } \boldsymbol{s}^- \text{ equal to } s^- = \boldsymbol{w}^\top \boldsymbol{x}^- \text{ for } \boldsymbol{x}^- \in \mathcal{X}^-. \end{aligned} \tag{2.6}$$

As t is a function of the scores $s = \boldsymbol{w}^\top \boldsymbol{x}$, problem (2.6) is a special case of (2.2).

TopPush from [14] replaces the false-negatives in (2.6) by their surrogate and adds a regularization term to arrive at

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^+}\,\overline{\mathrm{fn}}(\boldsymbol{w}, t) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & t = s^-_{[1]}, \\ & \text{components of } \boldsymbol{s}^- \text{ equal to } s^- = \boldsymbol{w}^\top \boldsymbol{x}^- \text{ for } \boldsymbol{x}^- \in \mathcal{X}^-. \end{aligned} \tag{2.7}$$

Note that this falls into the framework of (2.5).

As we will show in Section 2.2.4, *TopPush* is sensitive to outliers and mislabelled data. To robustify it, we follow the idea from [20] and propose to replace the largest negative score by the mean of k largest negative scores. This results in

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^+}\overline{\text{fn}}(\boldsymbol{w}, t) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & t = \frac{1}{k}(s^-_{[1]} + \cdots + s^-_{[k]}), \\ & \text{components of } \boldsymbol{s}^- \text{ equal to } s^- = \boldsymbol{w}^\top \boldsymbol{x}^- \text{ for } \boldsymbol{x}^- \in \mathcal{X}^-. \end{aligned} \tag{2.8}$$

We used the mean of highest k negative scores instead of the value of the k-th negative score to preserve convexity as shown in Section 2.2.2.

2.1.2 Accuracy at the Top

The previous category considers formulations which minimize the false-negatives below the highest-ranked negative. Accuracy at the Top [15] takes a different approach and minimizes false-positives above the top τ-quantile defined by

$$t_1(\boldsymbol{w}) = \max\{t \mid \text{tp}(\boldsymbol{w}, t) + \text{fp}(\boldsymbol{w}, t) \geq n\tau\}. \tag{2.9}$$

Then the Accuracy at the Top problem is defined by

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^-}\text{fp}(\boldsymbol{w}, t) \\ \text{subject to} \quad & t \text{ is the top } \tau\text{-quantile: it solves (2.9)}. \end{aligned} \tag{2.10}$$

Due to Lemma A.1 in the Appendix, the previous problem (2.10) is equivalent (up to a small theoretical issue) to

$$\begin{aligned} \text{minimize} \quad & \mu\,\text{fn}(\boldsymbol{w}, t) + (1-\mu)\,\text{fp}(\boldsymbol{w}, t) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & t \text{ is the top } \tau\text{-quantile: it solves (2.9)} \end{aligned} \tag{2.11}$$

for any $\mu \in [0, 1]$. This problem with $\mu = 0$ equals to (2.10), with $\mu = 1$ it falls into our framework (2.2), while with $\mu = \frac{n^-}{n}$ it corresponds to the original definition from [15].

Apart from the quantile (2.9), there are two other possible choices of the threshold

$$t_2(\boldsymbol{w}) = \frac{1}{n\tau}\sum_{i=1}^{n\tau} s_{[i]}, \tag{2.12}$$

$$t_3(\boldsymbol{w}) \quad \text{solves} \quad \frac{1}{n}\sum_{i=1}^{n} l(\beta(s_i - t)) = \tau. \tag{2.13}$$

We again use the vector of scores $\boldsymbol{s}$ with components $s_i = \boldsymbol{w}^\top \boldsymbol{x}_i$ and for the rest of the paper we assume, for simplicity, that $n\tau$ is an integer. The quantile (2.9) is sometimes denoted as VaR (value at risk) and (2.12) as CVaR (conditional value of risk). It is known is that the latter is the tightest convex approximation of the former. We will sometimes denote (2.13) as surrogate top τ-quantile. We will investigate the relations between these three objects as well as their properties such as convexity, differentiability or stability in Section 2.2.

Paper [3] builds on the Accuracy at the Top problem (2.11), where it replaces $\mathrm{fn}(\boldsymbol{w}, t)$ and $\mathrm{fp}(\boldsymbol{w}, t)$ in the objective by their surrogate counterparts $\overline{\mathrm{fn}}(\boldsymbol{w}, t)$ and $\overline{\mathrm{fp}}(\boldsymbol{w}, t)$. This leads to

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^+}\,\overline{\mathrm{fn}}(\boldsymbol{w}, t) + \frac{1}{n^-}\,\overline{\mathrm{fp}}(\boldsymbol{w}, t) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & t \text{ is the top } \tau\text{-quantile: it solves (2.9).} \end{aligned} \tag{2.14}$$

Based on the first author, we name this formulation *Grill*. The main purpose of (2.12) is to provide a convex approximation of the non-convex quantile (2.9). Putting it into the constraint results in a convex approximation problem, which we call *TopMeanK*

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^+}\,\overline{\mathrm{fn}}(\boldsymbol{w}, t) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & t = \frac{1}{n\tau}(s_{[1]} + \cdots + s_{[n\tau]}), \\ & \text{components of } \boldsymbol{s} \text{ equal to } s = \boldsymbol{w}^\top \boldsymbol{x} \text{ for } \boldsymbol{x} \in \mathcal{X}. \end{aligned} \tag{2.15}$$

Similarly, we can use the surrogate top quantile in the constraint to arrive at

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^+}\,\overline{\mathrm{fn}}(\boldsymbol{w}, t) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & t \text{ is the surrogate top } \tau\text{-quantile: it solves (2.13).} \end{aligned} \tag{2.16}$$

Note that *Grill* minimizes the convex combination of false-positives and false-negatives while (2.15) and (2.16) minimize only the false-negatives. The reason for this will be evident in Section 2.2.2 and amounts to preservation of convexity. Moreover, as will see later, problem (2.16) provides a good approximation to the Accuracy at the Top problem, it is easily solvable due to convexity and requires almost no tuning, we named it *Pat&Mat* (Precision At the Top & Mostly Automated Tuning).

2.1.3 Methods optimizing the Neyman-Pearson criterion

Another category falling into the framework of (2.2) and (2.5) is the Neyman-Pearson problem which is closely related to hypothesis testing, where null H_0 and alternative H_1 hypotheses are given. Type I error occurs when H_0 is true but is rejected, and type II error happens when H_0 is false, but it fails to be rejected. The standard technique is to minimize Type II error while a bound for Type I error is given.

In the Neyman-Pearson problem, the null hypothesis H_0 states that a sample $\boldsymbol{x}$ has the negative label. Then Type I error corresponds to false-positives while Type II error to false-negatives. If the bound on Type I error equals τ, we may write this as

$$t_1^{\mathrm{NP}}(\boldsymbol{w}) = \max\Big\{t \mid \mathrm{fp}(\boldsymbol{w}, t) \geq n^- \tau\Big\}. \tag{2.17}$$

Then, we may write the Neyman-Pearson problem as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^+}\,\mathrm{fn}(\boldsymbol{w}, t) \\ \text{subject to} \quad & t \text{ is Type I error at level } \tau\text{: it solves (2.17).} \end{aligned} \tag{2.18}$$

Since (2.18) differs from (2.11) only by counting only the false-positives in (2.17) instead of counting all positives in (2.9), we can derive its approximations in exactly the same

way as in Section 2.1.2. We therefore provide only their brief description and start with approximations of (2.17)

$$t_2^{\mathrm{NP}}(\boldsymbol{w}) = \frac{1}{n^-\tau} \sum_{i=1}^{n^-\tau} s^-_{[i]}, \tag{2.19}$$

$$t_3^{\mathrm{NP}}(\boldsymbol{w}) \quad \text{solves} \quad \frac{1}{n} \sum_{i=1}^{n^-} l(\beta(s_i^- - t)) = \tau. \tag{2.20}$$

Replacing the true counts by their surrogates results in the Neyman-Pearson variant *Grill-NP*

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^+} \overline{\mathrm{fn}}(\boldsymbol{w}, t) + \frac{1}{n^-} \overline{\mathrm{fp}}(\boldsymbol{w}, t) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & t \text{ is the Neyman-Pearson threshold: it solves (2.17).} \end{aligned} \tag{2.21}$$

Similarly, the Neyman-Pearson alternative to *TopMeanK* reads

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^+} \overline{\mathrm{fn}}(\boldsymbol{w}, t) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & t = \frac{1}{n^-\tau}(s^-_{[1]} + \cdots + s^-_{[n^-\tau]}), \\ & \text{components of } \boldsymbol{s}^- \text{ equal to } s^- = \boldsymbol{w}^\top \boldsymbol{x}^- \text{ for } \boldsymbol{x}^- \in \mathcal{X}. \end{aligned} \tag{2.22}$$

This problem already appeared in [23] under the name τ-*FPL*. Finally, *Pat&Mat-NP* reads

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n^+} \overline{\mathrm{fn}}(\boldsymbol{w}, t) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & t \text{ is the surrogate Neyman-Pearson threshold: it solves (2.20).} \end{aligned} \tag{2.23}$$

We may see (2.22) from two different viewpoints. First, τ-*FPL* provide convex approximations of *Grill-NP*. Second, τ-*FPL* has the same form as *TopPushK*. The only difference is that for τ-*FPL* we have $k = n^-\tau$ while for *TopPushK* the value of k is small. Thus, even though we started from two different problems, we arrived at two approximations which differ only in the value of one parameter. This shows a close relation of the ranking problem and the Neyman-Pearson problem and the need for a unified theory to handle these problems.

2.2 Theoretical Analysis of the Framework

In this section, we provide a theoretical analysis of the unified framework from Section 2.1. We consider purely the problem *formulations* and not individual *algorithms* which specify how to solve these formulations. We focus mainly on the following desirable properties:

- *Convexity* implies a guaranteed convergence for many optimization algorithms or their better convergence rates [24].
- *Differentiability* increases the speed of convergence.
- *Stability* is a general term, by which we mean that the global minimum is not at $\boldsymbol{w} = \boldsymbol{0}$. This actually happens for many formulations from Section 2.1 and results in the situation where the separating hyperplane is degenerate and does not actually exist.

For a nicer flow of text, we show the results only for formulations from Section 2.1.2. The results for methods from Section 2.1.3 are identical. For the same reason, we postpone the proofs to Appendix A.1.

2.2.1 Threshold value comparison

We start with the following proposition, which compares the threshold approximation quality.

Proposition 2.1: [23]

We always have

$$t_1(\boldsymbol{w}) \leq t_2(\boldsymbol{w}) \leq t_3(\boldsymbol{w}).$$

Whenever the objective contains only false-negatives, a lower threshold t means a lower objective function. Therefore, a lower threshold is preferred.

2.2.2 Convexity

Convexity is one of the most important properties in numerical optimization. It ensures that the optimization problem has neither stationary points nor local minima. All points of interest are global minima. Moreover, it allows for faster convergence rates. We present the following two results.

Proposition 2.2

Thresholds t_2 and t_3 are convex functions of the weights $\boldsymbol{w}$. The threshold function t_1 is non-convex.

Theorem 2.3

If the threshold t is a convex function of the weights $\boldsymbol{w}$, then function $f(\boldsymbol{w}) = \overline{\text{fn}}(\boldsymbol{w}, t(\boldsymbol{w}))$ is convex.

While the proof of Theorem 2.3 is simple, it points to the necessity of considering only false-negatives in the objective of the problems in Section 2.1. In such a case, *TopPush*, *TopPushK*, *TopMeanK*, *τ-FPL*, *Pat&Mat* and *Pat&Mat-NP* are convex problems. At the same time, *Grill* and *Grill-NP* are not convex problems.

2.2.3 Differentiability

Similarly to convexity, differentiability allows for faster convergence rate and in some algorithms, better termination criteria. The next theorem shows which formulations are differentiable.

Theorem 2.4

If the surrogate function l is differentiable, then threshold t_3 is a differentiable func-

tion of the weights $\boldsymbol{w}$ and its derivative equals to

$$\nabla t_3(\boldsymbol{w}) = \frac{\sum_{\boldsymbol{x}\in\mathcal{X}} l'(\beta(\boldsymbol{w}^\top \boldsymbol{x} - t_3(\boldsymbol{w})))\boldsymbol{x}}{\sum_{\boldsymbol{x}\in\mathcal{X}} l'(\beta(\boldsymbol{w}^\top \boldsymbol{x} - t_3(\boldsymbol{w})))}.$$

The threshold functions t_1 and t_2 are non-differentiable.

This theorem shows that the objective functions of *Pat&Mat* and *Pat&Mat-NP* are differentiable. This allows us to prove the convergence of the stochastic gradient descent for these two formulations in Section 2.3.

2.2.4 Stability

We first provide a simple example and show that many formulations from the previous section are degenerate for it. Then we analyze general conditions under which this degenerate behaviour happens.

Example of a Degenerate Behavior

We consider n negative samples uniformly distributed in $[-1, 0] \times [-1, 1]$, n positive samples uniformly distributed in $[0, 1] \times [-1, 1]$ and one negative sample at $(2, 0)$, see Figure 2.1 (left). We consider the hinge loss and no regularization. If n is large, the point at $(2, 0)$ is an outlier and the dataset is separable and the separating hyperplane has the normal vector $\boldsymbol{w} = (1, 0)$.

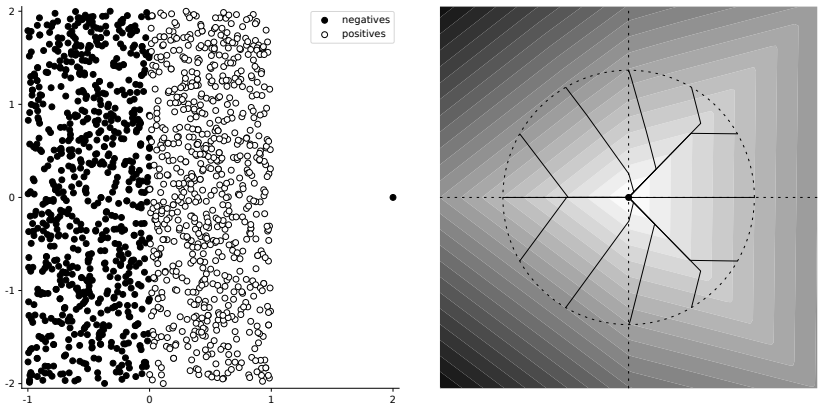


Figure 2.1: Left: distribution of positive (empty circle) and negative samples (full circles) for the example from Section 2.2.4. Right: contour plot for *TopPush* and its convergence to the zero vector from 12 initial points.

Table 2.1 shows the threshold t and the objective value f for two points $\boldsymbol{w}_1 = (0, 0)$ and $\boldsymbol{w}_2 = (1, 0)$. These two points are both important: $\boldsymbol{w}_1$ does not generate any separating hyperplane, while $\boldsymbol{w}_2$ generates the optimal separating hyperplane. We show the precise computation in Appendix A.2. Since the dataset is perfectly separable by $\boldsymbol{w}_2$, we expect that $\boldsymbol{w}_2$ provides a lower objective than $\boldsymbol{w}_1$. By shading the better objective in Table 2.1 by grey, we see that this did not happen for *TopPush* and *TopMeanK*.

It can be shown that $\boldsymbol{w}_1 = (0, 0)$ is even the global minimum for *TopPush* and *TopMeanK*. This raises the question of whether some tricks, such as early stopping or excluding a small ball around zero, cannot overcome this difficulty. The answer is negative

		$\boldsymbol{w}_1 = (0,0)$		$\boldsymbol{w}_2 = (1,0)$	
Name	Label	t	f	t	f
TopPush	(2.7)	0	1	2	2.5
TopPushK	(2.8)	0	1	$\frac{2}{k}$	$0.5 + \frac{2}{k}$
Grill	(2.14)	0	2	$1 - 2\tau$	$1.5 + 2\tau(1-\tau)$
TopMeanK	(2.15)	0	1	$1 - \tau$	$1.5 - \tau$
Pat&Mat	(2.16)	$\frac{1}{\beta}(1-\tau)$	$1 + \frac{1}{\beta}(1-\tau)$	$\frac{1}{\beta}(1-\tau)$	$0.5 + \frac{1}{\beta}(1-\tau)$

Table 2.1: Comparison of formulations on the very simple problem from Section 2.2.4. Two formulations have the global minimum (denoted by grey color) at $\boldsymbol{w}_1 = (0,0)$ which does not generate any separating hyperplane. The optimal separating hyperplane is generated by $\boldsymbol{w}_2 = (1,0)$.

as shown in Figure 2.1 (right). Here, we run *TopPush* from several starting points, and it always converges to zero from one of the three possible directions; all of them far from the normal vector to the separating hyperplane.

Stability and Global minimum at zero

The convexity derived in the previous section guarantees that there are no local minima. However, as we showed in the example above, the global minimum may be at $\boldsymbol{w} = \mathbf{0}$. This is highly undesirable since $\boldsymbol{w}$ is the normal vector to the separating hyperplane and the zero vector provides no information. In this section, we analyze when this situation happens. The first result states that if the threshold $t(\boldsymbol{w})$ is above a certain value, then zero has a better objective that $\boldsymbol{w}$. If this happens for all $\boldsymbol{w}$, then zero is the global minimum.

Theorem 2.5

Consider any of these formulations: *TopPush*, *TopPushK*, *TopMeanK* or τ-*FPL*. Fix any $\boldsymbol{w}$ and denote the corresponding threshold $t(\boldsymbol{w})$. If we have

$$t(\boldsymbol{w}) \geq \frac{1}{n^+} \sum_{\boldsymbol{x}^+ \in \mathcal{X}^+} \boldsymbol{w}^\top \boldsymbol{x}^+, \tag{2.24}$$

then $f(\mathbf{0}) \leq f(\boldsymbol{w})$. Specifically, denote the scores $s^+ = \boldsymbol{w}^\top \boldsymbol{x}^+$ for $\boldsymbol{x}^+ \in \mathcal{X}^+$ and $s^- = \boldsymbol{w}^\top \boldsymbol{x}^-$ for $\boldsymbol{x}^- \in \mathcal{X}^-$ and the ordered variants with decreasing components of

$\boldsymbol{s}^-$ by $\boldsymbol{s}^-_{[\cdot]}$. Then

$$\begin{aligned} s^-_{[1]} \geq \frac{1}{n^+}\sum_{i=1}^{n^+} s^+_i \implies f(\mathbf{0}) \leq f(\boldsymbol{w}) \text{ for } \textit{TopPush}, \\ \frac{1}{k}\sum_{i=1}^{k} s^-_{[i]} \geq \frac{1}{n^+}\sum_{i=1}^{n^+} s^+_i \implies f(\mathbf{0}) \leq f(\boldsymbol{w}) \text{ for } \textit{TopPushK}, \\ \frac{1}{n^-\tau}\sum_{i=1}^{n^-\tau} s^-_{[i]} \geq \frac{1}{n^+}\sum_{i=1}^{n^+} s^+_i \implies f(\mathbf{0}) \leq f(\boldsymbol{w}) \text{ for } \tau\textit{-FPL}. \end{aligned} \tag{2.25}$$

We can use this result immediately to deduce that some formulations have the global minimum at $\boldsymbol{w} = \mathbf{0}$. More specifically, *TopPush* fails if there are outliers, and *TopMeanK* fails whenever there are many positive samples.

Corollary 2.6

Consider the *TopPush* formulation. If the positive samples lie in the convex hull of negative samples, then $\boldsymbol{w} = \mathbf{0}$ is the global minimum.

Corollary 2.7

Consider the *TopMeanK* formulation. If $n^+ \geq n\tau$, then $\boldsymbol{w} = \mathbf{0}$ is the global minimum.

The proof of Theorem 2.5 employs the fact that all formulations in the theorem statement have only false-negatives in the objective. If $\boldsymbol{w}_0 = \mathbf{0}$, then $\boldsymbol{w}_0^\top \boldsymbol{x} = 0$ for all samples $\boldsymbol{x}$, the threshold equals to $t = 0$ and the objective equals to one. If the threshold is large for $\boldsymbol{w}$, many positives are below the threshold, and the false-negatives have the average surrogate value larger than one. In such a case, $\boldsymbol{w}_0 = \mathbf{0}$ becomes the global minimum. There are two fixes to this situation:

- Include false-positives to the objective. This approach is taken by *Grill* and *Grill-NP* and necessarily results in the loss of convexity.
- Move the threshold away from zero even when all scores $\boldsymbol{w}^\top \boldsymbol{x}$ are zero. This approach is taken by our formulations *Pat&Mat* and *Pat&Mat-NP* and keeps convexity.

The next theorem shows the advantage of the second approach.

Theorem 2.8

Consider the *Pat&Mat* or *Pat&Mat-NP* formulation with the hinge surrogate and no regularization. Assume that for some $\boldsymbol{w}$ we have

$$\frac{1}{n^+}\sum_{\boldsymbol{x}^+ \in \mathcal{X}^+} \boldsymbol{w}^\top \boldsymbol{x}^+ > \frac{1}{n^-}\sum_{\boldsymbol{x}^- \in \mathcal{X}^-} \boldsymbol{w}^\top \boldsymbol{x}^-. \tag{2.26}$$

Then there is a scaling parameter β_0 from (2.13) such that $f(\boldsymbol{w}) < f(\mathbf{0})$ for all $\beta \in (0, \beta_0)$.

These theorem shed some light on the behaviour of the formulations. Theorem 2.5 states that the stability of τ-*FPL* requires

$$\frac{1}{n^-\tau} \sum_{i=1}^{n^-\tau} s^-_{[i]} < \frac{1}{n^+} \sum_{i=1}^{n^+} s^+_i, \tag{2.27}$$

while Theorem 2.8 states that the stability of *Pat&Mat-NP* is ensured by

$$\frac{1}{n^-} \sum_{i=1}^{n^-} s^-_{[i]} < \frac{1}{n^+} \sum_{i=1}^{n^+} s^+_i. \tag{2.28}$$

The right-hand sides of (2.27) and (2.28) are the same, while the left-hand side of (2.28) is always smaller than the left-hand side of (2.27). This implies that if τ-*FPL* is stable, then *Pat&Mat-NP* is stable as well.

At the same time, there may be a huge difference in the stability of both formulations. Since the scores of positive samples should be above the scores of negative samples, the scores s may be interpreted as performance. Then formula (2.27) states that if the mean performance of a *small number of the best* negative samples is larger than the average performance of *all* positive samples, then τ-*FPL* fails. On the other hand, formula (2.28) states that if the average performance of *all* positive samples is better than the average performance of *all* negative samples, then *Pat&Mat-NP* is stable. The former may well happen as accuracy at the top is interested in a good performance of only a small number of positive samples.

2.2.5 Method comparison

We provide a summary of the obtained results in Table 2.2. There we give basic characterizations of the formulations such as their definition label, their source, the hyperparameters, whether the formulation is differentiable and convex, and whether it has stability problems with $\boldsymbol{w} = \mathbf{0}$ being the global minimum.

A similar comparison is performed in Figure 2.2. Methods in green and grey are convex, while formulations in white are non-convex. Based on Theorem 2.5, four formulations in grey are vulnerable to have the global minimum at $\boldsymbol{w} = 0$. This theorem states that the higher the threshold, the more vulnerable the formulation is. The full arrows depict this dependence. If it points from one formulation to another, the latter one has a smaller threshold and thus is less vulnerable to this undesired global minima. The dotted arrows indicate that this holds usually but not always, the precise formulation is provided in Appendix A.1.5. This complies with Corollaries 2.6 and 2.7 which state that *TopPush* and *TopMeanK* are most vulnerable. At the same time, it says that τ-*FPL* is the best one from the grey-coloured formulations. Finally, even though *Pat&Mat-NP* has a worse approximation of the true threshold than τ-*FPL* due to Theorem 2.5, it is more stable due to the discussion after Theorem 2.8.

2.3 Convergence of stochastic gradient descent

The previous section analyzed the formulations from Section 2.1 but did not consider any optimization algorithms. In this section, we show a basic version of the stochastic

Name	Source	Definition	Hyperpars	Convex	Differentiable	Stable
TopPush	[14]	(2.7)	λ	✓	✗	✗
TopPushK	ours	(2.8)	λ, k	✓	✗	✗
Grill	[3]	(2.14)	λ	✗	✗	✓
Pat&Mat	ours	(2.16)	β, λ	✓	✓	✓
TopMeanK	-	(2.15)	λ	✓	✗	✗
Grill-NP	-	(2.21)	λ	✗	✗	✓
Pat&Mat-NP	ours	(2.23)	β, λ	✓	✓	✓
τ*-FPL*	[23]	(2.22)	λ	✓	✗	✗

Table 2.2: Summary of the formulations from Section 2.1. The table shows their definition label, the source or the source they are based on, the hyperparameters, whether the formulation is differentiable, convex and stable (in the sense of having problems with $\boldsymbol{w} = \mathbf{0}$).

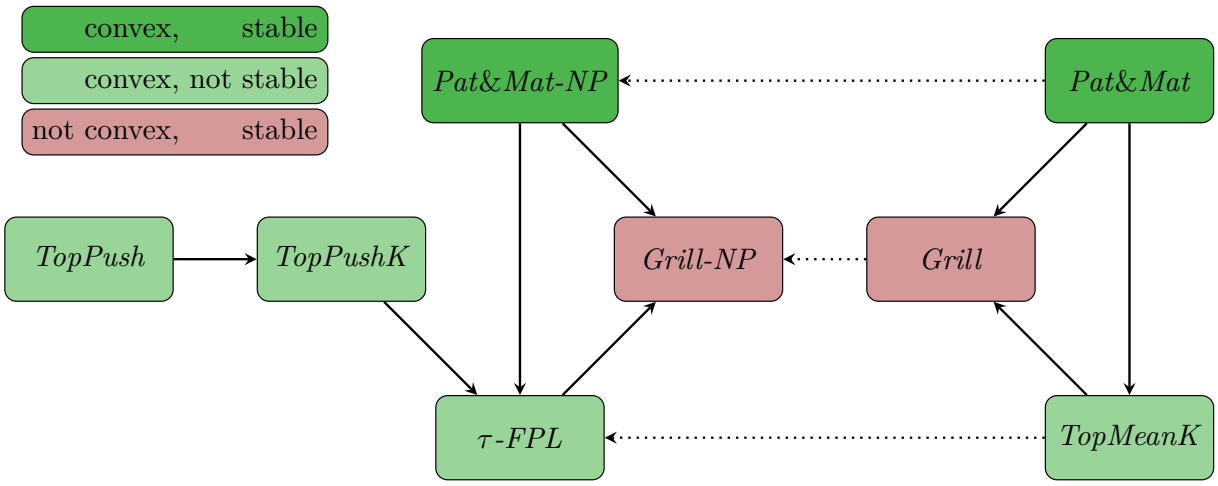


Figure 2.2: Summary of the formulations from Section 2.1. Methods in green and grey are convex, while formulations in white are non-convex. Methods in grey are vulnerable to have the global minimum at $\boldsymbol{w} = 0$. Full (dotted) arrow pointing from one formulation to another show that the latter formulation has always (usually) smaller threshold.

gradient descent and then show its convergent version. Since due to considering the threshold, gradient computed on a minibatch is a biased estimate of the true gradient, we need to use variance reduction techniques, and the proof is rather complex.

2.3.1 Stochastic gradient descent: Basic

Many optimization algorithms for solving the formulations from Section 2.1 use primal-dual or purely dual formulations. [18] introduced dual variables and used alternating optimization to the resulting min-max problem. [14] and [23] dualized the problem and solved it with the steepest gradient ascent. [25] followed the same path but added kernels

to handle non-linearity. We follow the ideas of [22] and [26] and solve the problems directly in their primal formulations. Therefore, even though we use the same formulation for *TopPush* as [14] or for τ-*FPL* as [23], our solution process is different. However, due to convexity, both algorithms should converge to the same point.

