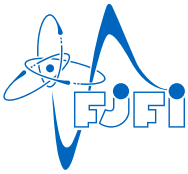
Czech Technical University in Prague
Faculty of Nuclear Sciences and
Physical Engineering

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

Todo list

1

Introduction to Binary Classification

change font (??? libertine) and font size (??? 11pt)

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. 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 and the negtative class using label 0 (for neural networks) or -1 (for SVM-like algorithms [9]).

Notation 1.1: Dataset

In the rest of the work,, we follow the notation used for neural networks, i.e. we use 1 as positive label and 0 as negative label. Moreover, by dataset of size $n \in \mathbb{N}$ we mean set in the form

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

where $\boldsymbol{x}_i \in \mathbb{R}^d$ represents samples, $d \in \mathbb{N}$ its dimension and $y_i \in \{0, 1\}$ represents corresponding labels. To simplify future notation, we denote set of all indices of dataset $\mathcal{D}$ as $\mathcal{I} = \mathcal{I}_- \cup \mathcal{I}_+$, where

$$\mathcal{I}_- = \{i \mid i \in \{1, 2, \ldots, n\} \wedge y_i = 0\},$$
$$\mathcal{I}_+ = \{i \mid i \in \{1, 2, \ldots, n\} \wedge y_i = 1\}.$$

We also denote the number of negative samples in $\mathcal{D}$ as $n_- = |\mathcal{I}_-|$ and the number of positive samples in $\mathcal{D}$ as $n_+ = |\mathcal{I}_+|$, i.e. total number of samples is $n = n_- + n_+$.

The goal of any classification problem is to classify given samples with the highest possible accuracy or in other words with the lowest possible error. In the case of binary classification, there are two types of error: positive samples classified as negative and vice versa. Formally, using the Notation 1.1, the minimization of these two types of errors can be written as follows

$$\begin{aligned} \underset{\boldsymbol{w}, t}{\text{minimize}} \quad & \lambda_1 \sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} + \lambda_2 \sum_{i \in \mathcal{I}_+} \mathbb{1}_{[s_i < t]} \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \end{aligned} \tag{1.1}$$

where $\lambda_1, \lambda_2 \in \mathbb{R}$, the function $f : \mathbb{R}^d \to \mathbb{R}$ and $\mathbb{1}_{[.]}$ is Iverson function that is used to counts misclassified samples and is defined as

$$\mathbb{1}_{[x]} = \begin{cases} 0 & \text{if } x \text{ is false,} \\ 1 & \text{if } x \text{ is true.} \end{cases} \tag{1.2}$$

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, the decision threshold is set to $t = 0.5$ and the sample is classified as positive if its classification score is larger than this threshold. In Notation 1.2, we summarize the notation that is used in the rest of the work.

Notation 1.2: Classifier

By classifier, we always mean pair of model f and corresponding decision threshold $t \in \mathbb{R}$. By model, we mean a function $f\colon \mathbb{R}^d \to \mathbb{R}$ which maps samples $\boldsymbol{x}$ to its classification scores s, i.e. for all $i \in \mathcal{I}$ the classification score is defined as

$$s_i = f(\boldsymbol{x}_i;\, \boldsymbol{w}),$$

where $\boldsymbol{w}$ represents trainable parameters (weights) of the model. Predictions are defined $i \in \mathcal{I}$ in the following way

$$\hat{y}_i = \begin{cases} 1 & \text{if } s_i \geq t, \\ 0 & \text{otherwise.} \end{cases}$$

Add description of differrent binary classification problems such as SVm, logistric regression ...

1.1 Performance Evaluation

In the previous section we defined general binary classification problem 1.1. However, we did not discuss yet how to measure the performance of the resulting classifier. In this section, we will introduce basic approaches that are used to measure the performance of binary classifiers.

1.1.1 Confusion Matrix

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 model f as $\boldsymbol{s} \in \mathbb{R}^n$, where $s_i = f(\boldsymbol{x}_i; \boldsymbol{w})$ for all $i \in \mathcal{I}$, 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 $\mathrm{tp} = \mathrm{tp}(\boldsymbol{s}, t)$ (and similar notation for 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

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

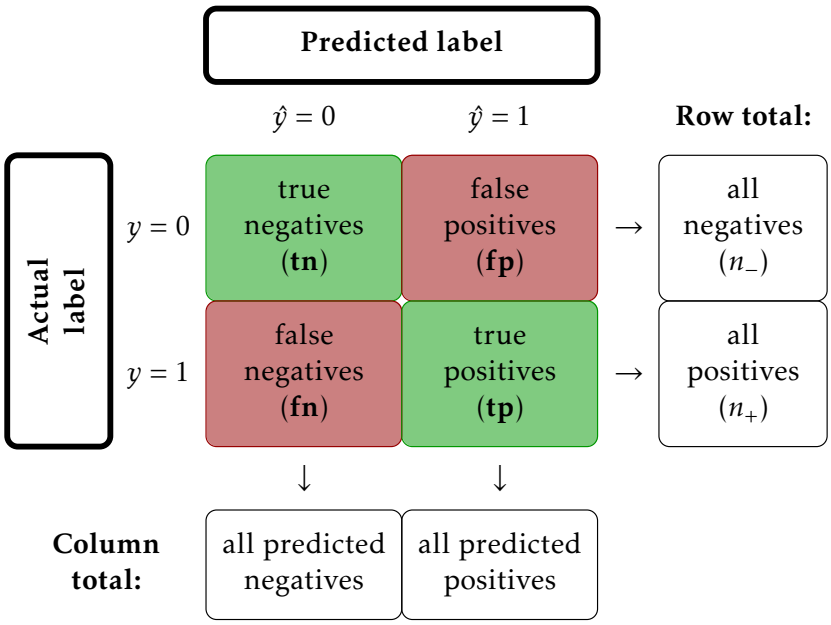


Figure 1.1: Representation of the confusion matrix for the binary classification problem, where the negative class has label 0 and the positive class has label 1. The true (target) label is denoted as y and predicted label is denoted as $\hat{y}$.

Figure 1.2 show the relation between classification rates and the decision threshold. The blue and red curves represent theoretical distribution of the scores of negative and positive samples samples respectively. The position 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 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 (1.1), 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 (1.4) and rewrite the problem (1.1) to the following form

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

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

In addition to the confusion matrix, there are many other classification metrics, and many of them are derived directly from the confusion matrix. As an example, we can mention accuracy and the balanced accuracy. Accuracy is defined as the ratio of correctly classified samples from all samples [11]

$$\mathrm{acc} = \frac{\mathrm{tp} + \mathrm{tn}}{n}.$$

However, the accuracy is not suitable for unbalanced datasets, i.e. for dataset where the number of samples in one class is significantly higher then the number of samples in the other class.

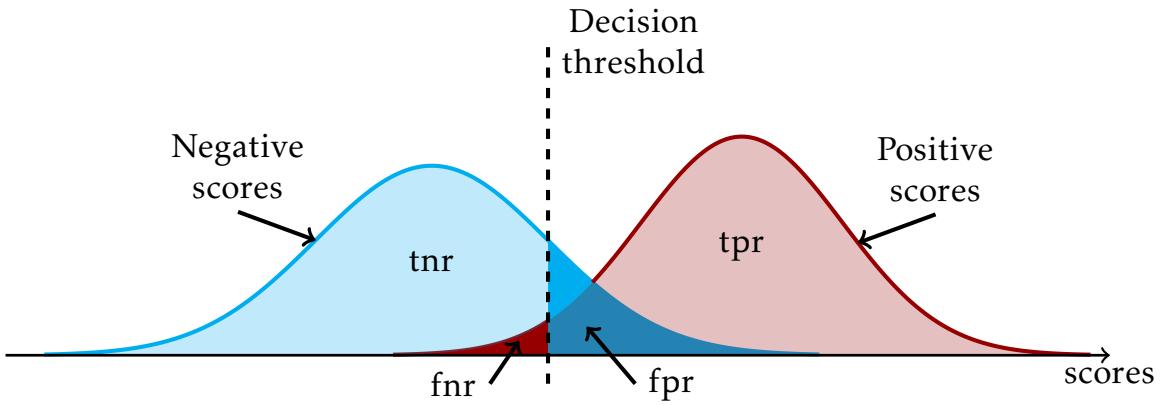


Figure 1.2: The relation between classification scores and rates. The blue curve represents theoretical distribution of the scores of negative samples and the red curve the same for the score of positive samples. Filled areas with light blue or red color represent true-negative and true-positive rates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

In such a case, the balanced accuracy is better. The balanced accuracy is defined as an average of true-positive and true-negative rate [12]

$$\text{bacc} = \frac{1}{2}(\text{tpr} + \text{tnr}).$$

The difference can be easily demonstrated on a simple example. Let us suppose that we have 100 samples and 10 of them is negatives and the rest is positive. If we use simple classifier that all samples clasify as positive we will get the following accuracy

$$\text{acc} = \frac{90 + 0}{100} = 0.9.$$

Even though we know, that the classifier totally ignores negative samples, the accuracy is still 90%. The reason is, that the used classifier is biased towards the more frequent class. Balanced accuracy solves this problem by using true-positive and true-negative rates instead of counts, which leads to the following results for the given example

$$\text{bacc} = \left(\frac{90}{90} + \frac{0}{10}\right) = 0.5.$$

In this case the balanced accuracy is only 50% which is very poor, but is more relevant to the unbalanced dataset. There are many more classification metrics that are based on the confusion matrix [10, 11, 12, 13]. In this work, however, we will use mainly those that we have presented in this section. For simplicity, Table 1.1 provides a summary of binary classification metrics used in this work.

1.1.2 ROC Analysis

In the previous section, we defined general binary clasification problem as a minimization task with objective that consists of a weighted sum of the false-positive and false-negative counts (1.5). For fixed model f and decision threshold t, the results can be visualized in the Receiver Operating Characteristic space [14].

Finish roc section

Name	Aliases	Formula
true negatives	correct rejection	tn
false positives	Type I error, false alarm	$\mathrm{fp} = n_- - \mathrm{tn}$
true positives	hity	tp
false negatives	Type II error	$\mathrm{fn} = n_+ - \mathrm{tp}$
true negative rate	specificity, selectivity	$\mathrm{tnr} = \frac{\mathrm{tn}}{n_-}$
false positive rate	fall-out	$\mathrm{fpr} = \frac{\mathrm{fp}}{n_-} = 1 - \mathrm{tnr}$
true positive rate	sensitivity, recall, hit rate	$\mathrm{tpr} = \frac{\mathrm{tp}}{n_+}$
false negative rate	miss rate	$\mathrm{fnr} = \frac{\mathrm{fn}}{n_+} = 1 - \mathrm{tpr}$
accuracy	—	$\mathrm{acc} = \frac{\mathrm{tp}+\mathrm{tn}}{n}$
balanced accuracy	—	$\mathrm{bacc} = \frac{\mathrm{tpr}+\mathrm{tnr}}{2}$
precision	positive predictive value	$\mathrm{precision} = \frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fp}}$

Table 1.1: Summary of classification metrics derived from confusion matrix.

1.2 Related Problems

The aim of classical binary classification is to separate positive and negative samples with the highest possible accuracy. However, 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. The rest of the samples is irrelevant and therefore the performance on them is not important. Figure 1.3 shows the difference between the standard classifier (classifier 1) that maximizes the accuracy and the classifier that focuses only on the classification at the top (classifier 2). In this particular case, the classifier 2 tries to maximize the number of positive samples that are ranked higher than the worst negative sample, i.e. the negative sample with the highest score. Formally, we can define metric

$$\mathrm{pos@top}(\boldsymbol{s}) = \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} \mathbb{1}_{[s_i \geq \max_{j \in \mathcal{I}_-} \{s_j\}]}.$$

While classifier 1 has good total acc, its pos@top metric is subpar because of the few negative outliers. On the other hand, classifier 2 has worse total acc, but its pos@top metric is extremely good because more than half of the positive samples are ranked higher than the worst negative sample. While classifier 1 selected different thresholds for the acc and pos@top metrics, these thresholds coincide for classifier 2. In the rest of the chapter, we will present three main categories of problems that are closely related to the binary classification but do not focus on optimizing overall performance.

1.2.1 Ranking problems

Add proper introduction to ranking problems

Ranking problems: Ranking problems [15, 16, 17, 18] 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. As an example, we

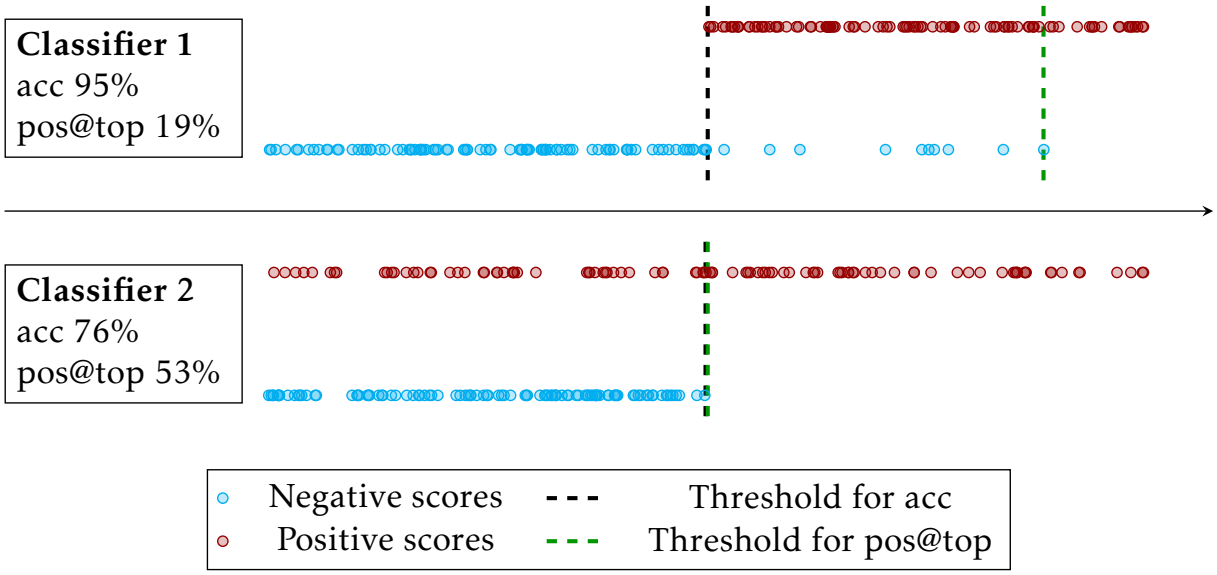


Figure 1.3: Difference between standard classifiers (**Classifier 1**) and classifiers maximizing pos@top metric (**Classifier 2**). While the former has a good total acc, the latter has a pos@top metric.

can mention search engines such as Google, DucDucGo or Yahoo. In such a case, the goal is to provide most relevant results on the first two or three pages. The results on page 50 are usually of no interest to anyone, so it is important to move the most relevant results to the few first pages [19].

The first category of problems that is tightly related to binary classification at the top, is the category of ranking problems. Ranking problems have become very important in many different fields

- **Information retrieval systems:** The goal of the information retrieval systems is to rank documents according to relevance to a given query.
- **Recommendation systems:** The goal is to rank and recommend products based on the user's previous behavior.
-

All the examples above can be formulated as bipartite ranking problem [15, 20, 16], where the goal is to rank the relevant (positive) samples higher than the non-relevant (negative) ones.

Ranking problems [15, 16, 17, 18] 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. As an example, we can mention search engines such as Google, DucDucGo or Yahoo. In such a case, the goal is to provide most relevant results on the first two or three pages. The results on page 50 are usually of no interest to anyone, so it is important to move the most relevant results to the few first pages [19].

A prototypical example is the RankBoost [15] maximizing the area under the ROC curve, the Infinite Push [16] or the p-norm push [17] 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 [18], 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.

Many methods, such as *RankBoost* [15], *Infinite Push* [16] or *p-norm push* [17] employ a pairwise comparison of samples, which makes them infeasible for larger datasets. This was alleviated in *TopPush* [18] 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* [21] by considering the threshold to be the mean of K largest scores of negative samples.

1.2.2 Accuracy at the Top

Add proper introduction to Accuracy a the Top

Accuracy at the Top: Accuracy at the Top [22, 3] is similar to ranking problems. However, instead of ranking the most relevant samples, it only maximizes the number of positive samples (equivalently minimizes the misclassification) above the top τ-quantile of scores. The Accuracy at the Top can be very useful for search engines or in applications where identified samples undergo expensive post-processing such as human evaluation. As an example, we can mention cyber security [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.

Accuracy at the Top (τ-quantile) was formally defined in [22] 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, [23] 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. [22] 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. [24] 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. [25] suggested new formulations for various criteria and argued that they keep desired properties such as convexity. [26] showed that accuracy at the top is maximized by thresholding the posterior probability of the relevant class. The closest approach to our framework is [27, 28], where the authors considered multi-class classification problems, and their goal was to optimize the performance on the top few classes and [29], where the authors implicitly removed some variables and derived an efficient algorithm.

Accuracy at the Top [22] 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 [22], 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.

1.2.3 Hypothesis Testing

Add proper introduction to Hypothesis testing

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.

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.

2

Binary Classiciation at the Top

In the previous chapter, we introduced the general formulation (1.5) and fundamental evaluation matrices for the binary classification problems. Furthermore, in Section 1.2, we introduced three problems closely related to binary classification but focused on specific performance criteria. Namely: *Accuracy at the top* problem, *Ranking problems*, and the problem of *Hypothesis testing*. Even though these problems are usually considered separately, they have one crucial thing in common. All three problems aim to minimize the number of misclassified samples below (or above) a certain threshold. In the rest of the chapter, we focus on this common property. We show that all these problems fall into the following unified framework for binary classification at the top

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \lambda_1 \cdot \mathrm{fp}(\boldsymbol{s}, t) + \lambda_2 \cdot \mathrm{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s}, \boldsymbol{y}), \end{aligned} \tag{2.1}$$

where function $G\colon \mathbb{R}^n \times \{0,1\}^n \to \mathbb{R}$ takes the scores and labels of all samples and computes the decision threshold. The concrete form of the function G that defines the decision threshold depends on the used problem. As we show later in the chapter, all problems mentioned above differ only in the definition of the function G. Note the important distinction from the standard binary classification (1.5): the decision threshold is no longer fixed (as in the case of neural networks) or trained independently (as in SVM) but is a function of scores of all samples. Therefore, the minimization in problem (2.1) is performed only concerning the one variable $\boldsymbol{w}$.

2.1 Surrogate formulation

The objective function of problem (2.1) is a weighted sum of false-positive and false-negative counts. Since these counts are discontinuous due to the presence of the Iverson function (see (1.3)), the whole objective function is discontinuous too. Therefore, problem (2.1) is difficult to solve. One way how to simplify the problem is to derive its continuous approximation. Since the only discontinuous part of the objective function is the Iverson function, the usual approach is to employ a surrogate function to replace it [18, 3].

Notation 2.1: Surrogate function

In the text below, the symbol l denotes any convex non-negative non-decreasing function with $l(0) = 1$. As examples of such function we can mention the hinge loss function or the quadratic hinge loss functions defined as follows

$$l_{\text{hinge}}(s) = \max\{0, 1+s\}, \qquad l_{\text{quadratic}}(s) = (\max\{0, 1+s\})^2.$$

Moreover, parameter $\vartheta > 0$ is used to scale inputs to any surrogate function.

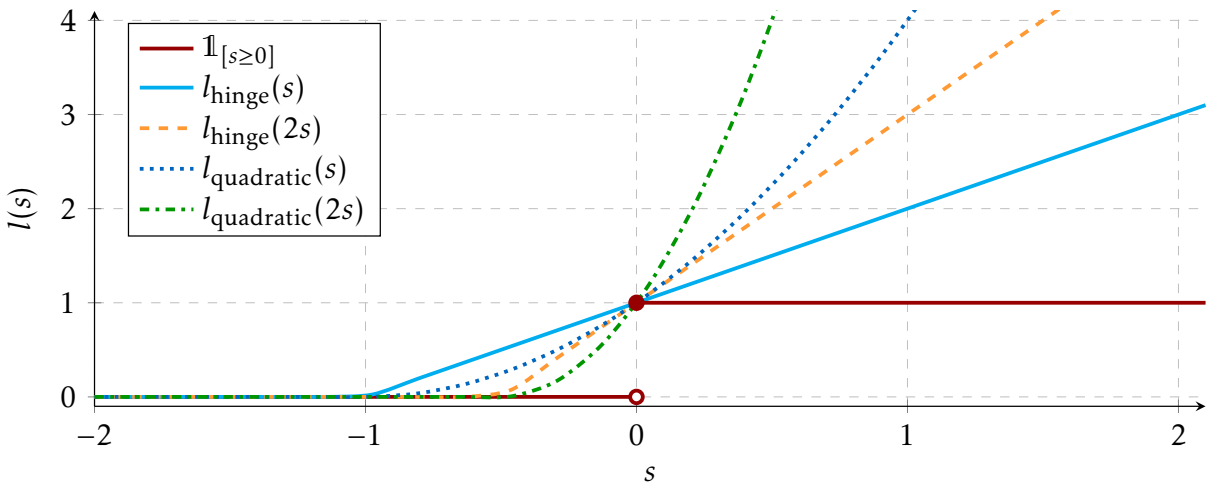


Figure 2.1: Comparison of the approximation quality of the Iverson function using different surrogate functions and scaling parameters.

Notation 2.1 summarizes all assumptions that a proper surrogate function must fulfill and introduces the two most often used surrogate functions: hinge and quadratic hinge loss functions. Moreover, Figure 2.1 compares these two surrogate functions with the Iverson function. It is clear that the surrogate function always provides an upper approximation of the Iverson function. In other words, if a surrogate function l satisfies assumptions from Notation 2.1, then $l(s) \geq \mathbb{1}_{[s\geq 0]}$ holds for any $s \in \mathbb{R}$. Besides that, Figure 2.1 shows how the scaling parameter ϑ affects the approximation quality of the surrogate function. If the scaling parameter is greater than 1, the surrogate function approximates the Iverson function better on interval $(-\infty, 0)$. In the opposite case, the approximation is better on interval $(0, \infty)$. The usual choice of scaling parameter is $\vartheta = 1$, and we used this choice for all surrogate functions used in the objective functions. However, we also use surrogate functions for approximation of the decision threshold. In such a case, the scaling parameter plays a crucial role for some theoretical guaranties, as shown in upcoming chapters.

With a properly defined surrogate function, we can define the surrogate approximation of the objective function of problem (2.1). To follow the notation from the previous chapter, we first replace the Iverson function in (2.1). Using any surrogate function l that satisfies assumptions from Notation 2.1, the true counts (2.1) may be approximated by their surrogate counterparts defined by

$$\begin{aligned} \overline{\text{tp}}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_+} l(s_i - t), & \overline{\text{fn}}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_+} l(t - s_i), \\ \overline{\text{tn}}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_-} l(t - s_i), & \overline{\text{fp}}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_-} l(s_i - t). \end{aligned} \tag{2.2}$$

Since the surrogate function provides upper approximation of the Iverson function, the surrogate counts (2.2) provide upper approximations of the true counts (1.3). By replacing the true counts in the objective function of (2.1) with their surrogate counterparts and adding a regularization for better numerical stability, we get

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \lambda_1 \cdot \overline{\text{fp}}(\boldsymbol{s}, t) + \lambda_2 \cdot \overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s}, \boldsymbol{y}). \end{aligned} \tag{2.3}$$

The resulting objective function is continuous, and therefore the problem is easier to solve than the original problem (2.1). No additional theoretical properties can be derived without

knowing the concrete form of model f and function G. Therefore, the rest of the chapter is dedicated to problems that fall into the general framework (2.3) and concrete form of G for such problems. More precisely, we focus on the three problems introduced in Section 1.2 and show how to rewrite them to our general formulation (2.3). Most of these problems are defined originally only for the linear model since this choice allows to derive nice theoretical properties and efficient solving algorithms. However, this chapter focuses on the problem formulation itself rather than on how to solve it. Therefore for all problems, we derive their version with general model f. The discussion of the theoretical properties for specific forms of f is provided in Chapter 3, 4, and 5.

Notation 2.2: Classification scores

In Notation 1.2, we defined vector $\boldsymbol{s} \in \mathbb{R}^n$ of scores of all samples with components defined for any $i \in \mathcal{I}$ as

$$s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I},$$

where $f : \mathbb{R}^d \to \mathbb{R}$ represents an arbitrary model. To simplify the upcoming sections, we define a sorted version of vector $\boldsymbol{s}$ with decreasing components and denote it as $\boldsymbol{s}_{[\cdot]}$. It means that components of $\boldsymbol{s}_{[\cdot]}$ fulfill

$$s_{[1]} \geq s_{[2]} \geq \cdots \geq s_{[n-1]} \geq s_{[n]}.$$

Moreover, we denote negative samples as $\boldsymbol{x}^-$ and positive samples as $\boldsymbol{x}^+$. Finally, we define vectors $\boldsymbol{s}^- \in \mathbb{R}^{n_-}$, $\boldsymbol{s}^+ \in \mathbb{R}^{n_+}$ of scores of all positive and negative samples with components defined as

$$s_j^- = f(\boldsymbol{x}_j^-; \boldsymbol{w}), \quad j = 1, 2, \ldots, n_-,$$
$$s_i^+ = f(\boldsymbol{x}_i^+; \boldsymbol{w}), \quad i = 1, 2, \ldots, n_+,$$

and their sorted versions $\boldsymbol{s}_{[\cdot]}^-$, $\boldsymbol{s}_{[\cdot]}^+$ with decreasing components.

2.2 Ranking problems

The first category of problems from Section 1.2 is a category of ranking problems. The general goal of problems from this category is to rank positive (relevant) samples higher than negative ones. That can be achieved in many different ways, but we focus only on the problems that concentrate on the high-ranked negative samples and try to push as many positive samples as possible above them. The simplest case is when the goal is to maximize the number of positive samples above the worst negative. Since the worst negative sample is the negative sample with the highest classification score, the decision threshold for such a case is the highest score corresponding to the negative sample. Then the aim is to maximize the number of true-positive samples above this threshold or, equivalently, minimize the number of false-negative negative below it, which may be written as

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{n_+} \operatorname{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = s_{[1]}^- \end{aligned} \tag{2.4}$$

Note that the decision threshold t in the previous formulation is a function of classification scores. Therefore, the formulation is just a special case of the general formulation (2.1) for $\lambda_1 = 0$ and $\lambda_2 = 1/n_+$. The authors in [18] proposed an efficient method to solve formulation (2.4) and called it *TopPush*. They replaced the true counts in the objective function of (2.4) with its

surrogate counterpart in the same way as we did in Section 2.1. The resulting formulation has the following form

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \bar{s}_{[1]}, \end{aligned} \tag{2.5}$$

which again falls into our framework (2.3). To stress the origin of this formulation, we denote it as *TopPush*. Unfortunately, *TopPush* formulation can be very sensitive to outliers, especially when the linear model is used, as shown in 3.3. To robustify the formulation, we follow the idea presented in [27] and replace the highest negative score by the mean of K highest negative scores. The resulting formulation is as follows

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \sum_{i=1}^{K} \bar{s}_{[i]}, \end{aligned} \tag{2.6}$$

To emphasize the similarity with the *TopPush*, we call this formulation *TopPushK*. It is also possible to use the value of K-th highest negative score as the threshold. Such a choice may be advantageous in some cases, and we will discuss it in Chapter 5. For now, we will stick to the formulation that uses the mean since it will allow us to derive some crucial theoretical properties, as shown in Section 3.1.

2.3 Accuracy at the Top

The second problem from Section 1.2 is the problem of Accuracy at the Top [22]. This problem aims to find an ordering of samples so that samples whose scores are among the top τ-quantile are as relevant as possible. The top τ-quantile of all scores is defined by

$$t_1(\boldsymbol{s}) = \max\left\{ t \;\middle|\; \frac{1}{n}\sum_{i \in \mathcal{I}} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}. \tag{2.7}$$

All relevant samples should be ranked above the quantile t_1 and all irrelevant samples below the quantile t_1 in an ideal case. Thus, the main difference to the ranking problems is that the problem of Accuracy at the Top considers both classification errors and does not focus only on false-negative samples. The original formulation [22] considers a balanced dataset with the same number of positive and negative samples. Paper [3] reformulated the problem for the unbalanced dataset and derived the following formulation

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{n_-}\text{fp}(\boldsymbol{s}, t) + \frac{1}{n_+}\text{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \max\left\{ t \;\middle|\; \frac{1}{n}\sum_{i \in \mathcal{I}} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}. \end{aligned} \tag{2.8}$$

This formulation already falls into our framework (2.1) for $\lambda_1 = 1/n_-$ and $\lambda_2 = 1/n_+$. Moreover, the authors of [22, 3] used the same surrogate trick to get rid of the discontinuous objec-

tive function, as we used in Section 2.1. Thus, by replacing replaces false-positive and false-negative counts in the objective function with their surrogate counterparts we get

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_-}\overline{\text{fp}}(\boldsymbol{s}, t) + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \max\left\{ t \;\middle|\; \frac{1}{n}\sum_{i \in \mathcal{I}} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}. \end{aligned} \tag{2.9}$$

This formulation falls into our framework (2.3) for $\lambda_1 = 1/n_-$ and $\lambda_2 = 1/n_+$. Even though the original formulation is presented in [22], we denote the previous formulation as *Grill* based on the name of the first author of [3]. There are two reasons for that. The first one is that we used an unbalanced dataset as in [3]. The second one is that we use an algorithm proposed in [3] for numerical experiments since the one from [22] is suitable only for a small dataset.

