FAST OPTIMIZATION ALGORITHMS FOR AUC MAXIMIZATION

by

Michael Natole, Jr.

A Dissertation

Submitted to the University at Albany, State University of New York

in Partial Fulfillment of

the Requirements for the Degree of

Doctor of Philosophy

College of Arts & Sciences

Department of Mathematics and Statistics

2020

Abstract

Stochastic optimizations algorithms like stochastic gradient descent (SGD) are favorable for large-scale data analysis because they update the model sequentially and with low per-iteration costs. Much of the existing work focuses on optimizing accuracy, however, it is known that accuracy is not an appropriate measure for class imbalanced data. Area under the ROC curve (AUC) is a standard metric that is used to measure classification performance for such a situation. Therefore, developing stochastic learning algorithms that maximize AUC in lieu of accuracy is of both theoretical and practical interest. However, AUC maximization presents a challenge since the learning objective function is defined over a pair of instances of opposite classes. Existing methods can overcome this issue and achieve online processing but with higher space and time complexity. In this thesis, we will develop two novel stochastic algorithms for AUC maximization. The first is an online method which is referred to as SPAM. In comparison to the previous literature, the algorithm can be applied to non-smooth penalty functions while achieving a convergence rate of $\mathcal{O}(\log T/T)$. The second is a batch learning method which is referred to as SPDAM. We establish a linear convergence rate for a sufficiently large batch size. We demonstrate the effectiveness of such algorithms on standard benchmark data sets as well as data sets for anomaly detection tasks.

Acknowledgements

I would like to thank the following people for whom without their help and support, this work would not have been possible.

First and foremost, I would like to thank my advisor Dr. Yiming Ying who first introduced me to the field of machine learning. I am grateful for his continual support and patience during my graduate studies and the course of my PhD.

I would also like to thank the faculty of the Mathematics & Statistics department. Specifically, I would like to thank Dr. Yunlong Feng, Dr. Karin Reinhold, and Dr. Boris Goldfarb for being a part of my committee. Also, I would like to thank Joan Mainwaring, JoAnna Aveyard, and Rosemary Bellanger.

I would also like to thank my parents, Michael and Paula, and my siblings, Michela, Amber, and Antonio for their continued support during my graduate studies.

Previous Publications

The material in chapter 4 appeared in [68] and the material in chapter 5 appeared in [69]. See `http://proceedings.mlr.press` and `https://www.frontiersin.org/legal/copyright-statement`, respectively, for permissions to publish. The work in both publications have been revised for this thesis and is being included since they were part of the line of research. For both publications, I was the lead author for each work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Motivations

Machine learning is an ever growing blend of mathematics and computer science to analyze data, identify patterns, and to make accurate predictions. Over the past few years, prediction algorithms have been applied to several application domains including computer vision [60], web search, natural language processing [8], and many others. This has resulted in technology (i.e. phones, cars, etc.) benefiting tremendously from learning algorithms. Voice assistants are now ubiquitous and cars are now learning to drive themselves.

A driving force behind such innovations has been *big data*. Since storage systems have become cheap and ubiquitous, it is now easy to store huge amounts of information that was not possible before. This has attracted the attention of numerous industries because they are intrigued by its potential. Businesses could use this information to make well-informed decisions to lower costs and increase sales. Even more so, the impact of big data goes beyond business. Political candidates use data to make better campaign decisions to try to gain more votes and determine who is the leading candidate. Companies, such as *Facebook* and *Twitter*, collect large amounts of information so that they can specifically target the most realistic voter. An important question to consider then is how is this done? How is a person labeled or categorized? *Machine learning* has become more and more crucial with the size of available data. Existing methods are not able to scale efficiently, and therefore new methods are of critical importance.

At the highest level, machine learning can be classified as either *unsupervised* or *supervised* learning. In unsupervised learning, the learner is given unlabeled training data, and the goal is to draw conclusions on future data. For example, given a number of social media posts, you might want to create an unsupervised learning algorithm that groups the posts into various topics. This task is an example known as *clustering*. *Dimensionality reduction* is also another example of unsupervised learning. In this problem, the data has a large number of features. It is then desirable to reduce the number of features to find a lower-dimensional

representation that maintains the properties of the original data. In supervised learning, the learner is given *labeled* training data with the intention to make predictions on future data points. For example, given the characteristics of a house, you would want to predict the sale price. This is an example of *regression*. *Classification* and *ranking* are also other examples of supervised learning. For classification, a category is assigned to an item while ranking is when items are to be ordered. Typically, ranking is based on a score based system, but preference-based settings are also used. Ranking is preferred over classification because a user will usually only examine the top ten or so documents. For example, a fraud detection officer can not investigate all suspected cases, but rather will consider the ones with the highest degree of certainty. In most scenarios, however, datasets are only assigned a label with no information on how they are ordered.