The decision variables in (2.5) are the normal vector of the separating hyperplane $\boldsymbol{w}$ and the threshold t. To apply an efficient optimization method, we need to compute gradients. The simplest idea [3] is to compute the gradient only with respect to $\boldsymbol{w}$ and then recompute t. A more sophisticated way is based on the chain rule. For each $\boldsymbol{w}$, the threshold t can be computed uniquely. We stress this dependence by writing $t(\boldsymbol{w})$ instead of t. By doing so, we effectively remove the threshold t from the decision variables and $\boldsymbol{w}$ remains the only decision variable. Note that the convexity is preserved. Then we can compute the derivative via the chain rule

$$\begin{aligned} f(\boldsymbol{w}) &= \frac{1}{n^+} \sum_{\boldsymbol{x} \in \mathcal{X}^+} l(t(\boldsymbol{w}) - \boldsymbol{w}^\top \boldsymbol{x}) + \frac{\lambda}{2} \|\boldsymbol{w}\|^2, \\ \nabla f(\boldsymbol{w}) &= \frac{1}{n^+} \sum_{\boldsymbol{x} \in \mathcal{X}^+} l'(t(\boldsymbol{w}) - \boldsymbol{w}^\top \boldsymbol{x})(\nabla t(\boldsymbol{w}) - \boldsymbol{x}) + \lambda \boldsymbol{w}. \end{aligned} \tag{2.29}$$

The only remaining part is the computation of $\nabla t(\boldsymbol{w})$. It is simple for $\nabla t_1(\boldsymbol{w})$ and $\nabla t_2(\boldsymbol{w})$ and Theorem 2.4 shows the computation for $\nabla t_3(\boldsymbol{w})$. Appendix A.3 provides an efficient computation method for $t_3(\boldsymbol{w})$.

Having derivative (2.29), deriving the stochastic gradient is simple. It partitions the dataset into minibatches and provides an update of the weights $\boldsymbol{w}$ based only on a minibatch, namely by replacing the mean over the whole dataset in (2.29) by a mean over the minibatch.

2.3.2 Stochastic gradient descent: Convergent for *Pat&Mat* and *Pat&Mat-NP*

For the convergence proof, we need differentiability which is due to Theorem 2.4 possessed only by *Pat&Mat* and *Pat&Mat-NP*. Therefore, we consider only these two formulations and for simplicity, show it only for *Pat&Mat*. We apply a variance reduction technique based on delayed values similar to SAG [27].

At iteration k we have the decision variable $\boldsymbol{w}^k$ and the active minibatch I^k. First, we update the score vector $\boldsymbol{s}^k$ only on the active minibatch by setting

$$s_i^k = \begin{cases} \boldsymbol{x}_i^\top \boldsymbol{w}^k & \text{for all } i \in I^k, \\ s_i^{k-1} & \text{for all } i \notin I^k. \end{cases} \tag{2.30}$$

We keep scores from previous minibatches intact. We use Appendix A.3 to compute the surrogate quantile t^k as the unique solution of

$$\sum_{i \in X} l(\beta(s_i^k - t^k)) = n\tau. \tag{2.31}$$

This is an approximation of the surrogate quantile $t(\boldsymbol{w}^k)$ from (2.13). The only difference from the true value $t(\boldsymbol{w}^k)$ is that we use delayed scores. Then we introduce artificial variable

$$\boldsymbol{a}^k = \sum_{i \in I^k} l'(\beta(s_i^k - t^k))\boldsymbol{x}_i. \tag{2.32}$$

Finally, we approximate the derivative $\nabla f(\boldsymbol{w}^k)$ from (2.29) by

$$g(\boldsymbol{w}^k) = \frac{1}{n_+^k} \sum_{i \in I_+^k} l'(t^k - s_i^k)(\nabla t^k - \boldsymbol{x}_i), \tag{2.33}$$

where ∇t^k is an approximation of $\nabla t(\boldsymbol{w}^k)$ from Theorem 2.4 defined by

$$\nabla t^k = \frac{\boldsymbol{a}^k + \boldsymbol{a}^{k-1} + \cdots + \boldsymbol{a}^{k-m+1}}{\sum_{i \in X} l'(\beta(s_i^k - t^k))}. \tag{2.34}$$

A perhaps more straightforward possibility would be to consider only $\boldsymbol{a}^k$ in the numerator of (2.34). However, choice (2.34) enables us to prove the convergence and it adds stability to the algorithm for small minibatches.

The whole procedure does not perform any vector operations outside of the current minibatch I^k. We summarize it in Algorithm 1. Note that a proper initialization for the first m iterations is needed. We finish the theoretical part by the convergence proof.

Algorithm 1 Stochastic gradient descent for maximizing accuracy at the top

Require: Dataset X, Minibatches $I^1, \ldots, I^m$, Stepsize α^k

1: Initialize weights $\boldsymbol{w}^0$
2: **for** $k = 0, 1, \ldots$ **do**
3: Select a minibatch I^k
4: Compute s_i^k for all $i \in I^k$ according to (2.30)
5: Compute t^k according to (2.31)
6: Compute $\boldsymbol{a}^k$ according to (2.32)
7: Compute ∇t^k according to (2.34)
8: Compute $g(\boldsymbol{w}^k)$ according to (2.33)
9: Set $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k - \alpha^k g(\boldsymbol{w}^k)$
10: **end for**

Theorem 2.9

Consider the *Pat&Mat* or *Pat&Mat-NP* formulation, stepsizes $\alpha^k = \frac{\alpha^0}{k+1}$ and piecewise disjoint minibatches $I^1, \ldots, I^m$ which cycle periodically $I^{k+m} = I^k$. If l is the smoothened (Huberized) hinge function, then Algorithm 1 converges to the global minimum of (2.16).

2.4 Numerical experiments

In this section, we present numerical results.

2.4.1 Implementational details and Hyperparameter choice

We recall that all methods fall into the framework of either (2.2) or (2.5). Since the threshold t depends on the weights $\boldsymbol{w}$, we can consider the decision variable to be only $\boldsymbol{w}$. Then to apply a method, we implemented the following iterative procedure. At iteration j, we have the weights $\boldsymbol{w}^j$ to which we compute the threshold $t^j = t(\boldsymbol{w}^j)$. Then according

to (2.29), we compute the gradient of the objective and apply the ADAM descent scheme [28]. All methods were run for 10000 iterations using the stochastic gradient descent. The minibatch size was 512 except for the sigillito1989classification and Spambase datasets where the full gradient was used. All methods used the hinge surrogate (2.3). The initial point is generated randomly.

We run the methods for the following hyperparameters

$$\begin{aligned} \beta &\in \{0.0001,\ 0.001,\ 0.01,\ 0.1,\ 1,\ 10\}, \\ \lambda &\in \{0,\ 0.00001,\ 0.0001,\ 0.001,\ 0.01,\ 0.1\}, \\ k &\in \{1, 3, 5, 10, 15, 20\}. \end{aligned} \tag{2.35}$$

For *TopPushK*, *Pat&Mat* and *Pat&Mat-NP* we fixed $\lambda = 0.001$ to have six hyperparameters for all methods. For all datasets, we choose the hyperparameter which minimized the criterion on the validation set. The results are computed on the testing set which was not used during training the methods.

TopPush and τ*-FPL* were originally implemented in the dual. However, to allow for the same framework and the stochastic gradient descent, we implemented it in the primal. These two approaches are equivalent.

2.4.2 Dataset description and Performance criteria

For the numerical results, we considered 10 datasets summarized in Table 2.3. They can be downloaded from the UCI repository. sigillito1989classification [29] and Spambase are small, baldi2016parameterized [30] contains a large number of samples while guyon2005result [31] contains a large number of features. We also considered six visual recognition datasets: MNIST, FashionMNIST, CIFAR10, CIFAR20, CIFAR100 and SVHN2. MNIST and FashionMNIST are grayscale datasets of digits and fashion items, respectively. CIFAR100 is a dataset of coloured images of items grouped into 100 classes. CIFAR10 and CIFAR20 merge these classes into 10 and 20 superclasses, respectively. SVHN2 contains coloured images of house numbers. As Table 2.3 shows, these datasets are imbalanced.

Each of the visual recognition datasets was converted into ten binary datasets by considering one of the classes $\{0, \dots, 9\}$ as the positive class and the rest as the negative class. The experiments were repeated ten times for each dataset from different seeds, which influenced the starting point and minibatch creation. We use tpr@fpr as the evaluation criterion. This describes the true-positive rate at a prescribed true-negative rate, usually of 1% or 5%. For the linear classifier $\boldsymbol{w}^\top \boldsymbol{x} - t$, it selects the threshold t so that the desired true-negative rate is satisfied and then computes the true-positive rate for this threshold.

2.4.3 Numerical results

Figure 2.3 presents the standard ROC (receiver operating characteristic) curves on selected datasets. Since all methods from this paper are supposed to work at low false-positive rates, the x axis is logarithmic. Both figures depict averages over ten runs with different seeds. The left column depicts CIFAR100 while the right one Hepmass. These are the two more complicated datasets. We selected four representative methods: *Pat&Mat* and *Pat&Mat-NP* as our methods and *TopPush* and τ*-FPL* as state-of-the-art methods.

		Training		Validation		Testing	
	m	n	$\frac{n^+}{n}$	n	$\frac{n^+}{n}$	n	$\frac{n^+}{n}$
Ionosphere	34	175	36.0%	88	36.4%	88	35.2%
Spambase	57	2300	39.4%	1150	39.4%	1151	39.4%
Gisette	5000	1000	50.0%	1500	50.0%	500	50.0%
Hepmass	28	5250000	50.0%	1750000	50.0%	3500000	50.0%
MNIST	$28 \times 28 \times 1$	44999	11.2%	15001	11.2%	10000	11.4%
FashionMNIST	$28 \times 28 \times 1$	45000	10.0%	15000	10.0%	10000	10.0%
CIFAR10	$32 \times 32 \times 3$	37500	10.0%	12500	10.0%	10000	10.0%
CIFAR20	$32 \times 32 \times 3$	37500	5.0%	12500	5.0%	10000	5.0%
CIFAR100	$32 \times 32 \times 3$	37500	1.0%	12500	1.0%	10000	1.0%
SVHN2	$32 \times 32 \times 3$	54944	18.9%	18313	18.9%	26032	19.6%

Table 2.3: Structure of the used datasets. The training, validation and testing sets show the number of features m, samples n and the fraction of positive samples $\frac{n^+}{n}$.

Even though all methods work well, *Pat&Mat-NP* seems to outperform the remaining methods on most levels of false-positive rate.

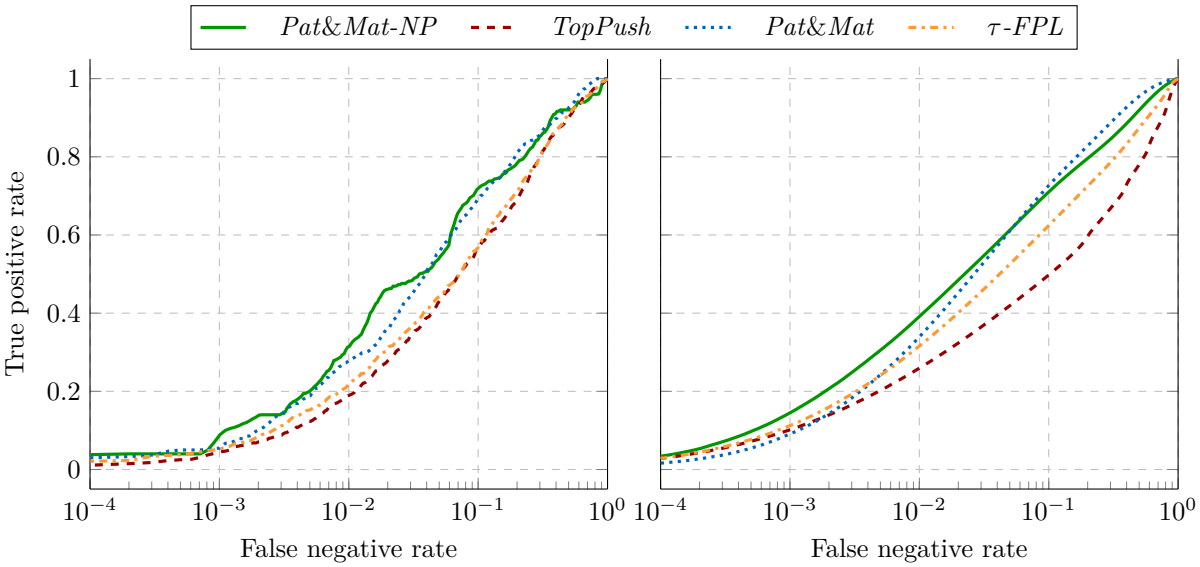


Figure 2.3: ROC curves (with logarithmic x axis) on CIFAR100 (left) and Hepmass (right).

While Figure 2.3 gave a glimpse of the behaviour of methods, Figures 2.4 and 2.5 provide a statistically more sound comparison. It employs the Nemenyi post hoc test for the Friedman test recommended in [32]. This test compares if the mean ranks of multiple methods are significantly different.

We consider 14 methods (we count different values of τ as different methods) as depicted in this table. For each dataset mentioned in Section 2.4.2 and each method, we evaluated the fpr@tpr metric and ranked all methods. Rank 1 refers to the best performance for given criteria, while rank 14 is the worst. The x-axis shows the average rank over all datasets. The Nemenyi test computes the critical difference. If two methods are within their critical difference, their performance is not deemed to be significantly different. Black wide horizontal lines group such methods.

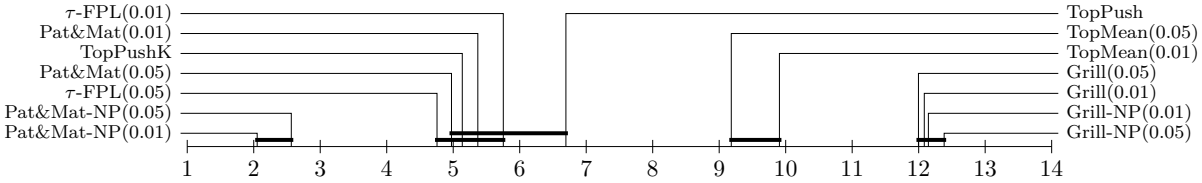


Figure 2.4: Critical difference (CD) diagrams (level of importance 0.05) of the Nemenyi post hoc test for the Friedman test. Each diagram shows the mean rank of each method, with rank 1 being the best. Black wide horizontal lines group together methods with the mean ranks that are not significantly different. The critical difference diagrams were computed for mean rank averages over all datasets of the tpr@fpr ($\tau = 0.01$) metric.

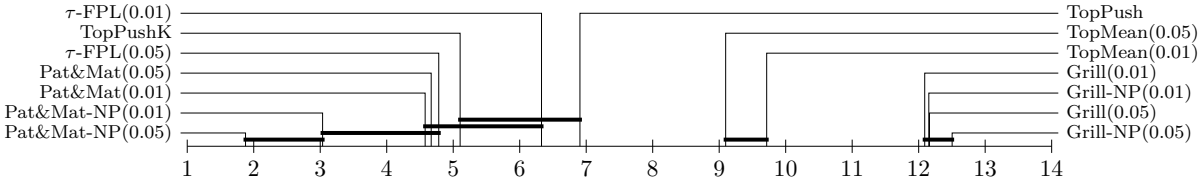


Figure 2.5: Critical difference (CD) diagrams (level of importance 0.05) of the Nemenyi post hoc test for the Friedman test. Each diagram shows the mean rank of each method, with rank 1 being the best. Black wide horizontal lines group together methods with the mean ranks that are not significantly different. The critical difference diagrams were computed for mean rank averages over all datasets of the tpr@fpr ($\tau = 0.05$) metric.

From this figure and table, we make several observations:

- *TopPushK* (rank 5.1) provides a slight improvement over *TopPush* (rank 6.7) even though this improvement is not statistically significant as both methods are connected by the black line in both Figures 2.4 and 2.5.
- Neither *Grill* (ranks 12.0 and 12.1) nor *Grill-NP* (ranks 12.1 and 12.4) perform well. We believe this happened due to the lack of convexity as indicated in Theorem 2.3 and the discussion after that.
- *TopMeanK* (ranks 9.2 and 9.9) does not perform well either. Since the thresholds τ are small, then $\boldsymbol{w} = 0$ is the global minimum as proved in Corollary 2.7.
- *Pat&Mat-NP* (rank 2.1 and 2.6) seems to outperform other methods.
- *Pat&Mat* (ranks 5.0 and 5.4), *τ-FPL* (ranks 4.8 and 5.8) and *TopPushK* (rank 5.1) perform similarly. Since they are connected, there is no statistical difference between their behaviours.

- *Pat&Mat-NP* at level 0.01 (rank 2.1) outperforms *Pat&Mat-NP* at level 0.05 (rank 2.6) for $\tau = 0.01$. *Pat&Mat-NP* at level 0.05 (rank 1.9 in Figure 2.5) outperforms *Pat&Mat-NP* at level 0.01 (rank 3.0 in Figure 2.5) for $\tau = 0.05$. This should be because these methods are optimized for the corresponding threshold. For τ-*FPL* we observed this behaviour for Figure 2.5 but not for Figure 2.4.

Figure 2.6 provides a similar comparison. Both axes are sorted from the best (left) to the worst (right) average ranks. The numbers in the graph show the p-value for the pairwise Wilcoxon signed-rank test, where the null hypothesis is that the mean tpr@fpr of both methods is the same. Even though Figure 2.4 employs a comparison of mean ranks and Figure 2.6 a pairwise comparison of fpr@tpr, the results are almost similar. Methods grouped by the black line in the former figure usually show a large p-value in the latter figure.

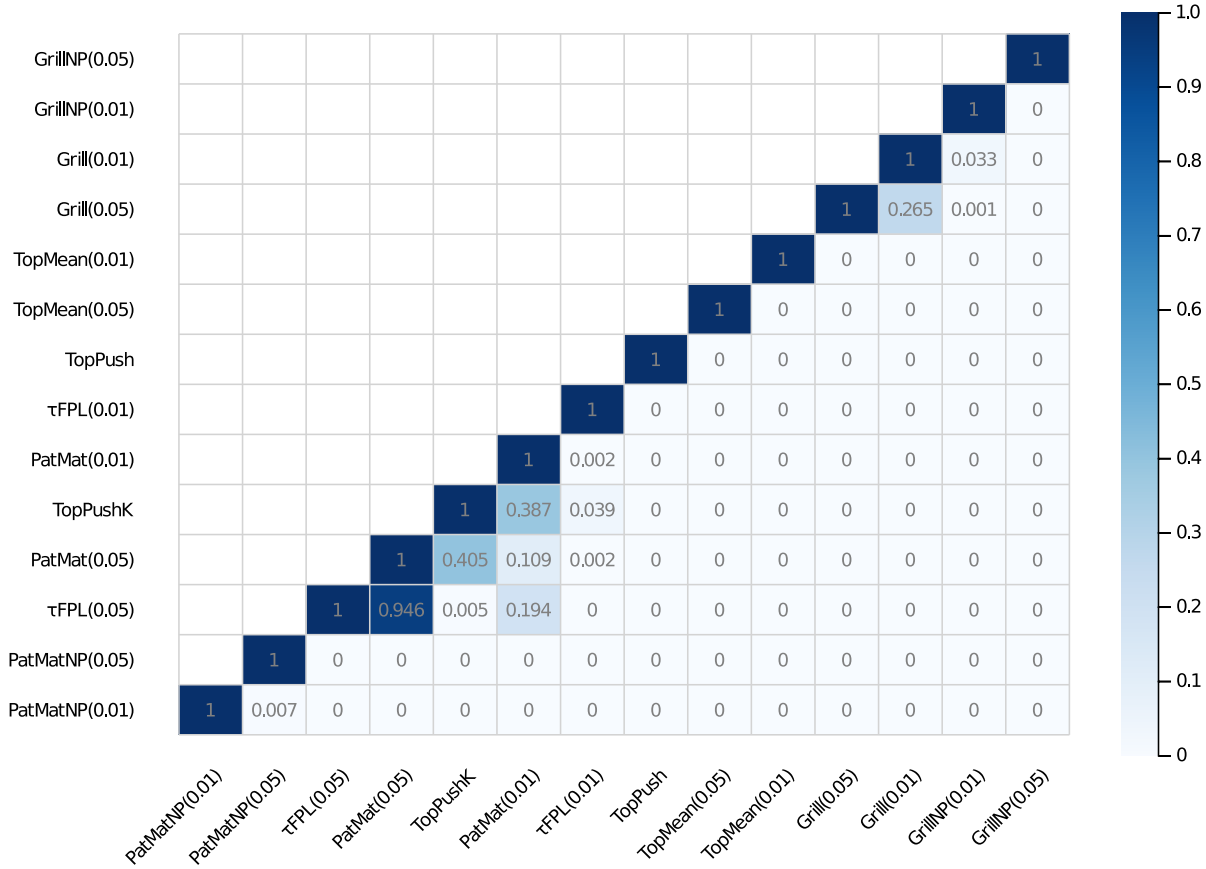


Figure 2.6: The p-value for the parwise Wilcoxon signed-rank test, where the null hypothesis is that the mean tpr@fpr(0.01) of both methods is the same. The methods are sorted by mean rank (left = better).

Table 2.4 investigates the impact of $\boldsymbol{w} = 0$ as a potential global minimum. Each method was optimized for six different values of hyperparameters. The table depicts the condition under which the final value has a lower objective than $\boldsymbol{w} = 0$. Thus, ✓ means that it is always better while ✗ means that the algorithm made no progress from the starting point $\boldsymbol{w} = 0$. The latter case implies that $\boldsymbol{w} = 0$ seems to be the global minimum. We make the following observations:

- *Pat&Mat* and *Pat&Mat-NP* are the only methods which succeeded at every dataset for some hyperparameter. Moreover, for each dataset, there was some β_0 such that these methods were successful if and only if $\beta \in (0, \beta_0)$. This is in agreement with Theorem 2.8.
- *TopMeanK* fails everywhere which agrees with Corollary 2.7.
- Figure 2.2 states that the methods from Section 2.1.2 has a higher threshold than their Neyman-Pearson variants from Section 2.1.3. This is documented in the table as the latter have a higher number of successes.

	Ionosphere	Hepmass	FashionMNIST	CIFAR100
TopPush	✓	✗	✓	✗
TopPushK	✓	✗	✓	✗
Grill $\tau = 0.01$	✗	✗	✗	✗
$\tau = 0.05$	✗	✗	✗	✗
Pat&Mat $\tau = 0.01$	✓	$\beta \leq 0.1$	$\beta \leq 1$	$\beta \leq 1$
$\tau = 0.05$	✓	$\beta \leq 1$	✓	✓
TopMeanK $\tau = 0.01$	✗	✗	✗	✗
$\tau = 0.05$	✗	✗	✗	✗
Grill-NP $\tau = 0.01$	✗	✗	✗	✗
$\tau = 0.05$	✗	✗	✗	✗
Pat&Mat-NP $\tau = 0.01$	✓	$\beta \leq 1$	✓	$\beta \leq 1$
$\tau = 0.05$	✓	✓	✓	$\beta \leq 1$
τ*-FPL* $\tau = 0.01$	✓	✗	✓	✗
$\tau = 0.05$	✓	✓	✓	$\lambda \leq 0.001$

Table 2.4: Necessary hyperparameter choice for the solution to have a better objective than zero. ✓ means that the solution was better than zero for all hyperparameters while ✗ means that it was worse for all hyperparameters.

2.5 Conclusion

In this paper, we achieved the following results:

- We presented a unified framework for the three criteria from Section 2.1. These criteria include ranking, accuracy at the top and hypothesis testing.
- We showed that several known methods (*TopPush*, *Grill*, τ*-FPL*) fall into our framework and derived some completely new methods (*Pat&Mat*, *Pat&Mat-NP*).

- We performed a theoretical analysis of the methods. We showed that known methods suffer from certain disadvantages. While *TopPush* and τ-*FPL* are sensitive to outliers, *Grill* is non-convex. We proved the global convergence of the stochastic gradient descent for *Pat&Mat* and *Pat&Mat-NP*.
- We performed a numerical comparison and we showed a good performance of our method *Pat&Mat-NP*.

3 Non-linear Classification at the Top

3.1 Introduction

The aim of classical linear binary classification is to separate positive and negative samples by a linear hyperplane. In many applications, it is desirable to separate only a certain number of samples. In such a case, the goal is not to maximize the performance on all samples but only the performance on the required samples with the highest relevance. Such classifiers have many applications. For example, in information retrieval systems, only the most relevant documents should be returned for a given query. Furthermore, they are useful in domains, where a large number of samples needs to be quickly screened and only a small subset of samples needs to be selected for further evaluation.

These problems can be generally written as pushing the positive samples above some decision threshold. The methods differ in the definition of the decision threshold. In our previous work [33], we introduced a general framework that unifies these methods. We showed that several problem classes, which were considered as separate problems so far, fit into the framework. As the most relevant we mention the following methods:

- *Ranking problems* focuses on ranking the positive samples higher than the negative ones. Many methods, such as *RankBoost* [11], *Infinite Push* [12] or *p-norm push* [13] employ a pairwise comparison of samples, which makes them infeasible for larger datasets. This was alleviated in *TopPush* [14] where the authors considered the limit $p \to \infty$. Since the l_∞ norm from *TopPush* is equal to the maximum, the decision threshold from our framework equals to the maximum of scores of negative samples. This was generalized into *TopPushK* [33] by considering the threshold to be the mean of K largest scores of negative samples.

- *Accuracy at the Top* [15] focuses on maximizing the number of positive samples above the top τ-quantile of scores. There are many methods on how to solve accuracy at the top. In [15], the authors assume that the top quantile is one of the samples, construct n unconstrained optimization problems with fixed thresholds, solve them and select the best solution. This method is computationally expensive. In [3] the authors propose a fast projected gradient descent method. In our previous paper, we proposed a convex approximation of the accuracy at the top called *Pat&Mat*. This method is reasonably fast and guaranteed the existence of global optimum.

The deficiency of methods from this framework is that they usually cover only linear classifiers. However, as many problems are not linearly separable, nonlinear classifiers are needed. In this work, we show how to extend our framework into nonlinear classification problems. To do so, we use the fact that our framework is similar to the primal formulation

of support vector machines [9]. The classical way to incorporate nonlinearity into SVM is to derive the dual formulation [24] and to employ the kernels method [34]. In this work, we follow this approach, derive dual formulations for the considered problems and add nonlinear kernels to them. Moreover, as dual problems are generally expensive to solve, we derive a quick method to solve them. This is a modification of the coordinate-wise dual ascent from [35]. For a review of other approaches see [36, 37].

The paper is organized as follows: In Section 3.2 we recall the unified framework derived in [33] and two class of problems that falls into it. Moreover, for selected methods, we derive their dual formulations. Namely, we focus on *TopPush*, *TopPushK* and *Pat&Mat*. In Section 3.3, we show how to add nonlinear kernels into dual formulations, derive a new method for solving these dual problems and perform its complexity analysis. Since our method depends on the chosen problem and surrogate function, we provide a concrete form of the solution for *TopPushK* with the truncated quadratic loss. Solutions for other problems are provided in Appendix B.3 and B.4. Finally, in Section 3.4 we present the description of performance criteria, choice of hyperparameters and description of datasets. The rest of the section is focused on the results of numerical experiments. Here we compare all methods in terms of overall accuracy and accuracy at a given threshold. We also discuss the convergence and time consumption of all methods. All our codes are available online.[1]

3.2 Derivation of dual problems

Linear binary classification is a problem of finding a linear hyperplane that separates a group of positive samples from a group of negative samples and achieves the lowest possible error. For a sample $\boldsymbol{x} \in \mathbb{R}^d$, the prediction for a linear classifier amounts to

$$\boldsymbol{x} \text{ has } \begin{cases} \text{positive label} & \text{if } \boldsymbol{w}^\top \boldsymbol{x} \geq t, \\ \text{negative label} & \text{otherwise.} \end{cases}$$

Here, $\boldsymbol{w} \in \mathbb{R}^d$ is the normal vector to the separating hyperplane and $t \in \mathbb{R}$ is a decision threshold. The well-known example of such a classifier is a support vector machine [9] where the decision threshold t is a free variable. However, many important binary classification problems maximize the performance only for a certain amount of samples with the highest scores $s = \boldsymbol{w}^\top \boldsymbol{x}$. In these cases, the threshold t is not a free variable but a function of the scores. In our previous work [33], we formulated a general framework for maximizing performance above the threshold t as

$$\begin{aligned} \underset{\boldsymbol{w},t}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n^+}[t - \boldsymbol{w}^\top \boldsymbol{x}_i^+] \\ \text{subject to} \quad & \text{threshold } t \text{ is a function of } \{\boldsymbol{w}^\top \boldsymbol{x}_i\}_{i=1}^n, \end{aligned} \tag{3.1}$$

where $C \in \mathbb{R}$ is a constant; $[\cdot]$ is the $0-1$ loss defined as 1 if the argument is true and 0 otherwise. To denote positive and negative samples, we use $+$ and $-$ symbols in the superscript, respectively. Note that $[t - \boldsymbol{w}^\top \boldsymbol{x}_i^+]$ counts the number of positive samples $\boldsymbol{x}_i^+$ whose score $\boldsymbol{w}^\top \boldsymbol{x}_i^+$ is below the threshold t. Since the objective is to be minimized, the positive samples should lie above the threshold t.