The *Grill* formulation (2.9) is still challenging to solve due to the form of the decision threshold (2.7). The authors of [22] removed the necessity to handle the difficult quantile constraint by setting quantile as one of the samples and solving n independent problems. However, such an approach is infeasible for a large number of samples. The authors of (2.9) proposed the projected gradient descent method, where after each gradient step, the quantile is recomputed. This approach is suitable for large data but lacks theoretical guarantees. In the following text, we propose two approximations of the true quantile (2.7) that can be used to simplify formulation (2.9). The first one is a simple approximation by the mean of $n\tau$ highest scores

$$t_2(\boldsymbol{s}) = \frac{1}{n\tau}\sum_{i=1}^{n\tau} s_{[i]}. \tag{2.10}$$

where for simplicity we assume, that $n\tau$ is an integer. The main purpose of (2.10) is to provide a convex approximation of the non-convex quantile (2.7). In fact, it is known is that it is the tightest convex approximation of (2.7). Putting (2.10) into the constraint results in the following problem, which we call *TopMeanK*

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \frac{1}{K}\sum_{i=1}^{K} s_{[i]}, \end{aligned} \tag{2.11}$$

where $K = n\tau$. Besides changing the form of the decision threshold, we also simplified the objective function. This change allows preserving the convexity of the formulation for the linear model as shown in Section 3.1. The resulting formulation is very similar to the *TopPushK* formulation from the previous section. The only difference is that the threshold for *TopMeanK* is computed from scores of all samples and not only from the negative ones.

The second option how to approximate the true quantile is to use surrogate counterparts to replace true counts in (2.7) and solve the following equality

$$t_3(\boldsymbol{s}) \quad \text{solves} \quad \frac{1}{n}\sum_{i \in \mathcal{I}} l(\vartheta(s_i - t)) = \tau, \tag{2.12}$$

where $\vartheta > 0$ is scaling parameter. Since this threshold uses the surrogate approximation, we denote it as surrogate top τ-quantile. We get the following formulation by replacing the true

quantile in the constrain and simplifying the objective function

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t \quad \text{solves} \quad \frac{1}{n}\sum_{i \in \mathcal{I}} l(\vartheta(s_i - t)) = \tau. \end{aligned} \tag{2.13}$$

This formulation also used only false negatives in the objective to preserve the convexity for the linear model. In such a case, the formulation is easily solvable due to the convexity and requires almost no tuning. Together with the fact that formulation (2.13) provides a good approximation to the Accuracy at the Top problem, we named it *Pat&Mat* (Precision At the Top & Mostly Automated Tuning).

2.4 Neyman-Pearson problem

The last problem that we introduce in Section 1.2 is the Neyman-Pearson problem, which is closely related to hypothesis testing. The hypothesis testing operates with null H_0 and alternative H_1 hypotheses. The goal is to decide to either reject the null hypotheses in favor of the alternative or not reject it. Since this problem is binary, two possible errors can occur. Type I occurs when H_0 is true but is rejected, and Type II error happens when H_0 is false but fails to be rejected. The Neyman-Pearson problem minimizes Type II error while keeping Type I error smaller than some predefined bound. Using our notation for the Neyman-Pearson problem, the null hypothesis H_0 states that sample $\boldsymbol{x}$ has a negative label. Then Type I error occurs when the sample is false-positive, while Type II error when the sample is false-negative, see Table 1.1. In other words, Type II corresponds to the false-negative rate, and Type I error false-positive rate. Therefore, if the bound on the Type I error is τ, we may write this as

$$t_1^{\text{NP}}(\boldsymbol{s}) = \max\left\{ t \;\middle|\; \frac{1}{n_-}\sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}. \tag{2.14}$$

Note that we only count the false-positive samples in (2.14) instead of counting all positives in (2.7). Then, we may write the Neyman-Pearson problem as

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{n_+}\text{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \max\left\{ t \;\middle|\; \frac{1}{n_-}\sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}. \end{aligned} \tag{2.15}$$

This problem falls within our framework for (2.1) for $\lambda_1 = 0$ and $\lambda_2 = 1/n_+$. Moreover, formulation (2.15) differs from (2.8) by two things. The first one is the absence of a false-positive rate in the objective function. The second one is that the threshold is computed from negative samples only. Therefore, we can use the same techniques to approximate both objective function and the decision threshold.

To follow the previous section, we first derive the Neyman-Pearson alternative to the *Grill* formulation. We need to add false-positive counts in the objective function to do that. Moreover, we also need to replace true counts with their surrogate counterparts and add the regu-

larization. The resulting formulation is as follows

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_-}\overline{\text{fp}}(\boldsymbol{s},t) + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s},t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i;\boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \max\left\{ t \;\middle|\; \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}. \end{aligned} \tag{2.16}$$

We denote this formulation as *Grill-NP* to emphasize the relation with the original *Grill* formulation and the Neyman-Pearson problem.

The second formulation (2.11) from the previous section, uses mean of $n\tau$ highest scores to approximate true quantile (2.7). In the same way, we can approximate true quantile (2.14) by the mean of $n_-\tau$ highest of scores corresponding to the negative samples

$$t_2^{\text{NP}}(\boldsymbol{s}) = \frac{1}{n_-\tau}\sum_{i=1}^{n_-\tau} s^-_{[i]}. \tag{2.17}$$

For simplicity, we again assume that $n_-\tau$ is an integer. Putting (2.17) into the constraint results in the Neyman-Pearson alternative to *TopMeanK* defined as

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s},t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i;\boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \frac{1}{n_-\tau}\sum_{i=1}^{n_-\tau} s^-_{[i]}. \end{aligned} \tag{2.18}$$

This problem already appeared in [30] under the name τ-*FPL*. Formulation (2.18) has almost the same form as formulation (2.11). 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 that differ only in the value of one parameter. This slight difference shows a close relationship between the ranking problems and the Neyman-Pearson problem and the need for a unified theory to handle these problems.

The last formulation (2.13) from the previous sections uses the surrogate approximation of the true quantile (2.7). The surrogate approximation of the true quantile (2.14) reads

$$t_3^{\text{NP}}(\boldsymbol{s}) \quad \text{solves} \quad \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} l(\vartheta(s_i - t)) = \tau. \tag{2.19}$$

Putting (2.19) into the constraint results in the Neyman-Pearson alternative to *Pat&Mat* in the following form

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s},t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i;\boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t \text{solves} \quad \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} l(\vartheta(s_i - t)) = \tau, \end{aligned} \tag{2.20}$$

We call this formulation *Pat&Mat-NP* to stress the similarity with *Pat&Mat*. The only difference between these two formulations is that only negative samples are involved in computing the decision threshold for *Pat&Mat-NP*, while *Pat&Mat* uses all samples.

2.5 Summary

Add summary of the framework and formulations that fall into the framework

Formulation	Label	Source	Ours	λ_1	λ_2	Threshold
TopPush	(2.5)	[18]	✗	0	$\frac{1}{n_+}$	$s^-_{[1]}$
TopPushK	(2.6)	[21]	✓	0	$\frac{1}{n_+}$	$\sum_{i=1}^{K} s^-_{[i]}$
Grill	(2.9)	[3]	✗	$\frac{1}{n_-}$	$\frac{1}{n_+}$	$\max\left\{t \mid \frac{1}{n}\sum_{i\in\mathcal{I}} \mathbb{1}_{[s_i\geq t]} \geq \tau\right\}$
TopMeanK	(2.11)	—	✗	0	$\frac{1}{n_+}$	$\frac{1}{K}\sum_{i=1}^{K} s_{[i]}$
Pat&Mat	(2.13)	[21]	✓	0	$\frac{1}{n_+}$	$\frac{1}{n}\sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) = \tau$
Grill-NP	(2.16)	—	✗	$\frac{1}{n_-}$	$\frac{1}{n_+}$	$\max\left\{t \mid \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} \mathbb{1}_{[s_i\geq t]} \geq \tau\right\}$
τ-*FPL*	(2.18)	[30]	✗	0	$\frac{1}{n_+}$	$\frac{1}{n_-\tau}\sum_{i=1}^{n_-\tau} s^-_{[i]}$
Pat&Mat-NP	(2.20)	[21]	✓	0	$\frac{1}{n_+}$	$\frac{1}{n_-}\sum_{i\in\mathcal{I}_-} l(\vartheta(s_i - t)) = \tau$

Table 2.1: Summary of problem fomrulations that fall in the framework (2.3). Column **Formulation** shows the name of the formulation that we use in this work. Column **Label** represents the label of the formulation in this text. Column **Source** is the citation of the work where the formulation was introduced. Column **Ours** shows whether the formulation was introduced in any of our previous papers. The last three columns show the values of parameters λ_1, λ_2 and the form of the decision threshold for framework (2.3).

3

Linear Classification at the Top

In this chapter, we focus on the special case when the model f is linear

$$f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x},$$

where $\boldsymbol{w} \in \mathbb{R}^d$ is the normal vector to the separating hyperplane. In such a case, the framework (2.3) simplifies into the following form

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \lambda_1 \cdot \overline{\text{fp}}(\boldsymbol{s},t) + \lambda_2 \cdot \overline{\text{fn}}(\boldsymbol{s},t) \\ \text{subject to} \quad & s_i = \boldsymbol{w}^\top \boldsymbol{x}_i, \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s},\boldsymbol{y}). \end{aligned}$$

In this chapter, we provide a theoretical analysis of the unified framework from Chapter 2 with linear classifier. We consider purely the problem *formulations* and not individual *algorithms* which specify how to solve these formulations. The overview of all formulations that falls into the framework (2.3) is in Table 2.1. We focus mainly on the following desirable properties:

- *Convexity* implies a guaranteed convergence for many optimization algorithms or their better convergence rates [31].
- *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 Chapter 2 and results in the situation where the separating hyperplane is degenerate and does not actually exist.

For a nicer flow of text, we postpone the proofs to Appendix 3. Moreover, we show the results only for formulations from Section 2.2 and 2.3. The results for methods from Section 2.4 are identical to the one for methods in Section 2.3.

3.1 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. Before we present any

results, we recall the form of the decision threshold from Section 2.3

$$t_1(\boldsymbol{w}) = \max\left\{ t \;\middle|\; \frac{1}{n}\sum_{i\in\mathcal{I}} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}$$
$$t_2(\boldsymbol{w}) = \frac{1}{K}\sum_{i=1}^{K} s_{[i]}$$
$$t_3(\boldsymbol{w}) \quad \text{solves} \quad \frac{1}{n}\sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) = \tau,$$

where we use Notation 2.2. Note that we denote the thresholds as functions of weights. This is true, since the scores $\boldsymbol{s}$ depends on the weights $\boldsymbol{w}$. The first result is summarized in the following proposition.

Proposition 3.1

Thresholds t_2 from (2.10) and t_3 from (2.12) are convex functions of the weights $\boldsymbol{w}$. The threshold function t_1 from (2.7) is non-convex.

The proposition says, that *Grill* uses non-convex threshold with respect to weights $\boldsymbol{w}$ while *TopMeanK*, *Pat&Mat* use the convex ones. Moreover, since τ*-FPL* and *TopPushK* formulations use almost the same threshold as *TopMeanK*, but computed only from negative scores, the resulting threshold is also convex function of weights. The same holds for formulations *Pat&Mat* and *Pat&Mat-NP*. Finally, the threshold for *TopPush* formulation is convex since maximum is convex function. The next theorem shows which formulations are convex.

Theorem 3.2

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

$$L(\boldsymbol{w}) = \overline{\mathrm{fn}}(\boldsymbol{s}, t)$$

is convex.

While the proof of Theorem 3.2 is simple, it points to the necessity of considering only false-negatives in the objective of the formulations in Chapter 2. 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.

3.2 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 3.3

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

$$\nabla t_3(\boldsymbol{w}) = \frac{\sum_{i\in\mathcal{I}} l'(\vartheta(\boldsymbol{w}^\top \boldsymbol{x}_i - t_3(\boldsymbol{w})))\boldsymbol{x}_i}{\sum_{i\in\mathcal{I}} l'(\vartheta(\boldsymbol{w}^\top \boldsymbol{x}_i - t_3(\boldsymbol{w})))}.$$

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 3.4.

3.3 Stability

We first provide a simple example and show that many formulations from Table 2.1 are degenerate for it. Then we analyze general conditions under which this degenerate behaviour happens.

Example 3.4: Degenerate Behaviour

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 3.1 (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 $\boldsymbol{w} = (1,0)$.

Table 3.1 shows the threshold t and the objective value L for two points $\boldsymbol{w}_0 = (0,0)$ and $\boldsymbol{w}_0 = (1,0)$. These two points are both important: $\boldsymbol{w}_0$ does not generate any separating hyperplane, while $\boldsymbol{w}_1$ generates the optimal separating hyperplane. We show the precise computation in Appendix A.3. Since the dataset is perfectly separable by $\boldsymbol{w}_0$, we expect that $\boldsymbol{w}_0$ provides a lower objective than $\boldsymbol{w}_0$. By highlithing the better objective in Table 3.1 by green, we see that this did not happen for *TopPush* and *TopMeanK*. It can be shown that $\boldsymbol{w}_0$ 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 as shown in Figure 3.1 (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.

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 $\boldsymbol{w} = \boldsymbol{0}$. This is highly undesirable since $\boldsymbol{w}$ is the normal vector to the separating hyperplane and the zero vector provides no information. In the rest of the section, we analyze when this situation happens. The first result states that if the decision threshold $t = t(\boldsymbol{w})$ 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 3.5

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

$$t(\boldsymbol{w}) \geq \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} \boldsymbol{w}^\top \boldsymbol{x}_i, \tag{3.1}$$

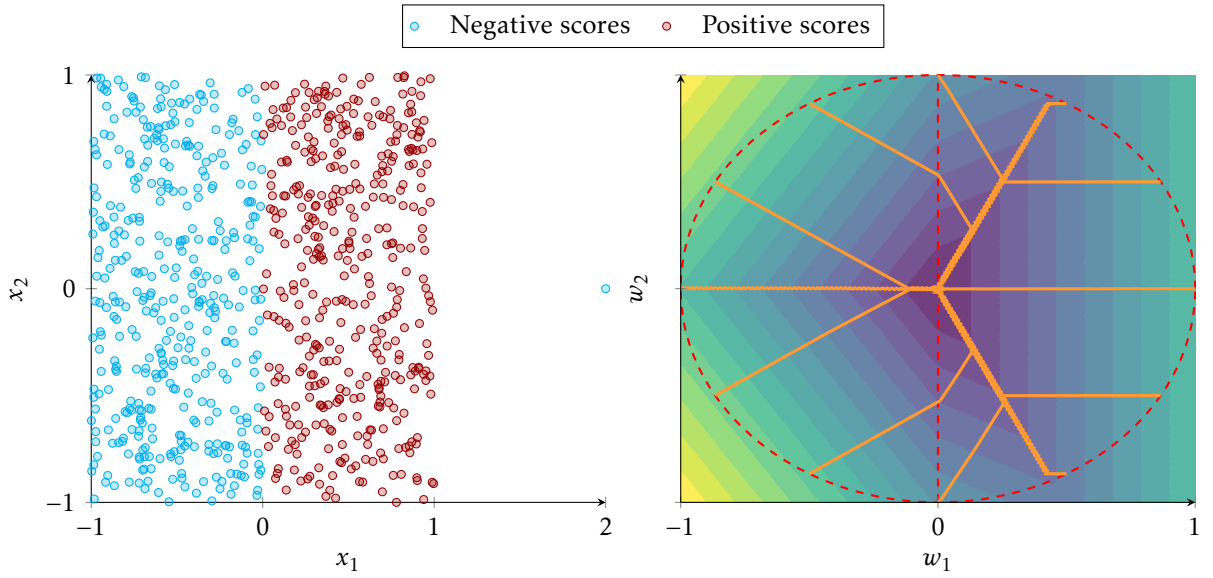


Figure 3.1: Left: distribution of positive (red circles) and negative samples (blue circles) for the example from Example 3.4 Right: contour plot of the objective function value for *TopPush* and its convergence (orange lines) to the zero vector from 12 initial points.

Name	**Label**	$\boldsymbol{w}_0 = (0,0)$		$\boldsymbol{w}_0 = (1,0)$	
		t	L	t	L
TopPush	(2.5)	0	1	2	2.5
TopPushK	(2.6)	0	1	$\frac{2}{K}$	$0.5 + \frac{2}{K}$
Grill	(2.9)	0	2	$1 - 2\tau$	$1.5 + 2\tau(1-\tau)$
TopMeanK	(2.11)	0	1	$1 - \tau$	$1.5 - \tau$
Pat&Mat	(2.13)	$\frac{1}{\vartheta}(1-\tau)$	$1 + \frac{1}{\vartheta}(1-\tau)$	$\frac{1}{\vartheta}(1-\tau)$	$0.5 + \frac{1}{\vartheta}(1-\tau)$

Table 3.1: Comparison of formulations on the very simple problem from Section 3.3. Two formulations have the global minimum (denoted by green color) at $\boldsymbol{w}_0 = (0,0)$ which does not generate any separating hyperplane. The optimal separating hyperplane is generated by $\boldsymbol{w}_1 = (1,0)$.

then $L(\mathbf{0}) \le L(\boldsymbol{w})$. Specifically, using notation 2.2 we get the following implications

$$\begin{aligned}
s^-_{[1]} \ge \frac{1}{n_+}\sum_{i=1}^{n_+} s^+_i &\implies L(\mathbf{0}) \le L(\boldsymbol{w}) \text{ for } \textit{TopPush},\\
\frac{1}{K}\sum_{i=1}^{K} s^-_{[i]} \ge \frac{1}{n_+}\sum_{i=1}^{n_+} s^+_i &\implies L(\mathbf{0}) \le L(\boldsymbol{w}) \text{ for } \textit{TopPushK},\\
\frac{1}{K}\sum_{i=1}^{K} s_{[i]} \le \sum_{i=1}^{n_+} s^+_i &\implies L(\mathbf{0}) \le L(\boldsymbol{w}) \text{ for } \textit{TopMeanK},\\
\frac{1}{n_-\tau}\sum_{i=1}^{n_-\tau} s^-_{[i]} \ge \frac{1}{n_+}\sum_{i=1}^{n_+} s^+_i &\implies L(\mathbf{0}) \le L(\boldsymbol{w}) \text{ for } \tau\textit{-FPL}.
\end{aligned} \tag{3.2}$$

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 3.6

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 3.7

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

The proof of Theorem 3.5 employs the fact that all formulations in the theorem statement have only false-negatives in the objective. If $\boldsymbol{w}_0 = \mathbf{0}$, then $\boldsymbol{w}_0^\top \boldsymbol{x}_i = 0$ for all $i \in \mathcal{I}$, 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 as shown in Section 3.1.
- Move the threshold away from zero even when all scores $\boldsymbol{s}$ 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 3.8

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

$$\frac{1}{n_+}\sum_{i\in\mathcal{I}_+} \boldsymbol{w}^\top \boldsymbol{x}_i > \frac{1}{n_-}\sum_{j\in\mathcal{I}_-} \boldsymbol{w}^\top \boldsymbol{x}_j. \tag{3.3}$$

Then there is a scaling parameter ϑ_0 for the surrogate top τ-quantile (2.12) such that $L(\boldsymbol{w}) < L(\mathbf{0})$ for all $\vartheta \in (0, \vartheta_0)$.

This theorem shed some light on the behaviour of the formulations. Theorem 3.5 states that the stability of τ-*FPL* requires

$$\frac{1}{n_-\tau}\sum_{i=1}^{n_-\tau} s^-_{[i]} < \frac{1}{n_+}\sum_{i=1}^{n_+} s^+_i, \tag{3.4}$$

while Theorem 3.8 states that the stability of *Pat&Mat-NP* is ensured by

$$\frac{1}{n_-}\sum_{i=1}^{n_-} s^-_{[i]} < \frac{1}{n_+}\sum_{i=1}^{n_+} s^+_i. \tag{3.5}$$

The right-hand sides of (3.4) and (3.5) are the same, while the left-hand side of (3.5) is always smaller than the left-hand side of (3.4). 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 $\boldsymbol{s}$ may be interpreted as performance. Then formula (3.4) 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 (3.5) 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.

3.4 Convergence of stochastic gradient descent

The previous section analyzed the formulations from Chapter 2 but did not consider any optimization algorithms. In this section, we show a basic version of the stochastic 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.

Many optimization algorithms for solving the formulations from Chapter 2 use primal-dual or purely dual formulations. [25] introduced dual variables and used alternating optimization to the resulting min-max problem. [18] and [30] dualized the problem and solved it with the steepest gradient ascent. [32] followed the same path but added kernels to handle nonlinearity. We follow the ideas of [29] and [33] and solve the problems directly in their primal formulations. Therefore, even though we use the same formulation for *TopPush* as [18] or for τ-*FPL* as [30], our solution process is different. However, due to convexity, both algorithms should converge to the same point.

The decision variables in (2.3) 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

$$\begin{aligned} L(\boldsymbol{w}) &= \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l(t(\boldsymbol{w}) - \boldsymbol{w}^\top \boldsymbol{x}_i) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2, \\ \nabla L(\boldsymbol{w}) &= \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l'(t(\boldsymbol{w}) - \boldsymbol{w}^\top \boldsymbol{x}_i)(\nabla t(\boldsymbol{w}) - \boldsymbol{x}_i) + \lambda \boldsymbol{w}. \end{aligned} \tag{3.6}$$

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 3.3 shows the computation for $\nabla t_3(\boldsymbol{w})$. Moreover, Appendix A.5 provides an efficient

computation method for $t_3(\boldsymbol{w})$. Having derivative (3.6), 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 (3.6) by a mean over the minibatch.

3.4.1 SGD: Convergent for *Pat&Mat* and *Pat&Mat-NP*

For the convergence proof, we need differentiability which is due to Theorem 3.3 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 [34].

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

$$s_i^k = \begin{cases} \boldsymbol{x}_i^\top \boldsymbol{w}^k & \text{for all } i \in \mathcal{I}^k_{\text{mb}}, \\ s_i^{k-1} & \text{otherwise.} \end{cases} \tag{3.7}$$

We keep scores from previous minibatches intact. We use Appendix A.5 to compute the surrogate quantile t^k as the unique solution of

$$\sum_{i \in \mathcal{I}} l\Big(\vartheta(s_i^k - t^k)\Big) = n\tau. \tag{3.8}$$

This is an approximation of the surrogate quantile $t(\boldsymbol{w}^k)$ from (2.12). The only difference from the true value $t(\boldsymbol{w}^k)$ is that we use delayed scores. Then we introduce artificial variable

$$\boldsymbol{a}^k = \sum_{i \in \mathcal{I}^k_{\text{mb}}} l'\Big(\vartheta(s_i^k - t^k)\Big)\boldsymbol{x}_i. \tag{3.9}$$

Finally, we approximate the derivative $\nabla f(\boldsymbol{w}^k)$ from (3.6) by

$$g(\boldsymbol{w}^k) = \frac{1}{n^k_{\text{mb},+}} \sum_{i \in \mathcal{I}^k_{\text{mb},+}} l'(t^k - s_i^k)(\nabla t^k - \boldsymbol{x}_i), \tag{3.10}$$

where ∇t^k is an approximation of $\nabla t(\boldsymbol{w}^k)$ from Theorem 3.3 defined by

$$\nabla t^k = \frac{\boldsymbol{a}^k + \boldsymbol{a}^{k-1} + \cdots + \boldsymbol{a}^{k-m+1}}{\sum_{i \in \mathcal{I}} l'\Big(\vartheta(s_i^k - t^k)\Big)}. \tag{3.11}$$

A perhaps more straightforward possibility would be to consider only $\boldsymbol{a}^k$ in the numerator of (3.11). However, choice (3.11) 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 $\mathcal{I}^k_{\text{mb}}$. 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.

Theorem 3.9

Consider the *Pat&Mat* or *Pat&Mat-NP* formulation, stepsizes $\alpha^k = \alpha^0/k+1$ and piecewise disjoint minibatches $\mathcal{I}^1_{\text{mb}}$, $\mathcal{I}^2_{\text{mb}}$,…, $\mathcal{I}^m_{\text{mb}}$ which cycle periodically $\mathcal{I}^{k+m}_{\text{mb}} = \mathcal{I}^k_{\text{mb}}$. If l is the smoothened (Huberized) hinge function, then Algorithm 1 converges to the global minimum of (2.13).

Algorithm 1 Stochastic gradient descent for maximizing accuracy at the top

Require: Dataset $\mathcal{D}$, Minibatches $\mathcal{I}_{\text{mb}}^1, \mathcal{I}_{\text{mb}}^2, \ldots, \mathcal{I}_{\text{mb}}^m$, Stepsize α^k

Initialize weights $\boldsymbol{w}^0$
for $k = 0, 1, \ldots$ **do**
  Select a minibatch $\mathcal{I}_{\text{mb}}^k$
  Compute s_i^k for all $i \in \mathcal{I}_{\text{mb}}^k$ according to (3.7)
  Compute t^k according to (3.8)
  Compute $\boldsymbol{a}^k$ according to (3.9)
  Compute ∇t^k according to (3.11)
  Compute $g(\boldsymbol{w}^k)$ according to (3.10)
  Set $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k - \alpha^k g(\boldsymbol{w}^k)$
end for

3.5 Summary

We provide a summary of the obtained results in Table 3.2. 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.

Name	Definition	Hyperpars	Convex	Differentiable	Stable
TopPush	(2.5)	—	✓	✗	✗
TopPushK	(2.6)	K	✓	✗	✗
Grill	(2.9)	τ	✗	✗	✓
Pat&Mat	(2.13)	τ, ϑ	✓	✓	✓
TopMeanK	(2.11)	τ	✓	✗	✗
Grill-NP	(2.16)	τ	✗	✗	✓
Pat&Mat-NP	(2.20)	τ, ϑ	✓	✓	✓
τ-FPL	(2.18)	τ	✓	✗	✗

Table 3.2: Summary of the formulations from Chapter 2. The table shows their definition label, the hyperparameters, whether the formulation is differentiable, convex and stable (in the sense of having problems with $\boldsymbol{w} = \mathbf{0}$).

A similar comparison is performed in Figure 3.2. Methods in green and grey are convex, while formulations in white are non-convex. Based on Theorem 3.5, 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.4. This complies with Corollaries 3.6 and 3.7 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 3.5, it is more stable due to the discussion after Theorem 3.8.

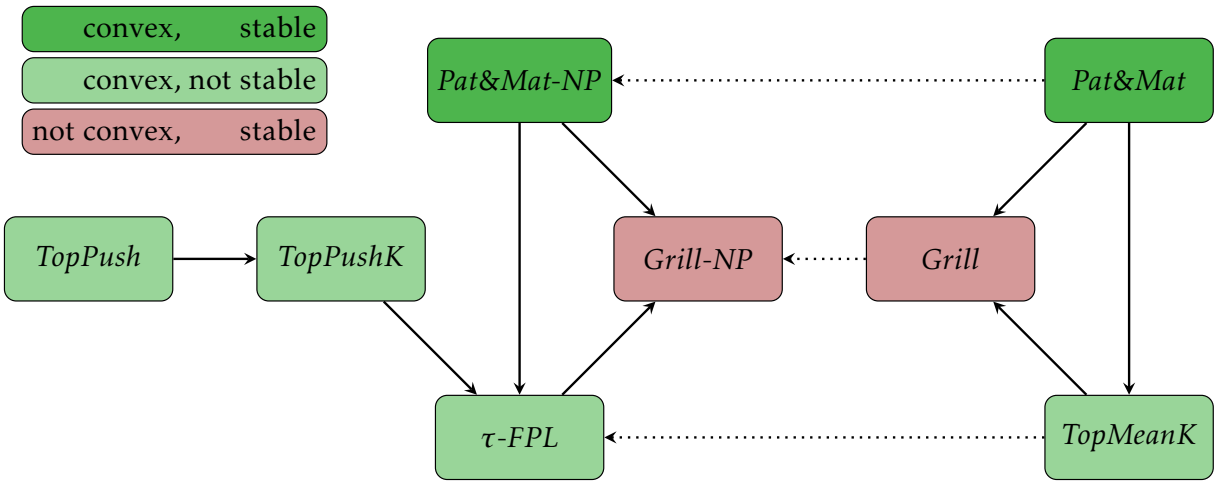


Figure 3.2: Summary of the formulations from Chapter 2. Methods in green are convex, while formulations in red are non-convex. Methods in light green are vulnerable to have the global minimum at $\boldsymbol{w} = 0$. Full (dotted) arrow pointing from one formulation to another show that the latter formulation has always (usually) smaller threshold.

4

Non-linear Classification at the Top

In the Chapter 2, we introduced general framework for binary classification at the top. Moreover, we showed that several problem classes, which were considered as separate problems so far, fit into the framework. As an example we can mention ranking problems of hypothesis testing. Summary of all formulations is in Table 2.1. In the Chapter 3 disscused special case, when the linear classifier is used. In such a case, many of the formulations from Table 2.1 have nice theoretical properties such as convexity or differentiability. However, as many problems are not linearly separable, nonlinear classifiers are needed. In this chapter, 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 [31] and to employ the kernels method [35]. In this chapter, 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 [36]. For a review of other approaches see [37, 38].