[1] `https://github.com/VaclavMacha/ClassificationOnTop_nonlinear.jl`

Since the objective function in (3.1) is discontinuous due to the $0-1$ loss, the optimization problem (3.1) is difficult. A typical approach to remove this unwanted feature is to approximate the $0-1$ loss by a surrogate function such as the truncated quadratic or the hinge functions

$$l_{\text{quadratic}}(s) = (\max\{0, 1+\vartheta s\})^2, \tag{3.2}$$
$$l_{\text{hinge}}(s) = \max\{0, 1+\vartheta s\}, \tag{3.3}$$

where $\vartheta > 0$ is a scaling parameter. In the following text, we use the symbol l to denote any convex non-negative non-decreasing function with $l(0) = 1$. Replacing the $0-1$ loss in (3.1) by a surrogate function results in

$$\begin{aligned} \underset{\boldsymbol{w},t}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n^+} l(t - \boldsymbol{w}^\top \boldsymbol{x}_i^+) \\ \text{subject to} & \text{threshold } t \text{ is a function of } \{\boldsymbol{w}^\top \boldsymbol{x_i}\}_{i=1}^n. \end{aligned} \tag{3.4}$$

Note that the objective function in (3.4) is continuous.

As we derived in [33], there are many problems belonging to the general framework (3.1). However, this framework handles only linear classification problems. As many problems are not linearly separable, this is often not sufficient. To generalize the framework to nonlinear classifiers, we realize that (3.4) is similar to the primal formulation of the SVM [9]. We will follow the standard way to incorporate nonlinearity into SVM by deriving the dual problem [24] and using the kernels methods [34].

In the remainder of this section, we recall two problem classes from [33] and their convex approximations, and for each of them, we derive its dual formulation. Namely, we will discuss *TopPushK* and *Accuracy at the Top* with its convex approximation *Pat&Mat*.

3.2.1 *TopPushK*

The first problem *TopPushK* is our modification of the *TopPush* method introduced in [14]. It selects the threshold t as the mean of the scores corresponding to K highest ranked negatives. By doing so, it enforces the positives to be ranked above negatives. Therefore, both *TopPush* ($K = 1$) and *TopPushK* ($K > 1$) fall into the category of ranking problems.

Writing this more formally, define vector $\boldsymbol{s}$ of all scores as $s_i = \boldsymbol{w}^\top \boldsymbol{x}_i$ for all $i = 1, 2, \ldots, n$ and its sorted version $\boldsymbol{s}_{[\cdot]}$ with decreasing components, i.e. $s_{[1]} \geq s_{[2]} \geq \cdots \geq s_{[n]}$. Then *TopPushK* reads

$$\begin{aligned} \underset{\boldsymbol{w},t,\boldsymbol{s}^-}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n^+} l(t - \boldsymbol{w}^\top \boldsymbol{x}_i^+) \\ \text{subject to} \quad & t = \frac{1}{K}\sum_{j=1}^{K} s_{[j]}^-, \\ & s_j^- = \boldsymbol{w}^\top \boldsymbol{x}_j^-, \quad \forall j = 1, 2, \ldots, n^-. \end{aligned} \tag{3.5}$$

It can be showed that this problem is convex and that for $K = 1$ we get the original *TopPush*.

In the following theorem, we denote the positive semidefinite kernel matrix $\mathbb{K}$ by

$$\mathbb{K} = \begin{pmatrix} \mathbb{X}^+ \\ -\mathbb{X}^- \end{pmatrix} \begin{pmatrix} \mathbb{X}^+ \\ -\mathbb{X}^- \end{pmatrix}^\top = \begin{pmatrix} \mathbb{X}^+\mathbb{X}^{+\top} & -\mathbb{X}^+\mathbb{X}^{-\top} \\ -\mathbb{X}^-\mathbb{X}^{+\top} & \mathbb{X}^-\mathbb{X}^{-\top} \end{pmatrix}. \tag{3.6}$$

and show the form of the *TopPushK* dual problem[2].

Theorem 3.1: *TopPushK* dual formulation

The dual problem corresponding to the problem (3.5) has the form

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix}^\top \mathbb{K} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} - C\sum_{i=1}^{n^+} l^\star\left(\frac{\alpha_i}{C}\right) \tag{3.7a}$$

$$\text{subject to} \sum_{i=1}^{n^+} \alpha_i = \sum_{j=1}^{n^-} \beta_j, \tag{3.7b}$$

$$0 \le \beta_j \le \frac{1}{K}\sum_{i=1}^{n^+} \alpha_i, \quad \forall j = 1, 2, \ldots, n^-, \tag{3.7c}$$

where $l^\star$ is a conjugate of the surrogate loss function l from (3.5) and $\mathbb{K}$ was defined in (3.6).

3.2.2 *Accuracy at the Top*

The second problem *Accuracy at the Top* was introduced in [15]. On the contrary to *TopPushK*, which focuses on the minimization of the number of positive samples below K highest ranked negatives, the *Accuracy at the Top* minimizes the number of positive samples below the top τ-quantile from negative samples defined as

$$t = \max\left\{ t \;\middle|\; \sum_{j=1}^{n^-} [\boldsymbol{w}^\top \boldsymbol{x}_j^- - t] \ge n\tau \right\}. \tag{3.8}$$

Then *Accuracy at the Top* is an optimization problem written as follows

$$\begin{aligned} \underset{\boldsymbol{w},t}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n^+} l_1(t - \boldsymbol{w}^\top \boldsymbol{x}_i^+) \\ \text{subject to} \quad & t \text{ is the surrogate top } \tau\text{-quantile: it solves (3.8).} \end{aligned} \tag{3.9}$$

Since it is known that the quantile function (3.8) is non-convex, we derived its convex surrogate approximation

$$t \quad \text{solves} \quad \sum_{j=1}^{n^-} l_2(\boldsymbol{w}^\top \boldsymbol{x}_j^- - t) \le n\tau. \tag{3.10}$$

[2]To keep the readability of the paper, we postpone all proofs to the Appendix

Replacing the true quantile (3.8) by its surrogate approximation (3.10) yields

$$\begin{aligned} \underset{\boldsymbol{w},t}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n^+} l_1(t - \boldsymbol{w}^\top \boldsymbol{x}_i^+) \\ \text{subject to} \quad & t \text{ is the top } \tau\text{-quantile: it solves (3.10).} \end{aligned} \tag{3.11}$$

In [33], we called the convex problem (3.11) *Pat&Mat*. The following theorem shows its dual form.

Theorem 3.2: *Pat&Mat* dual formulation

The dual problem corresponding to the problem (3.11) has the form

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} - C\sum_{i=1}^{n^+} l_1^\star\left(\frac{\alpha_i}{C}\right) - \delta\sum_{j=1}^{n^-} l_2^\star\left(\frac{\beta_j}{\delta}\right) - \delta n\tau \tag{3.12a}$$

$$\text{subject to} \sum_{i=1}^{n^+}\alpha_i = \sum_{j=1}^{n^-}\beta_j, \tag{3.12b}$$

$$\delta \geq 0, \tag{3.12c}$$

where $l_1^\star$, $l_2^\star$ are conjugates of the surrogate loss functions l_1, l_2 from (3.11) and $\mathbb{K}$ was defined in (3.6).

3.3 New method for solving dual problems

In the previous section, we derived the dual formulations for the *TopPushK* and *Pat&Mat* problems. These dual formulations allow us to incorporate nonlinearity using kernels [34] in the same way as in SVM. Since the dimension of (3.7) and (3.12) equals to the number of samples n, it is computationally expensive to use standard techniques such as the gradient descent. To handle this issue, the coordinate descent algorithm [38, 35] has been proposed in the context of SVMs. Since problems (3.7, 3.12) differ from original SVMs by additional constraints (3.7b, 3.12b), the key idea of our algorithm is to update two coordinates (instead of one) of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ at every iteration. To summarize, we will solve the original tasks (3.7, 3.12) by an iterative procedure where in every iteration we need to find a solution of a one-dimensional quadratic optimization problem. As we will show later, these one-dimensional problems have a closed form solution, which means that every iteration is cheap.

3.3.1 Adding kernels

To add kernels, we realize first that from the proofs of Theorems 3.1 and 3.2 for any $\boldsymbol{z} \in \mathbb{R}^d$ we have

$$\boldsymbol{w}^\top \boldsymbol{z} = \sum_{i=1}^{n^+}\alpha_i \boldsymbol{z}^\top \boldsymbol{x}_i^+ - \sum_{j=1}^{n^-}\beta_j \boldsymbol{z}^\top \boldsymbol{x}_j^- \tag{3.13}$$

Consider now any kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Using the standard trick, we replace the kernel matrix (3.6) by[3]

$$\mathbb{K} = \begin{pmatrix} k(\mathbb{X}^+, \mathbb{X}^+) & -k(\mathbb{X}^+, \mathbb{X}^-) \\ -k(\mathbb{X}^-, \mathbb{X}^+) & k(\mathbb{X}^-, \mathbb{X}^-) \end{pmatrix}, \tag{3.14}$$

where $k(\cdot, \cdot)$ is applied to all rows of both arguments. Then for a new sample $\boldsymbol{z}$, the prediction (3.13) is replaced by

$$\text{pred}(\boldsymbol{z}) = \sum_{i=1}^{n^+} \alpha_i k\left(\boldsymbol{z}, \boldsymbol{x}_i^+\right) - \sum_{j=1}^{n^-} \beta_j k\left(\boldsymbol{z}, \boldsymbol{x}_j^-\right), \tag{3.15}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the optimal solution of (3.7) or (3.12).

3.3.2 Update of dual optimization variables

Let us consider the kernel matrix $\mathbb{K}$ as in (3.14) and define the score vector $\boldsymbol{s}$ by

$$\boldsymbol{s} = \mathbb{K} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix}. \tag{3.16}$$

There are three possible update rules which modify two coordinates of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and which satisfy constraints (3.7b, 3.12b) and keep (3.16) satisfied. The first one updates two components of $\boldsymbol{\alpha}$

$$\alpha_k \to \alpha_k + \Delta, \qquad \alpha_l \to \alpha_l - \Delta, \qquad \boldsymbol{s} \to \boldsymbol{s} + (\mathbb{K}_{\bullet k} - \mathbb{K}_{\bullet l})\Delta, \tag{3.17a}$$

where $K_{\bullet i}$ denotes i-th column of $\mathbb{K}$. Note that the update rule for $\boldsymbol{s}$ does not use matrix multiplication but only vector addition. The second rule updates one component of $\boldsymbol{\alpha}$ and one component of $\boldsymbol{\beta}$

$$\alpha_k \to \alpha_k + \Delta, \qquad \beta_l \to \beta_l + \Delta, \qquad \boldsymbol{s} \to \boldsymbol{s} + (\mathbb{K}_{\bullet k} + \mathbb{K}_{\bullet l})\Delta, \tag{3.17b}$$

and the last one updates two components of $\boldsymbol{\beta}$

$$\beta_k \to \beta_k + \Delta, \qquad \beta_l \to \beta_l - \Delta, \qquad \boldsymbol{s} \to \boldsymbol{s} + (\mathbb{K}_{\bullet k} - \mathbb{K}_{\bullet l})\Delta. \tag{3.17c}$$

These three update rules hold true for any surrogate function. However, the calculation of the optimal Δ depends on the used problem formulation and surrogate function. In Subsection 3.3.4, we show the closed-form formula for Δ for *TopPushK* problem (3.7) with truncated quadratic surrogate function (3.2). Computation of Δ for the hinge surrogate or *Pat&Mat* is presented in Appendices B.3 and B.4.

[3] The first part of the objective of (3.7) and (3.12) amounts to

$$\begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix}^\top \mathbb{K} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix}^\top \begin{pmatrix} \mathbb{X}^+\mathbb{X}^{+\top} & -\mathbb{X}^+\mathbb{X}^{-\top} \\ -\mathbb{X}^-\mathbb{X}^{+\top} & \mathbb{X}^-\mathbb{X}^{-\top} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\alpha} \\ -\boldsymbol{\beta} \end{pmatrix}^\top \begin{pmatrix} \mathbb{X}^+\mathbb{X}^{+\top} & \mathbb{X}^+\mathbb{X}^{-\top} \\ \mathbb{X}^-\mathbb{X}^{+\top} & \mathbb{X}^-\mathbb{X}^{-\top} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ -\boldsymbol{\beta} \end{pmatrix},$$

from which (3.14) follows.

3.3.3 Algorithm summary and Complexity analysis

We summarize the whole procedure in Algorithm 2. We will describe it only for *Pat&Mat* (right column) as for *TopPushK* (left column) it is almost identical. In step 1 we initialize $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and δ to some feasible value and based on (3.16) compute $\boldsymbol{s}$. Each `repeat` loop in step 2 updates two coordinates as shown in (3.17). In step 3 we select a random index k and in the `for` loop in step 4 we compute the optimal (Δ_l, δ_l) for all possible combinations (k, l) as in (3.17). In step 7 we select the pair (Δ_l, δ_l) which maximizes the objective. Finally, based on (3.17) we update $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{s}$ and δ in steps 8 and 9.

Algorithm 2 Coordinate descent algorithm for *TopPushK* (left) and *Pat&Mat* (right).

1: set $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ feasible, set $\boldsymbol{s}$ based on (3.16)	1: set $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta)$ feasible, set $\boldsymbol{s}$ based on (3.16)
2: **repeat**	2: **repeat**
3: select random k from $\{1, 2, \ldots, n\}$	3: select random k from $\{1, 2, \ldots, n\}$
4: **for** $l \in \{1, 2, \ldots, n\}$ **do**	4: **for** $l \in \{1, 2, \ldots, n\}$ **do**
5: compute Δ_l	5: compute $(\Delta_l,\ \delta_l)$
6: **end for**	6: **end for**
7: select best Δ_l	7: select best $(\Delta_l,\ \delta_l)$
8: update $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{s}$ according to (3.17)	8: update $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{s}$ according to (3.17)
9:	9: set $\delta \leftarrow \delta_l$
10: **until** stopping criterion is satisfied	10: **until** stopping criterion is satisfied

Now we derive the computational complexity of each `repeat` loop from step 2. Since the computation of (Δ_l, δ_l) amounts to solving a quadratic optimization problem in one variable, there is a closed-form solution (see Section 3.3.4) and step 5 can be performed in $O(1)$. Since this is embedded in a `for` loop in step 4, the whole complexity of this loop is $O(n)$. Step 8 requires $O(1)$ for the update of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ while $O(n)$ for the update of $\boldsymbol{s}$. Since the other steps are $O(1)$, the total complexity of the `repeat` loop is $O(n)$. This holds true only if the kernel matrix $\mathbb{K}$ is precomputed. In the opposite case, all complexities must by multiplied by the cost of computation of components of $\mathbb{K}$ which is $O(d)$. This complexity analysis is summarized in Table 3.1.

Precomputed $\mathbb{K}$	Evaluation of Δ_l	Update of $\boldsymbol{s}$	Total per iteration
✓	$O(1)$	$O(n)$	$O(n)$
✗	$O(d)$	$O(nd)$	$O(nd)$

Table 3.1: Computational complexity of one `repeat` loop (which updates two coordinates of $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$) from Algorithm 2.

3.3.4 Computing Δ for *TopPushK* with truncated quadratic loss

In this section, we show how to compute the stepsize Δ from (3.17) for *TopPushK* (3.7) with the truncated quadratic surrogate function (3.2). Optimal Δ for *Pat&Mat* can be found in a similar way as we show in Appendix B.3. In Appendix B.4 we present the computation of optimal Δ for *TopPushK* and *Pat&Mat* with the hinge loss function (3.3).

Plugging the conjugate (B.2) of the truncated quadratic loss (3.2) into *TopPushK* dual formulation (3.7) yields

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}+\frac{1}{\vartheta}\sum_{i=1}^{n^+}\alpha_i-\frac{1}{4C\vartheta^2}\sum_{i=1}^{n^+}\alpha_i^2 \tag{3.18a}$$

$$\text{subject to}\sum_{i=1}^{n^+}\alpha_i=\sum_{j=1}^{n^-}\beta_j, \tag{3.18b}$$

$$\alpha_i\geq 0, \qquad \forall i=1,2,\ldots,n^+, \tag{3.18c}$$

$$0\leq\beta_j\leq\frac{1}{K}\sum_{i=1}^{n^+}\alpha_i, \quad \forall j=1,2,\ldots,n^-. \tag{3.18d}$$

This is a quadratic optimization problem. Moreover, for $K=1$ the upper bound in (3.18d) automatically follows from (3.18b) and the problem can be simplified. In the following theorem, we show that for each of update rules (3.17), problem (3.18) has a simple closed-form solution. For simplicity, we use $\text{clip}_{[a,\,b]}(c)$ to denote clipping (projecting) c to the interval $[a,b]$.

Theorem 3.3: Update rule for Δ^* for *TopPushK* with truncated quadratic loss

Consider problem (3.18). Then the optimal step $\Delta^\star$ equals to

$$\Delta^*=\text{clip}_{[\Delta_{lb},\,\Delta_{ub}]}(\gamma), \tag{3.19}$$

where there are the following three cases (each corresponding to one update rule in (3.17)):

- For any $1\leq k,l\leq n^+$ we have

$$\begin{aligned}\Delta_{lb}&=-\alpha_k,\\ \Delta_{ub}&=\alpha_l,\\ \gamma&=-\frac{s_k-s_l+\frac{1}{2C\vartheta^2}(\alpha_k-\alpha_l)}{\mathbb{K}_{kk}+\mathbb{K}_{ll}-\mathbb{K}_{kl}-\mathbb{K}_{lk}+\frac{1}{C\vartheta^2}}.\end{aligned}$$

- For any $1\leq k\leq n^+$ and $n^++1\leq l\leq n$ we define $\hat{l}=l-n^+$ and $\beta_{\max}=\max_{j\in\{1,2,\ldots,n^-\}\setminus\{\hat{l}\}}\beta_j$. Then we have

$$\Delta_{lb}=\begin{cases}\max\{-\alpha_k,\ -\beta_{\hat{l}}\} & \text{K}=1,\\ \max\left\{-\alpha_k,\ -\beta_{\hat{l}},\ K\beta_{\max}-\sum_{i=1}^{n^+}\alpha_i\right\} & \text{otherwise},\end{cases}$$

$$\Delta_{ub}=\begin{cases}+\infty & \text{K}=1,\\ \frac{1}{K-1}\left(\sum_{i=1}^{n^+}\alpha_i-K\beta_{\hat{l}}\right) & \text{otherwise},\end{cases}$$

$$\gamma=-\frac{s_k+s_l-\frac{1}{\vartheta}+\frac{1}{2C\vartheta^2}\alpha_k}{\mathbb{K}_{kk}+\mathbb{K}_{ll}+\mathbb{K}_{kl}+\mathbb{K}_{lk}+\frac{1}{2C\vartheta^2}}.$$

- For any $n^+ + 1 \leq k, l \leq n$ we define $\hat{k} = k - n^+$, $\hat{l} = l - n^+$ and then have

$$\Delta_{lb} = \begin{cases} -\beta_{\hat{k}} & \text{K} = 1, \\ \max\left\{-\beta_{\hat{k}},\ \beta_{\hat{l}} - \frac{1}{K}\sum_{i=1}^{n^+} \alpha_i\right\} & \text{otherwise}, \end{cases}$$

$$\Delta_{ub} = \begin{cases} \beta_{\hat{l}} & \text{K} = 1, \\ \min\left\{\beta_{\hat{l}},\ \frac{1}{K}\sum_{i=1}^{n^+} \alpha_i - \beta_{\hat{k}}\right\} & \text{otherwise}, \end{cases}$$

$$\gamma = -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}.$$

3.4 Numerical experiments

In this section, we present numerical results. All codes were implemented in the Julia language [39] and are available online.[4]

3.4.1 Performance criteria

For the evaluation of numerical experiments, we use precision and recall. For a threshold t they are defined by

$$\text{precision} = \frac{\sum_{i=1}^{n^+}[\boldsymbol{w}^\top \boldsymbol{x}_i^+ - t]}{\sum_{i=1}^{n}[\boldsymbol{w}^\top \boldsymbol{x}_i - t]}, \qquad \text{recall} = \frac{1}{n^+}\sum_{i=1}^{n^+}[\boldsymbol{w}^\top \boldsymbol{x}_i^+ - t]. \tag{3.20}$$

We also use the Precision-Recall (PR) curve that are commonly used for unbalanced data [40] and precision at a certain level of recall which we denote by Precision@Recall.

3.4.2 Hyperparameter choice

In Section 3.3 we introduced Algorithm 2 for solving dual problems (3.7, 3.12). We let it run for 20000 `repeat` loops, which corresponds to 40000 updates of coordinates of $(\boldsymbol{\alpha}, \boldsymbol{\beta})$. We use the linear and Gaussian kernels defined by

$$k_{\text{lin}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}, \tag{3.21}$$

$$k_{\text{gauss}}(\boldsymbol{x}, \boldsymbol{y}) = \exp\{-\sigma\|\boldsymbol{x} - \boldsymbol{y}\|_2^2\} \tag{3.22}$$

and the truncated quadratic loss (3.2) with $\vartheta = 1$ as a surrogate.

The classifiers were trained on the training set. We selected the optimal hyperparameter from

$$\tau \in \{0.01,\ 0.05,\ 0.1\}, \quad K \in \{5,\ 10\}, \quad C \in \{0.1,\ 1,\ 10\}, \quad \sigma \in \{0.01,\ 0.05\}$$

which gave the best performance on the validation set. All presented result are shown on the testing set which was not part of the training process.

[4] All codes are available at https://github.com/VaclavMacha/ClassificationOnTop_new.jl

			Training		Validation		Testing	
	y^+	d	n	$\frac{n^+}{n}$	n	$\frac{n^+}{n}$	n	$\frac{n^+}{n}$
sigillito1989classification	–	34	176	64.2%	87	64.4%	88	63.6%
Spambase	–	57	2301	39.4%	1150	39.4%	1150	39.4%
WhiteWineQuality	7, 8, 9	11	2449	21.6%	1224	21.7%	1225	21.6%
RedWineQuality	7, 8	11	800	13.5%	400	13.8%	399	13.5%
Fashion-MNIST	0	784	50000	10.0%	10000	10.0%	10000	10.0%

Table 3.2: Summary of the used datasets. It shows which original labels y^+ were selected as the positive class, the number of features d, samples n, and the fraction of positive samples $\frac{n^+}{n}$.

3.4.3 Dataset description

For numerical experiments, we consider the FashionMNIST dataset [41] and four smaller datasets from the UCI repository [42]: sigillito1989classification [29], Spambase, WhiteWineQuality [43] and RedWineQuality [43]. Datasets that do not contain testing set were randomly divided into a training (50%), validation (25%) and testing (25%) sets. For datasets that contain a testing set, the training set was randomly divided into a training and a validation set, where the validation set has the same size as the testing set. FashionMNIST dataset was converted to binary classification tasks by selecting class with label 0 as the positive class and the rest as the negative class. All datasets are summarized in Table 3.2.

3.4.4 Experiments

In Figure 3.1 we present the PR curves for all methods with two different kernels evaluated on the FashionMNIST dataset. The left column corresponds to the linear kernel (3.21) while the right one to the Gaussian kernel (3.22) with $\sigma = 0.01$. The nonlinear Gaussian kernel significantly outperforms the linear kernel. This will be confirmed later in Table 3.3 where we present a comparison from multiple datasets.

For a better illustration of how the methods from Figure 3.1 work, we present density estimates of scores $\boldsymbol{s}$ from (3.16). High scores predict positive labels while low scores predict negative labels. The rows of Figure 3.2 depict the linear (3.21) and the Gaussian kernels (3.22) with $\sigma = 0.01$ while each column corresponds to one method. The black vertical lines depict the top 5%-quantile of all scores (on the testing set). Since a smaller overlap of scores of samples with positive and negative labels implies a better separation, we deduce the benefit of the Gaussian over the linear kernel.

In Table 3.3 we present the precision of all methods across all datasets from Table 3.2. For each dataset, we trained each method and computed precision at certain levels of recall. The depicted values are averages over all datasets. For each kernel and each level of recall, the best precision is highlighted in light green. Moreover, the best overall precision for each level of recall is depicted in dark green. We can make several observations from Table 3.3:

- All methods perform better with the Gaussian kernels than with the linear kernel.

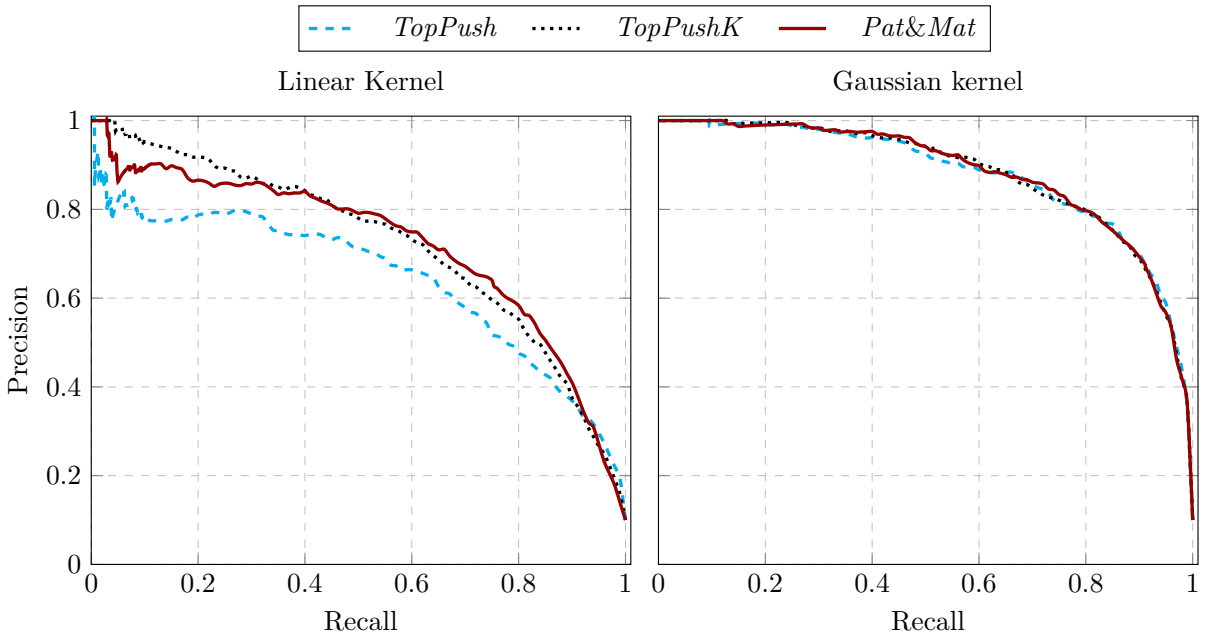


Figure 3.1: PR curves for all methods and FashionMNIST dataset. The left column corresponds to the linear kernel (3.21) and the right column corresponds to the Gaussian kernel (3.22).

- *TopPush* and *TopPushK* perform better for sufficiently small recall. This happened because they consider the threshold to be the maximal K negative scores and small recall corresponds to high threshold. However, for the same reason, *TopPush* is not robust.
- *Pat&Mat* is the best for all kernels if the recall is sufficiently large. The reason is again the form of the decision threshold.

In Figure 3.3, we investigate the convergence of methods. In each column, we show the convergence of primal and dual problems for one method. To solve the primal problem, we use the gradient method proposed in [33]. For the dual problem, we use our Algorithm 2. Since [33] considers only linear kernels, we present them. Moreover, since the computation of the objective is expensive, the results are presented for the sigillito1989classification dataset. We can see that *TopPush* and *TopPushK* converge to the same objective for primal and dual problems. This means that the problem was solved to optimality. However, there is a little gap between optimal solution of primal and dual problems for *Pat&Mat*.