4.1 Derivation of dual problems

In this section, we derive dual forms of formulations from Table 2.1. Since many of these formulations are very similar, we divide them into two families. The first one is a family of *TopPushK* formulations that consists *TopPush*, *TopPushK*, *TopMeanK* and τ-*FPL* formulations. All these formulations use false-negative rate as an objective function and the decision threshold is a mean of K largest scores of all or negative samples. The second family is a family of *Pat&Mat* formulations that consists *Pat&Mat* and *Pat&Mat* formulations. Also these two formulations use false-negative rate as an objective function, but the decision threshold is a surrogate approximation of top τ-quantile of scores of all or negative samples. In other words, we have two families that share the same objective function and the form of the decision threshold, even thogh the decision threshold may be computed from different samples.

Notation 4.1: Kernel Matrix

To simplify the future notation, we use matrix $\mathbb{X}$ of all samples with rows defined for all $i \in \mathcal{I}$ as

$$\mathbb{X}_{i,\bullet} = \boldsymbol{x}_i^\top.$$

In other words, each row of $\mathbb{X}$ represents one sample. Similarly, we defined matrices $\mathbb{X}^+$, $\mathbb{X}^-$ of all negative and positive samples with rows defined as

$$\mathbb{X}^-_{i,\bullet} = \boldsymbol{x}_i^\top \quad i = 1, , 2, \ldots, n^-,$$
$$\mathbb{X}^+_{i,\bullet} = \boldsymbol{x}_i^\top \quad i = 1, , 2, \ldots, n^+.$$

Moreover, for *TopPush*, *TopPushK*, τ-*FPL* and *Pat&Mat-NP* formulations we define the positive semidefinite kernel matrix $\mathbb{K}^-$ as

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

Simlarly, for *TopMeanK* and *Pat&Mat* formulations we define the positive semidefinite kernel matrix $\mathbb{K}^\pm$ as

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

4.1.1 Family of *TopPushK* formulations

As we mentioned before, the first family is a family of *TopPushK* formulations that consists *TopPush*, *TopPushK*, *TopMeanK* and τ-*FPL* formulations. All these formulations can be written as follows

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in\mathcal{I}_+} l(t - \boldsymbol{w}^\top \boldsymbol{x}_i) \\ \text{subject to} \quad & \tilde{s}_j = \boldsymbol{w}^\top \boldsymbol{x}_j, \quad j \in \tilde{\mathcal{I}}, \\ & t = \sum_{j=1}^{K} \tilde{s}_{[j]}, \end{aligned} \tag{4.1}$$

where $C \in \mathbb{R}$ is a constant and $\tilde{\mathcal{I}}$ and K is defined as follows

$$\tilde{\mathcal{I}} = \begin{cases} \mathcal{I} & \text{for } \textit{TopMeanK}, \\ \mathcal{I}_- & \text{otherwise.} \end{cases} \qquad K = \begin{cases} 1 & \text{for } \textit{TopPush}, \\ n\tau & \text{for } \textit{TopMeanK}, \\ n_-\tau & \text{for } \tau\textit{-FPL}. \end{cases}$$

It means, that for *TopMeanK*, the threshold is computed from all samples and otherwise only from negative ones. Also note, that we use linear classifier and we also use this alternative formulation with constant C, since it is more similar to the standard SVM. The following theorem show the dual formulation of (4.1). To keep the readability as simple as possible, we postpone all proofs to the Appendix B.

Theorem 4.2: Dual formulation for *TopPushK* family

Consider formulations *TopPush*, *TopPushK*, *TopMeanK* and τ-*FPL* from Table 2.1 with linear model, surrogate function l and Notation 4.1. Then the corresponding dual problem has the following form

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix}^\top \mathbb{K} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} - C\sum_{i=1}^{n_+} l^\star\left(\frac{\alpha_i}{C}\right) \tag{4.2a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+} \alpha_i = \sum_{j=1}^{\tilde{n}} \beta_j, \tag{4.2b}$$

$$0 \le \beta_j \le \frac{1}{K}\sum_{i=1}^{n_+} \alpha_i, \quad j = 1, 2, \ldots, \tilde{n}, \tag{4.2c}$$

where $l^\star$ is conjugate function of l and

$$\mathbb{K} = \begin{cases} \mathbb{K}^{\pm} & \text{for } \textit{TopMeanK}, \\ \mathbb{K}^{-} & \text{otherwise}, \end{cases} \qquad \tilde{n} = \begin{cases} n & \text{for } \textit{TopMeanK}, \\ n_- & \text{otherwise}. \end{cases}$$

Moreover, the variable K is defined as follows

$$K = \begin{cases} 1 & \text{for } \textit{TopPush}, \\ n\tau & \text{for } \textit{TopMeanK}, \\ n_-\tau & \text{for } \tau\text{-FPL}. \end{cases}$$

Finally, if $K = 1$, the upper bound in the second constrainet vanishes due to the first constraint.

4.1.2 Family of *Pat&Mat* formulations

Similarly to the previous section, we introduce the family of *Pat&Mat* formulations that consists of *Pat&Mat* and *Pat&Mat-NP* formulations and can be written as follows

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i \in \mathcal{I}_+} l(t - \boldsymbol{w}^\top \boldsymbol{x}_i) \\ \text{subject to} \quad & t \quad \text{solves} \quad \frac{1}{\tilde{n}} \sum_{i \in \tilde{\mathcal{I}}} l\Big(\vartheta(\boldsymbol{w}^\top \boldsymbol{x}_j - t)\Big) = \tau. \end{aligned} \tag{4.3}$$

where $C \in \mathbb{R}$ is a constant and $\tilde{\mathcal{I}}$ and $\tilde{n}$ is defined as follows

$$\tilde{\mathcal{I}} = \begin{cases} \mathcal{I} & \text{for } \textit{Pat\&Mat}, \\ \mathcal{I}_- & \text{otherwise}. \end{cases} \qquad \tilde{n} = \begin{cases} n & \text{for } \textit{Pat\&Mat}, \\ n_- & \text{otherwise}. \end{cases}$$

Again, we use linear classifier and the alternative formulation with constant C. The following theorem show the dual formulation of (4.3).

Theorem 4.3: Dual formulation for *Pat&Mat* family

Consider formulations *Pat&Mat* and *Pat&Mat-NP* from Table 2.1 with linear model, surrogate function l and Notation 4.1. Then the corresponding dual problem has the following form

$$\underset{\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix}^\top \mathbb{K} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} - C \sum_{i=1}^{n_+} l^\star\left(\frac{\alpha_i}{C}\right) - \delta \sum_{j=1}^{\tilde{n}} l^\star\left(\frac{\beta_j}{\delta \vartheta}\right) - \delta \tilde{n} \tau \tag{4.4a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+} \alpha_i = \sum_{j=1}^{\tilde{n}} \beta_j, \tag{4.4b}$$

$$\delta \geq 0, \tag{4.4c}$$

where $l^\star$ is conjugate function of l, $\vartheta > 0$ is a scaling parameter and

$$\mathbb{K} = \begin{cases} \mathbb{K}^{\pm} & \text{for } \textit{Pat\&Mat}, \\ \mathbb{K}^{-} & \text{otherwise}, \end{cases} \qquad \tilde{n} = \begin{cases} n & \text{for } \textit{Pat\&Mat}, \\ n_- & \text{otherwise}. \end{cases}$$

4.1.3 Adding kernels

As we mentioned in the beggining of the chapter, our goal is to to extend our framework into nonlinear classification problems. In the previous sections we derived dual formulations for the *TopPushK* and *Pat&Mat* family of formulations. In this sections we show, how to employ the kernels method [35] to introduce nonlinearity into the formulations. Firstly, consider any formulation that computes the decision threshold only from negative samples and therefore uses $\mathbb{K}^-$ as a kernel matrix. To add kernels, we first realize that the classification score s_j for any sample $\boldsymbol{x}_j \in \mathbb{R}^d$ is given by

$$s_j = \boldsymbol{w}^\top \boldsymbol{x}_j = \sum_{i=1}^{n_+} \alpha_i \boldsymbol{x}_j^\top \boldsymbol{x}_i^+ - \sum_{j=1}^{n_-} \beta_j \boldsymbol{x}_j^\top \boldsymbol{x}_j^-, \tag{4.5}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{n_+}$, $\boldsymbol{\beta} \in \mathbb{R}^{n_-}$ are dual variables. This relation yields from the proof of Theorems 4.2. Consider now any kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Then the first part of the objective function (4.2a) amounts to

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

Using the standard trick, we can replace the kernel matrix $\mathbb{K}^-$ with

$$\mathbb{K}^- = \begin{pmatrix} k(\mathbb{X}^+, \mathbb{X}^+) & -k(\mathbb{X}^+, \mathbb{X}^-) \\ -k(\mathbb{X}^-, \mathbb{X}^+) & k(\mathbb{X}^-, \mathbb{X}^-) \end{pmatrix}, \tag{4.6}$$

where $k(\cdot, \cdot)$ is applied to all rows of both arguments. Then for any sample $\boldsymbol{x}_j$, the classification score (4.5) is replaced by

$$s_j = \sum_{i=1}^{n_+} \alpha_i k\left(\boldsymbol{x}_j, \boldsymbol{x}_i^+\right) - \sum_{j=1}^{n_-} \beta_j k\left(\boldsymbol{x}_j, \boldsymbol{x}_j^-\right).$$

Similarly, we can use non-linear kernel matrix for any formulation that uses $\mathbb{K}^\pm$ and also for all formulations from Theorem 4.3.

4.2 New Coordinate Descent Algorithm

In the previous sections, we showed that that dual formulations of *TopPush*, *TopPushK*, *TopMeanK* and τ-*FPL* are very similary and can be written in general form summarized in Theorem 4.2. Similarly, dual formulations of *Pat&Mat* and *Pat&Mat-NP* are very similary and can be written in general form summarized in Theorem 4.3. We also showed, that these dual formulations allow us to incorporate nonlinearity using kernels [35] in the same way as in SVM. However, their dimension is at least equal to the number of all samples n and therefore it is computationally expensive to use standard techniques such as the gradient descent. To handle this issue, the coordinate descent algorithm [39, 36] has been proposed in the context of SVMs. Our goal in thsi section is to derived coordinate descent algorithm suitable for dual problems (4.2, 4.4). Since these problems differ from original SVMs by additional constraints (4.2b, 4.4b), the key idea of our algorithm is to update two coordinates (instead of one) of dual bariables $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ at every iteration. It will allow us to derive 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.

4.2.1 Family of *TopPushK* Formulations

Consider dual formulation (4.2) from Theorem 4.2 and fixed feasible dual variables $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$. Let us define vector of scores $\boldsymbol{s}$ by

$$\boldsymbol{s} = \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}. \tag{4.7}$$

As we said before, dual formulation (4.2) differs from original SVMs by additional constraintst (4.2b). Due to this constraint, we always have to update (at least) two coordinates of dual variables $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ to not violate the constraintst (4.2b). Moreover, there are only three update rules which modify two coordinates of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and which satisfy constraints (4.2b) and also keep (4.7) satisfied. The first one updates two components of $\boldsymbol{\alpha}$

$$\alpha_{\hat{k}} \to \alpha_{\hat{k}} + \Delta, \qquad \alpha_{\hat{l}} \to \alpha_{\hat{l}} - \Delta, \qquad \boldsymbol{s} \to \boldsymbol{s} + (\mathbb{K}_{\bullet,k} - \mathbb{K}_{\bullet,l})\Delta, \tag{4.8a}$$

where $\mathbb{K}_{\bullet,i}$ denotes i-th column of $\mathbb{K}$ and indices $\hat{k}$, $\hat{l}$ are defined in Notation 4.4. 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 $\boldsymbol{\beta}$

$$\alpha_{\hat{k}} \to \alpha_{\hat{k}} + \Delta, \qquad \beta_{\hat{l}} \to \beta_{\hat{l}} + \Delta, \qquad \boldsymbol{s} \to \boldsymbol{s} + (\mathbb{K}_{\bullet,k} + \mathbb{K}_{\bullet,l})\Delta, \tag{4.8b}$$

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

$$\beta_{\hat{k}} \to \beta_{\hat{k}} + \Delta, \qquad \beta_{\hat{l}} \to \beta_{\hat{l}} - \Delta, \qquad \boldsymbol{s} \to \boldsymbol{s} + (\mathbb{K}_{\bullet,k} - \mathbb{K}_{\bullet,l})\Delta. \tag{4.8c}$$

Using any of the update rules defined above, the problem (4.2) can be written as a one-dimensional quadratic 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} are constants with respect to Δ. The optimal solution to this problem is

$$\Delta^\star = \operatorname{clip}_{[\Delta_{lb},\,\Delta_{ub}]}(\gamma), \tag{4.9}$$

where $-b/a$ and $\operatorname{clip}_{[a,\,b]}(x)$ amounts to clipping (projecting) x to interval $[a,b]$. Since we assume one of the update rules (4.8), the constrain (4.2b) is always satisfied after the update. Evethough all three update rules hold true for any surrogate, the calculation of the optimal $\Delta^\star$ depends on the concrete form of surrogate function. In the following text, we show the closed-form formula for $\Delta^\star$, when the hinge loss or quadratic hinge loss is used as surrogate.

Notation 4.4

Consider any index l that satisfies $1 \le l \le n_+ + \tilde{n}$. Since the length of $\boldsymbol{\alpha}$ is always n_+, we define auxiliary index $\hat{l}$ as

$$\hat{l} = \begin{cases} l & \text{if } l \le n_+, \\ l - n_+ & \text{otherwise.} \end{cases}$$

Then the index l without hat can be safely used for kernel matrix $\mathbb{K}$ or vector of scores $\boldsymbol{s}$ while its corresponding version with hat $\hat{l}$ for $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$.

Hinge loss

We start with the hinge loss function from Notation 2.1. Plugging the conjugate (B.1) of the hinge loss into the dual formulation from Theorem 4.2 yields

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}+\sum_{i=1}^{n_+}\alpha_i \tag{4.10a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, \tag{4.10b}$$

$$0 \le \alpha_i \le C, \qquad i = 1,2,\ldots,n_+, \tag{4.10c}$$

$$0 \le \beta_j \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j = 1,2,\ldots,\tilde{n}. \tag{4.10d}$$

The form of $\mathbb{K}$ and $\tilde{n}$ depends on the used formulations as discussed in Theorem 4.2. Moreover, the upper limit in (4.10d) can be ommited for $K = 1$. Since we know the form of the optimal solution (4.9), we only need to show the concrete form of Δ_{lb}, Δ_{ub} and γ for all update rules (4.8) when the hinge loss is used. The following three lemmas provide closed-form formulas all considered update rules. To keep the presentation as simple as possible, we postpone all proofs to Appendix B.3.1.

Lemma 4.5: Update rule (4.8a) for problem (4.10)

Consider problem (4.10), update rule (4.8a), indices $1 \le k \le n_+$ and $1 \le l \le n_+$ and Notation 4.4. Then the optimal solution $\Delta^\star$ is given by (4.9) where

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

Lemma 4.6: Update rule (4.8b) for problem (4.10)

Consider problem (4.10), update rule (4.8b), indices $1 \le k \le n_+$ and $n_+ + 1 \le l \le \tilde{n}$ and Notation 4.4. Let us define

$$\beta_{\max} = \max_{j \in \{1,2,\ldots,\tilde{n}\}\setminus\{\hat{l}\}} \beta_j.$$

Then the optimal solution $\Delta^\star$ is given by (4.9) where

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

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

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

Lemma 4.7: Update rule (4.8c) for problem (4.10)

Consider problem (4.10), update rule (4.8c), indices $n_+ + 1 \le k \le \tilde{n}$ and $n_+ + 1 \le l \le \tilde{n}$ and Notation 4.4. Then the optimal solution $\Delta^\star$ is given by (4.9) where

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

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

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

Quadratic hinge loss

The second considered suroagte function is the quadratic hinge loss from Notation 2.1. Plugging the conjugate (B.1) of the quadratic hinge loss into the dual formulation from Theorem 4.2 yields

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad & -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} + \sum_{i=1}^{n_+}\alpha_i - \frac{1}{4C}\sum_{i=1}^{n_+}\alpha_i^2 && \text{(4.11a)}\\
\text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, && \text{(4.11b)}\\
& 0 \le \alpha_i, \qquad i = 1,2,\ldots,n_+, && \text{(4.11c)}\\
& 0 \le \beta_j \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j = 1,2,\ldots,\tilde{n}, && \text{(4.11d)}
\end{aligned}$$

Similarly to the previous case, the form of $\mathbb{K}$ and $\tilde{n}$ depends on the used formulations and the upper limit in (4.11d) can be ommited for $K = 1$. For simplicity, we postpone formulas for update rules (4.8) and their proofs to Appendix B.3.1. More specificaly, all update rules can be found in Lemma B.10-B.12.

Initialization

For all update rules (4.8) we assumed that the current solution $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ is feasible. So to create an iterative algorithm that solves problem (4.10) or (4.11), we need to have a way how to initialize the algorithm. Such a task can be formally written as a projection of initial solution $\boldsymbol{\alpha}^0$, $\boldsymbol{\beta}^0$ to the feasible set of solutions

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{minimize}} \quad & \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\right\|^2\\
\text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j,\\
& 0 \le \alpha_i \le C_1, \quad i = 1,2,\ldots,n_+,,\\
& 0 \le \beta_j \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j = 1,2,\ldots,\tilde{n},
\end{aligned} \qquad \text{(4.12)}$$

where the upper bound in the second constraint depends on the used surrogate function and is defined as follows

$$C_1 = \begin{cases} C & \text{for hinge loss,} \\ +\infty & \text{for quadratic hinge loss.} \end{cases}$$

To solve problem (4.12), we will follow the same approach as in [40]. In the following theorem, we show that problem (4.12) can be written as a system of two equations of two variables (λ, μ). Moreover, the theorem shows the concrete form of feasible solution $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ that depends only on (λ, μ).

Theorem 4.8

Consider problem (4.12) and some initial solution $\boldsymbol{\alpha}^0$, $\boldsymbol{\beta}^0$ and denote the sorted version (in non-decreasing order) of $\boldsymbol{\beta}^0$ as $\boldsymbol{\beta}^0_{[\cdot]}$. Then if the following condition holds

$$\sum_{j=1}^{K} \left(\beta^0_{[\tilde{n}-K+j]} + \max_{i=1,\dots,n_+} \alpha_i^0 \right) \leq 0, \tag{4.13}$$

the optimal solution of (4.12) amounts to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$. In the opposite case, the following system of two equations

$$\sum_{i=1}^{n_+} \operatorname{clip}_{[0,\, C_1]} \left(\alpha_i^0 - \lambda + \frac{1}{K} \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\infty)} \left(\beta_j^0 + \lambda - \mu \right) \right) - K\mu = 0, \tag{4.14a}$$

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu]} \left(\beta_j^0 + \lambda \right) - K\mu = 0, \tag{4.14b}$$

has a solution (λ, μ) with $\mu > 0$, and the optimal solution of (4.12) equals to

$$\alpha_i = \operatorname{clip}_{[0,\, C_1]} \left(\alpha_i^0 - \lambda + \frac{1}{K} \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\infty)} \left(\beta_j^0 + \lambda - \mu \right) \right),$$

$$\beta_j = \operatorname{clip}_{[0,\,\mu]} \left(\beta_j^0 + \lambda \right).$$

In the following text, we show that the number of variables in the system of equations (4.14) can be reduced to one. For any fixed μ, we denote the function on the left-hand side of (4.14b) by

$$g(\lambda; \mu) := \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu]} \left(\beta_j^0 + \lambda \right) - K\mu.$$

Then g is non-decreasing in λ but not necessarily strictly increasing. We denote by $\lambda(\mu)$ any such λ solving (4.14b) for a fixed μ. Denote $\boldsymbol{z}$ the sorted version of $-\boldsymbol{\beta}^0 s$. Then we have

$$g(\lambda; \mu) = \sum_{\{j \,|\, \lambda - z_j \in [0,\mu)\}} (\lambda - z_j) + \sum_{\{j \,|\, \lambda - z_j \geq \mu\}} \mu - K\mu.$$

Now we can easily compute $\lambda(\mu)$ by solving $g(\lambda(\mu); \mu) = 0$ for fixed μ using Algorithm 2. The algorithm can be described as follows: Index i will run over $\boldsymbol{z}$ while index j will run over $\boldsymbol{z} + \mu$. At every iteration, we know the values of $g(z_{i-1}; \mu)$ and $g(z_{j-1} + \mu; \mu)$ and we want to evaluate g at the next point. We denote number of indices j such that $\lambda - z_j \in [0, \mu)$ by d. If $z_i \leq z_j + \mu$, then we consider $\lambda = z_i$ and since one index enters the set $\left\{ j \,\middle|\, \lambda - z_j \in [0, \mu) \right\}$, we increase d by

one. On the other hand, if $z_i > z_j + \mu$, then we consider $\lambda = z_j + \mu$ and since one index leaves the set $\left\{j \,\middle|\, \lambda - z_j \in [0, \mu)\right\}$, we decrease d by one. In both cases, g is increased by d times the difference between the new λ and old λ. Once g exceeds 0, we stop the algorithm and linearly interpolate between the last two values. To prevent an overflow, we set $z_{m+1} = \infty$. Concerning the initial values, since $z_1 \leq z_1 + s\mu$, we set $i = 2$, $j = 1$ and $d = 1$.

Algorithm 2 For computing $\lambda(\mu)$ from (4.14)

Require: vector $-\boldsymbol{\beta}^0$ sorted into $\boldsymbol{z}$
$i \leftarrow 2, j \leftarrow 1, d \leftarrow 1$
$\lambda \leftarrow z_1, g \leftarrow -K\mu$
while $g < 0$ **do**
  if $z_i \leq z_j + \mu$ **then**
    $g \leftarrow g + d(z_i - \lambda)$
    $\lambda \leftarrow z_i, d \leftarrow d + 1, i \leftarrow i + 1$
  else
    $g \leftarrow g + d(z_j + \mu - \lambda)$
    $\lambda \leftarrow z_j + \mu, d \leftarrow d - 1, j \leftarrow j + 1$
  end if
end while
return linear interpolation of the last two values of λ

Since $\lambda(\mu)$ can be computed for fixed μ using Algorithm 2, we can reduce system (4.14) into one equation

$$h(\mu) := \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\, C_1]}\left(\alpha_i^0 - \lambda(\mu) + \frac{1}{K}\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\infty)}\left(\beta_j^0 + \lambda(\mu) - \mu\right)\right) - K\mu = 0 \tag{4.15}$$

which needs to be solved for μ. The following lemma describes properties of h. Since h is decreasing in μ on $(0, \infty)$, we can use root finding algorithms such as Bisection or Newton method to find the solution.

Lemma 4.9

Even though $\lambda(\mu)$ is not unique, function h is well-defined in the sense that it gives the same value for every choice of $\lambda(\mu)$. Moreover, h is decreasing in μ on $(0, \infty)$.

Add figures of h and g

4.2.2 Family of *Pat&Mat* Formulations

In the beginning of this chapter we derived general dual formulation (4.4) for the family of *Pat&Mat* formulations. Similarly to the dual formulation (4.2), we can use update rules (4.8) to solve this dual formulation. In this case, however, we have to also consider the third primal variable δ. Then the dual formulation (4.4) can be rewritten as a quadratic one-dimensional problem

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

where a, b, c, Δ_{lb}, Δ_{ub} are constants with respect to Δ. The form of the optimal solution is the same as for problem (4.2) and reads

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

where $-b/a$. Since we assume one of the update rule (4.8), the constrain (4.4b) is always satisfied after the update. The exact form of the update rules depend on the surrogate function. More, the form of optimal δ also depends on the surrogate function. In the following text, we derive update rules for hinge loss and quadratic hinge loss function.

Hinge Loss

We again start with the hinge loss function from Notation 2.1. Plugging the conjugate (B.1) of the hinge loss into the dual formulation from Theorem 4.3 yields

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad & -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} + \sum_{i=1}^{n_+}\alpha_i + \frac{1}{\vartheta}\sum_{j=1}^{\tilde{n}}\beta_j - \delta\tilde{n}\tau && \text{(4.16a)}\\
\text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, && \text{(4.16b)}\\
& 0 \le \alpha_i \le C, \quad i = 1,2,\dots,n_+, && \text{(4.16c)}\\
& 0 \le \beta_j \le \delta\vartheta, \quad j = 1,2,\dots,\tilde{n}, && \text{(4.16d)}\\
& \delta \ge 0. && \text{(4.16e)}
\end{aligned}$$

This is a convex quadratic problem and can be solved optimized using update rules (4.8). In such a case, we do not perform a joint maximization in $(\alpha_{\hat{k}}, \beta_l, \delta)$ but perform a maximization with respect to $(\alpha_{\hat{k}}, \beta_l)$, update these two values and then optimize the objective with respect to δ. Then for fixed feasible solution $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, maximizing objective function (4.16a) with respect to δ yields

$$\begin{aligned}
\underset{\delta}{\text{maximize}} \quad & -\tilde{n}\tau\delta\\
\text{subject to} \quad & 0 \le \beta_j \le \delta\vartheta, \quad j = 1,2,\dots,\tilde{n},\\
& \delta \ge 0.
\end{aligned}$$

Since $\tilde{n}\tau \ge 0$, to maximize the objective function with respect to the δ, we have to find the smallest possible δ that satisfies the constrains. Such δ is in the following form

$$\delta^* = \frac{1}{\vartheta}\max_{j\in\{1,2,\dots,\tilde{n}\}}\beta_j. \tag{4.17}$$

Since the formulas for optimal update rules are rather long, we postpone them to Appendix B.3.2. More specificaly, all update rules can be found in Lemma B.15-B.17.

Quadratic Hinge Loss

The second choice of the surrogate function is the quadratic hinge loss function from Notation 2.1. Plugging the conjugate (B.2) of the quadratic hinge loss into the dual formulation

from Theorem 4.3 yields

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}+\sum_{i=1}^{n_+}\alpha_i-\frac{1}{4C}\sum_{i=1}^{n_+}\alpha_i^2 \tag{4.18a}$$

$$+\frac{1}{\vartheta}\sum_{j=1}^{\tilde{n}}\beta_j-\frac{1}{4\delta\vartheta^2}\sum_{j=1}^{\tilde{n}}\beta_j^2-\delta\tilde{n}\tau \tag{4.18b}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i=\sum_{j=1}^{\tilde{n}}\beta_j, \tag{4.18c}$$

$$\alpha_i\geq 0, \qquad i=1,2,\ldots,n_+, \tag{4.18d}$$

$$\beta_j\geq 0, \qquad j=1,2,\ldots,\tilde{n}, \tag{4.18e}$$

$$\delta\geq 0, \tag{4.18f}$$

Again we get a convex quadratic problem that can be solved optimized using update rules (4.8). In such a case, we again perform maximization only with respect to $(\alpha_{\hat{k}},\ \beta_l)$ and we need to maximazie the objective with respect to δ separately. For fixed feasible solution $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, maximizing objective function (4.18a-4.18b) with respect to δ leads to the following problem

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

with the optimal solution that equals to

$$\delta^*=\sqrt{\frac{1}{4\vartheta^2\tilde{n}\tau}\sum_{j=1}^{\tilde{n}}\beta_j^2}. \tag{4.19}$$

As in the previous section, we postpone the formulas for optimal update rules to Appendix B.3.2. More specificaly, all update rules can be found in Lemma B.18-B.20.

Initialization

As in the case of problem (4.2), for all update rules (4.8) we assumed that the current solution $\boldsymbol{\alpha},\ \boldsymbol{\beta}\ \delta$ is feasible. So to create an iterative algorithm that solves problem (4.16) or (4.18), we need to have a way how to initialize the algorithm. Such a task can be formally written as a projection of initial solution $\boldsymbol{\alpha}^0,\ \boldsymbol{\beta}^0\ \delta^0$ to the feasible set of solutions

$$\begin{aligned}\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{minimize}} \quad & \frac{1}{2}\left\|\boldsymbol{\alpha}-\boldsymbol{\alpha}^0\right\|^2+\frac{1}{2}\left\|\boldsymbol{\beta}-\boldsymbol{\beta}^0\right\|^2+\frac{1}{2}(\delta-\delta^0)^2\\ \text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i=\sum_{j=1}^{\tilde{n}}\beta_j,\\ & 0\leq\alpha_i\leq C_1, \quad i=1,2,\ldots,n_+,\\ & 0\leq\beta_j\leq C_2\delta, \quad j=1,2,\ldots,\tilde{n},,\\ & \delta\geq 0,\end{aligned} \tag{4.20}$$

where the upper bounds in the second and third constraints depend on the used surrogate function and are defined as follows

$$C_1=\begin{cases}C & \text{for hinge loss,}\\ +\infty & \text{for quadratic hinge loss,}\end{cases} \qquad C_2=\begin{cases}\delta\vartheta & \text{for hinge loss,}\\ +\infty & \text{for quadratic hinge loss.}\end{cases}$$

Again, we will follow the same approach as in [40] to solve problem (4.20). In the following theorem, we show that problem (4.20) can be written as a system of two equations of two variables (λ, μ). Moreover, the theorem shows the concrete form of feasible solution $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ δ sthat depends only on (λ, μ).

Theorem 4.10

Consider problem (4.20) and some initial solution $\boldsymbol{\alpha}^0$, $\boldsymbol{\beta}^0$ and δ^0. Then if the following condition holds

$$\delta^0 \leq -C_2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\infty)}\left(\beta_j^0 + \max_{i=1,\dots,n_+} \alpha_i^0\right). \tag{4.21}$$

the optimal solution of (4.12) amounts to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$ and $\delta^0 = 0$. In the opposite case, the following system of two equations

$$0 = \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda\right) - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\lambda+\mu]}\left(\beta_j^0 + \lambda\right), \tag{4.22a}$$

$$\lambda = C_2\delta^0 + C_2^2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\infty)}\left(\beta_j^0 - \mu\right) - \mu. \tag{4.22b}$$

has a solution (λ, μ) with $\lambda + \mu > 0$ and the optimal solution of (4.20) equals to

$$\begin{aligned} \alpha_i &= \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda\right), \\ \beta_j &= \operatorname{clip}_{[0,\,\lambda+\mu]}\left(\beta_j^0 + \lambda\right), \\ C_2\delta &= \lambda + \mu. \end{aligned}$$

System (4.22) is relatively simple to solve, since equation (4.22b) provides an explicit formula for λ. Let us denote it as $\lambda(\mu)$, then we denote the right-hand side of (4.22a) as

$$h(\mu) := \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda(\mu)\right) - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\lambda(\mu)+\mu]}\left(\beta_j^0 + \mu\right). \tag{4.23}$$

finally, system (4.22) is equivalent to solving $h(\mu) = 0$.

Lemma 4.11

Function h is non-decreasing in μ on $(0, \infty)$.

The previous lemma states that h is a non-decreasing functionin μ on $(0, \infty)$ and thus the equation $h(\mu) = 0$ is simple to solve numerically using any root finding method such as Bisection or Newton method. Note that if $\delta^0 < 0$, then it may happen that $\lambda + \mu < 0$ if the initial μ is chosen large. In such a case, it suffices to decrease μ until $\lambda + \mu$ is positive.

Add figures of h and g

4.2.3 Complexity analysis

In the previous sections, we derived dual formulations for two families of problems. Moreover, we showed, that these dual formulations can be solved iteratively using simple update rules (4.8). Since these update rules assume initial feasible solution, we also showed how to find such initial solution. Then the final algorithm can be summarized as in Algorithm 3.

Algorithm 3 Coordinate descent algorithm for *TopPushK* family of formulations (left) and *Pat&Mat* family of formulations (right).

1: set $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ using Theorem 4.8	1: set $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta)$ using Theorem 4.10
2: set $\boldsymbol{s}$ based on (4.7)	2: set $\boldsymbol{s}$ based on (4.7)
3: **repeat**	3: **repeat**
4: random k from $\{1, 2, \ldots, n_+ + \tilde{n}\}$	4: random k from $\{1, 2, \ldots, n_+ + \tilde{n}\}$
5: **for** $l \in \{1, 2, \ldots, n_+ + \tilde{n}\}$ **do**	5: **for** $l \in \{1, 2, \ldots, n_+ + \tilde{n}\}$ **do**
6: compute Δ_l	6: compute (Δ_l, δ_l)
7: **end for**	7: **end for**
8: select best Δ_l	8: select best (Δ_l, δ_l)
9: update $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{s}$ according to (4.8)	9: update $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{s}$ according to (4.8)
10:	10: set $\delta \leftarrow \delta_l$
11: **until** stopping criterion is satisfied	11: **until** stopping criterion is satisfied

The left column in Algorithm 3 describe the algorithm for *TopPushK* family of formulations and the right for *Pat&Mat* family of formulations. In step 2 we initialize $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and δ to some feasible value using Theorem 4.8 for *TopPushK* family and Theorem 4.10 for *Pat&Mat* family. Then, based on (4.7) we compute $\boldsymbol{s}$. Each `repeat` loop in step 3 updates two coordinates as shown in (4.8). In step 4 we select a random index k and in the `for` loop in step 5 we compute the optimal (Δ_l, δ_l) for all possible combinations (k, l) as in (4.8). In step 8 we select the best pair (Δ_l, δ_l) which maximizes the coresponding objective function. Finally, based on the selected update rule we update $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{s}$ and δ in steps 9 and 10.

Now we derive the computational complexity of each `repeat` loop from step 3. The computation of (Δ_l, δ_l) amounts to solving a quadratic optimization problem in one variable. As we showed in Sections 4.2.1 and 4.2.2, there is a closed-form solution and step 6 can be performed in $O(1)$. Since this is embedded in a `for` loop in step 5, the whole complexity of this loop is $O(n_+ + \tilde{n})$. Step 9 requires $O(1)$ for the update of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ while $O(n_+ + \tilde{n})$ for the update of $\boldsymbol{s}$. Since the other steps are $O(1)$, the total complexity of the `repeat` loop is $O(n_+ + \tilde{n})$. 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 4.1.

Operation	$\mathbb{K}$ precomputed	$\mathbb{K}$ not precomputed
Evaluation of Δ_l	$O(1)$	$O(d)$
Update of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$	$O(1)$	$O(1)$
Update of $\boldsymbol{s}$	$O(n_+ + \tilde{n})$	$O((n_+ + \tilde{n})d)$
Total per iteration	$O(n_+ + \tilde{n})$	$O((n_+ + \tilde{n})d)$

Table 4.1: Computational complexity of one `repeat` loop (which updates two coordinates of $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$) from Algorithm 3.

5

Deep

Since this task considers only scores above the threshold, [22] 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 5.1 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 [41].

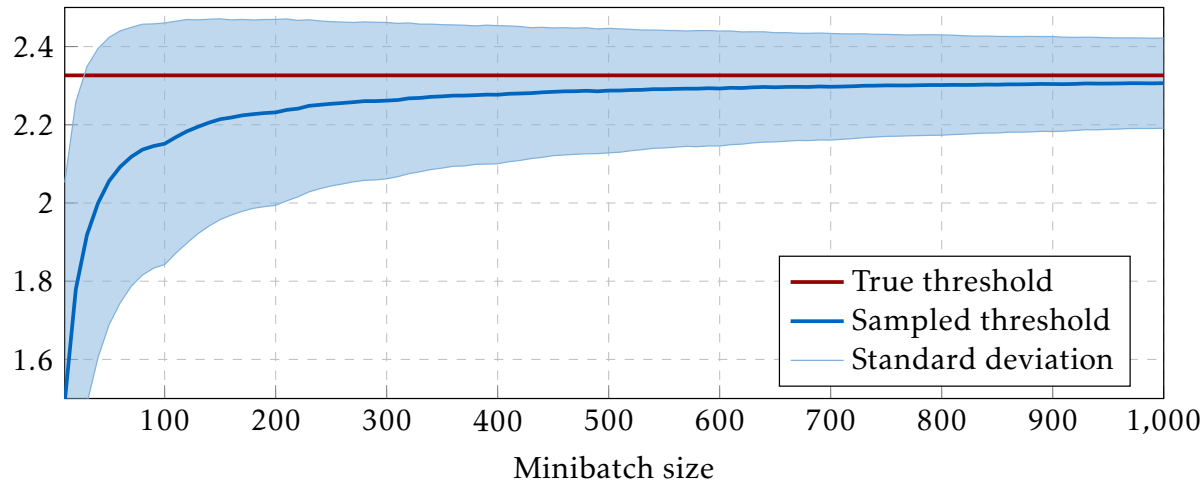


Figure 5.1: The bias between the sampled and true thresholds computed from scores following the standard normal distribution. The threshold separates the top 1% of samples with the highest scores.

Our method mitigates this bias. It is based on several results. [18] proposed the TopPush formulation of the accuracy at the top and solved it in its dual formulation. [21] 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 [22], 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

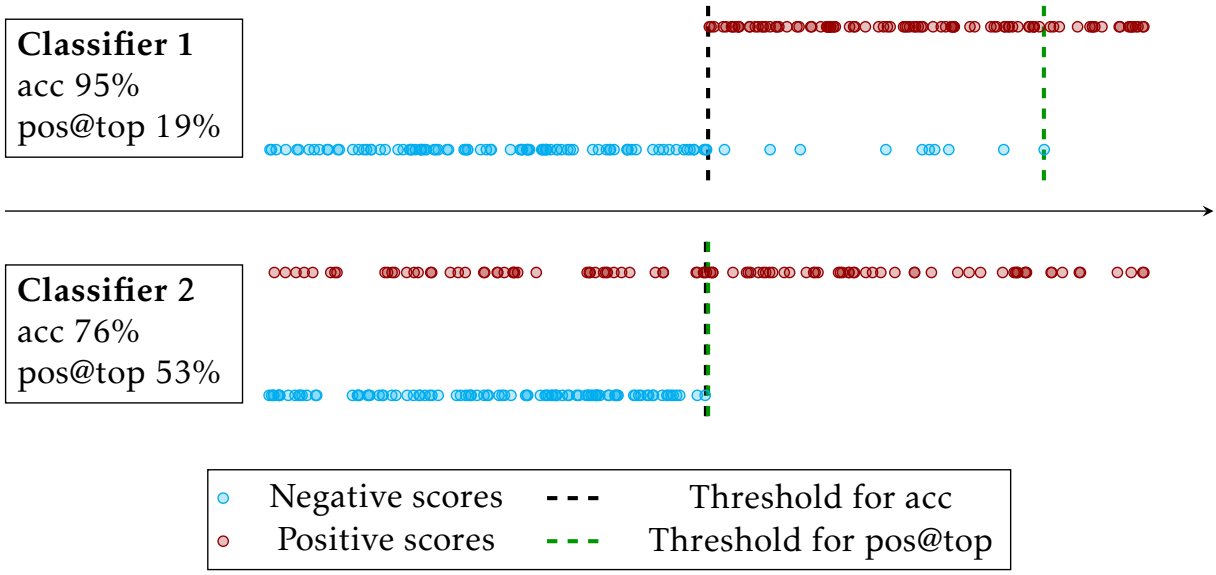


Figure 5.2: Difference between standard classifiers (top row) and classifiers maximizing accuracy at the top (bottom row). While the former has a good total accuracy, the latter has a good top acccuracy.

the idea of [33] and enhance 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 5.1 introduces a general formulation of accuracy at the top. Section 5.2 derives formulas for the bias of the sampled threshold and proposes *DeepTopPush* to minimize it. Section 6.3 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.

5.1 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

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{s},t}{\text{minimize}} \quad & \lambda_1 \cdot \mathrm{fp}(\boldsymbol{s},t) + \lambda_2 \cdot \mathrm{fn}(\boldsymbol{s},t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i;\boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s},\boldsymbol{y}). \end{aligned} \tag{5.1}$$

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 $\mathbb{1}_{[\cdot]}$ is the characteristic (0/1) function counting how many times the argument is satisfied. Setting (5.1) includes TopPush [18] which minimizes the number of positive samples below the highest-ranked negative sample. This fits into (5.1) with $\lambda_1 = 0$, $\lambda_2 = 1$ and $t = \max_{i \in I^-} z_i$.

Figure 5.2 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 2.1 shows other special cases of (5.1) including maximizing precision at a given level of recall [29] 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 [21] minimizes the false negative rate (equivalently maximizes the true positive rate) under the constraint that the false positive rate is at most τ.

5.1.1 Related works

There is a close connection between accuracy at the top and ranking problems [37, 38]. This was, together with similarities to the Neyman-Pearson problem, showed in [21]. A special case of the ranking problems attempts to rank positive samples above negative samples. Several approaches, such as RankBoost [15], Infinite Push [16] or p-norm push [17] employ a positive-negative pairwise comparison of scores, which can handle only small datasets. TopPush [18] converts the pairwise sum into a single sum and minimizes the 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 (5.1) 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 [22] 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. [25, 21, 42] write the threshold as a constraint and replace both the objective and the constraint via surrogates. [25] uses Lagrange multipliers to obtain a minimax problem, [29] implicitly removes the threshold as an optimization variable and uses the chain rule to compute the gradient while [32] 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* [43] solves a general class of constrained problems via a minimax reformulation. In the second approach, SoDeep [44] or SmoothI [45] 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* [46] 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.

5.2 DeepTopPush as a method for maximizing accuracy at the top

This section first shows a basic algorithm to solve (5.1). 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.

5.2.1 Basic algorithm for solving accuracy at the top

Even though the presented technique can be applied to any formulation from Table **??**, 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 (5.1). Since the function $\mathbb{1}_{[\cdot]}$ in the formulation (5.1) is discontinuous, it is usually replaced by a general surrogate function l which is continuous and non-decreasing. This leads to

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{s},t}{\text{minimize}} \quad & \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l(t-s_i) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i;\boldsymbol{w}), \quad i\in\mathcal{I}, \\ & t = G(\boldsymbol{s},\boldsymbol{y}). \end{aligned} \tag{5.2}$$

To apply the stochastic gradient descent, we need to compute the gradient. The core idea follows [29] which was proposed in a more general context in [33]. It rewrites problem (5.2) into its equivalent form

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l\Big(G\Big(\big\{(f(\boldsymbol{x}_j;\boldsymbol{w}),y_j)\big\}_{j\in\mathcal{I}}\Big)\Big) - f(\boldsymbol{x}_i;\boldsymbol{w}). \tag{5.3}$$

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 (5.2). In all cases from Table **??**, the threshold t always equals to one of the scores, let it have index j^* and then $t = z_{j^*}$. Denoting the objective of (5.3) by $L(\boldsymbol{w})$, the chain rule implies that the gradient of the objective from (5.3) equals to

$$\nabla L(\boldsymbol{w}) = \frac{1}{n_+}\sum_{i\in I^+} l'(t-z_i)\Big(\nabla_w f(\boldsymbol{w};\boldsymbol{x}_{j^*}) - \nabla_w f(\boldsymbol{w};\boldsymbol{x}_i)\Big). \tag{5.4}$$

The stochastic gradient descent replaces the sum over all positive samples I^+ with a sum over all positive samples in a minibatch $\mathcal{I}_{\text{mb},+}$. 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 $n_{\text{mb},+}$, we replace the true gradient (5.4) by the *sampled gradient*

$$\nabla\hat{L} = \frac{1}{n_{\text{mb},+}}\sum_{i\in\mathcal{I}_{\text{mb},+}} l'(\hat{t}-z_i)\Big(\nabla_w f(\boldsymbol{w};\boldsymbol{x}_{\hat{j}}) - \nabla_w f(\boldsymbol{w};\boldsymbol{x}_i)\Big), \tag{5.5}$$

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 4.

5.2.2 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 [41]. This means that

$$\text{bias}(\boldsymbol{w}) := \nabla L(\boldsymbol{w}) - \mathbb{E}\nabla\hat{L}(\boldsymbol{w}) \tag{5.6}$$

equals to 0 for all $\boldsymbol{w}$. A comparison of (5.4) and (5.5) 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 4 Basic algorithm for solving (5.1)

Initialize weights $\boldsymbol{w}$
repeat
  Select minibatch $\mathcal{I}_{\text{mb}}$

  Compute $z_i \leftarrow f(\boldsymbol{w}; \boldsymbol{x}_i)$ for $i \in \mathcal{I}_{\text{mb}}$
  Set $\hat{t} \leftarrow G(\{(z_i, y_i)\}_{i \in \mathcal{I}_{\text{mb}}})$

  Compute $\nabla \hat{L}$ based on $\mathcal{I}_{\text{mb}}$
  Make a gradient step
until stopping criterion is satisfied

Algorithm 5 DeepTopPush as an efficient method for maximizing accuracy at the top.

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

Proposition 5.1: [47]

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 5.1 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 5.2

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

$$\nabla \hat{L}(\boldsymbol{w}) = \frac{1}{n_{\text{mb},+}} \sum_{i \in \mathcal{I}_{\text{mb},+}} l'(t - z_i)\left(\nabla_w f(\boldsymbol{w}; \boldsymbol{x}_{j^*}) - \nabla_w f(\boldsymbol{w}; \boldsymbol{x}_i)\right). \tag{5.7}$$

The summands are identical to the ones in (5.4). 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 5.3

Under the assumptions of Lemma 5.2, the bias of the sampled gradient from (5.6) satisfies

$$\text{bias}(\boldsymbol{w}) = \mathbb{P}(\hat{j} \neq j^*)\Big(\nabla L(\boldsymbol{w}) - \mathbb{E}\Big(\nabla \hat{L}(\boldsymbol{w}) \mid \hat{j} \neq j^*\Big)\Big). \tag{5.8}$$

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 (5.6) and Lemma 5.2. ■

The assumptions of Theorem 5.3 holds for all methods from Table **??** with the exception of Rec@K. For this method, the bias contains an additional term, as we show in the appendix.

The bias (5.8) 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.

5.2.3 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 [48]. 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 (5.8). Moreover, Proposition 5.1 ensures that the difference between the sampled threshold $\hat{t}$ and the true threshold t is small. Then the difference between the true gradient (5.4) and the sampled gradient (5.5) decreases as well. This reduces the second term in (5.8). This approach is applicable to any method from Table **??**.

5.2.4 Bias reduction: Incorporating delayed values

Various reasons may enforce the use of small minibatches. Then Algorithm 4 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 (5.8), reduces the bias of the sampled gradient.

We summarize the procedure in Algorithm 5 and show it next to Algorithm 4 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 5 uses the same formulation as *TopPush* [18] but can handle an arbitrary classifier, we name it *DeepTopPush*. We provide empirical evidence of why our technique works later in Section 6.3.6.

6 Numerical Experiments

6.1 Linear Model

In this section, we present numerical results.

6.1.1 Implementational details and Hyperparameter choice

We recall that all methods fall into the framework of either (2.1) or (2.3). 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 (3.6), we compute the gradient of the objective and apply the ADAM descent scheme [49]. 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 (**??**). The initial point is generated randomly.

We run the methods for the following hyperparameters

$$\begin{aligned} &\beta \in \{0.0001,\ 0.001,\ 0.01,\ 0.1,\ 1,\ 10\}, \\ &\lambda \in \{0,\ 0.00001,\ 0.0001,\ 0.001,\ 0.01,\ 0.1\}, \\ &k \in \{1, 3, 5, 10, 15, 20\}. \end{aligned} \tag{6.1}$$

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.

6.1.2 Dataset description and Performance criteria

For the numerical results, we considered 10 datasets summarized in Table 6.1. They can be downloaded from the UCI repository. sigillito1989classification [50] and Spambase are small, baldi2016parameterized [51] contains a large number of samples while guyon2005result [52] 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 6.1 shows, these datasets are imbalanced.

Each of the visual recognition datasets was converted into ten binary datasets by considering one of the classes $\{0, \ldots, 9\}$ 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.

Dataset	m	Train		Validation		Test	
		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	2 300	39.4%	1 150	39.4%	1 151	39.4%
Gisette	5 000	1 000	50.0%	1 500	50.0%	500	50.0%
Hepmass	28	5 250 000	50.0%	1 750 000	50.0%	3 500 000	50.0%
MNIST	$28 \times 28 \times 1$	44 999	11.2%	15 001	11.2%	10 000	11.4%
FashionMNIST	$28 \times 28 \times 1$	45 000	10.0%	15 000	10.0%	10 000	10.0%
CIFAR10	$32 \times 32 \times 3$	37 500	10.0%	12 500	10.0%	10 000	10.0%
CIFAR20	$32 \times 32 \times 3$	37 500	5.0%	12 500	5.0%	10 000	5.0%
CIFAR100	$32 \times 32 \times 3$	37 500	1.0%	12 500	1.0%	10 000	1.0%
SVHN2	$32 \times 32 \times 3$	54 944	18.9%	18 313	18.9%	26 032	19.6%

Table 6.1: 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 $\frac{n_+}{n}$.

6.1.3 Numerical results

Figure 6.1 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. Even though all methods work well, *Pat&Mat-NP* seems to outperform the remaining methods on most levels of false-positive rate.

While Figure 6.1 gave a glimpse of the behaviour of methods, Figures 6.2 and 6.3 provide a statistically more sound comparison. It employs the Nemenyi post hoc test for the Friedman test recommended in [53]. 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 6.1.2 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.

From this figure and table, we make several observations:

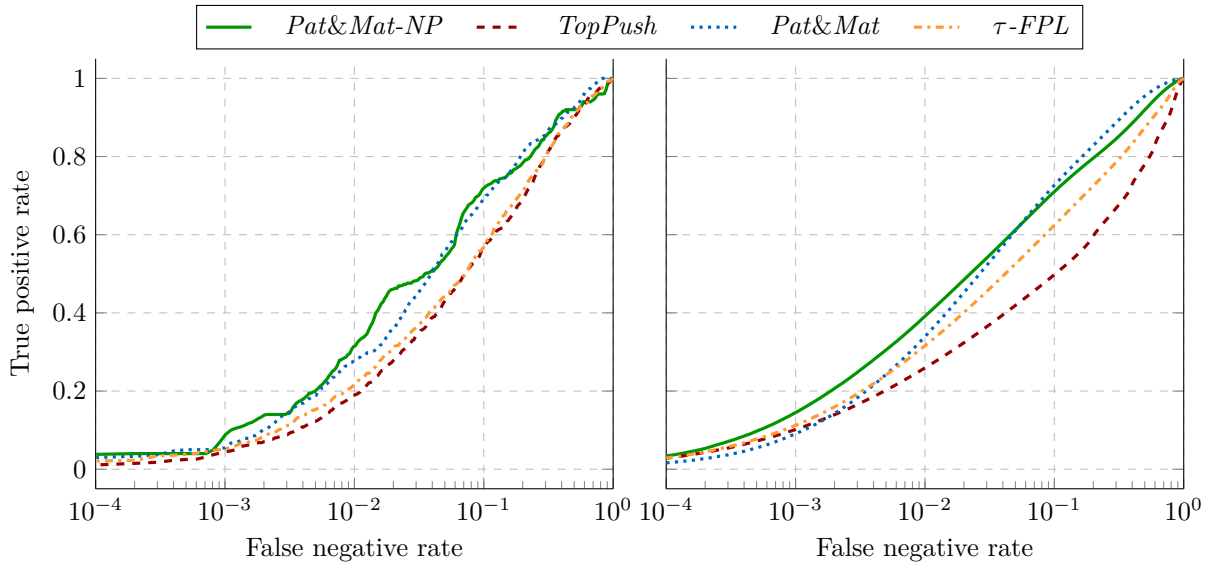


Figure 6.1: ROC curves (with logarithmic x axis) on CIFAR100 (left) and Hepmass (right).

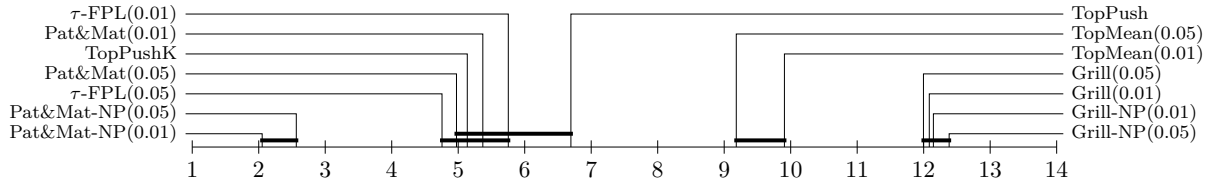


Figure 6.2: Critical difference (CD) diagrams (level of importance 0.05) of the Nemenyi post hoc test for the Friedman test. Each diagram shows the mean rank of each method, with rank 1 being the best. Black wide horizontal lines group together methods with the mean ranks that are not significantly different. The critical difference diagrams were computed for mean rank averages over all datasets of the tpr@fpr ($\tau = 0.01$) metric.

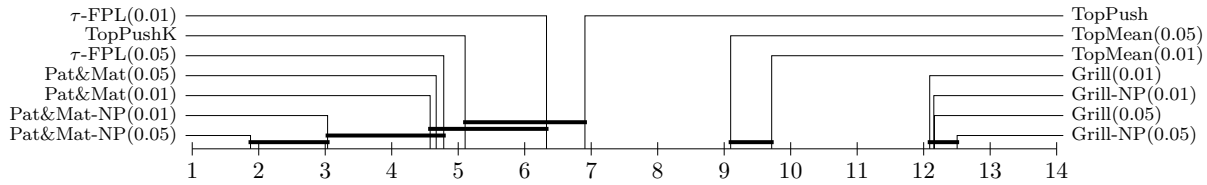


Figure 6.3: Critical difference (CD) diagrams (level of importance 0.05) of the Nemenyi post hoc test for the Friedman test. Each diagram shows the mean rank of each method, with rank 1 being the best. Black wide horizontal lines group together methods with the mean ranks that are not significantly different. The critical difference diagrams were computed for mean rank averages over all datasets of the tpr@fpr ($\tau = 0.05$) metric.

- *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 6.2 and 6.3.
- 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 3.2 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 3.7.

- *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 6.3) outperforms *Pat&Mat-NP* at level 0.01 (rank 3.0 in Figure 6.3) 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 6.3 but not for Figure 6.2.

Figure 6.4 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 6.2 employs a comparison of mean ranks and Figure 6.4 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.

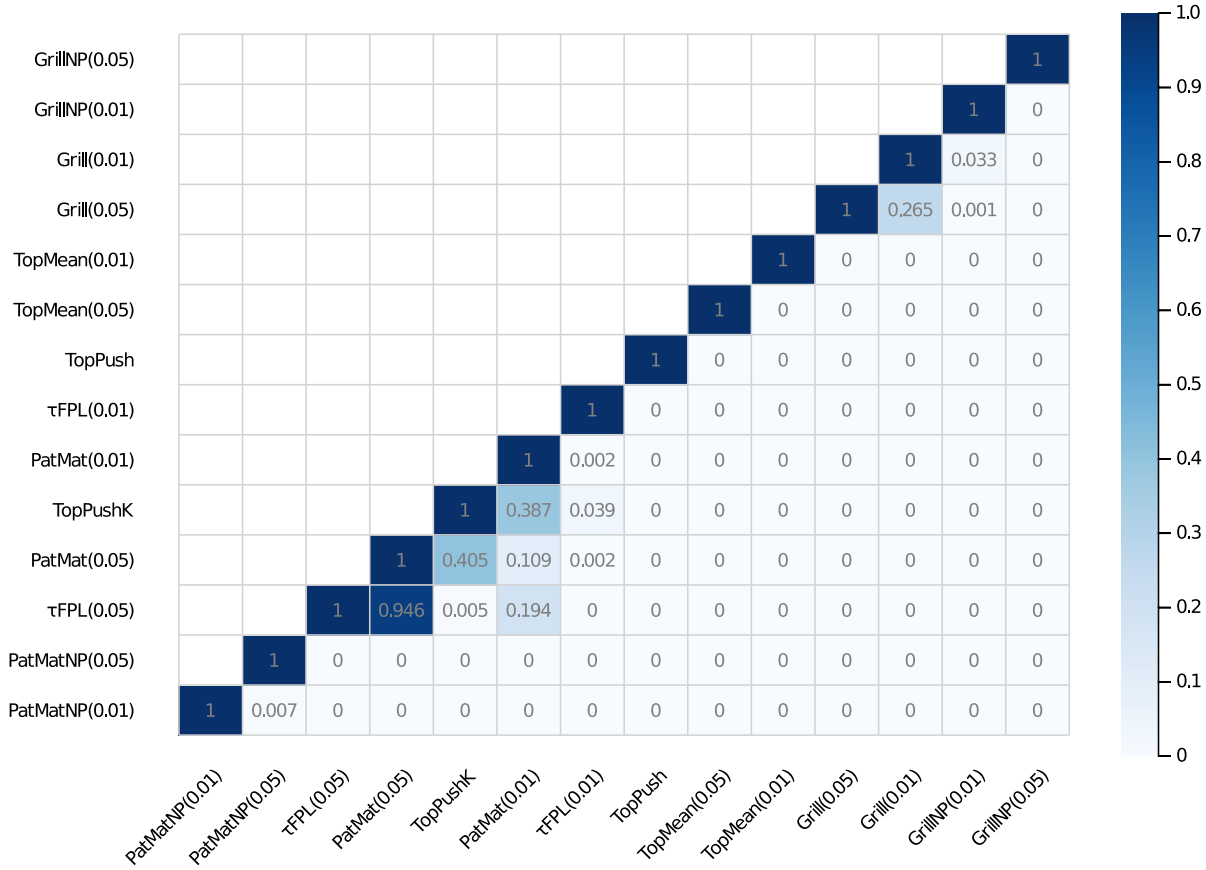


Figure 6.4: The p-value for the parwise Wilcoxon signed-rank test, where the null hypothesis is that the mean tpr@fpr(0.01) of both methods is the same. The methods are sorted by mean rank (left = better).

Table 6.2 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 3.8.
- *TopMeanK* fails everywhere which agrees with Corollary 3.7.
- Figure 3.2 states that the methods from Section 2.3 has a higher threshold than their Neyman-Pearson variants from Section 2.4. This is documented in the table as the latter have a higher number of successes.

Method		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 6.2: 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.

6.2 Dual

In this section, we present numerical results. All codes were implemented in the Julia language [54] and are available online.[1]

6.2.1 Performance criteria

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

$$\text{precision} = \frac{\sum_{i=1}^{n_+} [w^\top x_i^+ - t]}{\sum_{i=1}^{n} [w^\top x_i - t]}, \qquad \text{recall} = \frac{1}{n_+} \sum_{i=1}^{n_+} [w^\top x_i^+ - t]. \tag{6.2}$$

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

We also use the Precision-Recall (PR) curve that are commonly used for unbalanced data [55] and precision at a certain level of recall which we denote by Precision@Recall.