Finally, Table 3.4 depicts the time comparison for all methods and all datasets. It shows the average time in milliseconds needed for one `repeat` loop in Algorithm 2. The time is relatively stable and for most of the datasets it is below one millisecond. Since we run all experiments for 20000 `repeat` loops, the evaluation of one method with one hyperparameter setting takes a few seconds for smaller datasets and approximately 7 minutes for FashionMNIST. The average time for one Δ_l in step 5 in Algorithm 2 took between $1.7 \cdot 10^{-7}$ and $3.1 \cdot 10^{-7}$ seconds for each methods. It is almost the same for all datasets, which corresponds to the fact that the complexity of step 5 is independent of the size of the dataset. Note that in all experiments we used precomputed kernel matrix $\mathbb{K}$ saved on the hard drive and not in memory.

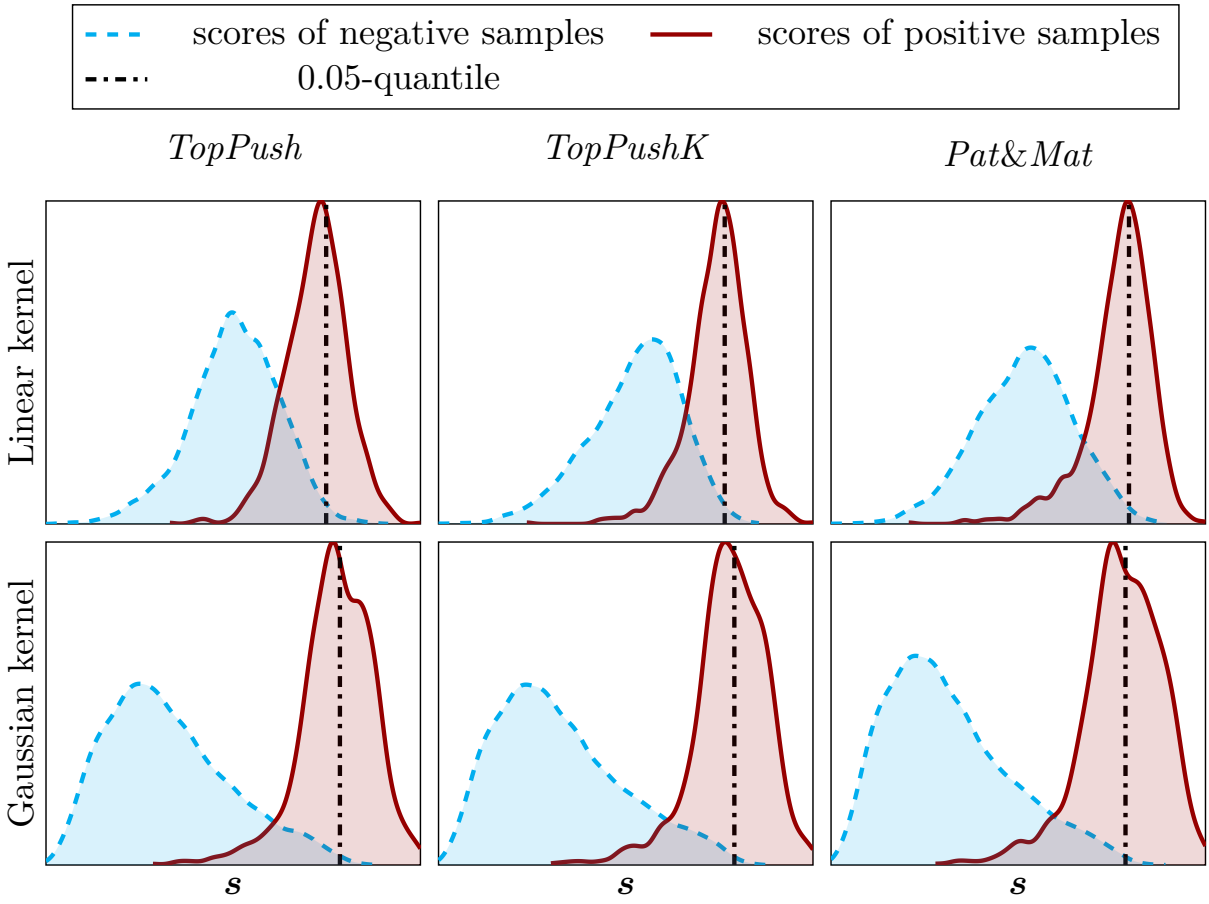


Figure 3.2: Density estimates for scores corresponding to samples with positive and negative labels for the FashionMNIST dataset.

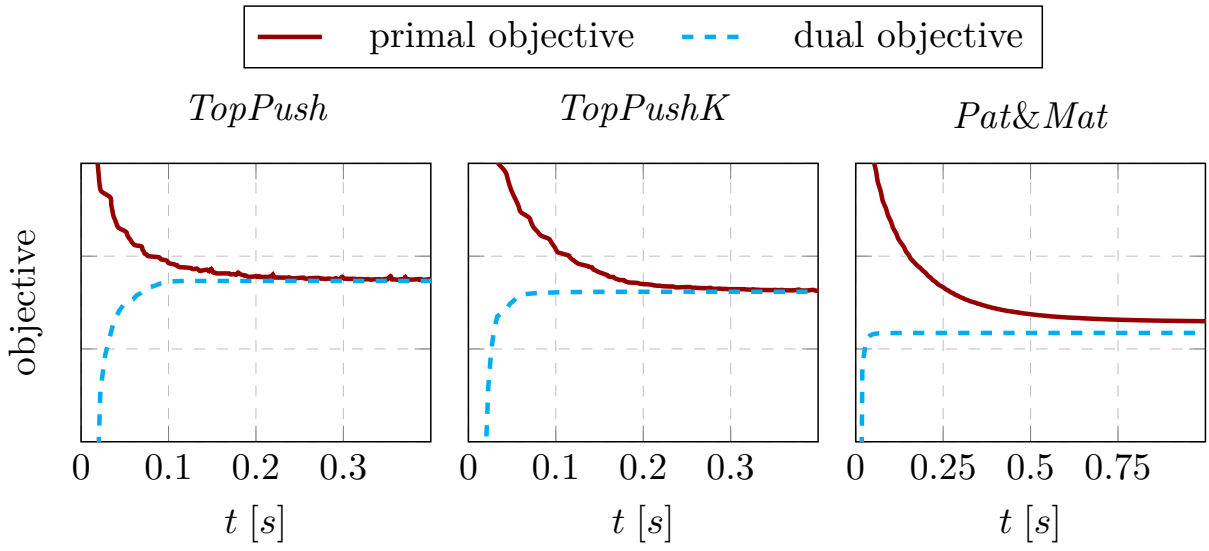


Figure 3.3: Convergence of the objectives for the primal (red line) and dual (blue line) problems for the sigillito1989classification dataset with linear kernel.

3.5 Conclusion

In this paper, we analyzed and extended the general framework for binary classification on top samples from [33] to nonlinear problems. Achieved results can be summarized as

			Precision@Recall					
			0.05	0.1	0.2	0.4	0.6	0.8
Linear kernel	*TopPush*		79.83	64.27	65.55	61.85	57.89	51.83
	TopPushK	$K = 5$	73.96	65.41	64.82	60.28	56.94	50.52
		$K = 10$	60.63	61.97	59.69	56.89	54.40	49.83
	Pat&Mat	$\tau = 0.01$	63.67	60.30	58.74	57.75	53.32	48.42
		$\tau = 0.05$	54.05	60.91	63.32	55.24	52.55	48.30
		$\tau = 0.1$	57.02	61.24	62.49	63.11	59.91	52.14
Gaussian kernel	*TopPush*		97.50	86.06	81.28	76.15	71.13	60.17
	TopPushK	$K = 5$	92.50	87.56	85.31	78.47	70.77	57.10
		$K = 10$	89.50	87.56	83.15	79.09	71.88	59.27
	Pat&Mat	$\tau = 0.01$	89.65	89.11	86.75	80.77	75.44	65.95
		$\tau = 0.05$	80.77	81.28	85.74	82.92	74.91	65.04
		$\tau = 0.1$	81.30	84.14	82.58	83.12	77.82	66.50

Table 3.3: The precision of all methods averaged across all datasets from Table 3.2. Each column represents precision at a certain level of recall. Light green depicts the best method for the given kernel and dark green depicts the best overall method.

follows:

- We derived the dual formulations for *TopPush*, *TopPushK* and *Pat&Mat*.
- We proposed a new method for solving the dual problems. We performed its complexity analysis. For selected surrogate functions we also derived the exact formulas needed in the method.
- We performed a numerical analysis of the proposed method. We showed its good convergence as well as improved performance of nonlinear kernels over the linear one.

Based on the numerical analysis from Section 3.4, we recommend using *TopPush* or *TopPushK* for problems where the resulting recall should be small. Otherwise, we recommend using *Pat&Mat* with an appropriately selected τ parameter.

		TopPush	*TopPushK*	*Pat&Mat*
One `repeat` loop [ms]	sigillito1989classification	0.04 ± 0.00	0.03 ± 0.00	0.03 ± 0.00
	Spambase	0.56 ± 0.02	0.49 ± 0.01	0.50 ± 0.01
	WhiteWineQuality	0.62 ± 0.03	0.53 ± 0.01	0.54 ± 0.01
	RedWineQuality	0.17 ± 0.01	0.14 ± 0.01	0.15 ± 0.01
	Fashion-MNIST	17.16 ± 0.74	15.95 ± 0.14	15.54 ± 0.80

Table 3.4: The average time with standard deviation (in milliseconds) for one `repeat` loop in Algorithm 2. The average time for one Δ_l in step 5 in Algorithm 2 took between $1.7 \cdot 10^{-7}$ and $3.1 \cdot 10^{-7}$ seconds for each methods.

4

Deep

Binary classifiers compute a score for each sample and compare it with a given threshold to predict whether the sample belongs to the positive or negative class. This score is often interpreted as the probability that the sample is of the positive class, and the threshold is usually 0.5. Such a task attempts to classify all samples correctly.

On the other hand, many applications need to classify only a small fraction of samples correctly. In information retrieval systems, the user is interested only in the top few queries. Whenever samples undergo manual processing, humans can process only a small fraction of all samples. Cybersecurity defensive mechanisms must have extremely low false positive rates; otherwise, they are removed by the user. In all these applications, the score determines the sample relevance. After computing the scores of all samples, they are compared with a threshold. All samples below the threshold are irrelevant. The small fraction of samples above the threshold is deemed positive. These selected samples are then the search results, items for further analysis, or the detected malware.

Since this task considers only scores above the threshold, [15] named it *Accuracy at the Top*. The important distinction from standard classifiers is that this threshold is no longer fixed, as in the case of 0.5, but depends on all samples. Therefore, the objective is non-additive and non-decomposable. This brings both theoretical and numerical issues. Standard machine learning algorithms use minibatch sampling. However, when the threshold is computed on a minibatch, it provides a lower estimate of the true threshold. Therefore, the sampled threshold is a biased estimate of the true threshold. Figure 4.1 illustrates this phenomenon. The bias between the true and sampled thresholds is large even for medium-sized minibatches. Backpropagation then propagates this sampling error through the whole gradient, and consequently, the minibatch gradient is a biased estimate of the true gradient. This brings numerical issues [44].

Our method mitigates this bias. It is based on several results. [14] proposed the TopPush formulation of the accuracy at the top and solved it in its dual formulation. [33] solved the TopPush formulation directly in its primal form for linear classifiers. Since we generalize the linear TopPush into non-linear classifiers, we name our method *DeepTopPush*. We stay in the primal form to be able to employ stochastic gradient descent. Due to non-decomposability, we need to propose a way of computing the gradient and reduce the bias mentioned above. Since the threshold always equals to one of the scores [15], its computation has a simple local formula. We implicitly remove some variables and apply the chain rule (backpropagation) to compute the gradient in an end-to-end manner. To reduce the bias, we need to improve the approximation quality of the sampled threshold. We employ again the fact that the true threshold corresponds to one sample. Since this sample changes slowly during optimization, we modify the idea of [26] and enhance

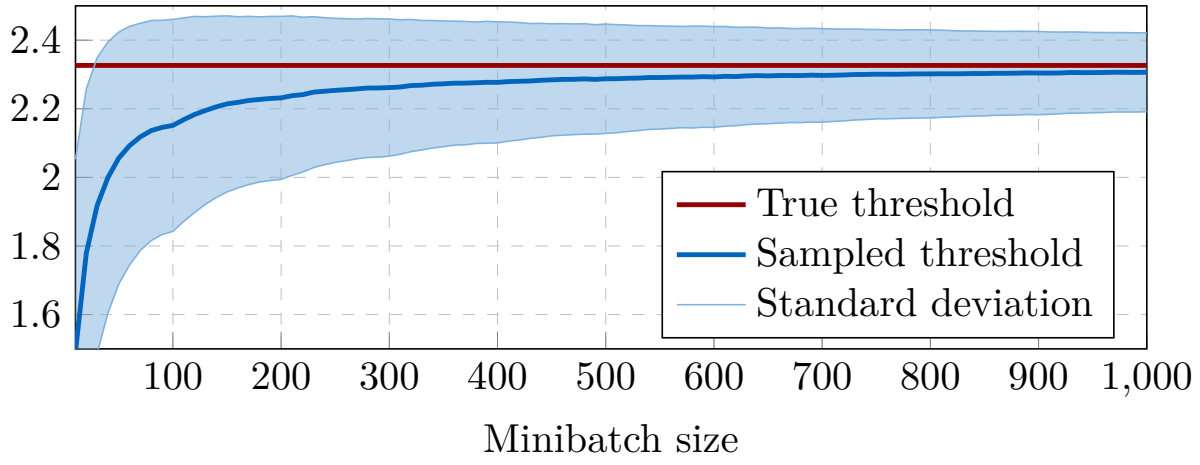


Figure 4.1: The bias between the sampled and true thresholds computed from scores following the standard normal distribution. The threshold separates the top 1% of samples with the highest scores.

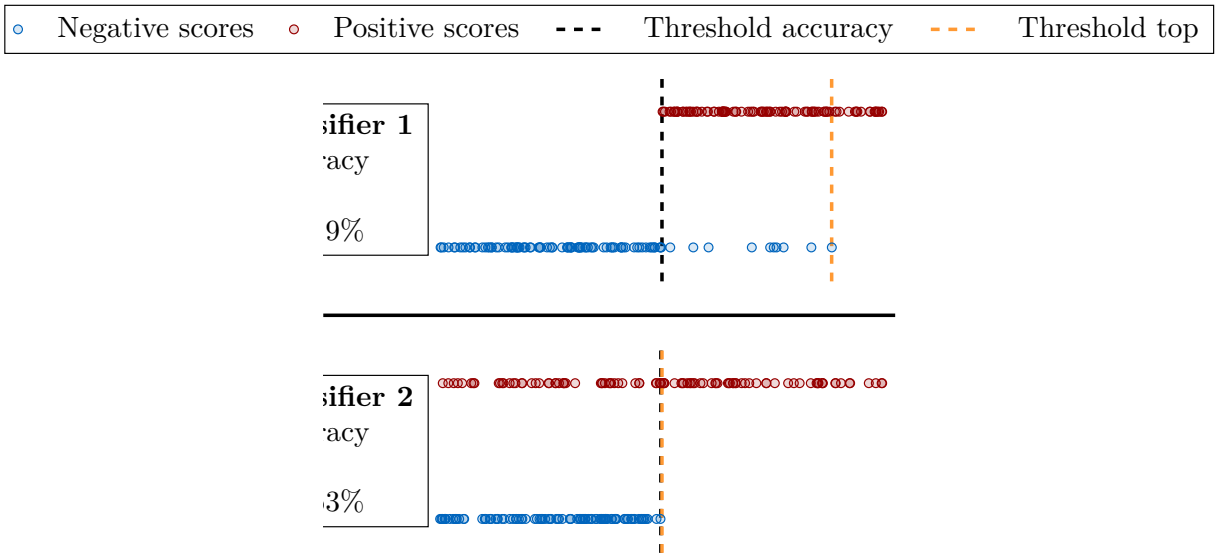


Figure 4.2: Difference between standard classifiers (top row) and classifiers maximizing accuracy at the top (bottom row). While the former has a good total accuracy, the latter has a good top acccuracy.

the current minibatch by the sample, which equalled the sampled threshold on the previous minibatch. As this added sample usually propagates across multiple minibatches, it tracks the threshold, and this trick mitigates the sampled threshold bias. The main contributions of the paper are as follows:

- We propose *DeepTopPush*, which is a simple and scalable method for accuracy at the top.
- We show that *DeepTopPush* increases the computational time only slightly, yet it achieves better performance than prior art methods.
- We show both theoretically and numerically that enhancing the minibatch by one sample reduces the bias of the sampled gradient.

The paper is organized as follows: Section 4.1 introduces a general formulation of accuracy at the top. Section 4.2 derives formulas for the bias of the sampled threshold and proposes *DeepTopPush* to minimize it. Section 4.3 shows the good performance of *DeepTopPush* on multiple images recognition datasets, a real-world medical application, and a malware detection dataset, where we detected 46% malware at an extremely low false alarm rate of 10^{-5}. To promote reproducibility, our codes are available online.

4.1 Accuracy at the top

This section introduces the accuracy at the top. A standard deep network f with weights $\boldsymbol{w}$ takes inputs $\boldsymbol{x}_i$, transforms them into scores z_i, and computes the total loss based on these scores and labels y_i. On the other hand, accuracy at the top solves

$$\begin{aligned} \underset{w,z,t}{\text{minimize}} \quad & \lambda_1 \sum_{i \in I^-} \mathbb{1}_{z_i \geq t} + \lambda_2 \sum_{i \in I^+} \mathbb{1}_{z_i < t} \\ \text{subject to} \quad & z_i = f(\boldsymbol{w}; \boldsymbol{x}_i), \\ & t = G(\{(z_i, y_i)\}_{i \in I}). \end{aligned} \tag{4.1}$$

Similarly to the standard network, the classifier f computes the score z_i for each sample $\boldsymbol{x}_i$. Then a general function G takes the scores and labels of **all** samples and computes the threshold t. This makes the problem non-decomposable. The objective function equals the weighted sum of false-positives (negative samples above the threshold) and false-negatives (positive samples below the threshold). Here, I, I^+ and I^- are the sets of all, positive and negative labels, respectively, and $\mathbb{1}$ is the characteristic (0/1) function counting how many times the argument is satisfied. Setting (4.1) includes TopPush [14] which minimizes the number of positive samples below the highest-ranked negative sample. This fits into (4.1) with $\lambda_1 = 0$, $\lambda_2 = 1$ and $t = \max_{i \in I^-} z_i$.

Figure 4.2 shows the difference between the standard approach with cross-entropy and accuracy at the top. While classifier 1 has good total accuracy, its top accuracy is subpar because of the few negative outliers. On the other hand, classifier 2 has worse total accuracy, but its top accuracy is extremely good because more than half of the positive samples are on the top. While classifier 1 selected different thresholds for the accuracy and top metrics, these thresholds coincide for classifier 2.

Table 4.1 shows other special cases of (4.1) including maximizing precision at a given level of recall [22] or recall at K. The threshold t always equals to the sample with the j^*-th highest score on all, positive, or negative samples. The problems differ only in j^* and from which samples the threshold is computed. For example, Pat&Mat-NP [33] minimizes the false negative rate (equivalently maximizes the true positive rate) under the constraint that the false positive rate is at most τ.

4.1.1 Related works

There is a close connection between accuracy at the top and ranking problems [36, 37]. This was, together with similarities to the Neyman-Pearson problem, showed in [33]. A special case of the ranking problems attempts to rank positive samples above negative samples. Several approaches, such as RankBoost [11], Infinite Push [12] or p-norm push [13] employ a positive-negative pairwise comparison of scores, which can handle only small datasets. TopPush [14] converts the pairwise sum into a single sum and minimizes the

Name	λ_1	λ_2	t computed from	j^*
Prec@Rec	1	0	positive samples	$n^+\tau$
Rec@K	0	1	all samples	K
TopPush	0	1	negative samples	1
Pat&Mat-NP	0	1	negative samples	$n^-\tau$

Table 4.1: Selected problems of setting (4.1). False-positives and false-negatives have weights λ_1 and λ_2, the threshold t equals to the sample with the j^*-th highest score on all, positive, or negative samples.

false-negatives below a threshold given by the maximum score corresponding to negative samples. Thus, it converts ranking into accuracy at the top problems.

Two approaches for solving (4.1) exist. The first approach considers the threshold constraint as it is, while the second approach uses heuristics to approximate it. In the first approach, Acc@Top [15] argues that the threshold equals one of the scores. They fix the index of a sample and solve as many optimization problems as there are samples. [18, 33, 45] write the threshold as a constraint and replace both the objective and the constraint via surrogates. [18] uses Lagrange multipliers to obtain a minimax problem, [22] implicitly removes the threshold as an optimization variable and uses the chain rule to compute the gradient while [25] solves an SVM-like dual formulation with kernels. [3] uses the same formulation but applies surrogates only to the objective and recomputes the threshold after each gradient step. TFCO [46] solves a general class of constrained problems via a minimax reformulation. In the second approach, SoDeep [47] or SmoothI [48] use the fact that the threshold may be easily computed from sorted scores. They approximate the sorting operator by a network trained on artificial data. AP-Perf [49] considers a general metric and hedges against the worst-case perturbation of scores. The authors argue that the problem is bilinear in scores and use duality arguments. However, the bilinearity is lost when optimizing with respect to the weights of the original network.

4.2 DeepTopPush as a method for maximizing accuracy at the top

This section first shows a basic algorithm to solve (4.1). We then argue that the stochastic gradient descent produces a biased estimate of the true gradient, and we mention two strategies for mitigating this bias. Based on one strategy, we propose the *DeepTopPush* algorithm. The whole section assumes that the classifier f is differentiable.

4.2.1 Basic algorithm for solving accuracy at the top

Even though the presented technique can be applied to any formulation from Table 4.1, for simplicity, we derive it only for the TopPush formulation, where $\lambda_1 = 0$ and $\lambda_2 = 1$. This amounts to minimizing the false-negatives in (4.1). Since the function $\mathbb{1}$ in the formulation (4.1) is discontinuous, it is usually replaced by a general surrogate function l

which is continuous and non-decreasing. This leads to

$$\begin{aligned} \underset{w,z,t}{\text{minimize}} \quad & \frac{1}{n^+} \sum_{i \in I^+} l(t - z_i) \\ \text{subject to} \quad & z_i = f(\boldsymbol{w}; \boldsymbol{x}_i), \\ & t = G(\{(z_i, y_i)\}_{i \in I}). \end{aligned} \tag{4.2}$$

To apply the stochastic gradient descent, we need to compute the gradient. The core idea follows [22] which was proposed in a more general context in [26]. It rewrites problem (4.2) into its equivalent form

$$\underset{w}{\text{minimize}} \quad \frac{1}{n^+} \sum_{i \in I^+} l(G(\{(f(\boldsymbol{w}; \boldsymbol{x}_j), y_j)\}_{j \in I}) - f(\boldsymbol{w}; \boldsymbol{x}_i)). \tag{4.3}$$

This form removed the constraints and it has the advantage that the only optimization variable is $\boldsymbol{w}$ instead of $(\boldsymbol{w}, \boldsymbol{z}, t)$ in (4.2). In all cases from Table 4.1, the threshold t always equals to one of the scores, let it have index j^* and then $t = z_{j^*}$. Denoting the objective of (4.3) by $L(\boldsymbol{w})$, the chain rule implies that the gradient of the objective from (4.3) equals to

$$\nabla L(\boldsymbol{w}) = \frac{1}{n^+} \sum_{i \in I^+} l'(t - z_i) \Big(\nabla_w f(\boldsymbol{w}; \boldsymbol{x}_{j^*}) - \nabla_w f(\boldsymbol{w}; \boldsymbol{x}_i) \Big). \tag{4.4}$$

The stochastic gradient descent replaces the sum over all positive samples I^+ with a sum over all positive samples in a minibatch I^+_{mb}. However, as both the threshold t and the index j^* depend on all scores z_i, they need to be approximated on the minibatch as well. We denote these approximations by $\hat{t}$ and $\hat{j}$, respectively. Denoting the number of positive samples in the minibatch by n^+_{mb}, we replace the true gradient (4.4) by the *sampled gradient*

$$\nabla \hat{L} = \frac{1}{n^+_{\text{mb}}} \sum_{i \in I^+_{\text{mb}}} l'(\hat{t} - z_i) \Big(\nabla_w f(\boldsymbol{w}; \boldsymbol{x}_{\hat{j}}) - \nabla_w f(\boldsymbol{w}; \boldsymbol{x}_i) \Big), \tag{4.5}$$

The most straightforward way is to choose the sampled threshold $\hat{t}$ by the same rule as the true threshold t. As an example, if t is the 100^{th} largest score on the whole dataset and $\frac{n}{n_{\text{mb}}} = 20$ is the ratio of sizes of the whole dataset and of the minibatch, we select the sampled threshold $\hat{t}$ as the 5^{th} largest score on the minibatch. We summarize this procedure in Algorithm 3.

4.2.2 Bias of the sampled gradient

Convergence proofs of the stochastic gradient descent require that the sampled gradient is an unbiased estimate of the true gradient [44]. This means that

$$\text{bias}(\boldsymbol{w}) := \nabla L(\boldsymbol{w}) - \mathbb{E} \nabla \hat{L}(\boldsymbol{w}) \tag{4.6}$$

equals to 0 for all $\boldsymbol{w}$. A comparison of (4.4) and (4.5) shows that a necessary condition is that the sampled threshold $\hat{t}$ is an unbiased estimate of the true threshold t. However, the sampled version underestimates the true value, which is evident for the maximum where the sampled maximum is always smaller or equal to the true maximum. The next result quantifies the difference between the sampled and true thresholds.

Algorithm 3 Basic algorithm for solving (4.1)

Initialize weights $\boldsymbol{w}$
repeat
    Select minibatch I_{mb}
    Compute $z_i \leftarrow f(\boldsymbol{w}; \boldsymbol{x}_i)$ for $i \in I_{\text{mb}}$
    Set $\hat{t} \leftarrow G(\{(z_i, y_i)\}_{i \in I_{\text{mb}}})$
    Compute $\nabla \hat{L}$ based on I_{mb}
    Make a gradient step
until stopping criterion is satisfied

Algorithm 4 DeepTopPush as an efficient method for maximizing accuracy at the top.

Initialize weights $\boldsymbol{w}$, random index j^*
repeat
    Select minibatch I_{mb}
    Enhance minibatch $I_{\text{mb}}^{\text{enh}} = I_{\text{mb}} \cup \{j^*\}$
    Compute $z_i \leftarrow f(\boldsymbol{w}; \boldsymbol{x}_i)$ for $i \in I_{\text{mb}}^{\text{enh}}$
    Set $\hat{t} \leftarrow \{\max z_i \mid i \in I_{\text{mb}}^{\text{enh}} \cap I^-\}$
    Find index j^* such that $t = z_{j^*}$
    Compute $\nabla \hat{L}$ based on $I_{\text{mb}}^{\text{enh}} \cap I^+$
    Make a gradient step
until stopping criterion is satisfied

Proposition 4.1: [50]

Let X be an absolutely continuous random variable with distribution function F, let $X_1, \ldots, X_n$ be iid samples from X and let $\tau \in (0, 1)$. Denote the true threshold $t = F^{-1}(1 - \tau)$ and the sampled threshold $\hat{t} = X_{\lfloor\lceil n\tau\rceil\rfloor}$. If F is differentiable with a positive gradient at t, then

$$\sqrt{n}(t - \hat{t}) \to N\left(0, \frac{\tau(1-\tau)}{F'(t)^2}\right),$$

where the convergence is in distribution and N denotes the normal distribution.

This proposition states that when the minibatch size increases to infinity, the variance of the sampled threshold is approximately $\frac{\tau(1-\tau)}{nF'(t)^2}$. Figure 4.1 in the introduction shows this empirically for the case where the scores follow the standard normal distribution and $\tau = 0.01$ is the desired top fraction. The approximation is poor with both large bias and standard deviation. Even though this result gives us insight into the bias of the sampled threshold, we are ultimately interested in the bias of the sampled gradient $\nabla \hat{L}(\boldsymbol{w})$. To do so, recall that j^* is the threshold index on the whole dataset ($t = z_{j^*}$) while $\hat{j}$ is the threshold index on the minibatch ($\hat{t} = z_{\hat{j}}$). We split the computation based on whether these two indices are identical.

Lemma 4.2

Let j^* be unique. Assume that the selection of positive and negative samples into the minibatch is independent and that the threshold is computed from negative samples while the objective is computed from positive samples. Then the conditional expectation of the sampled gradient satisfies

$$\mathbb{E}\left(\nabla \hat{L}(\boldsymbol{w}) \mid \hat{j} = j^*\right) = \nabla L(\boldsymbol{w}).$$

Proof:
If j^* is unique, then the true threshold t is a differentiable function. The differentiability of L and $\hat{L}$ follows from the chain rule. If $\hat{j} = j^*$ holds, then the sampled gradient equals to

$$\nabla \hat{L}(\boldsymbol{w}) = \frac{1}{n^+_{\text{mb}}} \sum_{i \in I^+_{\text{mb}}} l'(t - z_i)\Big(\nabla_w f(\boldsymbol{w}; \boldsymbol{x}_{j^*}) - \nabla_w f(\boldsymbol{w}; \boldsymbol{x}_i)\Big). \tag{4.7}$$

The summands are identical to the ones in (4.4). Since the sum is performed with respect to positive samples, the threshold is computed from negative samples, the lemma statement follows. ■

Now we present the main result about the bias.

Theorem 4.3

Under the assumptions of Lemma 4.2, the bias of the sampled gradient from (4.6) satisfies

$$\text{bias}(\boldsymbol{w}) = \mathbb{P}(\hat{j} \neq j^*)\Big(\nabla L(\boldsymbol{w}) - \mathbb{E}\Big(\nabla \hat{L}(\boldsymbol{w}) \mid \hat{j} \neq j^*\Big)\Big). \tag{4.8}$$

Proof:
The law of total expectation implies

$$\begin{aligned}\mathbb{E}\nabla \hat{L}(\boldsymbol{w}) = \mathbb{P}(\hat{j} = j^*)\mathbb{E}(\nabla \hat{L}(\boldsymbol{w}) \mid \hat{j} = j^*) \\ + \mathbb{P}(\hat{j} \neq j^*)\mathbb{E}(\nabla \hat{L}(\boldsymbol{w}) \mid \hat{j} \neq j^*),\end{aligned}$$

from where the statement follows due to definiton (4.6) and Lemma 4.2. ■

The assumptions of Theorem 4.3 holds for all methods from Table 4.1 with the exception of Rec@K. For this method, the bias contains an additional term, as we show in the appendix.

The bias (4.8) consists of a multiplication of two terms. We propose two strategies for reducing the bias. The first strategy reduces both terms, while the second strategy reduces only the first term.

4.2.3 Bias reduction: Increasing minibatches size

The natural choice to mitigate the bias is to work with large minibatches. Even though this is not a standard way, some works suggest this route [51]. When the minibatch is large, it contains more samples and the chance that $\hat{j}$ differs from j^* decreases. This reduces the first term in (4.8). Moreover, Proposition 4.1 ensures that the difference between the sampled threshold $\hat{t}$ and the true threshold t is small. Then the difference between the true gradient (4.4) and the sampled gradient (4.5) decreases as well. This reduces the second term in (4.8). This approach is applicable to any method from Table 4.1.

4.2.4 Bias reduction: Incorporating delayed values

Various reasons may enforce the use of small minibatches. Then Algorithm 3 is not suitable for a small fraction of top samples. For example, a minibatch of size 32 with 16 negative samples must have thresholds $\tau \geq \frac{100}{16} = 6.25\%$. However, we need to aim for much smaller thresholds.

We propose a simple fix based on the reasoning that when the weights $\boldsymbol{w}$ of a neural network are updated, the scores $\boldsymbol{z}$ usually do not change much, especially for a small learning rate. This means that if a sample has the largest score, it will likely have the largest score even after the gradient step. Since the threshold t for TopPush equals the largest score corresponding to negative samples, we can easily track it. We enhance the current minibatch by the negative sample from the previous minibatch with the highest score. This significantly increases the chance that the sampled threshold is the true threshold and, due to the first term in (4.8), reduces the bias of the sampled gradient.

We summarize the procedure in Algorithm 4 and show it next to Algorithm 3 to highlight the differences. In every iteration, it stores the index j^* of the sample, which equals the threshold (step 7). We add it to the enhanced minibatch (step 4). Since we can track only the maximum, we set the threshold as the maximum of scores from negative samples (step 6) and minimize false-positives. Since Algorithm 4 uses the same formulation as *TopPush* [14] but can handle an arbitrary classifier, we name it *DeepTopPush*. We provide empirical evidence of why our technique works later in Section 4.3.5.

4.3 Numerical experiments

This section presents numerical results for *DeepTopPush*. Table 4.1 shows that it is similar to *Pat&Mat-NP*. While the former maximizes the number of positives above the largest negative, while the latter maximizes the number of positives above the $n^-\tau$-largest negative. The former may be understood as requiring no false-positives, while the latter allows for false positive rate τ.

Section 4.2.3 showed that we can use large minibatches to obtain good results for *Pat&Mat-NP* for small fractions of top samples τ. Section 4.2.4 showed that *DeepTopPush* works well even with small minibatches if we track the threshold by enhancing the minibatch by one sample. We present numerical comparisons in several sections, each with a different purpose. Comparison with the prior art *TFCO* and *Ap-Perf* is performed on several visual recognition datasets and shows that *DeepTopPush* outperforms other methods. Then we present two real-world applications. The first one shows that *DeepTopPush* can handle ranking problems. The second one presents results on a complex malware detection problem. Finally, we show similarities between *DeepTopPush* and *Pat&Mat-NP* and explain why enhancing the minibatch in Algorithm 4 works.

4.3.1 Dataset description and Computational setting

We consider the following image recognition datasets: FashionMNIST [41], CIFAR100 [52], SVHN2 [53] and ImageNet [54]. These datasets were converted to binary classification tasks by selecting one class as the positive class and the rest as the negative class. ImageNet merged turtles and non-turtles. We also consider the 3A4 dataset [55] with molecules and their activity levels. Finally, malware analysis reports of executable files were provided by a cybersecurity company. This is an extremely tough dataset as individual samples are JSON files whose size ranges from 1kB to 2.5MB. Moreover, they contain different features, and their features may have variable lengths. All datasets are summarized in the appendix.

We use truncated quadratic loss $l(z) = (\max\{0, 1 + z\})^2$ as the surrogate function and $\tau = \frac{1}{n^-}$ and $\tau = 0.01$. This first one computes the true positive rate above the second highest-ranked negative, while the latter allows for the false positive rate of 1%.

All algorithms were run for 200 epochs on an NVIDIA P100 GPU card with balanced minibatches of 32 samples. The only exception was Malware Detection, which was run on a cluster in a distributed manner, and where the minibatch size was 20000. For the evaluation of numerical experiments, we use the standard receiver operating characteristic (ROC) curve. All results are computed from the test set. All codes were implemented in the Julia language [39]. The network structure was the same for all methods; we describe them in the online appendix.

4.3.2 Comparison with prior art

We compare our methods with *BaseLine*, which uses the weighted cross-entropy. Moreover, we use two prior art methods which have codes available online, namely *TFCO* [46, 56] and *Ap-Perf* [49]. We did not implement the original TopPush because its duality arguments restrict the classifiers to only linear ones. Table 4.2 shows the time requirement per epoch. All methods besides *Ap-Perf* have similar time requirements, while *Ap-Perf* is much slower. This difference increases drastically when the minibatch size increases, as noted in [49]. We do not present the results for SVHN for *Ap-Perf* because it was too slow and for *TFCO* because we encountered a TensorFlow memory error. All these methods are designed to maximize true-positives when the false positive rate is at most τ. This is the same as for *Pat&Mat-NP*.

	FashionMNIST	CIFAR100	SVHN
BaseLine	4.4s	5.1s	62.8s
DeepTopPush	4.8s	5.6s	66.6s
Pat&Mat-NP	4.8s	5.6s	66.6s
TFCO	7.2s	6.5s	-
AP-Perf	95.3s	81.2s	-

Table 4.2: Time requirements per epoch for investigated methods for minibatches of size $n_{\text{mb}} = 32$.

Table 4.3 shows the true positive rate (tpr) above the second-largest negative and at the prescribed false positive rate (fpr) $\tau = 0.01$. Using the second-largest negative, which corresponds to $\tau = \frac{1}{n^-}$, allows for one outlier. The results are averaged over ten independent runs except for AP-Perf, which is too slow. The best result for each metric (in columns) is highlighted. All methods are better than *BaseLine*. This is not surprising as all these methods are designed to work well for low false positive rates. *DeepTopPush* outperforms all other methods at the top, while it performs well at the low fpr of $\tau = 0.01$. There *Pat&Mat-NP*, which also falls into our framework, performs well. Both these methods outperform the state of the art methods.

Figure 4.4 **A)** shows the ROC curves on CIFAR100 averaged over ten independent runs. We use the logarithmic x axis to highlight low fpr modes. *DeepTopPush* performs significantly the best again whenever the false positive rate is smaller than 0.01.

As a further test, we performed a simple experiment on ImageNet. We modified the pre-trained EfficientNet B0 [57] by removing the last dense layer and adding another dense

	Dataset	BaseLine	DeepTopPush	Pat&Mat-NP	TFCO	AP-Perf
tpr@fpr $\tau = 1/n^-$	FashionMNIST	5.06 ± 1.41	27.30 ± 5.91	22.21 ± 5.62	11.30 ± 3.44	9.90
	CIFAR100	1.70 ± 0.46	14.40 ± 5.44	8.10 ± 3.45	7.70 ± 2.28	5.00
	3A4	2.58 ± 0.61	5.61 ± 1.70	3.79 ± 0.90	3.03 ± 1.52	3.03
	SVHN	6.51 ± 1.37	12.21 ± 5.39	12.07 ± 4.41	-	-
tpr@fpr $\tau = 0.01$	FashionMNIST	63.14 ± 1.39	75.37 ± 1.18	74.11 ± 1.00	73.27 ± 2.92	64.60
	CIFAR100	49.40 ± 4.90	70.20 ± 2.14	66.30 ± 2.33	67.30 ± 1.79	65.00
	3A4	57.80 ± 0.35	60.08 ± 3.35	65.91 ± 0.59	54.55 ± 10.22	63.64
	SVHN	84.72 ± 0.84	91.05 ± 1.45	91.07 ± 0.30	-	-

Table 4.3: The true positive rates (in %) at two levels of false positive rates averaged across ten indepenedent runs with standard deviation. The best methods are highlighted.

layer with one output. Then we retrained the newly added layer to perform well at the top. The original EfficientNet achieved 68.0% at the top, while *DeepTopPush* achieved 70.0% for the same metric. This shows that *DeepTopPush* can provide better accuracy at the top than pre-trained networks.

4.3.3 Application to ranking

The 3A4 dataset contains information about activity levels of approximately 50000 molecules, each with about 10000 descriptors. The activity level corresponds to the usefulness of the molecule for creating new drugs. Since medical scientists can focus on properly investigating only a small number of molecules, it is important to select a small number of molecules with high activity.

We converted the continuous activity level into binary by considering a threshold on the activity. Since the input is large-dimensional, and there is no spatial structure to use convolutional neural networks, we used PCA to reduce the dimension to 100. Then we created a network with two hidden layers and applied *DeepTopPush* to it. The test activity was evaluated at the continuous (and not binary level). Table 4.3 shows again the results at the top. *DeepTopPush* outperforms other methods. Figure 4.3 shows that high scores (output of the network) indeed correspond to high activity. Thus, even though the problem was "binarized" and its dimension reduced, our algorithm was able to select a small number of molecules with high activity levels. These molecules can be used for further manual (expensive) investigation.

4.3.4 Real-world application

This section shows a real-world application of the accuracy at the top. A renowned cybersecurity company provided malware analysis reports of executable files. Its structure is highly complicated because each sample has a different number of features, and features may have a complicated structure, such as a list of ports to which the file connects. This is in sharp contrast with standard datasets, where each sample has the same number of features, and each feature is a real number. We processed the data by a public implementation of hierarchical multi-instance learning (HMIL) [58]. Then we applied *DeepTopPush*

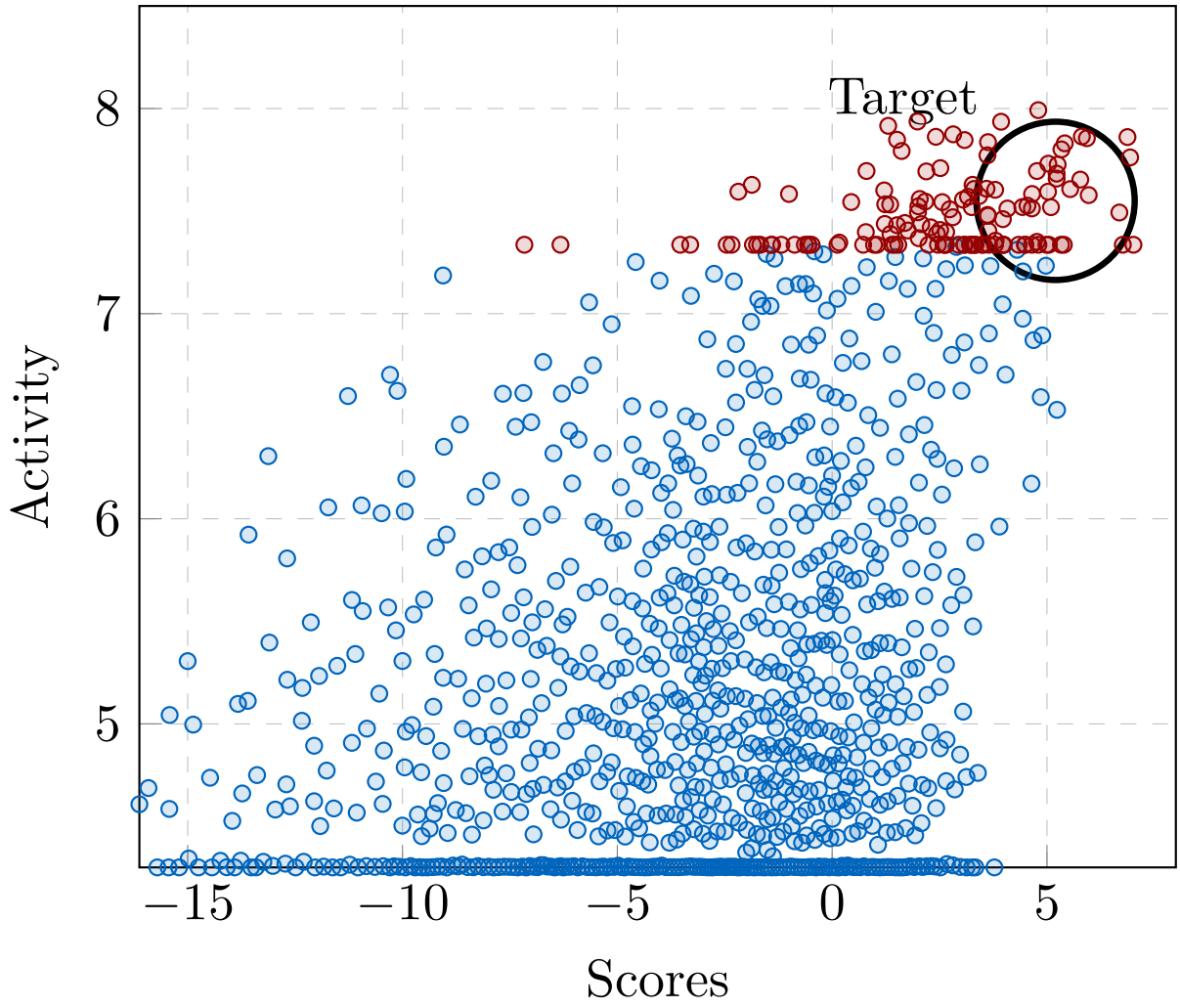


Figure 4.3: Results for the 3A4 dataset. The goal was to assign large scores to a few molecules with high activity (scores on top-right are preferred).

and *Pat&Mat-NP* at $\tau = 10^{-3}$ and $\tau = 10^{-2}$. The latter maximizes the true positives rate when the false positive rate is at most τ. The minibatch size was 20000, which allowed us to obtain precise threshold estimates and unbiased sampled gradients due to Section 4.2.3.

Figure 4.4 **B)** shows the performance on the test set. *DeepTopPush* is again the best at low false positive rates. This is extremely important in cybersecurity as it prevents false alarms for malware. Even at the extremely low false positive rate $\tau = 10^{-5}$, our algorithm correctly identified 46% of malware. The circles denote the thresholds for which the methods were optimized. *DeepTopPush* should have the best performance at the leftmost point, *Pat&Mat-NP* ($\tau = 10^{-3}$) at $\tau = 10^{-3}$ and similarly *Pat&Mat-NP* ($\tau = 10^{-2}$).

4.3.5 Impact of enhancing the minibatch

The crucial aspect of *DeepTopPush* is enhancing the minibatch by one sample. In all presented results with the exception of the Malware Detection, the minibatch contained only 32 samples. Then the discussion in Section 4.2.4 implies that *Pat&Mat-NP* equals

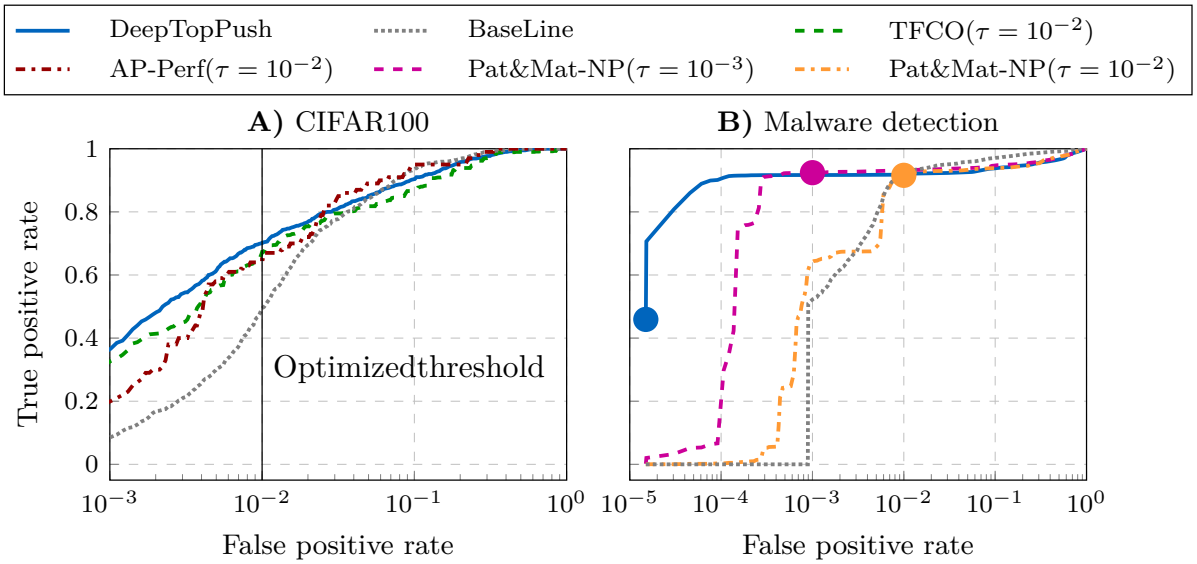


Figure 4.4: **A)** ROC curves averaged over ten runs on the CIFAR100 dataset. **B)** ROC curve for Malware Detection dataset. The circles show the thresholds the methods were optimized for.

to *DeepTopPush* without enhancing the minibatch. In other words, *Pat&Mat-NP* uses Algorithm 3 while *DeepTopPush* uses Algorithm 4. As Table 4.3 clearly shows that *DeepTopPush* ourperforms *Pat&Mat-NP*, this implies that using the delayed values is beneficial.

Figure 4.5 shows explanation for this behaviour. The full blue line shows the behaviour of *DeepTopPush* while the dotted grey line shows *Pat&Mat-NP*. As explained in the previous paragraph, their difference demonstrates the effect of enhancing the minibatch by one delayed value. The top subfigure compares thresholds with the true threshold (dashed black). While the threshold for *Pat&Mat-NP* jumps wildly, it is smooth for *DeepTopPush*, and it often equals the true threshold. Theorem 4.3 then implies that our sampled gradient is an unbiased estimate of the true gradient. This is even more pronounced in the bottom subfigure, which shows the angle between the true gradient and the computed gradient. This angle is important because [59] showed that if this angle is uniformly in the interval $[0, 90)$, then gradient descent schemes converge. This is precisely what happened for *DeepTopPush*. When the threshold is correct, the true and estimated gradients are parallel to each other, and the gradient descent moves in the correct direction.

4.4 Conclusions

We proposed *DeepTopPush* as an efficient method for solving the constrained non-decomposable problem of accuracy at the top, which focuses on the performance only above a threshold. We implicitly removed some optimization variables, created an unconstrained end-to-end network and used the stochastic gradient descent to train it. We modified the minibatch so that the sampled threshold (computed on a minibatch) is a good estimate of the true threshold (computed on all samples). We showed both theoretically and numerically that this procedure reduces the bias of the sampled gradient. The time increase over the standard method with no threshold is small. We demonstrated the usefulness of *DeepTopPush*

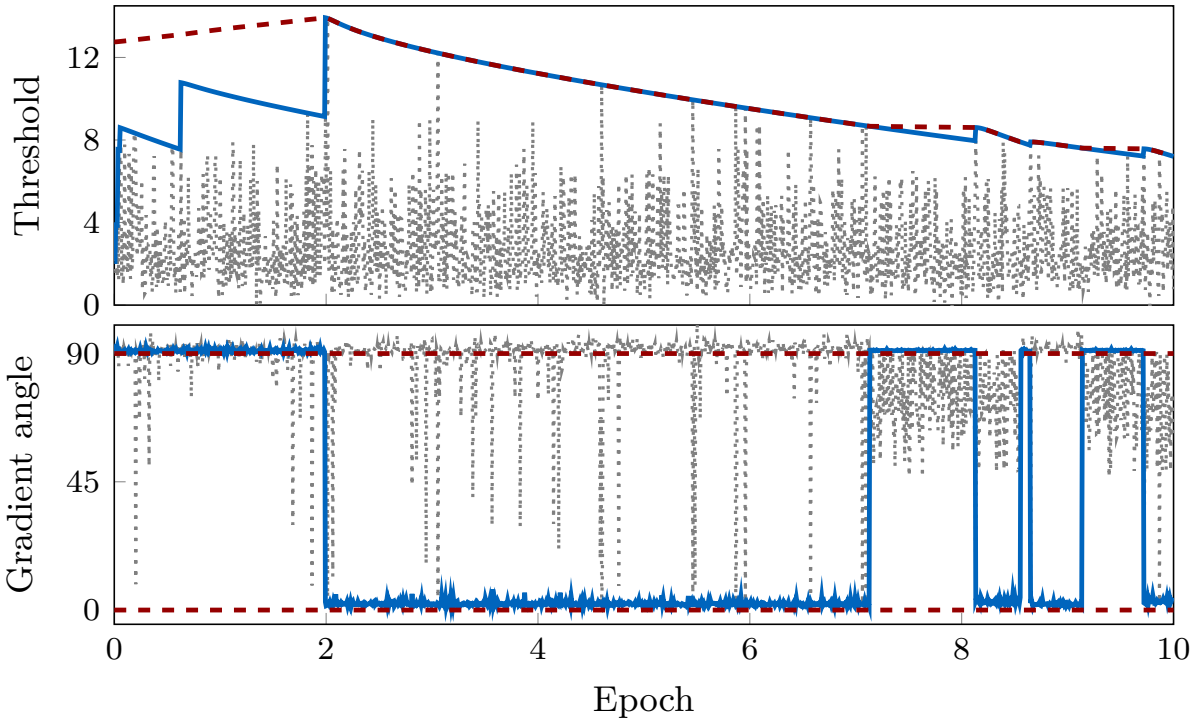


Figure 4.5: The thresholds (top) and angle between true and sampled gradients (bottom) for Algorithm 3 (full blue) and Algorithm 4 (dotted gray).

both on visual recognition datasets, a ranking problem and on a real-world application of malware detection.

Apendices

A

Linear case

A.1 Additional results and proofs

Here, we provide additional results and proofs of results mentioned in the main body. For convenience, we repeat the result statements.

A.1.1 Equivalence of (2.10) and (2.11)

To show this equivalence, we will start with an auxiliary lemma.

Lemma A.1

Denote by t the exact quantile from (2.9). Then for all $\mu \in [0, 1]$ we have

$$\mathrm{fp}(\boldsymbol{w}, t) = \mu\,\mathrm{fp}(\boldsymbol{w}, t) + (1-\mu)\,\mathrm{fn}(\boldsymbol{w}, t) + (1-\mu)(n\tau - n^+) + (1-\mu)(q-1), \quad \text{(A.1)}$$

where $q := \#\{\boldsymbol{x} \in \mathcal{X} |\ \boldsymbol{w}^\top \boldsymbol{x} = t\}$.

Proof:
By the definition of the quantile we have

$$\mathrm{tp}(\boldsymbol{w}, t) + \mathrm{fp}(\boldsymbol{w}, t) = n\tau + q - 1.$$

This implies

$$\mathrm{fp}(\boldsymbol{w}, t) = n\tau + q - 1 - \mathrm{tp}(\boldsymbol{w}, t) = n\tau + q - 1 - n^+ + \mathrm{fn}(\boldsymbol{w}, t).$$

From this relation we deduce

$$\begin{aligned}\mathrm{fp}(\boldsymbol{w}, t) &= \mu\,\mathrm{fp}(\boldsymbol{w}, t) + (1-\mu)\,\mathrm{fp}(\boldsymbol{w}, t) = \mu\,\mathrm{fp}(\boldsymbol{w}, t) + (1-\mu)\Big(\mathrm{fn}(\boldsymbol{w}, t) + n\tau - n^+ + q - 1\Big) \\ &= \mu\,\mathrm{fp}(\boldsymbol{w}, t) + (1-\mu)\,\mathrm{fn}(\boldsymbol{w}, t) + (1-\mu)\Big(n\tau - n^+\Big) + (1-\mu)(q-1),\end{aligned}$$

which is precisely the lemma statement. ■

The right-hand side of (A.1) consists of three parts. The first one is a convex combination of false-positives and false-negatives. The second one is a constant term which has no impact on optimization. Finally, the third term $(1-\mu)(q-1)$ equals the number of samples for which their classifier equals the quantile. However, this term is small in comparison with the true-positives and the false-negatives and can be neglected. Moreover, when the data are "truly" random such as when measurement errors are present, then $q = 1$ and this term vanishes completely. This gives the (almost) equivalence of (2.10) and (2.11). Note that term q is ignored in many papers.

A.1.2 Results related to convexity

Proposition 2.2

Thresholds t_2 and t_3 are convex functions of the weights $\boldsymbol{w}$. The threshold function t_1 is non-convex.

Proof Proposition 2.2 on page 13:
It is easy to show that the quantile t_1 is not convex. Due to [20], the mean of the k highest values of a vector is a convex function and therefore, t_2 is a convex function. It remains to analyze t_3. It is defined via an implicit equation, where we consider for simplicity $\beta = 1$,

$$g(\boldsymbol{w}, t) := \frac{1}{n} \sum_{\boldsymbol{x} \in \mathcal{X}} l(\boldsymbol{w}^\top \boldsymbol{x} - t) - \tau = 0.$$

Since l is convex, we immediately obtain that g is jointly convex in both variables.

To show the convexity, consider $\boldsymbol{w}$, $\hat{\boldsymbol{w}}$ and the corresponding $t = t_3(\boldsymbol{w})$, $\hat{t} = t_3(\hat{\boldsymbol{w}})$. Note that this implies $g(\boldsymbol{w}, t) = g(\hat{\boldsymbol{w}}, \hat{t}) = 0$. Then for any $\lambda \in [0, 1]$ we have

$$g(\lambda \boldsymbol{w} + (1-\lambda)\hat{\boldsymbol{w}},\ \lambda t + (1-\lambda)\hat{t}) \leq \lambda g(\boldsymbol{w}, t) + (1-\lambda) g(\hat{\boldsymbol{w}}, \hat{t}) = 0, \tag{A.2}$$

where the inequality follows from the convexity of g and the equality from $g(\boldsymbol{w}, t) = g(\hat{\boldsymbol{w}}, \hat{t}) = 0$. From the definition of the surrogate quantile function t_3 we have

$$g(\lambda \boldsymbol{w} + (1-\lambda)\hat{\boldsymbol{w}}, t_3(\lambda \boldsymbol{w} + (1-\lambda)\hat{\boldsymbol{w}})) = 0. \tag{A.3}$$

Since g is non-increasing in the second variable, from (A.2) and (A.3) we deduce

$$t_3(\lambda \boldsymbol{w} + (1-\lambda)\hat{\boldsymbol{w}}) \leq \lambda t + (1-\lambda)\hat{t} = \lambda t_3(\boldsymbol{w}) + (1-\lambda) t_3(\hat{\boldsymbol{w}}),$$

which implies that function $\boldsymbol{w} \mapsto t_3(\boldsymbol{w})$ is convex. ■

Theorem 2.3

If the threshold t is a convex function of the weights $\boldsymbol{w}$, then function $f(\boldsymbol{w}) = \overline{\text{fn}}(\boldsymbol{w}, t(\boldsymbol{w}))$ is convex.