6.2.2 Hyperparameter choice

In Section 4.2 we introduced Algorithm 3 for solving dual problems (B.4, B.6). 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

$$k_{\text{lin}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}, \tag{6.3}$$

$$k_{\text{gauss}}(\boldsymbol{x}, \boldsymbol{y}) = \exp\{-\sigma \left\| \boldsymbol{x} - \boldsymbol{y} \right\|_2^2\} \tag{6.4}$$

and the truncated quadratic loss (**??**) 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.

Dataset	y^+	d	Train		Validation		Test	
			n	$\frac{n_+}{n}$	n	$\frac{n_+}{n}$	n	$\frac{n_+}{n}$
Ionosphere	–	34	176	64.2%	87	64.4%	88	63.6%
Spambase	–	57	2 301	39.4%	1 150	39.4%	1 150	39.4%
WhiteWineQuality	7, 8, 9	11	2 449	21.6%	1 224	21.7%	1 225	21.6%
RedWineQuality	7, 8	11	800	13.5%	400	13.8%	399	13.5%
Fashion-MNIST	0	784	50 000	10.0%	10 000	10.0%	10 000	10.0%

Table 6.3: 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 $\frac{n_+}{n}$.

6.2.3 Dataset description

For numerical experiments, we consider the FashionMNIST dataset [56] and four smaller datasets from the UCI repository [57]: sigillito1989classification [50], Spambase, WhiteWineQuality [58] and RedWineQuality [58]. 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 6.3.

6.2.4 Experiments

In Figure 6.5 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 (6.3) while the right one to the Gaussian kernel (6.4) with $\sigma = 0.01$. The nonlinear Gaussian kernel significantly outperforms the linear kernel. This will be confirmed later in Table 6.4 where we present a comparison from multiple datasets.

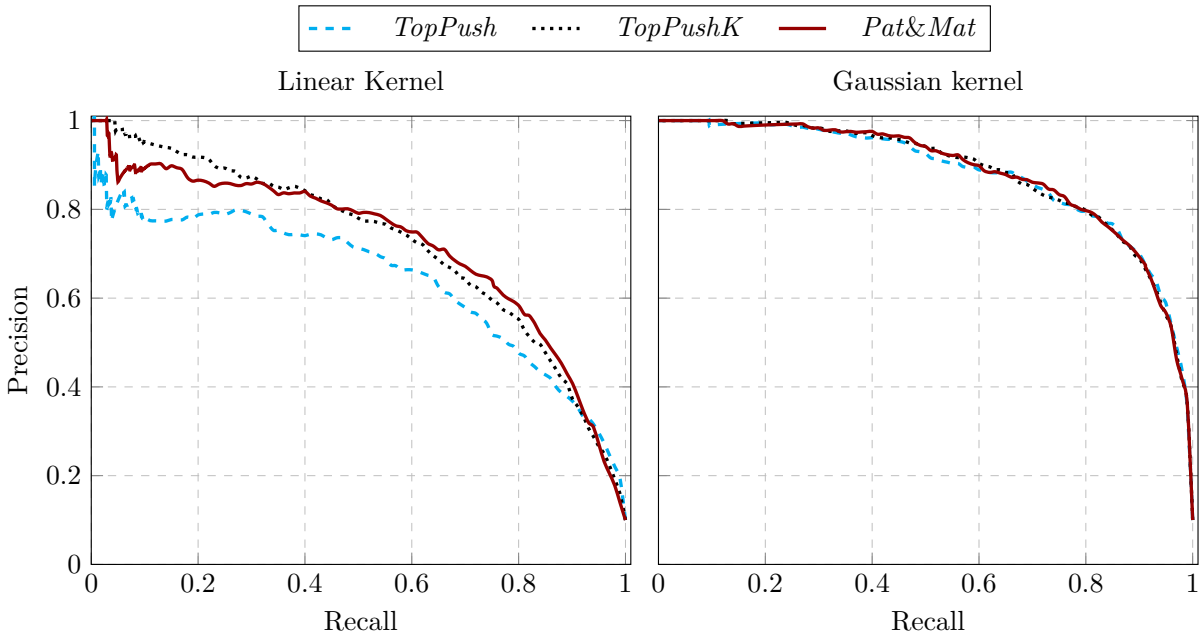


Figure 6.5: PR curves for all methods and FashionMNIST dataset. The left column corresponds to the linear kernel (6.3) and the right column corresponds to the Gaussian kernel (6.4).

For a better illustration of how the methods from Figure 6.5 work, we present density estimates of scores $\boldsymbol{s}$ from (**??**). High scores predict positive labels while low scores predict negative labels. The rows of Figure 6.6 depict the linear (6.3) and the Gaussian kernels (6.4) 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 6.4 we present the precision of all methods across all datasets from Table 6.3. 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 6.4:

- All methods perform better with the Gaussian kernels than with the linear kernel.
- *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 6.7, 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 [21]. For the dual problem, we use our Algorithm 3. Since [21] 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 6.5 depicts the time comparison for all methods and all datasets. It shows the average time in milliseconds needed for one `repeat` loop in Algorithm 3. The time is relatively

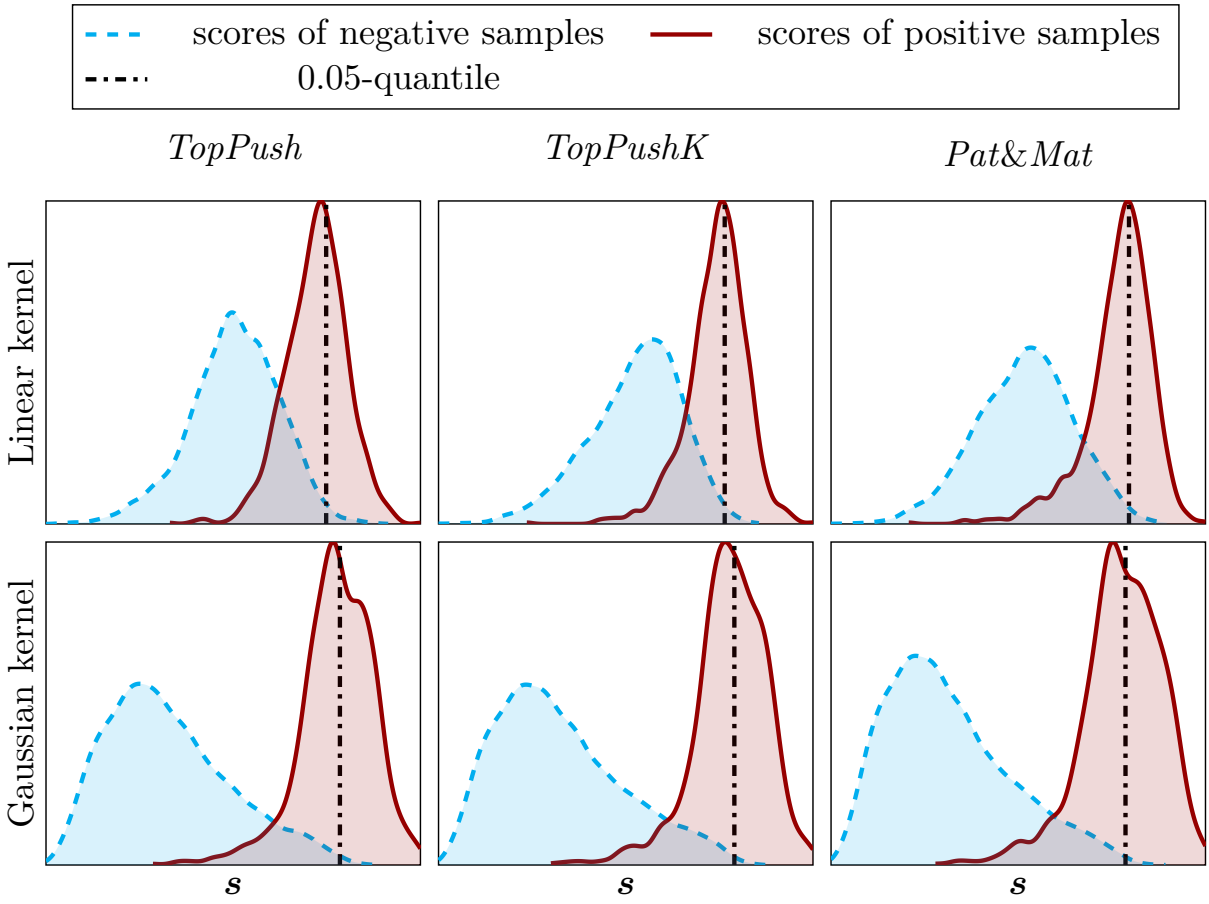


Figure 6.6: Density estimates for scores corresponding to samples with positive and negative labels for the FashionMNIST dataset.

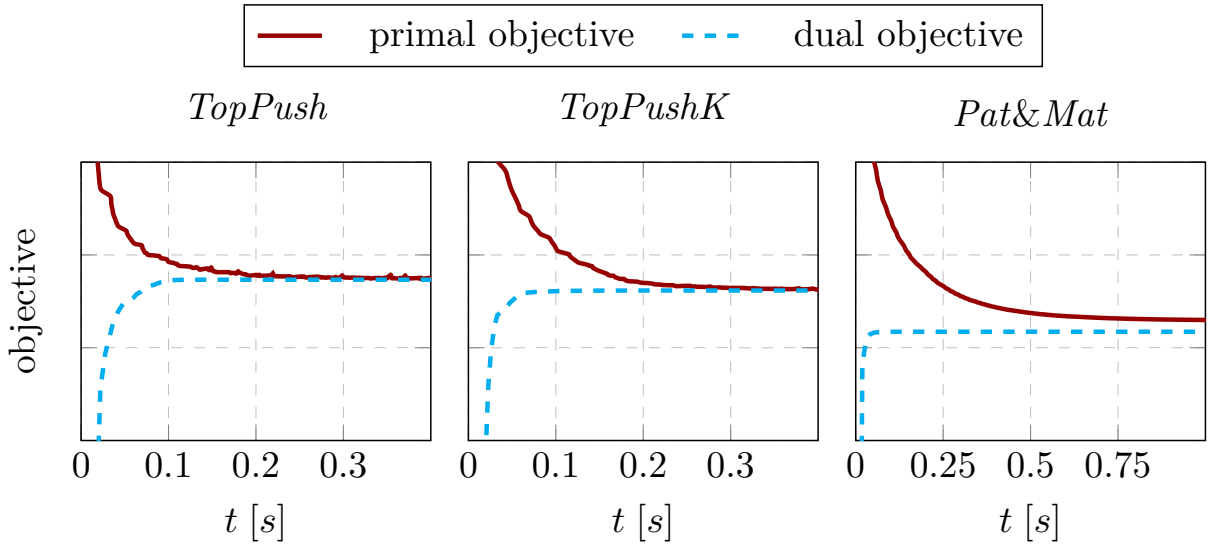


Figure 6.7: Convergence of the objectives for the primal (red line) and dual (blue line) problems for the sigillito1989classification dataset with linear kernel.

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

	Method		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 6.4: The precision of all methods averaged across all datasets from Table 6.3. 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.

time for one Δ_l in step 6 in Algorithm 3 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 6 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.

6.3 Neural Networks

This section presents numerical results for *DeepTopPush*. Table **??** 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_τ-largest negative. The former may be understood as requiring no false-positives, while the latter allows for false positive rate τ.

Section 5.2.3 showed that we can use large minibatches to obtain good results for *Pat&Mat-NP* for small fractions of top samples τ. Section 5.2.4 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 5 works.

	Dataset	*TopPush*	*TopPushK*	*Pat&Mat*
One repeat loop [ms]	Ionosphere	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 6.5: The average time with standard deviation (in milliseconds) for one `repeat` loop in Algorithm 3. The average time for one Δ_l in step 6 in Algorithm 3 took between $1.7 \cdot 10^{-7}$ and $3.1 \cdot 10^{-7}$ seconds for each methods.

Dataset	d	Train		Test		Licence
		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 6.6: 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.

6.3.1 Dataset description and Computational setting

We consider the following image recognition datasets: FashionMNIST [56], CIFAR100 [59], SVHN2 [60] and ImageNet [61]. 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 [62] 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. Table 6.6 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.

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 [54]. The network structure was the

same for all methods; we describe them in the online appendix.

6.3.2 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.

6.3.3 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* [43, 63] and *Ap-Perf* [46]. We did not implement the original TopPush because its duality arguments restrict the classifiers to only linear ones. Table 6.7 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 [46]. 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*.

Method	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 6.7: Time requirements per epoch for investigated methods for minibatches of size $n_{\text{mb}} = 32$.

Table 6.8 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,

	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 6.8: 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.

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 6.9 **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 [64] by removing the last dense layer and adding another dense 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.

6.3.4 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 6.8 shows again the results at the top. *DeepTopPush* outperforms other methods. Figure 6.8 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.

6.3.5 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

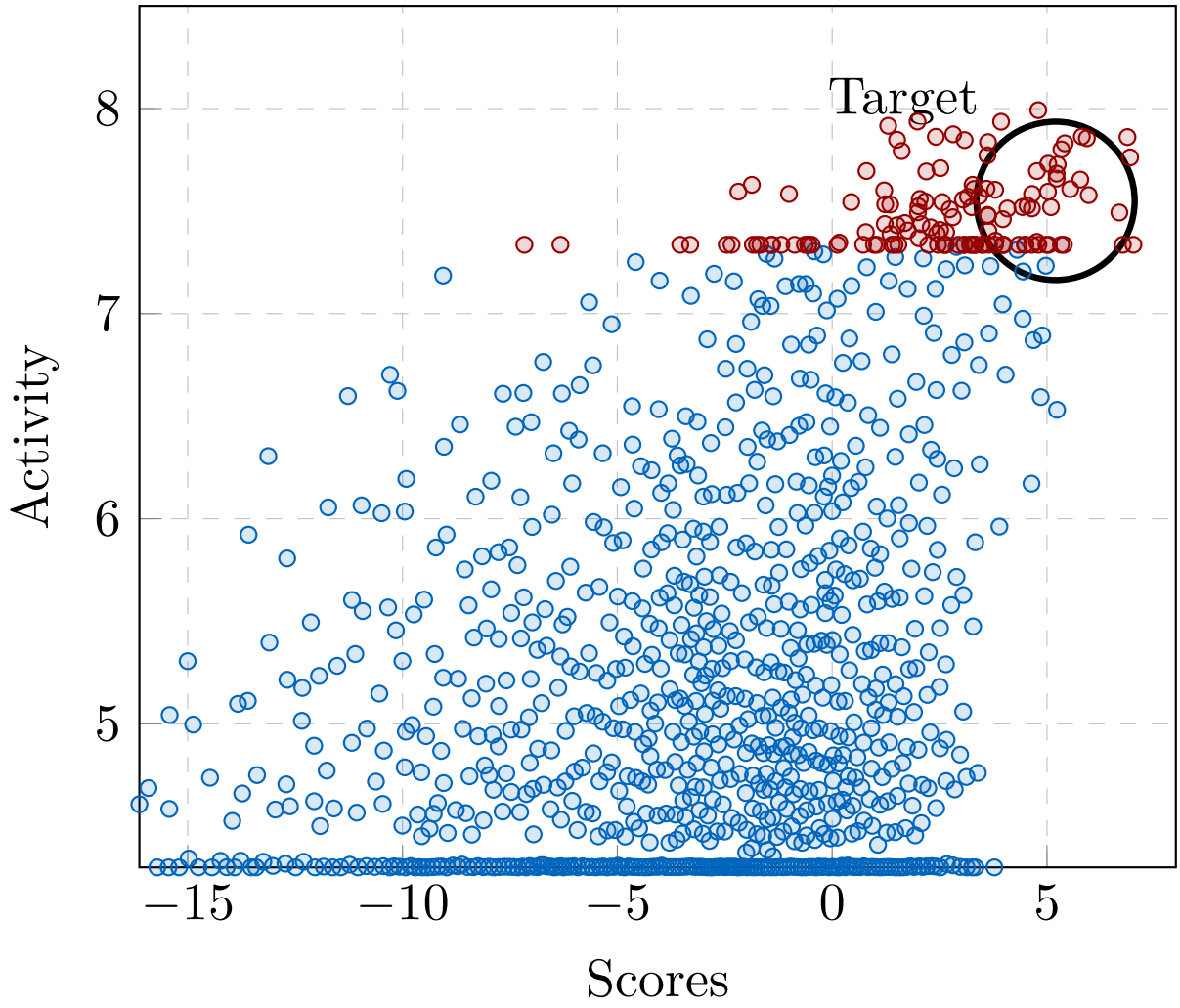


Figure 6.8: Results for the 3A4 dataset. The goal was to assign large scores to a few molecules with high activity (scores on top-right are preferred).

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) [65]. Then we applied *DeepTopPush* 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 5.2.3.

Figure 6.9 **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* ($\tau = 10^{-2}$).

6.3.6 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.

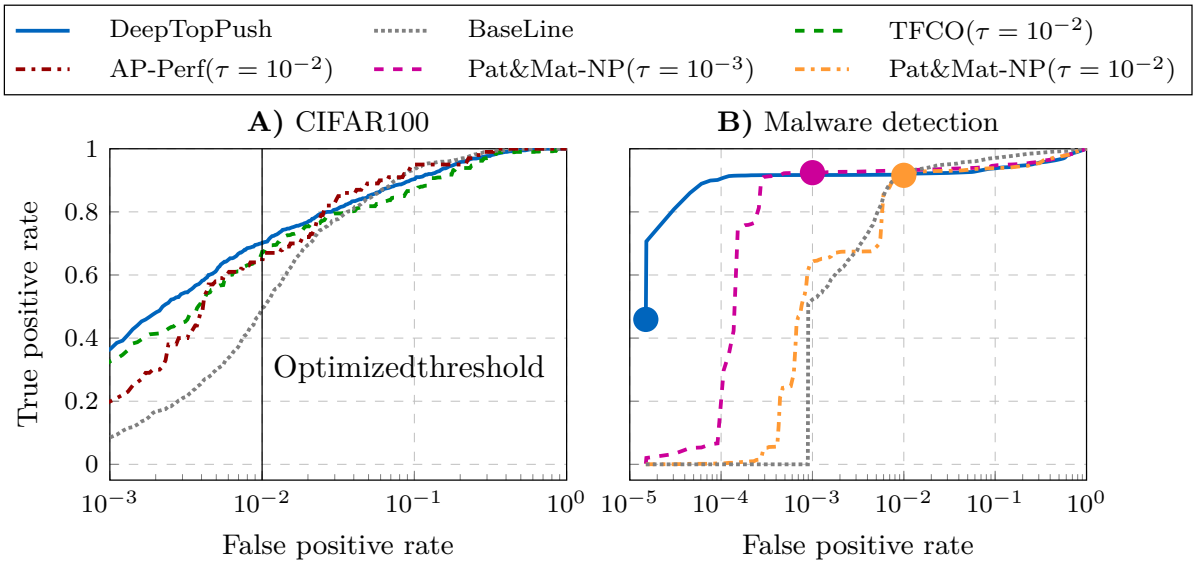


Figure 6.9: **A)** ROC curves averaged over ten runs on the CIFAR100 dataset. **B)** ROC curve for Malware Detection dataset. The circles show the thresholds the methods were optimized for.

Then the discussion in Section 5.2.4 implies that *Pat&Mat-NP* equals to *DeepTopPush* without enhancing the minibatch. In other words, *Pat&Mat-NP* uses Algorithm 4 while *DeepTopPush* uses Algorithm 5. As Table 6.8 clearly shows that *DeepTopPush* ourperforms *Pat&Mat-NP*, this implies that using the delayed values is beneficial.

Figure 6.10 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 5.3 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 [66] 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.

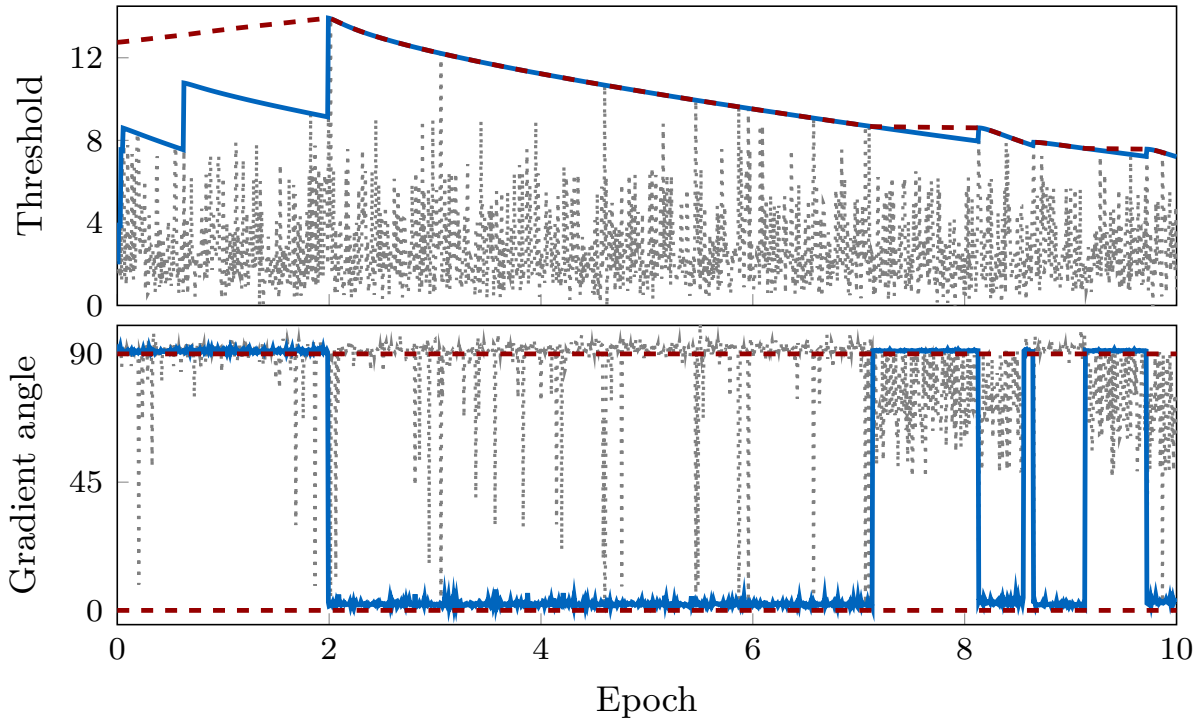


Figure 6.10: The thresholds (top) and angle between true and sampled gradients (bottom) for Algorithm 4 (full blue) and Algorithm 5 (dotted gray).

7 Conclusion

7.1 Linear Model

In this paper, we achieved the following results:

- We presented a unified framework for the three criteria from Chapter 2. 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*.

7.2 Dual

In this paper, we analyzed and extended the general framework for binary classification on top samples from [21] to nonlinear problems. Achieved results can be summarized as 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 6.2, 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.

7.3 Neural Networks

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

Apendices

A

Appendix for Chapter 3

Firstly we recall definitions of the thresholds defined in Section 2.3

$$t_1(\boldsymbol{w}) = \max\left\{ t \;\middle|\; \frac{1}{n}\sum_{i\in\mathcal{I}} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}$$
$$t_2(\boldsymbol{w}) = \frac{1}{K}\sum_{i=1}^{K} s_{[i]}$$
$$t_3(\boldsymbol{w}) \quad \text{solves} \quad \frac{1}{n}\sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) = \tau,$$

and also recall, that we assume linear classifier, i.e. scores are defined for all $i \in \mathcal{I}$ as $s_i = \boldsymbol{w}^\top \boldsymbol{x}_i$.

A.1 Convexity

Proposition 3.1

Thresholds t_2 from (2.10) and t_3 from (2.12) are convex functions of the weights $\boldsymbol{w}$. The threshold function t_1 from (2.7) is non-convex.

Proof Proposition 3.1 on page 22:
It is easy to show that the quantile t_1 is not convex. Due to [27], 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. Let us dfefine function g as follows

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

where we for simplicity set $\vartheta = 1$. Then t_3 is defined via an implicit equation $g(\boldsymbol{w}, t) = 0$. Moreover, since l is convex, we immediately obtain that g is jointly convex in both variables. To show the convexity, consider $\boldsymbol{w}$, $\hat{\boldsymbol{w}} \in \mathbb{R}^d$ and the corresponding thresholds $t = t_3(\boldsymbol{w})$, $\hat{t} = t_3(\hat{\boldsymbol{w}})$. Then for any $\lambda \in [0, 1]$ we have

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

where the inequality follows from the convexity of g and the equality from

$$g(\boldsymbol{w}, t) = g(\hat{\boldsymbol{w}}, \hat{t}) = 0,$$

which holds true since both t and $\hat{t}$ solves (2.12). From the definition of the surrogate quantile function t_3 we have

$$g(\lambda w + (1-\lambda)\hat{w}, t_3(\lambda w + (1-\lambda)\hat{w})) = 0. \tag{A.2}$$

Since g is non-increasing in the second variable, from (A.1) and (A.2) we deduce

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

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

Theorem 3.2

If the threshold $t = t(w)$ is a convex function of the weights w, then function

$$L(w) = \overline{\text{fn}}(s, t)$$

is convex.

Proof of Theorem 3.2 on page 22:
Due to the definition of the surrogate counts (2.2), the function L equals to

$$L(w) = \overline{\text{fn}}(s, t(w)) = \sum_{i \in \mathcal{I}_+} l\Big(t(w) - w^\top x_i\Big).$$

Here we write $t(w)$ to stress the dependence of t on w. Since $w \mapsto t(w)$ is a convex function, we also have that $w \mapsto t(w) - w^\top x$ 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.2 Differentiability

Theorem 3.3

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

$$\nabla t_3(w) = \frac{\sum_{i \in \mathcal{I}} l'(\vartheta(w^\top x_i - t_3(w)))x_i}{\sum_{i \in \mathcal{I}} l'(\vartheta(w^\top x_i - t_3(w)))}.$$

The threshold functions t_1 and t_2 are non-differentiable.

Proof of Theorem 3.3 on page 22:
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.3 Stability

In this section, we derive the results presented from Section 3.3 more properly.

Example 3.4: Degenerate Behaviour

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 3.1 (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 $\boldsymbol{w}=(1,0)$.

Moreover, 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

$$\tau=\frac{1}{n}\sum_{i\in\mathcal{I}} l\Big(\vartheta(\boldsymbol{w}_0^\top\boldsymbol{x}_i-t)\Big)=l(0-\vartheta t)=1-\vartheta t,$$

which implies

$$t=1-\tau/\vartheta$$

and consequently the value of the objective function is

$$L(\boldsymbol{w}_0)=\frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l(t-\boldsymbol{w}_0^\top\boldsymbol{x}_i)=l(t-0)=1+t, \tag{A.3}$$

where the last equality follows from definition of the hinge loss function and the fact that $t\geq 0$. This finishes the computation for $\boldsymbol{w}_0$. For $\boldsymbol{w}_1=(1,0)$ the computation goes similar. Since $\boldsymbol{w}_1^\top\boldsymbol{x}$ for $i\in\mathcal{I}$ has the uniform distribution on $[-1,1]$, we have

$$\tau=\frac{1}{n}\sum_{i\in\mathcal{I}} l\Big(\vartheta(\boldsymbol{w}_1^\top\boldsymbol{x}_i-t)\Big)\approx\frac{1}{2}\int_{-1}^{1} l(\vartheta(s-t))\,\mathrm{d}s=\frac{1}{2}\int_{-1}^{1}\max\{0,1+\vartheta(s-t)\}\,\mathrm{d}s$$

If $\vartheta\leq\tau$, then

$$1+\vartheta(s-t)\geq 1+\vartheta(-1-t)=1-\vartheta-1+\tau=\tau-\beta\geq 0.$$

Using this inequality, we can ignore the max operator in the relation for the τ above and get

$$\tau=\frac{1}{2}\int_{-1}^{1}(1+\vartheta(s-t))\,\mathrm{d}s=1-\vartheta t+\frac{\vartheta}{2}\int_{-1}^{1}s\,\mathrm{d}s=1-\vartheta t, \tag{A.4}$$

and thus again $t=1-\tau/\vartheta$. Finally, since $\boldsymbol{w}_1^\top\boldsymbol{x}$ for $i\in\mathcal{I}_+$ has the uniform distribution on $[0,1]$, we have

$$L(\boldsymbol{w}_1)=\frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l(t-\boldsymbol{w}_1^\top\boldsymbol{x}_i)\approx\int_0^1 l(t-s)\,\mathrm{d}s=\int_0^1(1+t-s)\,\mathrm{d}s=0.5+t.$$

Results for *Pat&Mat-NP* can be obtained in a similar way. In the rest of the section we provide proofs for all the theorems from Section 3.3.