Proof of Theorem 2.3 on page 13:
Due to the definition of the surrogate counts (2.4), the objective of (2.5) equals to

$$\frac{1}{n^+} \sum_{\boldsymbol{x} \in \mathcal{X}^+} l\Big(t(\boldsymbol{w}) - \boldsymbol{w}^\top \boldsymbol{x}\Big).$$

Here we write $t(\boldsymbol{w})$ to stress the dependence of t on $\boldsymbol{w}$. Since $\boldsymbol{w} \mapsto t(\boldsymbol{w})$ is a convex function, we also have that $\boldsymbol{w} \mapsto t(\boldsymbol{w}) - \boldsymbol{w}^\top \boldsymbol{x}$ is a convex function. From its definition, the surrogate function l is convex and non-decreasing. Since a composition of a convex function with a non-decreasing convex function is a convex function, this finishes the proof. ■

A.1.3 Results related to differentiability

Theorem 2.4

If the surrogate function l is differentiable, then threshold t_3 is a differentiable function of the weights $\boldsymbol{w}$ and its derivative equals to

$$\nabla t_3(\boldsymbol{w}) = \frac{\sum_{\boldsymbol{x}\in\mathcal{X}} l'(\beta(\boldsymbol{w}^\top\boldsymbol{x} - t_3(\boldsymbol{w})))\boldsymbol{x}}{\sum_{\boldsymbol{x}\in\mathcal{X}} l'(\beta(\boldsymbol{w}^\top\boldsymbol{x} - t_3(\boldsymbol{w})))}.$$

The threshold functions t_1 and t_2 are non-differentiable.

Proof of Theorem 2.4 on page 13:
The result for t_3 follows directly from the implicit function theorem. The non-differentiability of t_1 and t_2 happens whenever the threshold value is achieved at two different scores. ■

A.1.4 Results related to stability

Theorem 2.5

Consider any of these formulations: *TopPush*, *TopPushK*, *TopMeanK* or τ-*FPL*. Fix any $\boldsymbol{w}$ and denote the corresponding threshold $t(\boldsymbol{w})$. If we have

$$t(\boldsymbol{w}) \geq \frac{1}{n^+}\sum_{\boldsymbol{x}^+\in\mathcal{X}^+} \boldsymbol{w}^\top\boldsymbol{x}^+, \tag{2.24}$$

then $f(\mathbf{0}) \leq f(\boldsymbol{w})$. Specifically, denote the scores $s^+ = \boldsymbol{w}^\top\boldsymbol{x}^+$ for $\boldsymbol{x}^+ \in \mathcal{X}^+$ and $s^- = \boldsymbol{w}^\top\boldsymbol{x}^-$ for $\boldsymbol{x}^- \in \mathcal{X}^-$ and the ordered variants with decreasing components of $\boldsymbol{s}^-$ by $\boldsymbol{s}^-_{[\cdot]}$. Then

$$\begin{aligned}
s^-_{[1]} \geq \frac{1}{n^+}\sum_{i=1}^{n^+} s^+_i \implies f(\mathbf{0}) \leq f(\boldsymbol{w}) \text{ for } \textit{TopPush},\\
\frac{1}{k}\sum_{i=1}^{k} s^-_{[i]} \geq \frac{1}{n^+}\sum_{i=1}^{n^+} s^+_i \implies f(\mathbf{0}) \leq f(\boldsymbol{w}) \text{ for } \textit{TopPushK},\\
\frac{1}{n^-\tau}\sum_{i=1}^{n^-\tau} s^-_{[i]} \geq \frac{1}{n^+}\sum_{i=1}^{n^+} s^+_i \implies f(\mathbf{0}) \leq f(\boldsymbol{w}) \text{ for } \tau\textit{-FPL}.
\end{aligned} \tag{2.25}$$

Proof of Theorem 2.5 on page 15:
Due to $l(0) = 1$ and the convexity of l we have $l(s) \geq 1 + cs$, where c equals to the derivative of l at 0. Then we have

$$\begin{aligned}
f(\boldsymbol{w}) \geq \frac{1}{n^+}\overline{\mathrm{fn}}(\boldsymbol{w}, t) = \frac{1}{n^+}\sum_{\boldsymbol{x}\in\mathcal{X}^+} l(t - \boldsymbol{w}^\top\boldsymbol{x}) \geq \frac{1}{n^+}\sum_{\boldsymbol{x}\in\mathcal{X}^+}(1 + c(t - \boldsymbol{w}^\top\boldsymbol{x}))\\
= 1 + \frac{c}{n^+}\sum_{\boldsymbol{x}\in\mathcal{X}^+}(t - \boldsymbol{w}^\top\boldsymbol{x}) = 1 + ct - \frac{c}{n^+}\sum_{\boldsymbol{x}\in\mathcal{X}^+}\boldsymbol{w}^\top\boldsymbol{x} \geq 1,
\end{aligned}$$

where the last inequality follows from (2.24). Now we realize that for any formulation from the statement, the corresponding threshold for $\boldsymbol{w} = 0$ equals to $t = 0$, and thus $f(\mathbf{0}) = 1$. But then $f(\mathbf{0}) \leq f(\boldsymbol{w})$. The second part of the result follows from the form of thresholds $t(\boldsymbol{w})$. ■

Theorem 2.8

Consider the *Pat&Mat* or *Pat&Mat-NP* formulation with the hinge surrogate and no regularization. Assume that for some $\boldsymbol{w}$ we have

$$\frac{1}{n^+}\sum_{\boldsymbol{x}^+\in\mathcal{X}^+}\boldsymbol{w}^\top\boldsymbol{x}^+ > \frac{1}{n^-}\sum_{\boldsymbol{x}^-\in\mathcal{X}^-}\boldsymbol{w}^\top\boldsymbol{x}^-. \tag{2.26}$$

Then there is a scaling parameter β_0 from (2.13) such that $f(\boldsymbol{w}) < f(\boldsymbol{0})$ for all $\beta \in (0, \beta_0)$.

Proof of Theorem 2.8 on page 16:

Define first

$$s_{\min} = \min_{\boldsymbol{x}\in\mathcal{X}}\boldsymbol{w}^\top\boldsymbol{x}, \quad \bar{s} = \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{X}}\boldsymbol{w}^\top\boldsymbol{x}, \quad s_{\max} = \max_{\boldsymbol{x}\in\mathcal{X}}\boldsymbol{w}^\top\boldsymbol{x}.$$

Then we have the following chain of relations

$$\begin{aligned}\bar{s} &= \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{X}}\boldsymbol{w}^\top\boldsymbol{x} = \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{X}^+}\boldsymbol{w}^\top\boldsymbol{x} + \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{X}^-}\boldsymbol{w}^\top\boldsymbol{x} < \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{X}^+}\boldsymbol{w}^\top\boldsymbol{x} + \frac{n^-}{nn^+}\sum_{\boldsymbol{x}\in\mathcal{X}^+}\boldsymbol{w}^\top\boldsymbol{x} \\ &= \frac{1}{n}\left(1+\frac{n^-}{n^+}\right)\sum_{\boldsymbol{x}\in\mathcal{X}^+}\boldsymbol{w}^\top\boldsymbol{x} = \frac{1}{n}\frac{n^+ + n^-}{n^+}\sum_{\boldsymbol{x}\in\mathcal{X}^+}\boldsymbol{w}^\top\boldsymbol{x} = \frac{1}{n^+}\sum_{\boldsymbol{x}\in\mathcal{X}^+}\boldsymbol{w}^\top\boldsymbol{x}.\end{aligned} \tag{A.4}$$

The only inequality follows from (2.26) and the last equality follows from $n^+ + n^- = n$.

Due to (2.26) we have $s_{\min} < \bar{s} < s_{\max}$. Then we can define

$$\beta_0 = \min\left\{\frac{\tau}{\bar{s} - s_{\min}}, \frac{1-\tau}{s_{\max} - \bar{s}}, \tau\right\},$$

observe that $\beta_0 > 0$, fix any $\beta \in (0, \beta_0)$ and define

$$t = \frac{1-\tau}{\beta_0} + \bar{s}.$$

Then we obtain

$$1 + \beta(\boldsymbol{w}^\top\boldsymbol{x} - t) \geq 1 + \beta(s_{\min} - t) = 1 + \beta s_{\min} - 1 + \tau - \beta\bar{s} = \beta(s_{\min} - \bar{s}) + \tau \geq 0. \tag{A.5}$$

Here, the first equality follows from the definition of t and the last inequality from the definition of β_0. Moreover, we have

$$\begin{aligned}\frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{X}} l(\beta(\boldsymbol{w}^\top\boldsymbol{x} - t)) &= \frac{1s}{n}\sum_{\boldsymbol{x}\in\mathcal{X}}\max\{1 + \beta(\boldsymbol{w}^\top\boldsymbol{x} - t), 0\} = \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{X}}\Big(1 + \beta(\boldsymbol{w}^\top\boldsymbol{x} - t)\Big) \\ &= 1 - \beta t + \frac{\beta}{n}\sum_{\boldsymbol{x}\in\mathcal{X}}\boldsymbol{w}^\top\boldsymbol{x} = 1 - \beta t + \beta\bar{s} = \tau,\end{aligned}$$

where the second equality employs (A.5), the third one the definition of $\bar{s}$ and the last one the definition of t. But this means that t is the threshold corresponding to $\boldsymbol{w}$.

Similarly to (A.5) we get

$$1 + t - \boldsymbol{w}^\top\boldsymbol{x} \geq 1 + t - s_{\max} = 1 + \frac{1-\tau}{\beta} + \bar{s} - s_{\max} \geq \frac{1-\tau}{\beta} + \bar{s} - s_{\max} \geq 0, \tag{A.6}$$

where the last inequality follows from the definition of β_0. Then for the objective we have

$$\begin{aligned} f(\boldsymbol{w}) &= \frac{1}{n^+}\sum_{\boldsymbol{x}\in\mathcal{X}^+} l(t-\boldsymbol{w}^\top\boldsymbol{x}) = \frac{1}{n^+}\sum_{\boldsymbol{x}\in\mathcal{X}^+}\max\{1+t-\boldsymbol{w}^\top\boldsymbol{x},0\} = \\ &= \frac{1}{n^+}\sum_{\boldsymbol{x}\in\mathcal{X}^+}\left(1+t-\boldsymbol{w}^\top\boldsymbol{x}\right) = 1+t-\frac{1}{n^+}\sum_{\boldsymbol{x}\in\mathcal{X}^+}\boldsymbol{w}^\top\boldsymbol{x} < 1+t-\bar{s} \\ &= 1+\frac{1-\tau}{\beta}+\bar{s}-\bar{s} = 1+\frac{1-\tau}{\beta} = f(\boldsymbol{0}), \end{aligned}$$

where we the third equality follows from (A.6), the only inequality from (A.4) and the last equality from Appendix A.2. Thus, we finished the proof for *Pat&Mat*. The proof for *Pat&Mat-NP* can be performed in an identical way by replacing in the definition of $\bar{s}$ the mean with respect to all samples by the mean with respect to all negative samples. ■

A.1.5 Results related to threshold comparison

Lemma A.7

Define vector $\boldsymbol{s}^+$ with components $s^+ = \boldsymbol{w}^\top\boldsymbol{x}^+$ for $\boldsymbol{x}^+ \in \mathcal{X}^+$ and similarly define vector $\boldsymbol{s}^-$ with components $s^- = \boldsymbol{w}^\top\boldsymbol{x}^-$ for $\boldsymbol{x}^- \in \mathcal{X}^-$. Denote by $\boldsymbol{s}^+_{[\cdot]}$ and $\boldsymbol{s}^-_{[\cdot]}$ the sorted versions of $\boldsymbol{s}^+$ and $\boldsymbol{s}^-$, respectively. Then we have the following statements:

$$s^+_{[n^+\tau]} > s^-_{[n^-\tau]} \implies \textit{Grill} \text{ has larger threshold than } \textit{Grill-NP},$$

$$\frac{1}{n^+\tau}\sum_{i=1}^{n^+\tau} s^+_{[i]} > \frac{1}{n^-\tau}\sum_{i=1}^{n^-\tau} s^-_{[i]} \implies \textit{TopMeanK} \text{ has larger threshold than } \tau\textit{-FPL}.$$

Proof:
Since $\boldsymbol{s}^+$ and $\boldsymbol{s}^-$ are computed on disjunctive indices, we have

$$s_{[n\tau]} \geq \min\{s^+_{[n^+\tau]}, s^-_{[n^-\tau]}\}.$$

Since $s_{[n\tau]}$ is the threshold for *Grill* and $s^-_{[n^-\tau]}$ is the threshold for *Grill-NP*, the first statement follows. The second part can be shown in a similar way. ■
Since the goal of the presented formulations is to push s^+ above s^-, we may expect that the conditions in Lemma A.7 hold true.

A.2 Computation for Section 2.2.4

We derive the results presented in Section 2.2.4 more properly. We recall that we have n negative samples randomly distributed in $[-1,0]\times[-1,1]$, n positive samples randomly distributed in $[0,1]\times[-1,1]$ and one negative sample at $(2,0)$. We assume that n is large and the outlier may be ignored for the computation of thresholds which require a large number of points. Since the computation is simple for other formulations, we show it only for *Pat&Mat*.

For $\boldsymbol{w}_0 = (0,0)$, we get

$$\tau = \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{X}} l(\beta(\boldsymbol{w}_0^\top\boldsymbol{x}-t)) = l(0-\beta t) = 1-\beta t,$$

which implies $t = \frac{1}{\beta}(1-\tau)$ and consequently

$$\frac{1}{n^+}\overline{\mathrm{fn}}(\boldsymbol{w}_0, t) = \frac{1}{n^+}\sum_{\boldsymbol{x}\in\mathcal{X}^+} l(t-0) = l(t) = 1+t.$$

This finishes the computation for $\boldsymbol{w}_0$.

For $\boldsymbol{w}_1 = (1,0)$ the computation goes similar. Then $\boldsymbol{w}_1^\top \boldsymbol{x}^+$ has the uniform distribution on $[0,1]$ while $\boldsymbol{w}_1^\top \boldsymbol{x}$ has the uniform distribution on $[-1,1]$. If $\beta \le \tau$, then

$$\begin{aligned}\tau &= \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{X}} l(\boldsymbol{w}_1^\top\boldsymbol{x} - t) \approx \frac{1}{2}\int_{-1}^{1} l(s-t)\,\mathrm{d}s = \frac{1}{2}\int_{-1}^{1}\max\{0, 1+\beta(s-t)\}\,\mathrm{d}s \\ &= \frac{1}{2}\int_{-1}^{1}(1+\beta(s-t))\,\mathrm{d}s = 1-\beta t + \frac{\beta}{2}\int_{-1}^{1} s\,\mathrm{d}s = 1-\beta t,\end{aligned} \tag{A.7}$$

and thus again $t = \frac{1}{\beta}(1-\tau)$. Note that

$$1+\beta(s-t) \ge 1+\beta(-1-t) = 1-\beta-1+\tau = -\beta+\tau \ge 0$$

and we could have ignored the max operator in (A.7). Finally, we have

$$\frac{1}{n^+}\overline{\mathrm{fn}}(\boldsymbol{w}_1, t) \approx \int_0^1 l(t-s)\,\mathrm{d}s = \int_0^1 (1+t-s)\,\mathrm{d}s = 0.5+t.$$

A.3 Computing the threshold for *Pat&Mat*

We show how to efficiently compute the threshold (2.13) for *Pat&Mat* and the hinge surrogate (2.3). As always define the scores $s_i = \boldsymbol{w}^\top \boldsymbol{x}_i$ and consider function

$$h(t) = \sum_{i=1}^{n} l(\beta(s_i - t)) - n\tau. \tag{A.8}$$

Then solving (2.31) is equivalent to looking for $\hat{t}$ such that $h(\hat{t}) = 0$. We have the following properties of h.

Lemma A.8

Function h is continuous and strictly decreasing (until it hits the global minimum) with $h(t) \to \infty$ as $t \to -\infty$ and $h(t) \to -n\tau$ as $t \to \infty$. Thus, there is a unique solution to the equation $h(t) = 0$.

For sorted data, the following lemma gives advice on how to solve equation $h(t) = 0$.

Lemma A.9

Let $s_1 \le s_2 \le \cdots \le s_n$ be sorted. Define $\gamma = \frac{1}{\beta}$. Then

$$h(s_i + \gamma) = h(s_{i+1} + \gamma) + (n-i)\beta(s_{i+1} - s_i) \tag{A.9}$$

for all $i = n-1, \ldots, 1$ with the initial condition $h(s_n + \gamma) = -n\tau$.

Proof:
Observe first that

$$h(s_j+\gamma) = \sum_{i=1}^{n} l(\beta(s_i - s_j - \gamma)) - n\tau = \sum_{i=1}^{n} \max(0, \beta(s_i - s_j)) - n\tau = \sum_{i=j+1}^{n} \beta(s_i - s_j) - n\tau.$$

From here, we obtain $h(s_n + \gamma) = -n\tau$. Moreover, we have

$$\begin{aligned} h(s_j+\gamma) &= \sum_{i=j+1}^{n} \beta(s_i - s_j) - n\tau = \sum_{i=j+2}^{n} \beta(s_i - s_j) + \beta(s_{j+1} - s_j) - n\tau \\ &= \sum_{i=j+2}^{n} \beta(s_i - s_{j+1}) + \sum_{i=j+2}^{n} \beta(s_{j+1} - s_j) + \beta(s_{j+1} - s_j) - n\tau \\ &= \sum_{i=j+2}^{n} \beta(s_i - s_{j+1}) + (n-j)\beta(s_{j+1} - s_j) - n\tau \\ &= h(s_{j+1}+\gamma) + (n-j)\beta(s_{j+1} - s_j), \end{aligned}$$

which finishes the proof. ■

Thus, to solve $h(t) = 0$ with the hinge surrogate, we start with $t_n = s_n + \gamma$ and $h(t_n) = -n\tau$. Then we start decreasing t according to (A.9) until we find some $t_i = s_i + \gamma$ such that $h(t_i) > 0$. The desired t then lies between t_i and t_{i+1}. Since h is a piecewise linear function with

$$h(t) = h(t_i) + \frac{t - t_i}{t_{i+1} - t_i}(h(t_{i+1}) - h(t_i))$$

for $t \in [t_i, t_{i+1}]$, the precise value of $\hat{t}$ can be computed by a simple interpolation

$$\hat{t} = t_i - h(t_i)\frac{t_{i+1} - t_i}{h(t_{i+1}) - h(t_i)} = t_i - h(t_i)\frac{t_{i+1} - t_i}{-(n-i)\beta(t_{i+1} - t_i)} = t_i + \frac{h(t_i)}{\beta(n-i)}.$$

A.4 Proof of Theorem 2.9

The proof is divided into three parts. In Section A.4.1, we prove a general statement for convergence of stochastic gradient descent with a convex objective. In Section A.4.2 we apply it to Theorem 2.9. The proof is based on auxiliary results from Section A.4.3.

A.4.1 General result

Consider a differentiable objective function f and the optimization method

$$w^{k+1} = w^k - \alpha^k g(w^k), \tag{A.10}$$

where $\alpha^k > 0$ is a stepsize and $g(w^k)$ is an approximation of the gradient $\nabla f(w^k)$. Assume the following:

(A1) f is differentiable, convex and attains a global minimum;

(A2) $\left\|g(w^k)\right\| \leq B$ for all k;

(A3) the stepsize is non-increasing and satisfies $\sum_{k=0}^{\infty} \alpha^k = \infty$;

(A4) the stepsize satisfies $\sum_{k=0}^{\infty} (\alpha^k)^2 < \infty$;

(A5) the stepsize satisfies $\sum_{k=0}^{\infty}\left\|\alpha^{k+1}-\alpha^{k}\right\|<\infty$.

Assumptions (A3)-(A5) are satisfied for example for $\alpha^k = \alpha^0\frac{1}{k+1}$. We start with the general result.

Theorem A.10

Assume that (A1)-(A4) is satisfied. If there exists some C such that for some global minimum of w^* of f we have

$$\sum_{k=0}^{\infty}\alpha^k\langle g(w^k)-\nabla f(w^k), w^*-w^k\rangle \leq C, \tag{A.11}$$

then the sequence $\{w^k\}$ generated by (A.10) is bounded and $f(w^k)\to f(w^*)$. Thus, all its convergent subsequences converge to some global minimum of f.

Proof:

Note first that the convexity from (A1) implies

$$\langle \nabla f(w^k), w^*-w^k\rangle \leq f(w^*)-f(w^k). \tag{A.12}$$

Then we have

$$\begin{aligned}\left\|w^{k+1}-w^*\right\|^2 &= \left\|w^k-\alpha^k g(w^k)-w^*\right\|^2\\ &= \left\|w^k-w^*\right\|^2+2\alpha^k\langle g(w^k), w^*-w^k\rangle+(\alpha^k)^2\left\|g(w^k)\right\|^2\\ &\leq \left\|w^k-w^*\right\|^2+2\alpha^k\langle g(w^k)-\nabla f(w^k), w^*-w^k\rangle+2\alpha^k(f(w^*)-f(w^k))+(\alpha^k)^2B^2,\end{aligned}$$

where the inequality follows from (A.12) and assumption (A2). Summing this expression for all k and using (A.11) leads to

$$\limsup_k\left\|w^k-w^*\right\|^2 \leq \left\|w^0-w^*\right\|^2+2C+2\sum_{k=0}^{\infty}\alpha^k(f(w^*)-f(w^k))+\sum_{k=0}^{\infty}(\alpha^k)^2B^2.$$

Using assumption (A4) results in the existence of some $\hat{C}$ such that

$$\limsup_k\left\|w^k-w^*\right\|^2+2\sum_{k=0}^{\infty}\alpha^k(f(w^k)-f(w^*))\leq 2\hat{C}. \tag{A.13}$$

Since $\alpha^k>0$ and $f(w^k)\geq f(w^*)$ as w^* is a global minimum of f, we infer that sequence $\{w^k\}$ is bounded and (A.13) implies

$$\sum_{k=0}^{\infty}\alpha^k(f(w^k)-f(w^*))\leq\hat{C}.$$

Since $f(w^k)-f(w^*)\geq 0$, due to assumption (A3) we obtain $\lim f(w^k)\to f(w^*)$, which implies the theorem statement. ■

A.4.2 Proof of Theorem 2.9

For the proof, we will consider a general surrogate which satisfies:

(S1) $l(s) \geq 0$ for all $s \in \mathbb{R}$, $l(0) = 1$ and $l(s) \to 0$ as $s \to -\infty$;

(S2) l is convex and strictly increasing function on (s_0, ∞), where $s_0 := \sup\{s \mid l(s) = 0\}$;

(S3) $\frac{l'}{l}$ is a decreasing function on (s_0, ∞);

(S4) l' is a bounded function;

(S5) l' is a Lipschitz continuous function with modulus L.

All these reguirements are satisfied for the surrogate logistic or by the Huber loss, which is the hinge surrogate which is smoothened on an ε-neighborhood of zero.

Theorem 2.9

Consider the *Pat&Mat* or *Pat&Mat-NP* formulation, stepsizes $\alpha^k = \frac{\alpha^0}{k+1}$ and piecewise disjoint minibatches $I^1, \ldots, I^m$ which cycle periodically $I^{k+m} = I^k$. If l is the smoothened (Huberized) hinge function, then Algorithm 1 converges to the global minimum of (2.16).

Proof of Theorem 2.9 on page 20:

We intend to apply Theorem A.10 and thus, we need to verify its assumptions. Assumption (A1) is satisfied as f is convex due to Theorem 2.3. Assumption (A2) follows directly from Lemma A.13. Assumptions (A3),(A4) and (A5) are imposed directly in the statement of this theorem. It remains to verify (A.11).

For simplicity, we will do so only for $\beta = 1$ and for $s = 2$ minibatches of the same size. However, the proof would be identical for other values. This implies that there are some I^k and I^{k+1} which are pairwise disjoint, they cover all samples and that $I^k = I^{k+2}$ for all k. The assumptions imply that the number of positive samples in each minibatch equal to $n_+^k = \frac{1}{2}n_+$, where n_+ is the total number of positive samples.

First we estimate the difference between s_i^k defined in (2.30) and $x_i^\top w^k$. For any $i \in I^k$ due to the construction (2.30) we have

$$\begin{aligned} s_i^k &= x_i^\top w^k, \\ s_i^{k-1} = s_i^{k-2} &= x_i^\top w^{k-2} = x_i^\top \left(w^k + \alpha^{k-2} g(w^{k-2}) + \alpha^{k-1} g(w^{k-1})\right) \\ &= x_i^\top w^k + \alpha^{k-2} x_i^\top g(w^{k-2}) + \alpha^{k-1} x_i^\top g(w^{k-1}). \end{aligned} \tag{A.14}$$

Similarly, for $i \notin I^k$ we have

$$s_i^k = s_i^{k-1} = x_i^\top w^{k-1} = x_i^\top (w^k + \alpha^{k-1} g(w^{k-1})) = x_i^\top w^k + \alpha^{k-1} x_i^\top g(w^{k-1}). \tag{A.15}$$

Recall that we already verified (A1)-(A5). Combining (A2) with (A.14) and (A.15) yields the existence of some C_2 such that for all $i \in X$ we have

$$\begin{aligned} \left\| s_i^k - x_i^\top w^k \right\| &\leq C_2 \alpha^{k-1}, \\ \left\| s_i^{k-1} - x_i^\top w^k \right\| &\leq C_2 \left(\alpha^{k-1} + \alpha^{k-2}\right). \end{aligned} \tag{A.16}$$

This also immediately implies

$$\begin{aligned}\left\|t^k - t(w^k)\right\| &\leq C_2\alpha^{k-1},\\ \left\|t^{k-1} - t(w^k)\right\| &\leq C_2\left(\alpha^{k-1} + \alpha^{k-2}\right).\end{aligned} \tag{A.17}$$

Since l' is Lipschitz continuous with modulus L according to (S5), due to (A.16) and (A.17) we get

$$\left\|l'(t^k - s_i^k) - l'(t(w^k) - x_i^\top w^k)\right\| \leq L\left\|t^k - s_i^k - t(w^k) + x_i^\top w^k\right\| \leq 2C_2L\alpha^{k-1}. \tag{A.18}$$

In an identical way we can show

$$\begin{aligned}\left\|l'(t^{k-1} - s_i^{k-1}) - l'(t(w^k) - x_i^\top w^k)\right\| &\leq 2C_2L\left(\alpha^{k-1} + \alpha^{k-2}\right),\\ \left\|l'(s_i^k - t^k) - l'(x_i^\top w^k - t(w^k))\right\| &\leq 2C_2L\alpha^{k-1},\\ \left\|l'(s_i^{k-1} - t^{k-1}) - l'(x_i^\top w^k - t(w^k))\right\| &\leq 2C_2L\left(\alpha^{k-1} + \alpha^{k-2}\right).\end{aligned} \tag{A.19}$$

Now we need to estimate the distance between $\nabla t(w^k)$ and ∇t^k. We have

$$\begin{aligned}\nabla t^k &= \frac{\sum_{i\in I^k} l'(s_i^k - t^k)x_i + \sum_{i\in I^{k-1}} l'(s_i^{k-1} - t^{k-1})x_i}{\sum_{i\in X} l'(s_i^k - t^k)},\\ \nabla t(w^k) &= \frac{\sum_{i\in I^k} l'(x_i^\top w^k - t(w^k))x_i + \sum_{i\in I^{k-1}} l'(x_i^\top w^k - t(w^k))x_i}{\sum_{i\in X} l'(x_i^\top w^k - t(w^k))}.\end{aligned} \tag{A.20}$$

The first equality in (A.20) follows from (2.34) and (2.32) while the second equality in (A.20) follows from Theorem 2.4 and $X = I^k \cup I^{k-1}$. From Lemma A.12 we deduce that the denominators in (A.20) are bounded away from zero uniformly in k. Assumption (A4) implies $\alpha^k \to 0$. This allows us to use Lemma A.14 which together with (A.19) implies that there is some C_3 such that for all sufficiently large k we have

$$\left\|\nabla t^k - \nabla t(w^k)\right\| \leq C_3\left(\alpha^{k-1} + \alpha^{k-2}\right). \tag{A.21}$$

Using the assumptions above, we can simplify the terms for $g(w^k)$ and $\nabla f(w^k)$ to

$$\begin{aligned}g(w^k) &= \frac{2}{n_+}\sum_{i\in I_+^k} l'(t^k - s_i^k)(\nabla t^k - x_i),\\ g(w^{k+1}) &= \frac{2}{n_+}\sum_{i\in I_+^{k+1}} l'(t^{k+1} - s_i^{k+1})(\nabla t^{k+1} - x_i),\\ \nabla f(w^k) &= \frac{1}{n_+}\sum_{i\in \mathcal{X}_+} l'(t(w^k) - x_i^\top w^k)(\nabla t(w^k) - x_i),\\ \nabla f(w^{k+1}) &= \frac{1}{n_+}\sum_{i\in \mathcal{X}_+} l'(t(w^{k+1}) - x_i^\top w^{k+1})(\nabla t(w^{k+1}) - x_i).\end{aligned}$$

Due to the assumptions, we have $\mathcal{X}_+ = I_+^k \cup I_+^{k+1}$ and $\emptyset = I_+^k \cap I_+^{k+1}$, which allows us to

write

$$n^+\Big(g(w^k)+g(w^{k+1})-\nabla f(w^k)-\nabla f(w^{k+1})\Big) \tag{A.22a}$$
$$=\sum_{i\in I_+^k} l'(t^k-s_i^k)(\nabla t^k-x_i)-\sum_{i\in I_+^k} l'(t(w^k)-x_i^\top w^k)(\nabla t(w^k)-x_i) \tag{A.22b}$$
$$+\sum_{i\in I_+^k} l'(t^k-s_i^k)(\nabla t^k-x_i)-\sum_{i\in I_+^k} l'(t(w^{k+1})-x_i^\top w^{k+1})(\nabla t(w^{k+1})-x_i) \tag{A.22c}$$
$$+\sum_{i\in I_+^{k+1}} l'(t^{k+1}-s_i^{k+1})(\nabla t^{k+1}-x_i)-\sum_{i\in I_+^{k+1}} l'(t(w^k)-x_i^\top w^k)(\nabla t(w^k)-x_i) \tag{A.22d}$$
$$+\sum_{i\in I_+^{k+1}} l'(t^{k+1}-s_i^{k+1})(\nabla t^{k+1}-x_i)-\sum_{i\in I_+^{k+1}} l'(t(w^{k+1})-x_i^\top w^{k+1})(\nabla t(w^{k+1})-x_i). \tag{A.22e}$$

Then relations (A.21) and (A.18) applied to Lemma A.15 imply

$$\left\|\sum_{i\in I_+^k} l'(t^k-s_i^k)(\nabla t^k-x_i)-\sum_{i\in I_+^k} l'(t(w^k)-x_i^\top w^k)(\nabla t(w^k)-x_i)\right\|\le C_4\Big(\alpha^{k-1}+\alpha^{k-2}\Big)$$

for some C_4, which gives a bound for (A.22b). Bound for (A.22e) is obtained by increasing k by one. Bounds for (A.22c) and (A.22d) can be find similarly using (A.19). Altogether, we showed

$$\left\|g(w^k)+g(w^{k+1})-\nabla f(w^k)-\nabla f(w^{k+1})\right\|\le C_1(\alpha^{k-2}+\alpha^{k-1}+\alpha^k+\alpha^{k+1}) \tag{A.23}$$

for some C_1.

We now estimate

$$\alpha^k\langle g(w^k)-\nabla f(w^k),w^*-w^k\rangle+\alpha^{k+1}\langle g(w^{k+1})-\nabla f(w^{k+1}),w^*-w^{k+1}\rangle \tag{A.24a}$$
$$=\langle g(w^k)-\nabla f(w^k),\alpha^k(w^*-w^k)\rangle+\langle g(w^{k+1})-\nabla f(w^{k+1}),\alpha^{k+1}(w^*-w^{k+1})\rangle \tag{A.24b}$$
$$=\langle g(w^k)-\nabla f(w^k)+g(w^{k+1})-\nabla f(w^{k+1}),\alpha^k(w^*-w^k)\rangle \tag{A.24c}$$
$$+\langle g(w^{k+1})-\nabla f(w^{k+1}),\alpha^{k+1}(w^*-w^{k+1})-\alpha^k(w^*-w^k)\rangle. \tag{A.24d}$$

To estimate (A.24d), we make use of Lemma A.13 to obtain the existence of some C_5 such that

$$\begin{aligned}&\langle g(w^{k+1})-\nabla f(w^{k+1}),\alpha^{k+1}(w^*-w^{k+1})-\alpha^k(w^*-w^k)\rangle\\&\quad\le 2B\left\|\alpha^{k+1}(w^*-w^{k+1})-\alpha^k(w^*-w^k)\right\|\\&\quad=2B\left\|\alpha^{k+1}(w^*-w^k+\alpha^k g(w^k))-\alpha^k(w^*-w^k)\right\|\\&\quad=2B\left\|(\alpha^{k+1}-\alpha^k)w^*+(\alpha^k-\alpha^{k+1})w^k+\alpha^k\alpha^{k+1}g(w^k)\right\|\\&\quad\le C_5\left\|\alpha^{k+1}-\alpha^k\right\|+C_5(\alpha^k)^2+C_5(\alpha^{k+1})^2.\end{aligned} \tag{A.25}$$

In the last inequality we used the equality $2ab\le a^2+b^2$. To estimate (A.24c), we can apply (A.23) together with the boundedness of $\{w^k\}$ to obtain the existence of some C_6 such that

$$\begin{aligned}&\langle g(w^k)-\nabla f(w^k)+g(w^{k+1})-\nabla f(w^{k+1}),\alpha^k(w^*-w^k)\rangle\\&\quad\le C_6(\alpha^{k-2})^2+C_6(\alpha^{k-1})^2+C_6(\alpha^k)^2+C_6(\alpha^{k+1})^2.\end{aligned} \tag{A.26}$$

Plugging (A.25) and (A.26) into (A.24) and summing the terms yields (A.11). Then the assumptions of Theorem A.10 are verified and the theorem statement follows. ■

A.4.3 Auxiliary results

Lemma A.12

Let l satisfy (S1)-(S3). Then there exists some $\hat{C}$ such that for all k we have

$$\sum_{i \in X} l'(s_i^k - t^k) \geq \hat{C} > 0,$$
$$\sum_{i \in X} l'(x_i^\top w^k - t(w^k)) \geq \hat{C} > 0.$$

Proof:
First, we will find an upper bound of $s_i^k - t^k$. Fix any index i_0. Since l is nonnegative due to (S1), equation (2.31) implies

$$n\tau = \sum_{i \in X} l(s_i^k - t^k) \geq l(s_{i_0}^k - t^k).$$

Moreover, as l is a strictly increasing function due to (S2) and $n\tau > 0$, this means

$$l^{-1}(n\tau) \geq s_{i_0}^k - t^k. \tag{A.27}$$

Since i_0 was an arbitrary index, it holds true for all indices. Then (S3) which leads to a further estimate

$$\begin{aligned}\sum_{i \in X} l'(s_i^k - t^k) &= \sum_{i \in X} l(s_i^k - t^k) \frac{l'(s_i^k - t^k)}{l(s_i^k - t^k)} \geq \sum_{i \in X} l(s_i^k - t^k) \frac{l'(l^{-1}(n\tau))}{l(l^{-1}(n\tau))} \\ &= n\tau \frac{l'(l^{-1}(n\tau))}{l(l^{-1}(n\tau))} = l'(l^{-1}(n\tau)),\end{aligned}$$

where the inequality follows from (A.27) and the following equality from (2.31). Due to (S2) we obtain that $l'(l^{-1}(n\tau))$ is a positive number, which finishes the proof of the first part. The second part can be obtained in an identical way. ■

Lemma A.13

Let l satisfy (S1)-(S4). Then there exists some B such that for all k we have $\left\|\nabla f(w^k)\right\| \leq B$ and $\left\|g(w^k)\right\| \leq B$.

Proof:
Due to (S4) the derivative l' is bounded by some $\hat{B}$. Then Theorem 2.4 and Lemma A.12 imply

$$\left\|\nabla t(w^k)\right\| \leq \frac{\hat{B} \sum_{i \in X} \|x_i\|}{\sum_{i \in X} l'(x_i^\top w - t(w))} \leq \frac{\hat{B}}{\hat{C}} \sum_{i \in X} \|x_i\|,$$

which is independent of k. Then (2.29) and again the boundedness of l' imply the existence of some B such that $\left\|\nabla f(w^k)\right\| \leq B$ for all k. The proof for $g(w^k)$ can be performed identically. ■

Lemma A.14

Consider uniformly bounded positive sequences $c_1^k, c_2^k, d_1^k, d_2^k, \alpha^k$ and positive constants C_1, C_2 such that for all k we have $\left\|c_1^k - c_2^k\right\| \leq C_1\alpha^k$, $\left\|d_1^k - d_2^k\right\| \leq C_1\alpha^k$, $d_1^k \geq C_2$ and $d_2^k \geq C_2$. If $\alpha^k \to 0$, then there exists a constant C_3 such that for all sufficiently large k we have

$$\left\|\frac{c_1^k}{d_1^k} - \frac{c_2^k}{d_2^k}\right\| \leq C_3\alpha^k.$$

Proof:
Since d_1^k and d^k are bounded away from zero and since $\alpha^k \to 0$, we have

$$\left\|\frac{c_1^k}{d_1^k} - \frac{c_2^k}{d_2^k}\right\| \leq \max\left\{\frac{c_1^k}{d_1^k} - \frac{c_1^k + C_1\alpha^k}{d_1^k - C_1\alpha^k}, \frac{c_1^k}{d_1^k} - \frac{c_1^k - C_1\alpha^k}{d_1^k + C_1\alpha^k}\right\}.$$

The first term can be estimated as

$$\left\|\frac{c_1^k}{d_1^k} - \frac{c_1^k + C_1\alpha^k}{d_1^k - C_1\alpha^k}\right\| = \left\|\frac{(c_1^k + d_1^k)C_1\alpha^k}{d_1^k(d_1^k - C_1\alpha^k)}\right\| \leq \frac{(c_1^k + d_1^k)C_1\alpha^k}{C_2|d_1^k - C_1\alpha^k|}.$$

Since $\alpha^k \to 0$ by assumption, for large k we have $\left\|d_1^k - C_1\alpha^k\right\| \geq \frac{1}{2}C_2$. Since the sequences are uniformly bounded, the statement follows. ■

Lemma A.15

Consider scalars a_i, c_i and vectors b_i, d_i. If there is some $\hat{C}$ such that $\|a_i\| \leq \hat{C}$ and $\|d_i\| \leq \hat{C}$, then

$$\left\|\sum_{i=1}^n a_i b_i - \sum_{i=1}^n c_i d_i\right\| \leq \hat{C}\sum_{i=1}^n(\|a_i - c_i\| + \|b_i - d_i\|).$$

Proof:
It is simple to verify

$$\left\|\sum_{i=1}^n a_i b_i - \sum_{i=1}^n c_i d_i\right\| \leq \sum_{i=1}^n\|d_i\|\|a_i - c_i\| + \sum_{i=1}^n\|a_i\|\|b_i - d_i\|,$$

from which the statement follows. ■

B

Non-linear case

B.1 Convex conjugate

Recall that for a convex lower semi-continuous function f, its convex conjugate f^* is defined by

$$f^\star(\boldsymbol{y}) = \sup_{\boldsymbol{x}\in\operatorname{dom} f} \{\boldsymbol{y}^\top \boldsymbol{x} - f(\boldsymbol{x})\} = -\inf_{\boldsymbol{x}\in\operatorname{dom} f} \{f(\boldsymbol{x}) - \boldsymbol{y}^\top \boldsymbol{x}\}.$$

For details, see [24, page 91]. For the hinge loss function, it is well-known [60] that

$$\begin{aligned} l_{hinge}(x) &= \max\{0, 1 + \vartheta x\}, \\ l^\star_{hinge}(y) &= \begin{cases} -\dfrac{y}{\vartheta} & \text{if } y \in [0, \vartheta], \\ \infty & \text{otherwise.} \end{cases} \end{aligned} \tag{B.1}$$

Similarly, for the truncated quadratic loss function, it is well-known [61] that

$$\begin{aligned} l_{quad}(x) &= (\max\{0, 1 + \vartheta x\})^2, \\ l^\star_{quad}(y) &= \begin{cases} \dfrac{y^2}{4\vartheta^2} - \dfrac{y}{\vartheta} & \text{if } y \geq 0, \\ \infty & \text{otherwise.} \end{cases} \end{aligned} \tag{B.2}$$

B.2 Proofs for dual problems

Theorem 3.1: *TopPushK* dual formulation

The dual problem corresponding to the problem (3.5) has the form

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} - C\sum_{i=1}^{n^+} l^\star\left(\frac{\alpha_i}{C}\right) \tag{3.7a}$$

$$\text{subject to} \sum_{i=1}^{n^+} \alpha_i = \sum_{j=1}^{n^-} \beta_j, \tag{3.7b}$$

$$0 \leq \beta_j \leq \frac{1}{K}\sum_{i=1}^{n^+} \alpha_i, \quad \forall j = 1, 2, \ldots, n^-, \tag{3.7c}$$

where $l^\star$ is a conjugate of the surrogate loss function l from (3.5) and $\mathbb{K}$ was defined

in (3.6).

Proof of Theorem 3.1 on page 30:
Recall the primal formulation of the *TopPushK* problem (3.5)

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n^+} l\left(\frac{1}{K}\sum_{j=1}^{K} s_{[j]}^- - \boldsymbol{w}^\top \boldsymbol{x}_i^+\right).$$

Using [62, Lemma 1] and the fact that the surrogate function l is non-decreasing, we can write

$$\begin{aligned}\sum_{i=1}^{n^+} l\left(\frac{1}{K}\sum_{j=1}^{K} s_{[j]}^- - \boldsymbol{w}^\top \boldsymbol{x}_i^+\right) &= \sum_{i=1}^{n^+} l\left(\frac{1}{K}\min_t\left\{Kt + \sum_{j=1}^{n^-}\max\{0, s_j^- - t\}\right\} - \boldsymbol{w}^\top \boldsymbol{x}_i^+\right)\\ &= \min_t \sum_{i=1}^{n^+} l\left(t + \frac{1}{K}\sum_{j=1}^{n^-}\max\{0, \boldsymbol{w}^\top \boldsymbol{x}_j^- - t\} - \boldsymbol{w}^\top \boldsymbol{x}_i^+\right).\end{aligned}$$

Using auxiliary variables $\boldsymbol{y} \in \mathbb{R}^{n^+}$ and $\boldsymbol{z} \in \mathbb{R}^{n^-}$, we observe that (3.5) is equivalent to

$$\begin{aligned}\underset{\boldsymbol{w},t,\boldsymbol{y},\boldsymbol{z}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n^+} l(y_i)\\ \text{subject to} \quad & y_i = t + \frac{1}{K}\sum_{j=1}^{n^-} z_j - \boldsymbol{w}^\top \boldsymbol{x}_i^+, \quad && \forall i = 1, 2, \dots n^+,\\ & z_j \geq \boldsymbol{w}^\top \boldsymbol{x}_j^- - t, && \forall j = 1, 2, \dots n^-,\\ & z_j \geq 0, && \forall j = 1, 2, \dots n^-.\end{aligned}$$

The dual objective function is $g(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \min_{\boldsymbol{w},t,\boldsymbol{y},\boldsymbol{z}} L(\boldsymbol{w}, t, \boldsymbol{y}, \boldsymbol{z}; \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$, where the Lagrange function L is defined by

$$\begin{aligned}L(\boldsymbol{w}, t, \boldsymbol{y}, \boldsymbol{z}; \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n^+} l(y_i) + \sum_{i=1}^{n^+}\alpha_i\left(t + \frac{1}{K}\sum_{j=1}^{n^-} z_j - \boldsymbol{w}^\top \boldsymbol{x}_i^+ - y_i\right)\\ + \sum_{j=1}^{n^-}\beta_j(\boldsymbol{w}^\top \boldsymbol{x}_j^- - t - z_j) - \sum_{j=1}^{n^-}\gamma_j z_j\end{aligned} \tag{B.3}$$

Since this function is separable in primal variables, it can be minimized with respect to each variable separately. Optimality conditions with respect to t and $\boldsymbol{z}$ read

$$\begin{aligned}\sum_{i=1}^{n^+}\alpha_i - \sum_{j=1}^{n^-}\beta_j = 0,\\ \frac{1}{K}\sum_{i=1}^{n^+}\alpha_i - \beta_j - \gamma_j = 0, \quad \forall j = 1, 2, \dots n^-.\end{aligned}$$

From the first condition we deduce constraint (3.7b). Plugging the feasibility condition $\gamma_j \geq 0$ into the second optimal condition and combining it with the feasibility conditions $\beta_j \geq 0$ yields constraint (3.7c). By minimizing (B.3) with respect to $\boldsymbol{w}$ we deduce

$$\boldsymbol{w} = \sum_{i=1}^{n^+}\alpha_i \boldsymbol{x}_i^+ - \sum_{j=1}^{n^-}\beta_j \boldsymbol{x}_j^- = \begin{pmatrix}\mathbb{X}^+\\ -\mathbb{X}^-\end{pmatrix}^\top \begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix},$$

where $\mathbb{X}^+$, $\mathbb{X}^-$ are matrices of positive and negative samples respectively (each row corresponds to one sample). Finally, minimization of (B.3) with respect to $\boldsymbol{y}$ yields

$$C \min_{y_i}\left(l(y_i) - \frac{\alpha_i}{C} y_i\right) = -C l^\star\left(\frac{\alpha_i}{C}\right), \quad \forall i = 1, 2, \dots n^+.$$

Plugging all these relations into (B.3) yields the objective function of (3.7), which finishes the proof. ∎

Theorem 3.2: *Pat&Mat* dual formulation

The dual problem corresponding to the problem (3.11) has the form

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} - C\sum_{i=1}^{n^+} l_1^\star\left(\frac{\alpha_i}{C}\right) - \delta\sum_{j=1}^{n^-} l_2^\star\left(\frac{\beta_j}{\delta}\right) - \delta n\tau \tag{3.12a}$$

$$\text{subject to} \sum_{i=1}^{n^+}\alpha_i = \sum_{j=1}^{n^-}\beta_j, \tag{3.12b}$$

$$\delta \geq 0, \tag{3.12c}$$

where $l_1^\star$, $l_2^\star$ are conjugates of the surrogate loss functions l_1, l_2 from (3.11) and $\mathbb{K}$ was defined in (3.6).

Proof of Theorem 3.2 on page 31:
Let us first realize that the primal *Pat&Mat* problem (3.11) is equivalent to

$$\begin{aligned}\underset{\boldsymbol{w},t,\boldsymbol{y},\boldsymbol{z}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n^+} l_1(y_i)\\ \text{subject to} \quad & \sum_{j=1}^{n^-} l_2(z_j) \leq n\tau,\\ & y_i = t - \boldsymbol{w}^\top \boldsymbol{x}_i^+, \quad \forall i = 1, 2, \dots, n^+,\\ & z_j = \boldsymbol{w}^\top \boldsymbol{x}_j^- - t, \quad \forall j = 1, 2, \dots, n^-.\end{aligned}$$

Then the dual function is $g(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}) = \min_{\boldsymbol{w},t,\boldsymbol{y},\boldsymbol{z}} L(\boldsymbol{w}, t, \boldsymbol{y}, \boldsymbol{z}; \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$, where the Lagrange function L is defined by

$$\begin{aligned}L(\boldsymbol{w}, t, \boldsymbol{y}, \boldsymbol{z}; \boldsymbol{\alpha}, \boldsymbol{\beta}, \delta) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n^+} l_1(y_i) + \sum_{i=1}^{n^+}\alpha_i(t - \boldsymbol{w}^\top \boldsymbol{x}_i^+ - y_i)\\ + \sum_{j=1}^{n^-}\beta_j(\boldsymbol{w}^\top \boldsymbol{x}_j^- - t - z_j) + \delta\left(\sum_{j=1}^{n^-} l_2(z_j) - n\tau\right).\end{aligned} \tag{B.4}$$

Since this function is separable in primal variables, it can be minimized with respect to each variable separately. Optimality condition with respect to t read

$$\sum_{i=1}^{n^+}\alpha_i - \sum_{j=1}^{n^-}\beta_j = 0,$$

from which we deduce constraint (3.12b). By minimizing (B.4) with respect to $\boldsymbol{w}$ we deduce

$$\boldsymbol{w} = \sum_{i=1}^{n^+} \alpha_i \boldsymbol{x}_i^+ - \sum_{j=1}^{n^-} \beta_j \boldsymbol{x}_j^- = \begin{pmatrix} \mathbb{X}^+ \\ -\mathbb{X}^- \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix},$$

where $\mathbb{X}$, $\mathbb{X}^+$ are matrices of all and positive samples respectively (each row corresponds to one sample). Finally, minimization of (B.4) with respect to $\boldsymbol{y}$ and $\boldsymbol{z}$ yields

$$C \min_{y_i} \left(l_1(y_i) - \frac{\alpha_i}{C} y_i \right) = -C l_1^\star \left(\frac{\alpha_i}{C} \right),$$
$$\delta \min_{z_j} \left(l_2(z_j) - \frac{\beta_j}{\delta} z_j \right) = -\delta l_2^\star \left(\frac{\beta_j}{\delta} \right).$$

Plugging all these relations into (B.4) yields the objective function of (3.12), which finishes the proof. ■

B.3 Computing Δ^* with truncated quadratic surrogate

B.3.1 *TopPushK*

Theorem 3.3: Update rule for Δ^* for *TopPushK* with truncated quadratic loss

Consider problem (3.18). Then the optimal step $\Delta^\star$ equals to

$$\Delta^* = \mathrm{clip}_{[\Delta_{lb},\, \Delta_{ub}]}(\gamma), \tag{3.19}$$

where there are the following three cases (each corresponding to one update rule in (3.17)):

- For any $1 \leq k, l \leq n^+$ we have

$$\Delta_{lb} = -\alpha_k,$$
$$\Delta_{ub} = \alpha_l,$$
$$\gamma = -\frac{s_k - s_l + \frac{1}{2C\vartheta^2}(\alpha_k - \alpha_l)}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk} + \frac{1}{C\vartheta^2}}.$$

- For any $1 \leq k \leq n^+$ and $n^+ + 1 \leq l \leq n$ we define $\hat{l} = l - n^+$ and $\beta_{\max} =$

$\max_{j\in\{1,2,\dots,n^-\}\setminus\{\hat{l}\}} \beta_j$. Then we have

$$\Delta_{lb} = \begin{cases} \max\{-\alpha_k,\ -\beta_{\hat{l}}\} & \text{K} = 1, \\ \max\left\{-\alpha_k,\ -\beta_{\hat{l}},\ K\beta_{\max} - \sum_{i=1}^{n^+}\alpha_i\right\} & \text{otherwise}, \end{cases}$$
$$\Delta_{ub} = \begin{cases} +\infty & \text{K} = 1, \\ \frac{1}{K-1}\left(\sum_{i=1}^{n^+}\alpha_i - K\beta_{\hat{l}}\right) & \text{otherwise}, \end{cases}$$
$$\gamma = -\frac{s_k + s_l - \frac{1}{\vartheta} + \frac{1}{2C\vartheta^2}\alpha_k}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk} + \frac{1}{2C\vartheta^2}}.$$

- For any $n^+ + 1 \le k, l \le n$ we define $\hat{k} = k - n^+$, $\hat{l} = l - n^+$ and then have

$$\Delta_{lb} = \begin{cases} -\beta_{\hat{k}} & \text{K} = 1, \\ \max\left\{-\beta_{\hat{k}},\ \beta_{\hat{l}} - \frac{1}{K}\sum_{i=1}^{n^+}\alpha_i\right\} & \text{otherwise}, \end{cases}$$
$$\Delta_{ub} = \begin{cases} \beta_{\hat{l}} & \text{K} = 1, \\ \min\left\{\beta_{\hat{l}},\ \frac{1}{K}\sum_{i=1}^{n^+}\alpha_i - \beta_{\hat{k}}\right\} & \text{otherwise}, \end{cases}$$
$$\gamma = -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}.$$

Proof of Theorem 3.3 on page 34:
We will show that for each update rule (3.17) and for fixed $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, problem (3.18) can be rewritten as a quadratic one-dimensional problem

$$\begin{aligned} \underset{\Delta}{\text{maximize}} \quad & -\frac{1}{2}a(\boldsymbol{\alpha},\boldsymbol{\beta})\Delta^2 - b(\boldsymbol{\alpha},\boldsymbol{\beta})\Delta - c(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ \text{subject to} \quad & \Delta_{lb}(\boldsymbol{\alpha},\boldsymbol{\beta}) \le \Delta \le \Delta_{ub}(\boldsymbol{\alpha},\boldsymbol{\beta}). \end{aligned}$$

where a, b, c, Δ_{lb}, Δ_{ub} do not depend on Δ. The optimal solution to this problem is

$$\Delta^* = \text{clip}_{[\Delta_{lb},\,\Delta_{ub}]}\left(-\frac{b}{a}\right), \tag{B.5}$$

which amounts to (3.19). Before discussing the three update rules (3.17), we realize that (3.18b) is always satisfied after the update. For the three updates we have:

- For update rule (3.17a) with $1 \le k, l \le n^+$, constraint (3.18d) is satisfied since no β_j was updated and the sum of all α_i did not change. Constraint (3.18c) reads $-\alpha_k \le \Delta \le \alpha_l$ and objective (3.18a) can be rewritten as

$$-\frac{1}{2}\left[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk} + \frac{1}{C\vartheta^2}\right]\Delta^2 - \left[s_k - s_l + \frac{1}{2C\vartheta^2}(\alpha_k - \alpha_l)\right]\Delta + c(\boldsymbol{\alpha},\boldsymbol{\beta}).$$

- For update rule (3.17b) with $1 \le k \le n^+$ and $n^+ + 1 \le l \le n^+ + n^-$ we define $\hat{l} = l - n^+$. Constraint (3.18c) reads $\Delta \ge -\alpha_k$. Denoting $\beta_{\max} = \max_{j\in\{1,2,\dots,n^-\}\setminus\{\hat{l}\}} \beta_j$, then for any $K \ge 2$ constraint (3.18d) reads

$$\begin{aligned} 0 &\le \beta_{\hat{l}} + \Delta \le \frac{1}{K}\sum_{i=1}^{n^+}\alpha_i + \frac{1}{K}\Delta, \\ 0 &\le \beta_{\max} \le \frac{1}{K}\sum_{i=1}^{n^+}\alpha_i + \frac{1}{K}\Delta. \end{aligned} \tag{B.6}$$

If $K = 1$, the upper bounds for β_j may be omitted as discussed in Section 3.3.4. Combining this with $\Delta \geq -\alpha_k$ yields the lower and upper bound of Δ. Using update rule (3.17b), objective (3.18a) can be rewritten as

$$-\frac{1}{2}\left[\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk} + \frac{1}{2C\vartheta^2}\right]\Delta^2 - \left[s_k + s_l - \frac{1}{\vartheta} + \frac{1}{2C\vartheta^2}\alpha_k\right]\Delta + c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

- For update rule (3.17c) with $n^+ + 1 \leq k, l \leq n^+ + n^-$ we define $\hat{k} = k - n^+$, $\hat{l} = l - n^+$. Since no α_i was updated, constraint (3.18c) is always satisfied. Moreover, since we update only two coordinates of $\boldsymbol{\beta}$, constraint (3.18d) for any $K \geq 2$ reads

$$\begin{aligned} 0 &\leq \beta_{\hat{k}} + \Delta \leq \frac{1}{K}\sum_{i=1}^{n^+} \alpha_i, \\ 0 &\leq \beta_{\hat{l}} - \Delta \leq \frac{1}{K}\sum_{i=1}^{n^+} \alpha_i, \end{aligned} \tag{B.7}$$

  As in the previous case, the upper bounds for β_j may be omitted for $K = 1$. Combining the previous results yields the lower and upper bound of Δ. Using update rule (3.17c), objective (3.18a) can be rewritten as

$$-\frac{1}{2}[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}]\Delta^2 - [s_k - s_l]\Delta + c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

The proofs follows by plugging these cases into the solution (B.5). ■

B.3.2 *Pat&Mat*

Plugging the conjugate (B.2) of the truncated quadratic loss (3.2) into *Pat&Mat* dual formulation (3.12) yields

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad & -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} + \frac{1}{\vartheta_1}\sum_{i=1}^{n^+}\alpha_i - \frac{1}{4C\vartheta_1^2}\sum_{i=1}^{n^+}\alpha_i^2 \\ & + \frac{1}{\vartheta_2}\sum_{j=1}^{n^-}\beta_j - \frac{1}{4\delta\vartheta_2^2}\sum_{j=1}^{n^-}\beta_j^2 - \delta n\tau && \text{(B.8a)} \\ \text{subject to} \quad & \sum_{i=1}^{n^+}\alpha_i = \sum_{j=1}^{n^-}\beta_j, && \text{(B.8b)} \\ & \alpha_i \geq 0, \qquad \forall i = 1, 2, \ldots, n^+, && \text{(B.8c)} \\ & \beta_j \geq 0, \qquad \forall j = 1, 2, \ldots, n^-, && \text{(B.8d)} \\ & \delta \geq 0, && \text{(B.8e)} \end{aligned}$$

The following theorem provides a formula for the optimal step $\Delta^\star$ for the update rule (3.17). Note that we do not perform a joint minimization in $(\alpha_k, \beta_l, \delta)$ but perform a minimization with respect to (α_k, β_l), update these two values and then optimize the objective with respect to δ.