Theorem 3.5

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

$$t(\boldsymbol{w})\geq\frac{1}{n_+}\sum_{i\in\mathcal{I}_+}\boldsymbol{w}^\top\boldsymbol{x}_i, \tag{3.1}$$

then $L(\mathbf{0}) \leq L(\boldsymbol{w})$. Specifically, using notation 2.2 we get the following implications

$$\begin{aligned}
s^-_{[1]} \geq \frac{1}{n_+}\sum_{i=1}^{n_+} s^+_i &\implies L(\mathbf{0}) \leq L(\boldsymbol{w}) \text{ for } \textit{TopPush},\\
\frac{1}{K}\sum_{i=1}^{K} s^-_{[i]} \geq \frac{1}{n_+}\sum_{i=1}^{n_+} s^+_i &\implies L(\mathbf{0}) \leq L(\boldsymbol{w}) \text{ for } \textit{TopPushK},\\
\frac{1}{K}\sum_{i=1}^{K} s_{[i]} \leq \sum_{i=1}^{n_+} s^+_i &\implies L(\mathbf{0}) \leq L(\boldsymbol{w}) \text{ for } \textit{TopMeanK},\\
\frac{1}{n_\tau}\sum_{i=1}^{n_\tau} s^-_{[i]} \geq \frac{1}{n_+}\sum_{i=1}^{n_+} s^+_i &\implies L(\mathbf{0}) \leq L(\boldsymbol{w}) \text{ for } \tau\textit{-FPL}.
\end{aligned} \tag{3.2}$$

Proof of Theorem 3.5 on page 23:
All mentioned formulations use surogate approximation of the false-negative rate as the objective function L. For the linear classifier, the objective function has the following form

$$L(\boldsymbol{w}) = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l(t - \boldsymbol{w}^\top \boldsymbol{x}_i)$$

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

$$L(\boldsymbol{w}) \geq \frac{1}{n_+}\sum_{i\in\mathcal{I}_+}(1 + c(t - \boldsymbol{w}^\top \boldsymbol{x}_i)) = 1 + c\left(t - \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} \boldsymbol{w}^\top \boldsymbol{x}\right) \geq 1,$$

where the last inequality follows from assumption (3.1). Now we realize that for any formulation from the statement, the corresponding threshold for $\boldsymbol{w} = 0$ equals to $t = 0$, and thus $L(\mathbf{0}) = 1$. But then $L(\mathbf{0}) \leq L(\boldsymbol{w})$. The second part of the result follows from the form of thresholds $t(\boldsymbol{w})$. ■

Theorem 3.8

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

$$\frac{1}{n_+}\sum_{i\in\mathcal{I}_+} \boldsymbol{w}^\top \boldsymbol{x}_i > \frac{1}{n_-}\sum_{j\in\mathcal{I}_-} \boldsymbol{w}^\top \boldsymbol{x}_j. \tag{3.3}$$

Then there is a scaling parameter ϑ_0 for the surrogate top τ-quantile (2.12) such that $L(\boldsymbol{w}) < L(\mathbf{0})$ for all $\vartheta \in (0, \vartheta_0)$.

Proof of Theorem 3.8 on page 25:
Firstly recall thewe use linear model and Notation 2.2 and define the following auxilliary variables

$$s_{\min} = \min\{s_i \mid i \in \mathcal{I}\}, \qquad s_{\max} = \max\{s_i \mid i \in \mathcal{I}\}, \qquad \bar{s} = \frac{1}{n}\sum_{i\in\mathcal{I}} s_i.$$

Using the definition of $\bar{s}$ we get the following relation

$$\bar{s} = \frac{1}{n}\sum_{i\in\mathcal{I}} s_i = \frac{1}{n}\sum_{i\in\mathcal{I}_+} s_i + \frac{1}{n}\sum_{i\in\mathcal{I}_-} s_i < \frac{1}{n}\sum_{i\in\mathcal{I}_+} s_i + \frac{n_-}{nn_+}\sum_{i\in\mathcal{I}_+} s_i = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} s_i, \tag{A.5}$$

where the inequality follows from (3.3) and the last equality follows from

$$\frac{1}{n} + \frac{n_-}{nn_+} = \frac{1}{n}\left(1 + \frac{n_-}{n_+}\right) = \frac{1}{n}\frac{n_+ + n_-}{n_+} = \frac{1}{n}\frac{n}{n_+} = \frac{1}{n_+}.$$

Moreover, since the average of elements of the vector is smaller or equal to the maximum of elements of the same vector, we get the following relation

$$\bar{s} < \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} s_i \leq \max\{s_i \mid i \in \mathcal{I}_+\} \leq \max\{s_i \mid i \in \mathcal{I}\} = s_{\max}$$

where the first inequality follows from (A.5). The lower bound for $\bar{s}$ can be computed in a similar way. Altogether, we have $s_{\min} < \bar{s} < s_{\max}$. Then we can define

$$\vartheta_0 = \min\left\{\frac{\tau}{\bar{s} - s_{\min}}, \frac{1-\tau}{s_{\max} - \bar{s}}, \tau\right\},$$

observe that $\vartheta_0 > 0$, fix any $\vartheta \in (0, \vartheta_0)$ and define

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

Then we obtain for any $i \in \mathcal{I}$

$$1 + \vartheta(s_i - t) \geq 1 + \vartheta(s_{\min} - t) = 1 + \vartheta s_{\min} - 1 + \tau - \vartheta\bar{s} = \tau - \vartheta(\bar{s} - s_{\min}),$$

where the first equality follows from the definition of t. From the definition ϑ_0 we known the following

$$0 < \vartheta \leq \vartheta_0 \leq \frac{\tau}{\bar{s} - s_{\min}}.$$

Since $\bar{s} - s_{\min} > 0$, we get the following inequality

$$1 + \vartheta(s_i - t) = \tau - \vartheta(\bar{s} - s_{\min}) \geq \tau - \frac{\tau}{\bar{s} - s_{\min}}(\bar{s} - s_{\min}) = 0 \tag{A.6}$$

Moreover, combining the definition of the hinge loss function in Notation 2.1 and the inequality above, we have

$$l(\vartheta(s_i - t)) = \max\{0, 1 + \vartheta(s_i - t), 0\} = (1 + \vartheta(s_i - t)).$$

Finally, replacing the hinge loss in the left hand side of (2.12) leads to

$$\begin{aligned}\frac{1}{n}\sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) &= \frac{1}{n}\sum_{i\in\mathcal{I}}(1 + \vartheta(s_i - t)) \\ &= 1 - \vartheta t + \frac{\vartheta}{n}\sum_{i\in\mathcal{I}} s_i \\ &= 1 - \vartheta\left(\frac{1-\tau}{\vartheta} + \bar{s}\right) + \vartheta\bar{s} \\ &= \tau,\end{aligned}$$

where the third equality employs the definition of $\bar{s}$ and t. But this means that t is the threshold corresponding to $\boldsymbol{w}$, i.e. it solves (2.12). Similarly to (A.6) we get

$$1 + t - s_i \geq 1 + t - s_{\max} = 1 + \frac{1-\tau}{\vartheta} + \bar{s} - s_{\max} \geq \frac{1-\tau}{\vartheta} + \bar{s} - s_{\max} \geq 0, \tag{A.7}$$

where the last inequality follows from the definition of ϑ_0. Then for the objective we have

$$\begin{aligned}L(\boldsymbol{w}) = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l(t - s_i) &= \frac{1}{n_+}\sum_{i\in\mathcal{I}_+}(1 + t - s_i) \\ &= 1 + t - \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} s_i \\ &< 1 + \left(\frac{1-\tau}{\vartheta} + \bar{s}\right) - \bar{s} \\ &= 1 + \frac{1-\tau}{\vartheta} \\ &= L(\mathbf{0}),\end{aligned}$$

where the second equality follows from (A.7), the only inequality from (A.5) and the last equality from (A.3) and (A.3). 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.4 Threshold comparison

Whenever the objective contains only false-negatives, a lower threshold t means a lower objective function. Therefore, a lower threshold is preferred. The two following lemmas compares thresholds defined in Chapter 2 in terms of approximation quality.

Lemma A.7: Thresholds relation [30]

We always have

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

Lemma A.8

Consider the *Grill*, *Grill-NP*, *TopMeanK* and τ-*FPL* formulations and the notation from

Notation 2.2. 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.8 hold true.

A.5 Computing the threshold for *Pat&Mat*

We show how to efficiently compute the threshold (2.12) for *Pat&Mat* with linear model and the hinge surrogate from Notation 2.1. Consider function

$$h(t) = \sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) - n\tau. \tag{A.8}$$

Then solving (3.8) is equivalent to looking for $\hat{t}$ such that $h(\hat{t}) = 0$. 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

Consider vector of scores $\boldsymbol{s}$ and its sorted version $\boldsymbol{s}_{[\cdot]}$ with decreasing elements as defined in Notation 2.2. Define $\gamma = 1/\vartheta$. Then

$$h(s_{[j]} + \gamma) = h(s_{[j-1]} + \gamma) + (j-1)\vartheta(s_{[j-1]} - s_{[j]}) \tag{A.9}$$

for all $i = 2, 3, \ldots, n$ with the initial condition $h(s_{[1]} + \gamma) = -n\tau$.

Proof:
Observe first that

$$\begin{aligned} h(s_{[j]} + \gamma) &= \sum_{i\in\mathcal{I}} l\Big(\vartheta(s_i - (s_{[j]} + \gamma))\Big) - n\tau \\ &= \sum_{i\in\mathcal{I}} \max\left\{0,\ 1 + \vartheta\left(s_i - s_{[j]} - \frac{1}{\gamma}\right)\right\} - n\tau \\ &= \sum_{i=1}^{j-1} \vartheta(s_{[i]} - s_{[j]}) - n\tau, \end{aligned}$$

where the last equality holds since $\vartheta > 0$ and $s_{[i]} - s_{[j]} \leq 0$ for all $i \geq j$ From here, we ob-

tain $h(s_{[1]}+\gamma) = -n\tau$. Moreover, we have

$$\begin{aligned}
h(s_{[j]}+\gamma) &= \sum_{i=1}^{j-1} \vartheta(s_{[i]} - s_{[j]}) - n\tau \\
&= \sum_{i=1}^{j-2} \vartheta(s_{[i]} - s_{[j]}) + \vartheta(s_{[j-1]} - s_{[j]}) - n\tau \\
&= \sum_{i=1}^{j-2} \vartheta(s_{[i]} - s_{[j]} \pm s_{[j-1]}) + \vartheta(s_{[j-1]} - s_{[j]}) - n\tau \\
&= \sum_{i=1}^{j-2} \vartheta(s_{[i]} - s_{[j-1]}) + \sum_{i=1}^{j-2} \vartheta(s_{[j-1]} - s_{[j]}) + \vartheta(s_{[j-1]} - s_{[j]}) - n\tau \\
&= h(s_{[j-1]}+\gamma) + (j-1)\vartheta(s_{[j-1]} - s_{[j]}),
\end{aligned}$$

which finishes the proof. ∎

Thus, to solve $h(t) = 0$ with the hinge surrogate, we start with $t_1 = s_{[1]}+\gamma$ and $h(t_1) = -n\tau$. 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

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

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

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

A.6 Convergence of stochastic gradient descent

The proof is divided into three parts. In Section A.6.1, we prove a general statement for convergence of stochastic gradient descent with a convex objective. In Section A.6.2 we apply it to Theorem 3.9. The proof is based on auxiliary results from Section A.6.3.

A.6.1 General result

Consider a differentiable objective function L and the optimization method

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

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

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

(A2) $\left\|g(\boldsymbol{w}^k)\right\| \le B$ 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 $\alpha^k = \alpha^0/k+1$. 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 $\boldsymbol{w}^*$ of L we have

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

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

Proof:

Note first that the convexity of L from (A1) implies

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

Then we have

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

where the first inequality follows from assumption (A2) and the second on from the properties of inner product and (A.12). Summing this expression for all k and using (A.11) leads to

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

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

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

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

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

Since $L(\boldsymbol{w}^k) - L(\boldsymbol{w}^*) \geq 0$, due to assumption (A3) we obtain

$$\lim_{k\to\infty} L(\boldsymbol{w}^k) = L(\boldsymbol{w}^*),$$

which implies the theorem statement. ■

A.6.2 Proof of Theorem 3.9

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) l'/l is a decreasing function on (s_0, ∞);

(S4) l' is a bounded function;

(S5) l' is a Lipschitz continuous function with Lipschitz constant D.

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 3.9

Consider the *Pat&Mat* or *Pat&Mat-NP* formulation, stepsizes $\alpha^k = \alpha^0/k+1$ and piecewise disjoint minibatches $\mathcal{I}^1_{\text{mb}}$, $\mathcal{I}^2_{\text{mb}}$,..., $\mathcal{I}^m_{\text{mb}}$ which cycle periodically $\mathcal{I}^{k+m}_{\text{mb}} = \mathcal{I}^k_{\text{mb}}$. If l is the smoothened (Huberized) hinge function, then Algorithm 1 converges to the global minimum of (2.13).

Proof of Theorem 3.9 on page 27:

We intend to apply Theorem A.10 and thus, we need to verify its assumptions. Assumption (A1) is satisfied as L is convex due to Theorem 3.2. 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 $\vartheta = 1$ and for 2 minibatches of the same size. However, the proof would be identical for other values. This implies that there are some $\mathcal{I}^k_{\text{mb}}$ and $\mathcal{I}^{k+1}_{\text{mb}}$ which are pairwise disjoint, they cover all samples and that $\mathcal{I}^k_{\text{mb}} = \mathcal{I}^{k+2}_{\text{mb}}$ for all k. The assumptions imply that the number of positive samples in each minibatch equal to $n^k_{\text{mb},+} = n_+/2$, where n_+ is the total number of positive samples.

First we estimate the difference between s^k_i defined in (3.7) and $\boldsymbol{x}_i^\top \boldsymbol{w}^k$. For any $i \in \mathcal{I}^k_{\text{mb}}$ we have

$$s^k_i = \boldsymbol{x}_i^\top \boldsymbol{w}^k$$

and since we have two disjoint minibatches, due to the construction (3.7) we get

$$\begin{aligned} s^{k-1}_i = s^{k-2}_i &= \boldsymbol{x}_i^\top \boldsymbol{w}^{k-2} \\ &= \boldsymbol{x}_i^\top \left(\boldsymbol{w}^k + \alpha^{k-2} g(\boldsymbol{w}^{k-2}) + \alpha^{k-1} g(\boldsymbol{w}^{k-1})\right) \\ &= \boldsymbol{x}_i^\top \boldsymbol{w}^k + \alpha^{k-2} \boldsymbol{x}_i^\top g(\boldsymbol{w}^{k-2}) + \alpha^{k-1} \boldsymbol{x}_i^\top g(\boldsymbol{w}^{k-1}). \end{aligned} \tag{A.14}$$

Similarly, due to the construction (3.7), for $i \notin \mathcal{I}^k_{\text{mb}}$ we have

$$s^k_i = s^{k-1}_i = \boldsymbol{x}_i^\top \boldsymbol{w}^{k-1} = \boldsymbol{x}_i^\top (\boldsymbol{w}^k + \alpha^{k-1} g(\boldsymbol{w}^{k-1})) = \boldsymbol{x}_i^\top \boldsymbol{w}^k + \alpha^{k-1} \boldsymbol{x}_i^\top g(\boldsymbol{w}^{k-1}). \tag{A.15}$$

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 $i \in \mathcal{I}$ we have

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

This also immediately implies

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

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

$$\left\|l'(t^k - s_i^k) - l'(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k)\right\| \leq D\left\|t^k - s_i^k - t(\boldsymbol{w}^k) + \boldsymbol{x}_i^\top \boldsymbol{w}^k\right\| \leq 2C_2 D\alpha^{k-1}. \tag{A.18}$$

In an identical way we can show

$$\begin{aligned}
\left\|l'(t^{k-1} - s_i^{k-1}) - l'(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k)\right\| &\leq 2C_2 D\left(\alpha^{k-1} + \alpha^{k-2}\right), \\
\left\|l'(s_i^k - t^k) - l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k))\right\| &\leq 2C_2 D\alpha^{k-1}, \\
\left\|l'(s_i^{k-1} - t^{k-1}) - l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k))\right\| &\leq 2C_2 D\left(\alpha^{k-1} + \alpha^{k-2}\right).
\end{aligned} \tag{A.19}$$

Now we need to estimate the distance between $\nabla t(\boldsymbol{w}^k)$ and ∇t^k. From (3.11) and (3.9), we have

$$\nabla t^k = \frac{\sum_{i \in \mathcal{I}_{\text{mb}}^k} l'(s_i^k - t^k)\boldsymbol{x}_i + \sum_{i \in \mathcal{I}_{\text{mb}}^{k-1}} l'(s_i^{k-1} - t^{k-1})\boldsymbol{x}_i}{\sum_{i \in \mathcal{I}} l'(s_i^k - t^k)}.$$

Moreover, using Theorem 3.3 and the fact that we have only two minibatches and therefore for any k we have $\mathcal{I} = \mathcal{I}_{\text{mb}}^k \cup \mathcal{I}_{\text{mb}}^{k-1}$, we get

$$\nabla t(\boldsymbol{w}^k) = \frac{\sum_{i \in \mathcal{I}_{\text{mb}}^k} l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k))\boldsymbol{x}_i + \sum_{i \in \mathcal{I}_{\text{mb}}^{k-1}} l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k))\boldsymbol{x}_i}{\sum_{i \in \mathcal{I}} l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k))}.$$

From Lemma A.12 we deduce that the denominators in the relations above 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

$$\left\|\nabla t^k - \nabla t(\boldsymbol{w}^k)\right\| \leq C_3\left(\alpha^{k-1} + \alpha^{k-2}\right). \tag{A.20}$$

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

$$\begin{aligned}
g(\boldsymbol{w}^k) &= \frac{2}{n_+} \sum_{i \in \mathcal{I}_{\text{mb},+}^k} l'(t^k - s_i^k)(\nabla t^k - \boldsymbol{x}_i), \\
g(\boldsymbol{w}^{k+1}) &= \frac{2}{n_+} \sum_{i \in \mathcal{I}_{\text{mb},+}^{k+1}} l'(t^{k+1} - s_i^{k+1})(\nabla t^{k+1} - \boldsymbol{x}_i), \\
\nabla L(\boldsymbol{w}^k) &= \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} l'(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k)(\nabla t(\boldsymbol{w}^k) - \boldsymbol{x}_i), \\
\nabla L(\boldsymbol{w}^{k+1}) &= \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} l'(t(\boldsymbol{w}^{k+1}) - \boldsymbol{x}_i^\top \boldsymbol{w}^{k+1})(\nabla t(\boldsymbol{w}^{k+1}) - \boldsymbol{x}_i).
\end{aligned}$$

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

write

$$n_+\Big(g(w^k)+g(w^{k+1})-\nabla f(w^k)-\nabla f(w^{k+1})\Big) \tag{A.21a}$$
$$= \sum_{i\in\mathcal{I}^k_{\mathrm{mb},+}} l'(t^k-s_i^k)(\nabla t^k-x_i)-\sum_{i\in\mathcal{I}^k_{\mathrm{mb},+}} l'(t(w^k)-x_i^\top w^k)(\nabla t(w^k)-x_i) \tag{A.21b}$$
$$+ \sum_{i\in\mathcal{I}^k_{\mathrm{mb},+}} l'(t^k-s_i^k)(\nabla t^k-x_i)-\sum_{i\in\mathcal{I}^k_{\mathrm{mb},+}} l'(t(w^{k+1})-x_i^\top w^{k+1})(\nabla t(w^{k+1})-x_i) \tag{A.21c}$$
$$+ \sum_{i\in\mathcal{I}^{k+1}_{\mathrm{mb},+}} l'(t^{k+1}-s_i^{k+1})(\nabla t^{k+1}-x_i)-\sum_{i\in\mathcal{I}^{k+1}_{\mathrm{mb},+}} l'(t(w^k)-x_i^\top w^k)(\nabla t(w^k)-x_i) \tag{A.21d}$$
$$+ \sum_{i\in\mathcal{I}^{k+1}_{\mathrm{mb},+}} l'(t^{k+1}-s_i^{k+1})(\nabla t^{k+1}-x_i)-\sum_{i\in\mathcal{I}^{k+1}_{\mathrm{mb},+}} l'(t(w^{k+1})-x_i^\top w^{k+1})(\nabla t(w^{k+1})-x_i). \tag{A.21e}$$

Then relations (A.20) and (A.18) applied to Lemma A.15 imply

$$\left\| \sum_{i\in\mathcal{I}^k_{\mathrm{mb},+}} l'(t^k-s_i^k)(\nabla t^k-x_i)-\sum_{i\in\mathcal{I}^k_{\mathrm{mb},+}} l'(t(w^k)-x_i^\top w^k)(\nabla t(w^k)-x_i)\right\| \leq C_4\Big(\alpha^{k-1}+\alpha^{k-2}\Big)$$

for some C_4, which gives a bound for (A.21b). Bound for (A.21e) is obtained by increasing k by one. Bounds for (A.21c) and (A.21d) can be find similarly using (A.19). Altogether, we showed

$$\left\|g(w^k)+g(w^{k+1})-\nabla L(w^k)-\nabla L(w^{k+1})\right\| \leq C_1(\alpha^{k-2}+\alpha^{k-1}+\alpha^k+\alpha^{k+1}) \tag{A.22}$$

for some C_1. We now estimate

$$\begin{aligned}
&\alpha^k\Big\langle g(w^k)-\nabla L(w^k),\, w^*-w^k\Big\rangle+\alpha^{k+1}\Big\langle g(w^{k+1})-\nabla L(w^{k+1}),\, w^*-w^{k+1}\Big\rangle\\
&\quad=\Big\langle g(w^k)-\nabla L(w^k),\, \alpha^k(w^*-w^k)\Big\rangle+\Big\langle g(w^{k+1})-\nabla L(w^{k+1}),\, \alpha^{k+1}(w^*-w^{k+1})\Big\rangle\\
&\quad=\Big\langle g(w^k)-\nabla L(w^k)+g(w^{k+1})-\nabla L(w^{k+1}),\, \alpha^k(w^*-w^k)\Big\rangle\\
&\quad+\Big\langle g(w^{k+1})-\nabla L(w^{k+1}),\, \alpha^{k+1}(w^*-w^{k+1})-\alpha^k(w^*-w^k)\Big\rangle.
\end{aligned} \tag{A.23}$$

To estimate the second part of the right hand side of (A.23), we make use of Lemma A.13 to obtain the existence of some C_5 such that

$$\begin{aligned}
&\Big\langle g(w^{k+1})-\nabla L(w^{k+1}),\, \alpha^{k+1}(w^*-w^{k+1})-\alpha^k(w^*-w^k)\Big\rangle\\
&\leq 2B\left\|\alpha^{k+1}(w^*-w^{k+1})-\alpha^k(w^*-w^k)\right\|\\
&= 2B\left\|\alpha^{k+1}(w^*-w^k+\alpha^k g(w^k))-\alpha^k(w^*-w^k)\right\|\\
&= 2B\left\|(\alpha^{k+1}-\alpha^k)w^*+(\alpha^k-\alpha^{k+1})w^k+\alpha^k\alpha^{k+1}g(w^k)\right\|\\
&\leq C_5\left\|\alpha^{k+1}-\alpha^k\right\|+C_5(\alpha^k)^2+C_5(\alpha^{k+1})^2.
\end{aligned} \tag{A.24}$$

In the last inequality we used the inequality $2ab\leq a^2+b^2$. To estimate the first part of the right hand side of (A.23), we can apply (A.22) together with the boundedness of $\{w^k\}$ to

obtain the existence of some C_6 such that

$$\left\langle g(\boldsymbol{w}^k) - \nabla L(\boldsymbol{w}^k) + g(\boldsymbol{w}^{k+1}) - \nabla L(\boldsymbol{w}^{k+1}),\ \alpha^k(\boldsymbol{w}^* - \boldsymbol{w}^k) \right\rangle \\ \leq C_6(\alpha^{k-2})^2 + C_6(\alpha^{k-1})^2 + C_6(\alpha^k)^2 + C_6(\alpha^{k+1})^2. \quad \text{(A.25)}$$

Plugging (A.24) and (A.25) into (A.23) and summing the terms yields (A.11). Then the assumptions of Theorem A.10 are verified and the theorem statement follows. ■

A.6.3 Auxiliary results

Lemma A.12

Let l satisfy (S1)-(S3). Then there exists some $\hat{C} > 0$ such that for all k we have

$$\hat{C} \leq \sum_{i \in \mathcal{I}} l'(s_i^k - t^k),$$
$$\hat{C} \leq \sum_{i \in \mathcal{I}} l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k)).$$

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 (3.8) implies

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

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

$$l^{-1}(n\tau) \geq s_{i_0}^k - t^k. \quad \text{(A.26)}$$

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

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

where the inequality follows from (A.26) and the following equality from (3.8). 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

$$\left\| \nabla L(\boldsymbol{w}^k) \right\| \leq B,$$
$$\left\| g(\boldsymbol{w}^k) \right\| \leq B.$$

Proof:
Due to (S4) the derivative l' is bounded by some $\hat{B}$. Then Theorem 3.3 and Lemma A.12 imply

$$\left\|\nabla t(\boldsymbol{w}^k)\right\| \leq \frac{\hat{B}\sum_{i\in\mathcal{I}}\|\boldsymbol{x}_i\|}{\sum_{i\in\mathcal{I}} l'(\boldsymbol{x}_i^\top \boldsymbol{w} - t(\boldsymbol{w}))} \leq \frac{\hat{B}}{\hat{C}}\sum_{i\in\mathcal{I}}\|\boldsymbol{x}_i\|,$$

which is independent of k. Then (3.6) and again the boundedness of l' imply the existence of some B such that $\left\|\nabla L(\boldsymbol{w}^k)\right\| \leq B$ for all k. The proof for $g(\boldsymbol{w}^k)$ 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

$$\left\|c_1^k - c_2^k\right\| \leq C_1\alpha^k, \qquad \left\|d_1^k - d_2^k\right\| \leq C_1\alpha^k, \qquad d_1^k \geq C_2, \qquad d_2^k \geq C_2.$$

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_2^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

Appendix for Chapter 4

In this chapter we provide proofs and additional results for the Chapter 4. In the first part, we introduce concept of conjugate functions. In the second part, we derive dual formulation to the formulations from Table 2.1. Finally, the last part focuses on how to efficiently solve these dual formulations.

B.1 Convex Conjugate

Definition B.1: Convex conjugate [31]

Let $l\colon \mathbb{R}^n \to \mathbb{R}$. The function $l^\star\colon \mathbb{R}^n \to \mathbb{R}$, defined as

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

is called conjugate function of l.

Recall the hinge loss and quadratic hinge loss function defined in Notation 2.1 as follows

$$l_{\text{hinge}}(s) = \max\{0, 1+s\},$$
$$l_{\text{quadratic}}(s) = (\max\{0, 1+s\})^2.$$

The conjugate for the hinge loss can be found in [67] and has the following form

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

Similarly, the conjugate for the quadratic hinge is defuined in [68] as

$$l^\star_{\text{quadratic}}(y) = \begin{cases} \frac{y^2}{4} - y & \text{if } y \geq 0, \\ \infty & \text{otherwise.} \end{cases} \tag{B.2}$$

B.2 Dual formulations

In Section 4.1 we divided all formulations from Table 2.1 into two families. All formulations in these families use the same objective function and also use the same form of the decision threshold. In Theorems 4.2 and 4.3 we derived the dual formulation for these two families. In this section, we derive dual formulations for formulations from Table 2.1. Then the Theorems 4.2 and 4.3 are direct consequence of the theorems presented in the following sections.

B.2.1 Ranking Problems

In this section, we derive the dual formulation of *TopPushK*. Table 2.1 shows, that *TopPush* is a special is a special case of the *TopPushK* for $K = 1$. Therefore, it is sufficient to show the dual form only for *TopPushK*.

Lemma B.2: *TopPushK* alternative formulation.

The problem (2.6) with linear classifier can be equivalently written as follows

$$\begin{aligned} \underset{\boldsymbol{w},t,\boldsymbol{y},\boldsymbol{z}}{\text{maximize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n_+} l(y_i) \\ \text{subject to} \quad & y_i = t + \frac{1}{K}\sum_{j=1}^{n_-} z_j - \boldsymbol{w}^\top \boldsymbol{x}_i^+, \quad i = 1,\, 2,\ldots,\, n_+., \\ & z_j \geq \boldsymbol{w}^\top \boldsymbol{x}_j^- - t, \qquad\qquad j = 1,\, 2,\ldots,\, n_-, \\ & z_j \geq 0, \qquad\qquad j = 1,\, 2,\ldots,\, n_- \end{aligned} \tag{B.3}$$

Proof:

Firstly, we rewrite the formula for the decision threshold from (2.6) using the Lemma 1 from [69]

$$\sum_{j=1}^{K} s_{[j]}^- = \min_t \left\{ Kt + \sum_{j=1}^{n_-} \max\{0,\, s_j^- - t\} \right\}.$$

Substituing this formula into the objective function from (2.6) we get

$$\begin{aligned} \sum_{i=1}^{n_+} l\left(\frac{1}{K}\sum_{j=1}^{K} s_{[j]}^- - s_i^+ \right) &= \sum_{i=1}^{n_+} l\left(\frac{1}{K} \min_t \left\{ Kt + \sum_{j=1}^{n_-} \max\left\{0,\, s_j^- - t\right\} \right\} - s_i^+ \right) \\ &= \min_t \sum_{i=1}^{n_+} l\left(t + \frac{1}{K}\sum_{j=1}^{n_-} \max\left\{0,\, s_j^- - t\right\} - s_i^+ \right). \end{aligned}$$

where the last equality follows from the fact, that the surrogate function is l is non-decreasing. The max operator can be replaced using auxiliary variable $\boldsymbol{z} \in \mathbb{R}^{n_-}$ which for all $j = 1,\, 2,\ldots,\, n_-$ fullfills $z_j \geq s_j^- - t$ and at the same time $z_j \geq 0$. Moreover, we introduce new variable $\boldsymbol{y} \in \mathbb{R}^{n_-}$ defined for all $i = 1,\, 2,\ldots,\, n_+$ as

$$y_i = t + \frac{1}{K}\sum_{j=1}^{n_-} z_j - s_i^+.$$

Altogether, we get the formulation (B.3), where we use the fact, that we have linear model and therefore $s_j^- = \boldsymbol{w}^\top \boldsymbol{x}_j^-$ for all $j = 1,\, 2,\ldots,\, n_-$ and $s_i^+ = \boldsymbol{w}^\top \boldsymbol{x}_i^+$ for all $i = 1,\, 2,\ldots,\, n_+$. ■

Theorem B.3: Dual formulation of *TopPush* and *TopPushK*

Consider *TopPushK* formulation (2.5) with linear model, surrogate function l and Nota-

tion 4.1. Then the corresponding dual problem has the following form

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}^{-}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} - C\sum_{i=1}^{n_+} l^{\star}\left(\frac{\alpha_i}{C}\right) \tag{B.4a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{n_-}\beta_j, \tag{B.4b}$$

$$0 \le \beta_j \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j = 1,2,\ldots,n_-, \tag{B.4c}$$

where $l^{\star}$ is conjugate function of l. If $K = 1$, the upper bound in the second constrainet vanishes due to the first constraint and we get the dual form of *TopPush*.

Proof:
In Lemma B.2 we derived alternative fomrulation of *TopPushK* with Lagrangian in the following form

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

with feasibility conditions $\beta_j \ge 0$ and $\gamma_j \ge 0$ for all $j = 1, 2,\ldots, n_-$. Then the corresponding dual objective function reads

$$g(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = \min_{\boldsymbol{w},t,\boldsymbol{y},\boldsymbol{z}} \mathcal{L}(\boldsymbol{w},t,\boldsymbol{z};\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}),$$

Since the Lagrangian $\mathcal{L}$ is separable in primal variables, it can be minimized with respect to each variable separately, i.e., the dual function can be rewritten as follows

$$\begin{aligned} g(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = &\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \boldsymbol{w}^{\top}\left(\sum_{i=1}^{n_+}\alpha_i\boldsymbol{x}_i^{+} - \sum_{j=1}^{n_-}\beta_j\boldsymbol{x}_j^{-}\right) \\ &+ \min_{t} t\left(\sum_{i=1}^{n_+}\alpha_i - \sum_{j=1}^{n_-}\beta_j\right) \\ &+ \min_{\boldsymbol{y}} C\sum_{i=1}^{n_+}\left(l(y_i) - \frac{\alpha_i}{C}y_i\right) \\ &+ \min_{\boldsymbol{z}} \sum_{j=1}^{n_-}\left(\sum_{i=1}^{n_+}\alpha_i - \beta_j - \gamma_j\right)z_j \end{aligned} \tag{B.5}$$

From optimality conditions 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 we use Notation 4.1. Using this relation, we get the first part of the objective function (B.4a)

$$\frac{1}{2}\|\boldsymbol{w}\|_2^2 - \boldsymbol{w}^\top\left(\sum_{i=1}^{n_+}\alpha_i \boldsymbol{x}_i^+ - \sum_{j=1}^{n_-}\beta_j \boldsymbol{x}_j^-\right) = -\frac{1}{2}\|\boldsymbol{w}\|_2^2 = -\frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} = -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}^-\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix},$$

where $\mathbb{K}^-$ is defined in Notation 4.1. Optimality condition with respect to t reads

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

and implies constrain in (B.4b). Similarly, Optimality condition with respect to $\boldsymbol{z}$ reads for all $j = 1, 2, \ldots, n_-$ as

$$\frac{1}{K}\sum_{i=1}^{n_+}\alpha_i - \beta_j - \gamma_j = 0.$$

Plugging the feasibility condition $\gamma_j \geq 0$ into this equality and combining it with the feasibility conditions $\beta_j \geq 0$ yields constraint (B.4c). Finally, minimization of the Lagrangian with respect to $\boldsymbol{y}$ yields for all $i = 1, 2, \ldots, n_+$

$$C\min_{y_i}\left(l(y_i) - \frac{\alpha_i}{C}y_i\right) = -Cl^\star\left(\frac{\alpha_i}{C}\right).$$

where the equality follows from Definition B.1. Plugging this back into the Lagrange function yields the second part of the objective function (B.4a). For *TopPush*, we have $K = 1$. From (B.4b) and non-negativity of β_j we deduce, that the upper bound in (B.4c) is always fulfilled and therefore can be ommited, which finishes the proof. ■

B.2.2 Accuracy at the Top

In Section 2.3 we derived three formulations that fall into our framework (2.3). In this section, we focus only on two of them that are convex for linear classifer as showed in Chapter 3. Namely, we focus on *TopMeanK* and *Pat&Mat*.

Theorem B.4: Dual formulation of *TopMeanK*

Consider *TopMeanK* formulation (2.11) with linear model, surrogate function l and Notation 4.1. Then the corresponding dual problem has the following form

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad & -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}^{\pm}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} - C\sum_{i=1}^{n_+} l^\star\left(\frac{\alpha_i}{C}\right)\\
\text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{n}\beta_j,\\
& 0 \leq \beta_j \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j = 1, 2, \ldots, n,
\end{aligned}$$

where $l^\star$ is conjugate function of l and $K = n\tau$.

Proof:
TopMeanK formulation is similar to the *TopPushK* and therefore also dual formulations are similar. The main difference is, that the decision threshold for *TopMeanK* is computed from all socres and not only from the negative ones as for *TopPushK*. Due to that, the dual variable $\boldsymbol{\beta}$ has different size and the kernel matrix has slightly different form as can be seen in Notation 4.1. Besides that dual formulations of *TopMeanK* and *TopMeanK* are identical and the proof of Theorem B.4 is almost identical to the proof of Theorem B.3. ■

Theorem B.5: Dual formulation of *Pat&Mat*

Consider *Pat&Mat* formulation (2.13) with linear model, surrogate function l and Notation 4.1. Then the corresponding dual problem has the following form

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}^{\pm}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} - C\sum_{i=1}^{n_+} l^{\star}\left(\frac{\alpha_i}{C}\right) - \delta\sum_{j=1}^{n} l^{\star}\left(\frac{\beta_j}{\delta\vartheta}\right) - \delta n\tau \tag{B.6a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{n}\beta_j, \tag{B.6b}$$

$$\delta \geq 0, \tag{B.6c}$$

where $l^{\star}$ is conjugate function of l and $\vartheta > 0$ is a scaling parameter.

Proof:
Let us first realize tha *Pat&Mat* formulation (2.13) with linear model is equivalent to

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

Corresponding Lagrangian is in the following form

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

with feasibility condition $\delta \geq 0$. Then the corresponding dual objective function reads

$$g(\boldsymbol{\alpha},\boldsymbol{\beta},\delta) = \min_{\boldsymbol{w},t,\boldsymbol{y},\boldsymbol{z}} \mathcal{L}(\boldsymbol{w},t,\boldsymbol{y},\boldsymbol{z};\boldsymbol{\alpha},\boldsymbol{\beta},\delta),$$

Since the Lagrangian $\mathcal{L}$ is separable in primal variables, it can be minimized with respect to

each variable separately, i.e., the dual function can be rewritten as follows

$$\begin{aligned} g(\boldsymbol{\alpha},\boldsymbol{\beta},\delta) = \min_{\boldsymbol{w}} \ & \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \boldsymbol{w}^\top\left(\sum_{i=1}^{n_+}\alpha_i\boldsymbol{x}_i^+ - \sum_{j=1}^{n}\beta_j\boldsymbol{x}_j\right) \\ & + \min_{t} \ t\left(\sum_{i=1}^{n_+}\alpha_i - \sum_{j=1}^{n}\beta_j\right) \\ & + \min_{\boldsymbol{y}} \ C\sum_{i=1}^{n_+}\left(l(y_i) - \frac{\alpha_i}{C}y_i\right) \\ & + \min_{\boldsymbol{z}} \ \delta\sum_{j=1}^{n}\left(l(\vartheta z_j) - \frac{\beta_j}{\delta}z_j\right) \\ & - \delta n\tau. \end{aligned}$$

Note that resulting dual function is very similar to the dual function (B.5) for *TopPushK*, i.e. minimization of the Lagrangian with respect to $\boldsymbol{w}$, t and $\boldsymbol{y}$ yields similar results. From optimality conditions 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 we use Notation 4.1. Using this relation, we get the first part of the objective function (B.6a)

$$\frac{1}{2}\|\boldsymbol{w}\|_2^2 - \boldsymbol{w}^\top\left(\sum_{i=1}^{n_+}\alpha_i\boldsymbol{x}_i^+ - \sum_{j=1}^{n}\beta_j\boldsymbol{x}_j\right) = -\frac{1}{2}\|\boldsymbol{w}\|_2^2 = -\frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} = -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha} \\ \boldsymbol{\beta}\end{pmatrix}^\top\mathbb{K}^{\pm}\begin{pmatrix}\boldsymbol{\alpha} \\ \boldsymbol{\beta}\end{pmatrix},$$

where $\mathbb{K}^{\pm}$ is defined in Notation 4.1. Optimality condition with respect to t reads

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

and implies constrain in (B.6b). The optimality condition with respect to $\boldsymbol{y}$ is identical to the one in the proof of Theorem B.3. Finally, inimization of the Lagrangian with respect to $\boldsymbol{z}$ yields for all $j = 1, 2, \ldots, n$

$$\delta\min_{\boldsymbol{z}}\left(l(\vartheta z_j) - \frac{\beta_j}{\delta\vartheta}\vartheta z_j\right) = -\delta l^\star\left(\frac{\beta_i}{\delta\vartheta}\right),$$

where the equality follows from Definition B.1. Plugging this back into the Lagrange function yields the second part of the objective function (B.6a), which finishes the proof. ■

B.2.3 Hypothesis Testing

In Section 2.4 we derived three problem formulations that fall into our framework (2.3). Namely: *Grill-NP*, *τ-FPL* and *Pat&Mat-NP*. Similarly to the previous section, we focus only on *τ-FPL* and *Pat&Mat-NP*. Since *τ-FPL* is a special case of *TopPushK* for $K = n\_\tau$, the dual formulation is identical to the one in B.3.

Theorem B.6: Dual formulation of *Pat&Mat-NP*

Consider *Pat&Mat-NP* formulation (2.20) with linear model, surrogate function l and Notation 4.1. Then the corresponding dual problem has the following form

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad & -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}^- \begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} - C\sum_{i=1}^{n_+} l^\star\left(\frac{\alpha_i}{C}\right) - \delta\sum_{j=1}^{n_-} l^\star\left(\frac{\beta_j}{\delta\vartheta}\right) - \delta n_- \tau \\ \text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{n_-}\beta_j, \\ & \delta \geq 0, \end{aligned}$$

where $l^\star$ is conjugate function of l and $\vartheta > 0$ is a scaling parameter.

Proof:
Pat&Mat-NP formulation is similar to the *Pat&Mat* and therefore also dual formulations are similar. The main difference is, that the decision threshold for *Pat&Mat-NP* is computed from all socres and not only from the negative ones as for *Pat&Mat*. Due to that, the dual variable $\boldsymbol{\beta}$ has different size and the kernel matrix has slightly different form as can be seen in Notation 4.1. Besides that dual formulations of *Pat&Mat-NP* and *Pat&Mat* are identical and the proof of Theorem B.6 is almost identical to the proof of Theorem B.5. ■

B.3 New Coordinate descent Algorithm

B.3.1 Family of *TopPushK* Formulations

Hinge Loss

For better readability we recall the form of the dual formulation (4.10)

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad & -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K} \begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} + \sum_{i=1}^{n_+}\alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, \\ & 0 \leq \alpha_i \leq C, \qquad i = 1,2,\ldots,n_+, \\ & 0 \leq \beta_j \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j = 1,2,\ldots,\tilde{n}. \end{aligned}$$

In the rest of the section, we provide closed-form formulas for all update rules from (4.8).

Lemma 4.5: Update rule (4.8a) for problem (4.10)

Consider problem (4.10), update rule (4.8a), indices $1 \leq k \leq n_+$ and $1 \leq l \leq n_+$ and Nota-

tion 4.4. Then the optimal solution $\Delta^\star$ is given by (4.9) where

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

Proof of Lemma 4.5 on page 36:
Constraint (4.10b) is always satisfied from the definition of the update rule (4.8a). Constraint (4.10d) is always satisfied since no β_j was updated and the sum of all α_i did not change. Constraint (4.10c) reads

$$\begin{aligned}0 \le \alpha_{\hat{k}} + \Delta \le C \quad &\Longrightarrow \quad -\alpha_{\hat{k}} \le \Delta \le C - \alpha_{\hat{k}}\\ 0 \le \alpha_{\hat{l}} - \Delta \le C \quad &\Longrightarrow \quad \alpha_{\hat{l}} - C \le \Delta \le \alpha_{\hat{l}}\end{aligned}$$

which gives the lower and upper bound of Δ. Using the update rule (4.8a), objective function (4.10a) can be rewritten as a quadratic function with respect to Δ as

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

Finally, the optimal solution $\Delta^\star$ is given by (4.9). ■

Lemma 4.6: Update rule (4.8b) for problem (4.10)

Consider problem (4.10), update rule (4.8b), indices $1 \le k \le n_+$ and $n_+ + 1 \le l \le \tilde{n}$ and Notation 4.4. Let us define

$$\beta_{\max} = \max_{j \in \{1,2,\ldots,\tilde{n}\} \setminus \{\hat{l}\}} \beta_j.$$

Then the optimal solution $\Delta^\star$ is given by (4.9) where

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

Proof of Lemma 4.6 on page 36:
Constraint (4.10b) is always satisfied from the definition of the update rule (4.8b). Constraint (4.10c) reads

$$0 \le \alpha_{\hat{k}} + \Delta \le C \quad \Longrightarrow \quad -\alpha_{\hat{k}} \le \Delta \le C - \alpha_{\hat{k}}.$$

Using the definition of $\beta_{\max}$, constraint (4.10d) for any $K \ge 2$ reads

$$\begin{aligned}0 \le \beta_{\max} \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i + \frac{\Delta}{K} \quad &\Longrightarrow \quad K\beta_{\max} - \sum_{i=1}^{n_+}\alpha_i \le \Delta\\ 0 \le \beta_{\hat{l}} + \Delta \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i + \frac{\Delta}{K} \quad &\Longrightarrow \quad -\beta_{\hat{l}} \le \Delta \quad \wedge \quad \Delta \le \frac{1}{K-1}\left(\sum_{i=1}^{n_+}\alpha_i - K\beta_{\hat{l}}\right)\end{aligned}$$

Combination of these bounds yealds the lower bound Δ_{lb} and upper bound Δ_{ub}. If $K = 1$, the upper bounds in (4.10d) is always satisfied due to (4.10b) and the lower and upper bound of Δ can be simplified. Using the update rule (4.8b), objective function (4.10a) can be rewritten as a quadratic function with respect to Δ as

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

Finally, the optimal solution $\Delta^\star$ is given by (4.9). ■

Lemma 4.7: Update rule (4.8c) for problem (4.10)

Consider problem (4.10), update rule (4.8c), indices $n_+ + 1 \leq k \leq \tilde{n}$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.4. Then the optimal solution $\Delta^\star$ is given by (4.9) where

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

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

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

Proof of Lemma 4.7 on page 37:

Constraint (4.10b) is always satisfied from the definition of the update rule (4.8c). Constraint (4.10c) is also always satisfied since no α_i is updated. Constraint (4.10d) for any $K \geq 2$ reads

$$0 \leq \beta_{\hat{k}} + \Delta \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i \quad \Longrightarrow \quad -\beta_{\hat{k}} \leq \Delta \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i - \beta_{\hat{k}}$$

$$0 \leq \beta_{\hat{l}} - \Delta \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i \quad \Longrightarrow \quad \beta_{\hat{l}} - \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i \leq \Delta \leq \beta_{\hat{l}}$$

which gives the lower and upper bound of Δ. If $K = 1$, the upper bounds in (4.10d) is always satisfied due to (4.10b) and the lower and upper bound of Δ can be simplified. Using the update rule (4.8c), objective function (4.10a) can be rewritten as a quadratic function with respect to Δ as

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

Finally, the optimal solution $\Delta^\star$ is given by (4.9). ■

Quadratic Hinge Loss

For better readability we recall the form of the dual formulation (4.11)

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad & -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}+\sum_{i=1}^{n_+}\alpha_i-\frac{1}{4C}\sum_{i=1}^{n_+}\alpha_i^2 \\
\text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i=\sum_{j=1}^{\tilde{n}}\beta_j, \\
& 0\le\alpha_i, \qquad\qquad i=1,2,\dots,n_+, \\
& 0\le\beta_j\le\frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j=1,2,\dots,\tilde{n}.
\end{aligned}$$

In the rest of the section, we provide closed-form formulas for all update rules from (4.8).

Lemma B.10: Update rule (4.8a) for problem (4.11)

Consider problem (4.11), update rule (4.8a), indeices $1\le k\le n_+$ and $1\le l\le n_+$ and Notation 4.4. Then the optimal solution $\Delta^\star$ is given by (4.9) where

$$\Delta_{lb}=-\alpha_{\hat{k}}, \qquad \Delta_{ub}=\alpha_{\hat{l}}, \qquad \gamma=-\frac{s_k-s_l+\frac{1}{2C}(\alpha_{\hat{k}}-\alpha_{\hat{l}})}{\mathbb{K}_{kk}+\mathbb{K}_{ll}-\mathbb{K}_{kl}-\mathbb{K}_{lk}+\frac{1}{C}}.$$

Proof:
Constraint (4.11b) is always satisfied from the definition of the update rule (4.8a). Constraint (4.11d) is also always satisfied since no β_j was updated and the sum of all α_i did not change. Constraint (4.11c) reads

$$\begin{aligned}
0\le\alpha_{\hat{k}}+\Delta \quad &\Longrightarrow \quad -\alpha_{\hat{k}}\le\Delta \\
0\le\alpha_{\hat{l}}-\Delta \quad &\Longrightarrow \quad \Delta\le\alpha_{\hat{l}}
\end{aligned}$$

which gives the lower and upper bound of Δ. Using the update rule (4.8a), objective function (4.11a) can be rewritten as a quadratic function with respect to Δ as

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

Finally, the optimal solution $\Delta^\star$ is given by (4.9). ■

Lemma B.11: Update rule (4.8b) for problem (4.11)

Consider problem (4.11), update rule (4.8b), indeices $1\le k\le n_+$ and $n_++1\le l\le\tilde{n}$ and Notation 4.4. Let us define

$$\beta_{\max}=\max_{j\in\{1,2,\dots,\tilde{n}\}\setminus\{\hat{l}\}}\beta_j.$$

Then the optimal solution $\Delta^\star$ is given by (4.9) where

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

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

$$\gamma = -\frac{s_k + s_l - 1 + \frac{1}{2C}\alpha_{\hat{k}}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk} + \frac{1}{2C}}.$$

Proof:
Constraint (4.11b) is always satisfied from the definition of the update rule (4.8b). Constraint (4.11c) reads

$$0 \le \alpha_{\hat{k}} + \Delta \quad \Longrightarrow \quad -\alpha_{\hat{k}} \le \Delta.$$

Using the definition of $\beta_{\max}$, constraint (4.11d) for any $K \ge 2$ reads

$$0 \le \beta_{\max} \le \frac{1}{K}\sum_{i=1}^{n_+} \alpha_i + \frac{\Delta}{K} \quad \Longrightarrow \quad K\beta_{\max} - \sum_{i=1}^{n_+} \alpha_i \le \Delta$$

$$0 \le \beta_{\hat{l}} + \Delta \le \frac{1}{K}\sum_{i=1}^{n_+} \alpha_i + \frac{\Delta}{K} \quad \Longrightarrow \quad -\beta_{\hat{l}} \le \Delta \quad \wedge \quad \Delta \le \frac{1}{K-1}\left(\sum_{i=1}^{n_+} \alpha_i - K\beta_{\hat{l}}\right)$$

Combination of these bounds yealds the lower bound Δ_{lb} and upper bound Δ_{ub}. If $K = 1$, the upper bounds in (4.11d) is always satisfied due to (4.11b) and the lower and upper bound of Δ can be simplified. Using the update rule (4.8b), objective function (4.11a) can be rewritten as a quadratic function with respect to Δ as

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

Finally, the optimal solution $\Delta^\star$ is given by (4.9). ■

Lemma B.12: Update rule (4.8c) for problem (4.11)

Consider problem (4.11), update rule (4.8c), indices $n_+ + 1 \le k \le \tilde{n}$ and $n_+ + 1 \le l \le \tilde{n}$ and Notation 4.4. Then the optimal solution $\Delta^\star$ is given by (4.9) where

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

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

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

Proof:
Constraint (4.11b) is always satisfied from the definition of the update rule (4.8c). Constraint (4.11c) is also always satisfied since no α_i is updated. Constraint (4.11d) for any $K \ge 2$

reads

$$0 \le \beta_{\hat{k}} + \Delta \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i \quad \Longrightarrow \quad -\beta_{\hat{k}} \le \Delta \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i - \beta_{\hat{k}}$$
$$0 \le \beta_{\hat{l}} - \Delta \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i \quad \Longrightarrow \quad \beta_{\hat{l}} - \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i \le \Delta \le \beta_{\hat{l}}$$

which gives the lower and upper bound of Δ. If $K = 1$, the upper bounds in (4.11d) is always satisfied due to (4.11b) and the lower and upper bound of Δ can be simplified. Using the update rule (4.8c), objective function (4.11a) can be rewritten as a quadratic function with respect to Δ as

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

Finally, the optimal solution $\Delta^\star$ is given by (4.9). ■

Initialization

For better readability we recall the form of problem (4.12)

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{minimize}} \quad & \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\right\|^2 \\
\text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, \\
& 0 \le \alpha_i \le C_1, \quad i = 1, 2, \ldots, n_+,, \\
& 0 \le \beta_j \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j = 1, 2, \ldots, \tilde{n},
\end{aligned}$$

Theorem 4.8

Consider problem (4.12) and some initial solution $\boldsymbol{\alpha}^0$, $\boldsymbol{\beta}^0$ and denote the sorted version (in non-decreasing order) of $\boldsymbol{\beta}^0$ as $\boldsymbol{\beta}^0_{[\cdot]}$. Then if the following condition holds

$$\sum_{j=1}^{K}\left(\beta^0_{[\tilde{n}-K+j]} + \max_{i=1,\ldots,n_+}\alpha_i^0\right) \le 0, \tag{4.13}$$

the optimal solution of (4.12) amounts to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$. In the opposite case, the following system of two equations

$$\sum_{i=1}^{n_+}\operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda + \frac{1}{K}\sum_{j=1}^{\tilde{n}}\operatorname{clip}_{[0,\,\infty)}\left(\beta_j^0 + \lambda - \mu\right)\right) - K\mu = 0, \tag{4.14a}$$
$$\sum_{j=1}^{\tilde{n}}\operatorname{clip}_{[0,\,\mu]}\left(\beta_j^0 + \lambda\right) - K\mu = 0, \tag{4.14b}$$

has a solution (λ, μ) with $\mu > 0$, and the optimal solution of (4.12) equals to

$$\alpha_i = \operatorname{clip}_{[0,\, C_1]}\left(\alpha_i^0 - \lambda + \frac{1}{K}\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\infty)}\left(\beta_j^0 + \lambda - \mu\right)\right),$$
$$\beta_j = \operatorname{clip}_{[0,\,\mu]}\left(\beta_j^0 + \lambda\right).$$

Proof of Theorem 4.8 on page 38:
The Lagrangian for (4.12) reads

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}; \lambda, \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\right\|^2 + \lambda\left(\sum_{i=1}^{n_+}\alpha_i - \sum_{j=1}^{\tilde{n}}\beta_j\right)$$
$$- \sum_{i=1}^{n_+} p_i\alpha_i + \sum_{i=1}^{n_+} q_j(\alpha_i - C_1) - \sum_{j=1}^{\tilde{n}} u_j\beta_j + \sum_{j=1}^{\tilde{n}} v_j\left(\beta_j - \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i\right)$$

The KKT conditions then amount to the optimality conditions

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = \alpha_i - \alpha_i^0 + \lambda - p_i + q_i - \frac{1}{K}\sum_{j=1}^{\tilde{n}} v_j = 0, \qquad i = 1, 2, \ldots, n_+, \tag{B.7a}$$
$$\frac{\partial \mathcal{L}(\cdot)}{\partial \beta_j} = \beta_j - \beta_j^0 - \lambda - u_j + v_j = 0, \qquad j = 1, 2, \ldots, \tilde{n} \tag{B.7b}$$

the primal feasibility conditions (4.12), the dual feasibility conditions $\lambda \in \mathbb{R}$, $p_i \geq 0$, $q_i \geq 0$, $u_j \geq 0$, $v_j \geq 0$ and finally the complementarity conditions

$$p_i\alpha_i = 0, \qquad i = 1, 2, \ldots, n_+, \tag{B.7c}$$
$$q_i(\alpha_i - C_1) = 0, \qquad i = 1, 2, \ldots, n_+, \tag{B.7d}$$
$$u_j\beta_j = 0, \qquad j = 1, 2, \ldots, \tilde{n}, \tag{B.7e}$$
$$v_j\left(\beta_j - \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i\right) = 0, \qquad j = 1, 2, \ldots, \tilde{n}. \tag{B.7f}$$

Case 1: The first case concerns when the optimal solution satisfies $\sum_i \alpha_i = 0$. From the primal feasibility conditions, we immediately get $\alpha_i = 0$ for all i and $\beta_j = 0$ for all j. Then (B.7d) implies $q_i = 0$ for all i and all complementarity conditions are satisfied. Moreover, optimality condition (B.7a) implies

$$\lambda = \alpha_i^0 + p_i + \frac{1}{K}\sum_{j=1}^{\tilde{n}} v_j.$$

Since the only condition on p_i is the non-negativity, this implies

$$\lambda \geq \max_{i=1,\ldots,n_+} \alpha_i^0 + \frac{1}{K}\sum_{j=1}^{\tilde{n}} v_j.$$

Similarly, from optimality condition (B.7b) we deduce

$$v_j = \beta_j^0 + \lambda + u_j \geq \beta_j^0 + \lambda \geq \beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0 + \frac{1}{K}\sum_{i=1}^{\tilde{n}} v_i.$$

Since we need to fulfill $v_j \geq 0$, this amounts to

$$v_j \geq \operatorname{clip}_{[0,\,\infty)}\left(\beta_j^0 + \max_{i=1,\dots,n_+} \alpha_i^0 + \frac{1}{K}\sum_{i=1}^{\tilde{n}} v_i\right).$$

Summing this with respect to j and using the substitution $\bar{v} = \frac{1}{K}\sum_i v_i$ results in

$$K\bar{v} - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\infty)}\left(\beta_j^0 + \max_{i=1,\dots,n_+} \alpha_i^0 + \bar{v}\right) = 0. \tag{B.8}$$

Denote by $\beta_{[j]}^0$ the sorted version of β_j^0. Then the function on the left-hand side of (B.8) as a function of $\bar{v}$ is increasing on $(-\infty,\, -\beta_{[n_+-K+1]}^0 - \max_i \alpha_i^0]$ and non-decreasing on $[-\beta_{[n_+-K+1]}^0 - \max_i \alpha_i^0, \infty)$. Thus, (B.8) can be satisfied if and only if its function value at $-\beta_{[n_+-K+1]}^0 - \max_i \alpha_i^0$ is non-negative. But this is precisely the violation of (4.13).

Case 2: If (4.13) holds true, then from the discussion above we obtain that the optimal solution satisfies $\sum_i \alpha_i > 0$. For simplicity, we define

$$\bar{\alpha} = \frac{1}{K}\sum_{i=1}^{n_+} \alpha_i, \qquad \bar{\beta} = \frac{1}{K}\sum_{j=1}^{\tilde{n}} \beta_j, \qquad \bar{v} = \frac{1}{K}\sum_{j=1}^{\tilde{n}} v_j.$$

For any fixed i, the standard trick is to combine the optimality condition (B.7a) with the primal feasibility condition $0 \leq \alpha_i \leq C_1$, the dual feasibility conditions $p_i \geq 0$, $q_i \geq 0$ and the complementarity conditions (B.7c, B.7d) to obtain

$$\alpha_i = \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda + \bar{v}\right). \tag{B.9}$$

Similarly for any fixed j, we combine the optimality condition (B.7b) with the primal feasibility condition $0 \leq \beta_j \leq \bar{\alpha}$, the dual feasibility conditions $u_j \geq 0$, $v_j \geq 0$ and the complementarity conditions (B.7e, B.7f) to obtain

$$\beta_j = \operatorname{clip}_{[0,\,\bar{\alpha}]}\left(\beta_j^0 + \lambda\right), \tag{B.10}$$

$$v_j = \operatorname{clip}_{[0,\,\infty)}\left(\beta_j^0 + \lambda - \bar{\alpha}\right). \tag{B.11}$$

Summing equations (B.9), (B.10) and (B.11) respectively with respect to i and j results in

$$K\bar{\alpha} = \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda + \bar{v}\right), \tag{B.12a}$$

$$K\bar{\beta} = \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\bar{\alpha}]}\left(\beta_j^0 + \lambda\right), \tag{B.12b}$$

$$K\bar{v} = \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\infty)}\left(\beta_j^0 + \lambda - \bar{\alpha}\right). \tag{B.12c}$$

We denote $\mu = \bar{\alpha}$. Then (4.14a) results by plugging (B.12c) into (B.12a) while (4.14b) follows from (B.12b) and $\sum_i \alpha_i = \sum_j \beta_j$. ■

Lemma 4.9

Even though $\lambda(\mu)$ is not unique, function h is well-defined in the sense that it gives the same value for every choice of $\lambda(\mu)$. Moreover, h is decreasing in μ on $(0,\infty)$.