Theorem B.4: Update rule for Δ^* for *Pat&Mat* with truncated quadratic loss

Consider problem (B.8). Then the optimal step $\Delta^\star$ equals to

$$\Delta^* = \mathrm{clip}_{[\Delta_{lb},\, \Delta_{ub}]}(\gamma), \tag{B.9}$$

where there are the following three cases (each correspoding to one update rule in (3.17)):

- If $1 \leq k, l \leq n^+$, then we have

$$\begin{aligned}
\Delta_{lb} &= -\alpha_k, \\
\Delta_{ub} &= \alpha_l, \\
\gamma &= -\frac{s_k - s_l + \frac{\alpha_k - \alpha_l}{2C\vartheta_1^2}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk} + \frac{1}{C\vartheta_1^2}}, \\
\delta^* &= \delta.
\end{aligned}$$

- If $1 \leq k \leq n^+$ and $n^+ + 1 \leq l \leq n$, then defining $\hat{l} = l - n^+$ we have

$$\begin{aligned}
\Delta_{lb} &= \max\{-\alpha_k, -\beta_{\hat{l}}\}, \\
\Delta_{ub} &= +\infty, \\
\gamma &= -\frac{s_k + s_l - \frac{1}{\vartheta_1} + \frac{\alpha_k}{2C\vartheta_1^2} - \frac{1}{\vartheta_2} + \frac{\beta_{\hat{l}}}{2\delta\vartheta_2^2}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk} + \frac{1}{2C\vartheta_1^2} + \frac{1}{2\delta\vartheta_2^2}}, \\
\delta^* &= \sqrt{\delta^2 + \frac{1}{4\vartheta_2 n\tau}(\Delta^{*2} + 2\Delta^*\beta_{\hat{l}})}.
\end{aligned}$$

- If $n^+ + 1 \leq k, l \leq n$, then defining $\hat{k} = k - n^+$, $\hat{l} = l - n^+$ we have

$$\begin{aligned}
\Delta_{lb} &= -\beta_{\hat{k}}, \\
\Delta_{ub} &= \beta_{\hat{l}}, \\
\gamma &= -\frac{s_k - s_l + \frac{\beta_{\hat{k}} - \beta_{\hat{l}}}{2\delta\vartheta_2^2}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk} + \frac{1}{\delta\vartheta_2^2}}, \\
\delta^* &= \sqrt{\delta^2 + \frac{1}{2\vartheta_2 n\tau}(\Delta^{*2} + \Delta^*(\beta_{\hat{k}} - \beta_{\hat{l}}))}.
\end{aligned}$$

Proof:

In the beginning of this subsection we derived problem (B.8). As in the proof of Theorem 3.3, we show, that for each of update rules (3.17) and for fixed $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, δ, this problem can be rewritten as a simple one-dimensional quadratic problem with bound constraints. In this case, however, we have to also consider the third primal variable δ. For fixed $\boldsymbol{\alpha}$

and $\boldsymbol{\beta}$,, maximizing objective function (B.8a) with respect to δ leads to the

$$\begin{aligned} \underset{\delta}{\text{maximize}} \quad & -(n\tau)\delta - \left(\frac{1}{4\vartheta_2^2}\sum_{j=1}^{n^-}\beta_j^2\right)\frac{1}{\delta}, \\ \text{subject to} & \delta \geq 0. \end{aligned}$$

The solution of this problem equals to

$$\delta^* = \sqrt{\frac{1}{4\vartheta_2^2 n\tau}\sum_{j=1}^{n}\beta_j^2}. \tag{B.10}$$

In the following list, we discuss each of update rules (3.17):

- For update rule (3.17a) and any $1 \leq k, l \leq n^+$, constraint (B.8d) is satisfied since no β_j was updated. Constraint (B.8c) reads $-\alpha_k \leq \Delta \leq \alpha_l$ while objective (B.8a) can be rewritten as

  $$-\frac{1}{2}\left[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk} + \frac{1}{C\vartheta_1^2}\right]\Delta^2 - \left[s_k - s_l + \frac{1}{2C\vartheta_1^2}(\alpha_k - \alpha_l)\right]\Delta + c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

  Since optimal δ is given by (B.10) and no β_j was updated, the optimal δ does not change.

- For update rule (3.17b) with $1 \leq k \leq n^+$ and $n^+ + 1 \leq l \leq n$ we define $\hat{l} = l - n^+$. In this case, constraints (B.8c,B.8d) can be written in a simple form $\Delta \geq \max\{-\alpha_k, -\beta_{\hat{l}}\}$ and Δ has no upper bound. Objective (B.8a) can be rewritten as

  $$\begin{aligned} -\frac{1}{2}\left[\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk} + \frac{1}{2C\vartheta_1^2} + \frac{1}{2\delta\vartheta_2^2}\right]\Delta^2 \ldots \\ -\left[s_k + s_l - \frac{1}{\vartheta_1} - \frac{1}{\vartheta_2} + \frac{\alpha_k}{2C\vartheta_1^2} + \frac{\beta_{\hat{l}}}{2\delta\vartheta_2^2}\right]\Delta + c(\boldsymbol{\alpha}, \boldsymbol{\beta}). \end{aligned}$$

  We know that the optimal δ^* is given by (B.10), then

  $$\delta^* = \sqrt{\frac{1}{4\vartheta_2^2 n\tau}\left(\sum_{j\neq\hat{l}}\beta_j^2 + (\beta_{\hat{l}} + \Delta^\star)^2\right)} = \sqrt{\delta^2 + \frac{1}{4\vartheta_2^2 n\tau}(\Delta^{\star 2} + 2\Delta^\star\beta_{\hat{l}})}.$$

- For update rule (3.17c) with $n^+ + 1 \leq k, l \leq n^+ + n^-$ we define $\hat{k} = k - n^+$, $\hat{l} = l - n^+$. Since no α_i was updated, constraint (B.8c) is always satisfied. Constraint (B.8d) can be written in a simple form $-\beta_{\hat{k}} \leq \Delta \leq \beta_{\hat{l}}$ and objective (B.8a) can be rewritten as

  $$-\frac{1}{2}\left[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk} + \frac{1}{2\delta\vartheta_2^2}\right]\Delta^2 - \left[s_k - s_l + \frac{\beta_{\hat{k}} - \beta_{\hat{l}}}{\delta\vartheta_2^2}\right]\Delta + c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

  We know that the optimal δ^* is given by (B.10), then

  $$\delta^* = \sqrt{\frac{1}{4\vartheta_2^2 n\tau}\left(\sum_{j\notin\{\hat{l},\hat{k}\}}\beta_j^2 + (\beta_{\hat{k}} + \Delta^\star)^2 + (\beta_{\hat{l}} - \Delta^\star)^2\right)} = \sqrt{\delta + \frac{1}{2\vartheta_2^2 n\tau}(\Delta^{\star 2} + \Delta^\star(\beta_{\hat{k}} - \beta_{\hat{l}}))}.$$

The proofs follows by plugging these cases into the solution (B.9). ■

B.4 Computing Δ^* with hinge loss function

In this section, we provide the results when the truncated quadratic surrogate is replaced by the hinge surrogate. Since the proofs are identical, we omit them.

B.4.1 *TopPushK*

Plugging the conjugate (B.1) of the hinge loss (3.3) into *TopPushK* dual formulation (3.7) yields

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} + \frac{1}{\vartheta}\sum_{i=1}^{n^+}\alpha_i \tag{B.11a}$$

$$\text{subject to} \sum_{i=1}^{n^+}\alpha_i = \sum_{j=1}^{n^-}\beta_j, \tag{B.11b}$$

$$0 \le \alpha_i \le C\vartheta, \qquad \forall i = 1,2,\dots,n^+, \tag{B.11c}$$

$$0 \le \beta_j \le \frac{1}{K}\sum_{i=1}^{n^+}\alpha_i, \quad \forall j = 1,2,\dots,n^-. \tag{B.11d}$$

This is a convex quadratic problem. Moreover, for $K = 1$, the upper limit in (B.11d) is always satisfied due to (B.11b) and the problem can be simplified. The following theorem provides a formula for optimal Δ for each of update rules (3.17).

Theorem B.5: Update rule for Δ^* for *TopPushK* with hinge loss

Consider problem (B.11). Then

$$\Delta^* = \operatorname{clip}_{[\Delta_{lb},\,\Delta_{ub}]}(\gamma),$$

where there are the following cases:

- For any $1 \le k, l \le n^+$ we have

$$\begin{aligned}\Delta_{lb} &= \min\{-\alpha_k,\ \alpha_l - C\vartheta\},\\ \Delta_{ub} &= \max\{C\vartheta - \alpha_k,\ \alpha_l\},\\ \gamma &= -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}.\end{aligned}$$

- For any $1 \le k \le n^+$ and $n^+ + 1 \le l \le n$ we define $\hat{l} = l - n^+$ and $\beta_{\max} = \max_{j\in\{1,2,\dots,n^-\}\setminus\{\hat{l}\}}\beta_j$. Then we have

$$\Delta_{lb} = \begin{cases}\min\{-\alpha_k,\ -\beta_{\hat{l}}\} & \text{K} = 1,\\ \min\left\{-\alpha_k,\ -\beta_{\hat{l}},\ K\beta_{\max} - \sum_{i=1}^{n^+}\alpha_i\right\} & \text{otherwise},\end{cases}$$

$$\Delta_{ub} = \begin{cases}C\vartheta - \alpha_k & \text{K} = 1,\\ \max\left\{C\vartheta - \alpha_k,\ \frac{1}{K-1}\left(\sum_{i=1}^{n^+}\alpha_i - K\beta_{\hat{l}}\right)\right\} & \text{otherwise}.\end{cases}$$

$$\gamma = -\frac{s_k + s_l - \frac{1}{\vartheta}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk}}.$$

- For any $n^+ + 1 \leq k, l \leq n^+ + n^-$ we define $\hat{k} = k - n^+$, $\hat{l} = l - n^+$ and then we have

$$\Delta_{lb} = \begin{cases} -\beta_{\hat{k}} & \text{K} = 1, \\ \min\left\{-\beta_{\hat{k}},\ \beta_{\hat{l}} - \frac{1}{K}\sum_{i=1}^{n^+}\alpha_i\right\} & \text{otherwise}, \end{cases}$$
$$\Delta_{ub} = \begin{cases} \beta_{\hat{l}} & \text{K} = 1, \\ \max\left\{\frac{1}{K}\sum_{i=1}^{n^+}\alpha_i - \beta_{\hat{k}},\ \beta_{\hat{l}}\right\} & \text{otherwise}. \end{cases}$$
$$\gamma = -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}.$$

B.4.2 *Pat&Mat*

Plugging the conjugate (B.1) of the hinge loss (3.3) into *Pat&Mat* dual formulation (3.12) yields

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} + \frac{1}{\vartheta_1}\sum_{i=1}^{n^+}\alpha_i + \frac{1}{\vartheta_2}\sum_{j=1}^{n}\beta_j - \delta n\tau \tag{B.12a}$$
$$\text{subject to} \sum_{i=1}^{n^+}\alpha_i = \sum_{j=1}^{n}\beta_j, \tag{B.12b}$$
$$0 \leq \alpha_i \leq C\vartheta_1, \quad \forall i = 1, 2, \ldots, n^+, \tag{B.12c}$$
$$0 \leq \beta_j \leq \delta\vartheta_2, \quad \forall j = 1, 2, \ldots, n, \tag{B.12d}$$
$$\delta \geq 0, \tag{B.12e}$$

The following theorem provides a formula for optimal Δ for each of update rules (3.17).

Theorem B.6: Update rule for Δ^* for *Pat&Mat* with hinge loss

Consider problem (B.12). Then

$$\Delta^* = \text{clip}_{[\Delta_{lb},\,\Delta_{ub}]}(\gamma),$$

where there are the following cases:

- For any $1 \leq k, l \leq n^+$ we have

$$\Delta_{lb} = \min\{-\alpha_k,\ \alpha_l - C\vartheta_1\},$$
$$\Delta_{ub} = \max\{C\vartheta_1 - \alpha_k,\ \alpha_l\},$$
$$\gamma = -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}},$$
$$\delta^* = \delta.$$

- For any $1 \leq k \leq n^+$ and $n^+ + 1 \leq l \leq n^+$ we define $\hat{l} = l - n^+$ and $\beta_{\max} = \max_{j\in\{1,2,\ldots,n\}\setminus\{\hat{l}\}}\beta_j$. Then we have

$$\Delta_{lb} = \max\{-\alpha_k,\ -\beta_{\hat{l}}\}, \qquad \Delta_{ub} = C\vartheta_1 - \alpha_k$$

and the optimal solution is one of the two following possibilities which maximizes the original objective:

1. If $\beta_{\hat{l}} + \Delta^* \leq \beta_{\max}$, then

$$\gamma = -\frac{s_k + s_l - \frac{1}{\vartheta_1} - \frac{1}{\vartheta_2}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk}},$$
$$\delta^* = \frac{\beta_{\max}}{\vartheta_2}.$$

2. If $\beta_{\hat{l}} + \Delta^* \geq \beta_{\max}$, then

$$\gamma = -\frac{s_k + s_l - \frac{1}{\vartheta_1} - \frac{1-n\tau}{\vartheta_2}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk}},$$
$$\delta^* = \frac{\beta_{\hat{l}} + \Delta^*}{\vartheta_2}.$$

- For any $n^+ + 1 \leq k, l \leq n$ we define $\hat{k} = k - n^+$, $\hat{l} = l - n^+$. Then we have

$$\Delta_{lb} = -\beta_{\hat{k}}, \qquad \Delta_{ub} = \beta_{\hat{l}},$$

and the optimal solution is one of the three following possibilities which maximizes the original objective:

1. If $\beta_{\max} \geq \max\{\beta_{\hat{k}} + \Delta^*, \beta_{\hat{l}} - \Delta^*\}$, then

$$\gamma = -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}},$$
$$\delta^* = \frac{\beta_{\max}}{\vartheta_2}.$$

2. If $\beta_{\hat{k}} + \Delta^* \geq \max\{\beta_{\max}, \beta_{\hat{l}} - \Delta^*\}$, then

$$\gamma = -\frac{s_k - s_l + \frac{n\tau}{\vartheta_2}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}},$$
$$\delta^* = \frac{\beta_{\hat{k}} + \Delta}{\vartheta_2}.$$

3. If $\beta_{\hat{l}} - \Delta^* \geq \max\{\beta_{\hat{k}} + \Delta^*, \beta_{\max}\}$, then

$$\gamma = -\frac{s_k - s_l - \frac{n\tau}{\vartheta_2}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}},$$
$$\delta^* = \frac{\beta_{\hat{k}} - \Delta^*}{\vartheta_2}.$$

C

Deep

C.1 Code online

To promote reproducibility, we share all our code online. We follow the NeurIPS instructions which allow sharing only anonymized repositories. We provide one respository with the code[1] and one repository with numerical experiments.[2]

C.2 Theorem 4.3 for Rec@K

The assumption of Theorem 4.3 requires that the threshold is computing from negative samples and the objective for positive samples. This does not hold for Rec@K. We will show that we can obtain a similar result even for this case.

The proof of Theorem 4.3 is based on Lemma 4.2. We will now obtain the variant of Lemma 4.2 for Rec@K. First, we realize that if the threshold index j^* corresponds to a negative sample, the computation will not change and therefore

$$\mathbb{E}\Big(\nabla\hat{L}(\boldsymbol{w}) \mid \hat{j} = j^* \text{ is an index of a negative sample}\Big) = \nabla L(\boldsymbol{w}).$$

On the other hand, when j^* corresponds to a positive sample, it needs to be always present in the minibatch selection and there are effectively only $n^+_{\text{mb}} - 1$ positive samples in the minibatch. Then

$$\mathbb{E}\Big(\nabla\hat{L}(\boldsymbol{w}) \mid \hat{j} = j^* \text{ is an index of a positive sample}\Big) = \frac{n^+_{\text{mb}} - 1}{n^+_{\text{mb}}}\nabla L(\boldsymbol{w}).$$

Denote now p the probability that the threshold corresponds to a positive sample. Then we have

$$\begin{aligned}\mathbb{E}\Big(\nabla\hat{L}(\boldsymbol{w}) \mid \hat{j} = j^*\Big) &= (1-p)\nabla L(\boldsymbol{w}) + p\frac{n^+_{\text{mb}} - 1}{n^+_{\text{mb}}}\nabla L(\boldsymbol{w}) \\ &= \nabla L(\boldsymbol{w}) - \frac{p}{n^+_{\text{mb}}}\nabla L(\boldsymbol{w}).\end{aligned}$$

Theorem 4.3 will then be modified into

$$\begin{aligned}\text{bias}(\boldsymbol{w}) = \mathbb{P}(\hat{j} \neq j^*)\Big(\nabla L(\boldsymbol{w}) - \mathbb{E}\Big(\nabla\hat{L}(\boldsymbol{w}) \mid \hat{j} \neq j^*\Big)\Big) \\ - \mathbb{P}(\hat{j} = j^*)\frac{p}{n^+_{\text{mb}}}\nabla L(\boldsymbol{w}).\end{aligned}$$

[1] https://anonymous.4open.science/r/AccuracyAtTop-7562
[2] https://anonymous.4open.science/r/AccuracyAtTop_DeepTopPush-834E

We changed the result by adding the last term. Usually the training set contains much less positive than negative samples. This implies that p is assumed to be small and the extra term is small as well. Thefore, this change should have a negligible impact on the theorem implications.

C.2.1 Used network architecture

For 3A4, we preprocessed the input with 9491 into a 100-dimensional input by PCA. Then we used two dense layers of size 100×50 and 50×25 with batch-normalization after these layers. The last layer was dense.

For FashionMNIST, we used a network alternating two hidden convolutional layers with two max-pooling layers finished with a dense layer. The convolutional layers used kernels 5×5 and had 20 and 50 channels, respectively. For CIFAR100 and SVHN2, we increased the number of hidden and max-pooling layers from two to three. The convolutional layers used kernels 3×3 and had 64, 128, and 128 channels, respectively. A more detailed description can be found in our codes online. We are fully aware that these architectures are suboptimal. Since the accuracy at the top needs to select only a few relevant samples and the rest of the dataset's performance is irrelevant, such a network can be used. Moreover, using a simpler network has the advantage of faster experiments.

For ImageNet, we merged all turtles into the positive class and all non-turtles into the negative class. Then we used the pre-trained EfficientNet B0, where we replaced the last dense layer with 1000 outputs by a dense layer into a scalar output.

C.3 Dataset summary

Table C.1 summarizes the used datasets. The Malware Detection dataset was represented by JSONs, which contain varying number of features. Moreover, many features are not scalar but have some hierarchical structure as well.

		Train		Test		License
Dataset	d	n	$\frac{n^+}{n}$	n	$\frac{n^+}{n}$	
FashionMNIST	$28 \times 28 \times 1$	60 000	10.00%	10 000	10.00%	MIT
CIFAR100	$32 \times 32 \times 3$	50 000	1.00%	10 000	1.00%	not specified
SVHN2 extra	$32 \times 32 \times 3$	604 388	17.28%	26 032	19.59%	not specified
ImageNet	62720×1	1 281 167	0.51%	50000	0.50%	registration
3A4	9491×1	37 241	0.98%	37 241	1.07%	CC BY 4.0
Malware Detection	variable	6 580 166	87.22%	800 346	91.80%	proprietary

Table C.1: Summary of the used datasets with the number of features d, number of samples n and the fraction of positive samples $\frac{n^+}{n}$ in the training set.

Bibliography

[1] Giuseppe Viale. The current state of breast cancer classification. *Annals of Oncology*, 23:x207–x210, 2012.

[2] Daniel Lévy and Arzav Jain. Breast mass classification from mammograms using deep convolutional neural networks. *arXiv preprint arXiv:1612.00542*, 2016.

[3] Martin Grill and Tomáš Pevný. Learning combination of anomaly detectors for security domain. *Computer Networks*, 107:55–63, 2016.

[4] Karen Scarfone, Peter Mell, et al. Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007):94, 2007.

[5] Giorgio Giacinto and Fabio Roli. Intrusion detection in computer networks by multiple classifier systems. In *Object recognition supported by user interaction for service robots*, volume 2, pages 390–393. IEEE, 2002.

[6] Shashank Shanbhag and Tilman Wolf. Accurate anomaly detection through parallelism. *IEEE network*, 23(1):22–28, 2009.

[7] Frederick Kaefer, Carrie M Heilman, and Samuel D Ramenofsky. A neural network application to consumer classification to improve the timing of direct marketing activities. *Computers & Operations Research*, 32(10):2595–2615, 2005.

[8] Xi-Zheng Zhang. Building personalized recommendation system in e-commerce using association rule-based mining and classification. In *2007 International Conference on Machine Learning and Cybernetics*, volume 7, pages 4113–4118. IEEE, 2007.

[9] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[10] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[11] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003.

[12] Shivani Agarwal. The infinite push: A new support vector ranking algorithm that directly optimizes accuracy at the absolute top of the list. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 839–850. SIAM, 2011.

[13] Cynthia Rudin. The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list. *J. Mach. Learn. Res.*, 10:2233–2271, December 2009.

[14] Nan Li, Rong Jin, and Zhi-Hua Zhou. Top rank optimization in linear time. In *Advances in neural information processing systems*, NIPS'14, pages 1502–1510, Cambridge, MA, USA, 2014. MIT Press.

[15] Stephen Boyd, Corinna Cortes, Mehryar Mohri, and Ana Radovanovic. Accuracy at the top. In *Advances in neural information processing systems*, pages 953–961, 2012.

[16] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 377–384, New York, NY, USA, 2005. ACM.

[17] Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Surrogate functions for maximizing precision at the top. In *International Conference on Machine Learning*, pages 189–198, 2015.

[18] Elad ET Eban, Mariano Schain, Alan Mackey, Ariel Gordon, Rif A Saurous, and Gal Elidan. Scalable learning of non-decomposable objectives. In *Artificial Intelligence and Statistics*, pages 832–840, 2017.

[19] Dirk Tasche. A plug-in approach to maximising precision at the top and recall at the top. *arXiv preprint arXiv:1804.03077*, 2018.

[20] Maksim Lapin, Matthias Hein, and Bernt Schiele. Top-k multiclass svm. In *Advances in Neural Information Processing Systems*, pages 325–333, 2015.

[21] Maksim Lapin, Matthias Hein, and Bernt Schiele. Analysis and optimization of loss functions for multiclass, top-k, and multilabel classification. *IEEE transactions on pattern analysis and machine intelligence*, 40(7):1533–1554, 2018.

[22] Alan Mackey, Xiyang Luo, and Elad Eban. Constrained classification and ranking via quantiles. *arXiv preprint arXiv:1803.00067*, 2018.

[23] Ao Zhang, Nan Li, Jian Pu, Jun Wang, Junchi Yan, and Hongyuan Zha. *tau*-fpl: Tolerance-constrained learning in linear time. *arXiv preprint arXiv:1801.04701*, 2018.

[24] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[25] Václav Mácha, Lukáš Adam, and Václav Šmídl. Nonlinear classifiers for ranking problems based on kernelized svm. *arXiv preprint arXiv:2002.11436*, 2020.

[26] Lukáš Adam and Martin Branda. Machine learning approach to chance-constrained problems: An algorithm based on the stochastic gradient descent. *arXiv preprint arXiv:1905.10986*, 2019.

[27] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[28] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Vincent G Sigillito, Simon P Wing, Larrie V Hutton, and Kile B Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10(3):262–266, 1989.

[30] Pierre Baldi, Kyle Cranmer, Taylor Faucett, Peter Sadowski, and Daniel Whiteson. Parameterized neural networks for high-energy physics. *The European Physical Journal C*, 76(5):235, 2016.

[31] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in neural information processing systems*, pages 545–552, 2005.

[32] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7:1–30, 2006.

[33] Lukáš Adam, Václav Mácha, Václav Šmídl, and Tomáš Pevný. General framework for binary classification on top samples. *accepted to Optimization Methods and Software*, 2021.

[34] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[35] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.

[36] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review*, 52(1):1–37, 2019.

[37] Tino Werner. A review on ranking problems in statistical learning. *arXiv preprint arXiv:1909.02998*, 2019.

[38] Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate descent method for large-scale l2-loss linear support vector machines. *Journal of Machine Learning Research*, 9(Jul):1369–1398, 2008.

[39] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[40] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[41] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[42] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[43] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[44] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[45] Abhishek Kumar, Harikrishna Narasimhan, and Andrew Cotter. Implicit rate-constrained optimization of non-decomposable objectives. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5861–5871. PMLR, 2021.

[46] Andrew Cotter, Heinrich Jiang, Maya R Gupta, Serena Wang, Taman Narayan, Seungil You, and Karthik Sridharan. Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *Journal of Machine Learning Research*, 20(172):1–59, 2019.

[47] Martin Engilberge, Louis Chevallier, Patrick Pérez, and Matthieu Cord. Sodeep: a sorting deep net to learn ranking loss surrogates. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10792–10801, 2019.

[48] Thibaut Thonet, Yagmur Gizem Cinar, Eric Gaussier, Minghan Li, and Jean-Michel Renders. Smoothi: Smooth rank indicators for differentiable ir metrics. *arXiv preprint arXiv:2105.00942*, 2021.

[49] Rizal Fathony and J Zico Kolter. Ap-perf: Incorporating generic performance metrics in differentiable learning. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[50] Peter W. Glynn. Importance sampling for monte carlo estimation of quantiles. In *Proc. 2nd St. Petersburg Workshop on Simulation*, pages 180–185, St Petersburg, Russia, 1996. Publishing House of Saint Petersburg University.

[51] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

[52] Alex Krizhevsky, Geoffrey Hinton, et al. *Learning multiple layers of features from tiny images*. Citeseer, 2009.

[53] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in neural information processing systems*, NIPS'11, pages 1502–1510, 2011.

[54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision (IJCV)*, 115(3):211–252, 2015.

[55] Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274, 2015.

[56] Harikrishna Narasimhan, Andrew Cotter, and Maya Gupta. Optimizing generalized rate metrics with three players. In *Advances in Neural Information Processing Systems*, pages 10747–10758, 2019.

[57] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[58] Tomáš Pevný and Petr Somol. Using neural network formalism to solve multiple-instance problems. In *International Symposium on Neural Networks*, pages 135–142. Springer, 2017.

[59] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[60] Shai Shnlev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *31st International Conference on Machine Learning, ICML 2014*, volume 1, page 111, 2014.

[61] Takafumi Kanamori, Akiko Takeda, and Taiji Suzuki. Conjugate relation between loss functions and uncertainty sets in classification problems. *The Journal of Machine Learning Research*, 14(1):1461–1504, 2013.

[62] Wlodzimierz Ogryczak and Arie Tamir. Minimizing the sum of the k largest functions in linear time. *Information Processing Letters*, 85(3):117–122, 2003.