Proof of Lemma 4.9 on page 39:
Recall that based on (4.14b) we defined

$$g(\lambda;\mu) := \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu]}\left(\beta_j^0 + \lambda\right) - K\mu,$$

and solutions of $g(\lambda;\mu) = 0$ for a fixed μ are denoted by $\lambda(\mu)$. Function $g(\cdot;\mu)$ is non-decreasing and since K is an integer, the only case when the solution to $g(\lambda) = 0$ is not unique happens when the optimal solution $\lambda(\mu)$ satisfies

$$\beta_{[j]}^0 + \lambda(\mu) \begin{cases} \geq \mu & \text{for } j = \tilde{n} - K + 1, \ldots, \tilde{n}, \\ \leq 0 & \text{otherwise.} \end{cases} \tag{B.13}$$

Here, we again denote $\boldsymbol{\beta}_{[\cdot]}^0$ to be the sorted version of $\boldsymbol{\beta}_j^0$. Then h defined in (4.15) equals to

$$\begin{aligned} h(\mu) &= \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda(\mu) + \frac{1}{K} \sum_{j=\tilde{n}-K+1}^{\tilde{n}} \left(\beta_j^0 + \lambda(\mu) - \mu\right)\right) - K\mu \\ &= \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \mu + \frac{1}{K} \sum_{j=\tilde{n}-K+1}^{\tilde{n}} \beta_j^0\right) - K\mu. \end{aligned}$$

This implies the first statement of the lemma stating that h is independent of the choice of $\lambda(\mu)$.

Now we need to show that h is a decreasing function. Fix any $\mu_2 > \mu_1 > 0$. From (4.14b) we have

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) - K\mu_1 = 0, \tag{B.14}$$

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_2]}\left(\beta_j^0 + \lambda(\mu_2)\right) - K\mu_2 = 0. \tag{B.15}$$

Equation (B.14) implies that at most K values of $\beta_j^0 + \lambda(\mu_1)$ are greater or equal than μ_1. If we increase the upper bound in the projection, at most K values can increase, which results in

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_2]}\left(\beta_j^0 + \lambda(\mu_1)\right) \leq \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) + K(\mu_2 - \mu_1) = K\mu_2, \tag{B.16}$$

where the equality follows from (B.14). Comparing (B.15) and (B.16) yields $\lambda(\mu_2) \geq \lambda(\mu_1)$. Now define

$$J = \left\{ j \;\middle|\; \beta_j^0 + \lambda(\mu_1) \geq 0 \right\}$$

and observe that due to (B.14) we have $|J| \geq K$. Moreover, the definition of J and equation (B.14) yield

$$\sum_{j \in J} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) - K\mu_1 = \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) - K\mu_1 = 0. \tag{B.17}$$

Then we have

$$\begin{aligned}
\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_2]}\left(\beta_j^0 + \lambda(\mu_1) + \mu_2 - \mu_1\right) &\geq \sum_{j \in J} \operatorname{clip}_{[0,\,\mu_2]}\left(\beta_j^0 + \lambda(\mu_1) + \mu_2 - \mu_1\right) \\
&= \sum_{j \in J} \operatorname{clip}_{[\mu_2-\mu_1,\,\mu_2]}\left(\beta_j^0 + \lambda(\mu_1) + \mu_2 - \mu_1\right) \\
&= \sum_{j \in J} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) + |J|(\mu_2 - \mu_1) \\
&= K\mu_1 + |J|(\mu_2 - \mu_1) \\
&\geq K\mu_1 + K(\mu_2 - \mu_1) \\
&= K\mu_2,
\end{aligned} \tag{B.18}$$

where the first equality follows from the definition of J and the second equality is a shift by a $\mu_2 - \mu_1$. The third equality follows from (B.17) and finally, the last inequality follows from $|J| \geq K$. Chain (B.18) together with (B.15) implies $\lambda(\mu_2) - \mu_2 \leq \lambda(\mu_1) - \mu_1$. Combining this with $\mu_2 > \mu_1$ and $\lambda(\mu_2) \geq \lambda(\mu_1)$, this implies that h from (4.15) is non-increasing which is precisely the lemma statement. ∎

B.3.2 Family of *Pat&Mat* Formulations

Hinge Loss

For better readability we recall the form of the dual formulation (4.16)

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad & -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top} \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} + \sum_{i=1}^{n_+} \alpha_i + \frac{1}{\vartheta}\sum_{j=1}^{\tilde{n}} \beta_j - \delta\tilde{n}\tau \\
\text{subject to} \quad & \sum_{i=1}^{n_+} \alpha_i = \sum_{j=1}^{\tilde{n}} \beta_j, \\
& 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, n_+, \\
& 0 \leq \beta_j \leq \delta\vartheta, \quad j = 1, 2, \ldots, \tilde{n}, \\
& \delta \geq 0.
\end{aligned}$$

In the rest of the section, we provide closed-form formulas for all update rules from (4.8).

Lemma B.15: Update rule (4.8a) for problem (4.16)

Consider problem (4.16), update rule (4.8a), indices $1 \leq k \leq n_+$ and $1 \leq l \leq n_+$ and Notation 4.4. Then the optimal solution $\Delta^\star$ is given by (4.9) where

$$\Delta_{lb} = \min\{-\alpha_{\hat{k}},\ \alpha_{\hat{l}} - C\}, \qquad \Delta_{ub} = \max\{C - \alpha_{\hat{k}},\ \alpha_{\hat{l}}\},$$
$$\gamma = -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}, \qquad \delta^\star = \delta.$$

Proof:
Constraint (4.16b) is always satisfied from the definition of the update rule (4.8a). Constraint (4.16d) is also always satisfied since no β_j was updated and the sum of all α_i did not change. Constraint (4.16c) reads

$$\begin{aligned} 0 \leq \alpha_{\hat{k}} + \Delta \leq C \quad &\Longrightarrow \quad -\alpha_{\hat{k}} \leq \Delta \leq C - \alpha_{\hat{k}} \\ 0 \leq \alpha_{\hat{l}} - \Delta \leq C \quad &\Longrightarrow \quad \alpha_{\hat{l}} - C \leq \Delta \leq \alpha_{\hat{l}} \end{aligned}$$

which gives the lower and upper bound of Δ. Using the update rule (4.8a), objective function (4.16a) can be rewritten as a quadratic function with respect to Δ 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 optimal solution $\Delta^\star$ is given by (4.9). Finally, since optimal δ is given by (4.17) and no β_j was updated, the optimal δ does not change. ■

Lemma B.16: Update rule (4.8b) for problem (4.16)

Consider problem (4.16), update rule (4.8b), indices $1 \leq k \leq n_+$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.4. Let us define

$$\beta_{\max} = \max_{j \in \{1,2,\ldots,\tilde{n}\} \setminus \{\hat{l}\}} \beta_j.$$

Then the bounds from (4.9) are defined as $\Delta_{lb} = \max\{-\alpha_{\hat{k}}, -\beta_{\hat{l}}\}$ and $\Delta_{ub} = C - \alpha_{\hat{k}}$ and there are two possible solutions

1. $\Delta_1^\star$ is feasible if $\beta_{\hat{l}} + \Delta_1^\star \leq \beta_{\max}$ and is given by (4.9) where

$$\gamma = -\frac{s_k + s_l - 1 - \frac{1}{\vartheta}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk}}, \qquad \delta_1^* = \frac{\beta_{\max}}{\vartheta}.$$

2. $\Delta_2^\star$ is feasible if $\beta_{\hat{l}} + \Delta_2^\star \geq \beta_{\max}$ and is given by (4.9) where

$$\gamma = -\frac{s_k + s_l - 1 - \frac{1 - \tilde{n}\tau}{\vartheta}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk}}, \qquad \delta_2^* = \frac{\beta_{\hat{l}} + \Delta_2^\star}{\vartheta}.$$

The optimal solution $\Delta^\star$ is equal to the one of them which maximizes the original objective and is feasible.

Proof:
Constraint (4.16b) is always satisfied from the definition of the update rule (4.8b). Constraint (4.16c) reads

$$0 \leq \alpha_{\hat{k}} + \Delta \leq C \quad \Longrightarrow \quad -\alpha_{\hat{k}} \leq \Delta \leq C - \alpha_{\hat{k}}. \tag{B.19}$$

Using the definition of $\beta_{\max}$, constraint (4.16d) reads

$$\begin{aligned} \beta_{\max} &\leq \delta\vartheta \\ 0 \leq \beta_{\hat{l}} + \Delta &\leq \delta\vartheta \end{aligned}$$

Since the optimal δ is given by (4.17), there are only two possible choices

$$\delta = \frac{\beta_{\max}}{\vartheta}, \qquad \delta = \frac{\beta_{\hat{l}} + \Delta}{\vartheta}. \tag{B.20}$$

If we use any of these choices which is feasible, all upper bounds in constraint (4.16d) hold, i.e. we can simplify the constraints to

$$0 \leq \beta_{\hat{l}} + \Delta \quad \Longrightarrow \quad -\beta_{\hat{l}} \leq \Delta,$$

which in combination with (B.19) gives the lower and upper bound of Δ. Now let us discuss how to select optimal δ :

1. Using $\delta_1^\star$ from (B.20) and the update rule (4.8a), objective function (4.16a) can be rewritten as a quadratic function with respect to Δ as

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

   The optimal solution $\Delta_1^\star$ is given by (4.9) and is feasible if $\beta_{\hat{l}} + \Delta_1^\star \leq \beta_{\max}$.

2. Using $\delta_2^\star$ from (B.20) and the update rule (4.8a), objective function (4.16a) can be rewritten as a quadratic function with respect to Δ as

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

   The optimal solution $\Delta_2^\star$ is given by (4.9) and is feasible if $\beta_{\hat{l}} + \Delta_2^\star \geq \beta_{\max}$.

The final optimal solution is the one that is feasible and that maximizes the original objective function (4.16a). ■

Lemma B.17: Update rule (4.8c) for problem (4.16)

Consider problem (4.16), update rule (4.8c), indices $n_+ + 1 \leq k \leq \tilde{n}$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.4. Let us define

$$\beta_{\max} = \max_{j \in \{1,2,\ldots,\tilde{n}\} \setminus \{\hat{k},\hat{l}\}} \beta_j.$$

Then the bounds from (4.9) are defined as $\Delta_{lb} = -\beta_{\hat{k}}$ and $\Delta_{ub} = \beta_{\hat{l}}$ and there are three possible solutions

1. $\Delta_1^\star$ is feasible if $\beta_{\max} \geq \max\{\beta_{\hat{k}} + \Delta_1^\star, \beta_{\hat{l}} - \Delta_1^\star\}$ and is given by (4.9) where

$$\gamma = -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}, \qquad \delta_1^* = \frac{\beta_{\max}}{\vartheta}.$$

2. $\Delta_2^\star$ is feasible if $\beta_{\hat{k}} + \Delta_2^\star \geq \max\{\beta_{\max}, \beta_{\hat{l}} - \Delta_2^\star\}$ and is given by (4.9) where

$$\gamma = -\frac{s_k - s_l + \frac{\tilde{n}\tau}{\vartheta}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}, \qquad \delta_2^* = \frac{\beta_{\hat{k}} + \Delta_2^\star}{\vartheta}.$$

3. $\Delta_3^\star$ is feasible if $\beta_{\hat{l}} - \Delta_3^\star \geq \max\{\beta_{\hat{k}} + \Delta_3^\star, \beta_{\max}\}$ and is given by (4.9) where

$$\gamma = -\frac{s_k - s_l - \frac{\tilde{n}\tau}{\vartheta}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}, \qquad \delta_3^* = \frac{\beta_{\hat{k}} - \Delta_3^\star}{\vartheta}.$$

The optimal solution $\Delta^\star$ is equal to the one of them which maximizes the original objective and is feasible.

Proof:
Constraint (4.16b) is always satisfied from the definition of the update rule (4.8c). Constraint (4.16c) is also always satisfied since no α_i is updated. Using the definition of $\beta_{\max}$, constraint (4.16d) reads

$$\begin{aligned} \beta_{\max} &\leq \delta\vartheta, \\ 0 \leq \beta_{\hat{k}} + \Delta &\leq \delta\vartheta, \\ 0 \leq \beta_{\hat{l}} - \Delta &\leq \delta\vartheta. \end{aligned}$$

Since the optimal δ is given by (4.17), there are only two possible choices

$$\delta_1^\star = \frac{\beta_{\max}}{\vartheta}, \qquad \delta_2^\star = \frac{\beta_{\hat{k}} + \Delta}{\vartheta}, \qquad \delta_3^\star = \frac{\beta_{\hat{l}} - \Delta}{\vartheta}. \tag{B.21}$$

If we use any of these choices which is feasible, all upper bounds in constraint (4.16d) hold, i.e. we can simplify the constraints to

$$\begin{aligned} 0 \leq \beta_{\hat{k}} + \Delta \quad &\Longrightarrow \quad -\beta_{\hat{k}} \leq \Delta, \\ 0 \leq \beta_{\hat{l}} - \Delta \quad &\Longrightarrow \quad \Delta \leq \beta_{\hat{l}}, \end{aligned}$$

which gives the lower and upper bound of Δ. Now let us discuss how to select optimal δ :

1. Using $\delta_1^\star$ from (B.21) and the update rule (4.8a), objective function (4.16a) can be rewritten as a quadratic function with respect to Δ 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 optimal solution $\Delta_1^\star$ is given by (4.9) and is feasible if

   $$\beta_{\max} \geq \max\{\beta_{\hat{k}} + \Delta_1^\star,\ \beta_{\hat{l}} - \Delta_1^\star\}.$$

2. Using $\delta_2^\star$ from (B.21) and the update rule (4.8a), objective function (4.16a) can be rewritten as a quadratic function with respect to Δ as

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

   The optimal solution $\Delta_2^\star$ is given by (4.9) and is feasible if

   $$\beta_{\hat{k}} + \Delta_2^\star \geq \max\{\beta_{\max}, \beta_{\hat{l}} - \Delta_2^\star\}.$$

3. Using $\delta_3^\star$ from (B.21) and the update rule (4.8a), objective function (4.16a) can be rewritten as a quadratic function with respect to Δ as

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

   The optimal solution $\Delta_3^\star$ is given by (4.9) and is feasible if

   $$\beta_{\hat{l}} - \Delta_3^\star \geq \max\{\beta_{\max}, \beta_{\hat{k}} + \Delta_3^\star\}.$$

The final optimal solution is the one that is feasible and that maximizes the original objective function (4.16a). ■

Quadratic Hinge Loss

For better readability we recall the form of the dual formulation (4.18)

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad & -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}+\sum_{i=1}^{n_+}\alpha_i-\frac{1}{4C}\sum_{i=1}^{n_+}\alpha_i^2 \\
& +\frac{1}{\vartheta}\sum_{j=1}^{\tilde{n}}\beta_j-\frac{1}{4\delta\vartheta^2}\sum_{j=1}^{\tilde{n}}\beta_j^2-\delta\tilde{n}\tau \\
\text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i=\sum_{j=1}^{\tilde{n}}\beta_j, \\
& \alpha_i\geq 0, \qquad\qquad i=1,2,\ldots,n_+, \\
& \beta_j\geq 0, \qquad\qquad j=1,2,\ldots,\tilde{n}, \\
& \delta\geq 0,
\end{aligned}$$

In the rest of the section, we provide closed-form formulas for all update rules from (4.8).

Lemma B.18: Update rule (4.8a) for problem (4.18)

Consider problem (4.18), update rule (4.8a), indices $1\leq k\leq n_+$ and $1\leq l\leq n_+$ and Notation 4.4. Then the optimal solution $\Delta^\star$ is given by (4.9) where

$$\begin{aligned}
\Delta_{lb} &= -\alpha_{\hat{k}}, \\
\Delta_{ub} &= \alpha_{\hat{l}}, \\
\gamma &= -\frac{s_k-s_l+\frac{1}{2C}(\alpha_{\hat{k}}-\alpha_{\hat{l}})}{\mathbb{K}_{kk}+\mathbb{K}_{ll}-\mathbb{K}_{kl}-\mathbb{K}_{lk}+\frac{1}{C}}, \\
\delta^\star &= \delta.
\end{aligned}$$

Proof of Lemma B.18 on page 104:
Constraint (4.18c) is always satisfied from the definition of the update rule (4.8a). Constraint (4.18e) is also always satisfied since no β_j was updated and the sum of all α_i did not change. Constraint (4.18d) reads

$$\begin{aligned}
0\leq\alpha_{\hat{k}}+\Delta \quad &\Longrightarrow \quad -\alpha_{\hat{k}}\leq\Delta \\
0\leq\alpha_{\hat{l}}-\Delta \quad &\Longrightarrow \quad \Delta\leq\alpha_{\hat{l}}
\end{aligned}$$

which gives the lower and upper bound of Δ. Using the update rule (4.8a), objective function (4.18a-4.18b) can be rewritten as a quadratic function with respect to Δ as

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

The optimal solution $\Delta^\star$ is given by (4.9). Finally, since optimal δ is given by (4.19) and no β_j was updated, the optimal δ does not change. ■

Lemma B.19: Update rule (4.8b) for problem (4.18)

Consider problem (4.18), update rule (4.8b), indices $1\leq k\leq n_+$ and $n_++1\leq l\leq\tilde{n}$ and

Notation 4.4. Then the optimal solution $\Delta^\star$ is given by (4.9) where

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

Proof of Lemma B.19 on page 104:
Constraint (4.18c) is always satisfied from the definition of the update rule (4.8b). Constraints (4.18d) and (4.18e) reads

$$\begin{aligned}
0 \le \alpha_{\hat{k}} + \Delta \quad &\Longrightarrow \quad -\alpha_{\hat{k}} \le \Delta,\\
0 \le \beta_{\hat{l}} + \Delta \quad &\Longrightarrow \quad -\beta_{\hat{l}} \le \Delta,
\end{aligned}$$

which gives the lower bound of Δ. In this case, Δ has no upper bound. Using the update rule (4.8b), objective function (4.18a-4.18b) can be rewritten as a quadratic function with respect to Δ as

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

The optimal solution $\Delta^\star$ is given by (4.9). We know that the optimal δ^* is given by (4.19), then

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

■

Lemma B.20: Update rule (4.8c) for problem (4.18)

Consider problem (4.18), update rule (4.8c) indices $n_+ + 1 \le k \le \tilde{n}$ and $n_+ + 1 \le l \le \tilde{n}$ and Notation 4.4. Then the optimal solution $\Delta^\star$ is given by (4.9) where

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

Proof of Lemma B.20 on page 105:
Constraint (4.18c) is always satisfied from the definition of the update rule (4.8c). Con-

straint (4.18d) is also always satisfied since no α_i is updated. Constraint (4.18e) reads

$$\begin{aligned} 0 \le \beta_{\hat{k}} + \Delta \quad &\Longrightarrow \quad -\beta_{\hat{k}} \le \Delta, \\ 0 \le \beta_{\hat{l}} + \Delta \quad &\Longrightarrow \quad -\beta_{\hat{l}} \le \Delta, \end{aligned}$$

which gives the lower and upper bound of Δ. Using the update rule (4.8c), objective function (4.18a-4.18b) can be rewritten as a quadratic function with respect to Δ as

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

The optimal solution $\Delta^\star$ is given by (4.9). We know that the optimal δ^* is given by (4.19), then

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

■

Initialization

For better readability we recall the form of problem (4.3)

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{minimize}} \quad & \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\right\|^2 + \frac{1}{2}(\delta - \delta^0)^2 \\ \text{subject to} \quad & \sum_{i=1}^{n_+} \alpha_i = \sum_{j=1}^{\tilde{n}} \beta_j, \\ & 0 \le \alpha_i \le C_1, \quad i = 1, 2, \ldots, n_+, \\ & 0 \le \beta_j \le C_2\delta, \quad j = 1, 2, \ldots, \tilde{n},, \\ & \delta \ge 0, \end{aligned}$$

Theorem 4.10

Consider problem (4.20) and some initial solution $\boldsymbol{\alpha}^0$, $\boldsymbol{\beta}^0$ and δ^0. Then if the following condition holds

$$\delta^0 \le -C_2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\infty)}\left(\beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0\right). \tag{4.21}$$

the optimal solution of (4.12) amounts to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$ and $\delta^0 = 0$. In the opposite case, the following system of two equations

$$0 = \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda\right) - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\lambda+\mu]}\left(\beta_j^0 + \lambda\right), \tag{4.22a}$$

$$\lambda = C_2\delta^0 + C_2^2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\infty)}\left(\beta_j^0 - \mu\right) - \mu. \tag{4.22b}$$

has a solution (λ, μ) with $\lambda + \mu > 0$ and the optimal solution of (4.20) equals to

$$\begin{aligned} \alpha_i &= \operatorname{clip}_{[0,\, C_1]}\left(\alpha_i^0 - \lambda\right), \\ \beta_j &= \operatorname{clip}_{[0,\, \lambda+\mu]}\left(\beta_j^0 + \lambda\right), \\ C_2\delta &= \lambda + \mu. \end{aligned}$$

Proof of Theorem 4.10 on page 42:
The Lagrangian for (4.20) reads

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}; \lambda, \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\right\|^2 + \frac{1}{2}(\delta - \delta^0)^2 + \lambda\left(\sum_{i=1}^{n_+}\alpha_i - \sum_{j=1}^{\tilde{n}}\beta_j\right) \\ - \sum_{i=1}^{n_+} p_i\alpha_i + \sum_{i=1}^{n_+} q_j(\alpha_i - C_1) - \sum_{j=1}^{\tilde{n}} u_j\beta_j + \sum_{j=1}^{\tilde{n}} v_j(\beta_j - C_2\delta). \end{aligned}$$

The KKT conditions then amount to the optimality conditions

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = \alpha_i - \alpha_i^0 + \lambda - p_i + q_i = 0, \qquad i = 1, 2, \ldots, n_+, \tag{B.22a}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \beta_j} = \beta_j - \beta_j^0 - \lambda - u_j + v_j = 0, \qquad j = 1, 2, \ldots, \tilde{n} \tag{B.22b}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \delta} = \delta - \delta^0 - C_2\sum_{j=1}^{\tilde{n}} v_j = 0, \tag{B.22c}$$

the primal feasibility conditions (4.20), the dual feasibility conditions $\lambda \in \mathbb{R}$, $p_i \geq 0$, $q_i \geq 0$, $u_j \geq 0$, $v_j \geq 0$ and finally the complementarity conditions

$$p_i\alpha_i = 0, \qquad i = 1, 2, \ldots, n_+, \tag{B.22d}$$

$$q_i(\alpha_i - C_1) = 0, \qquad i = 1, 2, \ldots, n_+, \tag{B.22e}$$

$$u_j\beta_j = 0, \qquad j = 1, 2, \ldots, \tilde{n}, \tag{B.22f}$$

$$v_j\left(\beta_j - C_2\delta\right) = 0, \qquad j = 1, 2, \ldots, \tilde{n}. \tag{B.22g}$$

Case 1: The first case concerns when the optimal solution satisfies $\delta = 0$. From the primal feasibility conditions, we immediately get $\alpha_i = 0$ for all i and $\beta_j = 0$ for all j. Then (B.22e) implies $q_i = 0$ and all complementarity conditions are satisfied. Moreover, (B.22a) implies for all i

$$\lambda = \alpha_i^0 + p_i.$$

Since the only condition on p_i is the non-negativity, this implies $\lambda \geq \max_i \alpha_i^0$. Similarly, from (B.22b) we deduce

$$v_j = \beta_j^0 + \lambda + u_j \geq \beta_j^0 + \lambda \geq \beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0.$$

Since we also have the non-negativity constraint on u_j, this implies

$$v_j \geq \operatorname{clip}_{[0,\,\infty)}\left(\beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0\right).$$

Condition (B.22c) implies

$$\delta^0 = -C_2 \sum_{j=1}^{\tilde{n}} v_j \leq -C_2 \sum_{j=1}^{\tilde{n}} \text{clip}_{[0,\,\infty)}\Big(\beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0\Big).$$

This corresponds to the first case in the theorem statement and the violation of condition (4.21).

Case 2: If (4.21) holds true, then from the discussion above we obtain that the optimal solution satisfies $\delta > 0$. For any fixed i, the standard trick is to combine the optimality condition (B.22a) with the primal feasibility condition $0 \leq \alpha_i \leq C_1$, the dual feasibility conditions $p_i \geq 0$, $q_i \geq 0$ and the complementarity conditions (B.22d, B.22e) to obtain

$$\alpha_i = \text{clip}_{[0,\,C_1]}\big(\alpha_i^0 - \lambda\big). \tag{B.23}$$

Similarly for any fixed j, we combine the optimality condition (B.22b) with the primal feasibility condition $0 \leq \beta_j \leq C_2\delta$, the dual feasibility conditions $u_j \geq 0$, $v_j \geq 0$ and the complementarity conditions (B.22f, B.22g) to obtain

$$\beta_j = \text{clip}_{[0,\,C_2\delta]}\big(\beta_j^0 + \lambda\big), \tag{B.24}$$

$$v_j = \text{clip}_{[0,\,\infty)}\big(\beta_j^0 + \lambda - C_2\delta\big). \tag{B.25}$$

Note that we now obtain the following system

$$\sum_{i=1}^{n_+} \text{clip}_{[0,\,C_1]}\big(\alpha_i^0 - \lambda\big) - \sum_{j=1}^{\tilde{n}} \text{clip}_{[0,\,C_2\delta]}\big(\beta_j^0 + \lambda\big) = 0,$$

$$\delta - \delta^0 - C_2 \sum_{j=1}^{\tilde{n}} \text{clip}_{[0,\,\infty)}\big(\beta_j^0 + \lambda - C_2\delta\big) = 0.$$

Here, the first equation follows from plugging (B.23) and (B.24) into the feasibility condition $\sum_i \alpha_i = \sum_j \beta_j$ while the second equation follows from plugging (B.25) into (B.22c). Finally, system (4.22) follows after making the substitution $C_2\delta = \lambda + \mu$. ■

Lemma 4.11

Function h is non-decreasing in μ on $(0, \infty)$.

Proof of Lemma 4.11 on page 42:
Consider any $\mu_1 < \mu_2$. Then from (4.22b) we obtain both $\lambda(\mu_1) \geq \lambda(\mu_2)$ and $\mu_1 + \lambda(\mu_1) \geq \mu_2 + \lambda(\mu_2)$. The statement then follows from the definition of h in (4.23). ■

C

Appendix for Chapter 5

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 5.3 for Rec@K

The assumption of Theorem 5.3 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 5.3 is based on Lemma 5.2. We will now obtain the variant of Lemma 5.2 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 5.3 will then be modified into

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

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.

[1] https://anonymous.4open.science/r/AccuracyAtTop-7562

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

Bibliography

[1] Giuseppe Viale. The current state of breast cancer classification. *Annals of Oncology*, 23:207–210, 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] Charles E Metz. Basic principles of roc analysis. In *Seminars in nuclear medicine*, volume 8, pages 283–298. Elsevier, 1978.

[12] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*, pages 3121–3124. IEEE, 2010.

[13] Mohammad Hossin and Md Nasir Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1, 2015.

[14] James P Egan and James Pendleton Egan. *Signal detection theory and ROC-analysis*. Academic press, 1975.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] Corinna Cortes and Mehryar Mohri. Auc optimization vs. error rate minimization. *Advances in neural information processing systems*, 16, 2003.

[20] Shivani Agarwal, Thore Graepel, Ralf Herbrich, Sariel Har-Peled, Dan Roth, and Michael I Jordan. Generalization bounds for the area under the roc curve. *Journal of Machine Learning Research*, 6(4), 2005.

[21] Lukáš Adam, Václav Mácha, Václav Šmídl, and Tomáš Pevný. General framework for binary classification on top samples. *Optimization Methods and Software*, pages 1–32, 2021.

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

[23] 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.

[24] 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.

[25] 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.

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

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

[28] 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.

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

[30] 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.

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

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

[33] 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.

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

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

[36] 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.

[37] 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.

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

[39] 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.

[40] Lukáš Adam and V Mácha. Projections onto the canonical simplex with additional linear inequalities. *Optimization Methods and Software*, pages 1–29, 2020.

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

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

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

[50] 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.

[51] 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.

[52] 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.

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

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

[55] 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.

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

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

[58] 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.

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

[60] 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.

[61] 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.

[62] 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.

[63] 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.

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

[65] 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.

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

[67] 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.

[68] 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.

[69] Wlodzimierz Ogryczak and Arie Tamir. Minimizing the sum of the k largest functions in linear time. *Information Processing Letters*, 85(3):117–122, 2003.