Throughout this thesis, we will focus on the supervised learning problem of classification. This problem type is very popular among the machine learning community and is ubiquitous in our daily lives. For example, email is a popular method of communication, but not every email is legitimate. Our email systems categorize email as either spam or not spam. In other aspects of our lives, we are influenced by meaningful classification algorithms. Popular companies such as *Netflix* and *Amazon* rely heavily on having deep recommendation systems for their users. Netflix wants to present content that a user will watch based upon past viewing history while Amazon recommends products based upon previous purchases. Therefore, it is of critical importance to determine the effectiveness of these methods. In this thesis, we will focus on designing algorithms that are both effective and efficient in the problem of binary classification.

1.1 Binary Classification

In supervised learning, the goal is to learn a function $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the input space that is typically a domain in $\mathbb{R}^d$ (d is the number of features), and $\mathcal{Y}$ is the output space. In classification, the features of an object are used to determine a label in the output space. We will restrict this work to the binary case. Specifically, we will consider data $\{(\mathbf{x}_t, y_t)\}_{t=1}^T$ where $\mathbf{x}_t \in \mathcal{X}$ and $y_t \in \{-1, 1\}$. Let z_t denote the pairing $(\mathbf{x}_t, y_t)$. Throughout this thesis, we will let $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})$, where $\mathbf{w} \in \mathbb{R}^d$ is the parameter to be learned.

For binary classification, and machine learning in general, it is common to solve the

Name	**Loss Function** $\ell(f(x), y)$
Hinge [3]	$\max(0, 1 - yf(x))$
Squared Hinge [44]	$\frac{1}{2}\max(0, 1 - yf(x))^2$
Exponential [19]	$\exp(-yf(x))$
Logistic [12]	$\log(1 + \exp(-yf(x)))$
Least Squares [107]	$\frac{1}{2}(f(x) - y)^2$
Least Absolute Deviation	$\lvert f(x) - y \rvert$
Quantile Regression [47]	$\max(\tau(f(x) - y), (1 - \tau)(y - f(x)))$
ϵ-Insensitive [104]	$\max(0, \lvert f(x) - y \rvert - \epsilon)$
Huber's Robust Loss [66]	$\frac{1}{2}(f(x) - y)^2$ if $\lvert f(x) - y \rvert \leq 1$, else $\lvert f(x) - y \rvert - \frac{1}{1}$
Poisson Regression [16]	$\exp(f(x)) - yf(x)$

Table 1.1: Various Loss Functions

following empirical risk minimization (ERM) problem:

$$\min_{\mathbf{w}} J(\mathbf{w}) = R_{\text{emp}}(\mathbf{w}) + \Omega(\mathbf{w}), \tag{1.1}$$

where $R_{\text{emp}}(\mathbf{w})$ defines the empirical error and $\Omega(\mathbf{w})$ is the regularizer. The empirical error is the average loss incurred by some loss function ℓ after T observations:

$$R_{\text{emp}}(\mathbf{w}) = \frac{1}{T}\sum_{t=1}^{T} \ell(z_t, \mathbf{w}).$$

Typically, an indicator function is used for detecting when a misclassification made by a model has occured, i.e. $\ell(z_t, \mathbf{w}) = \mathbb{1}_{[y_t \mathbf{w}^T \mathbf{x}_t < 0]}$. This results in the following problem to be solved:

$$\min_{\mathbf{w}} \frac{1}{T}\sum_{t=1}^{T} \ell(z_t, \mathbf{w}) + \Omega(\mathbf{w}). \tag{1.2}$$

A common procedure in machine learning is to replace the indicator function by a convex surrogate loss function. There are many different common loss functions used in machine learning with each being used for a different application area. For example, least absolute deviation is used with image restoration [117]. A few of them are shown graphically in Figure 1.1 and many others are summarized in Table 1.1.

The second important consideration to make is the selection of a regularizer for the $\Omega(\mathbf{w})$ term. Regularization is used to avoid overfitting of the data. The most popular choice is the L_2 regularization, i.e., $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$. However, many other choices exist, such as

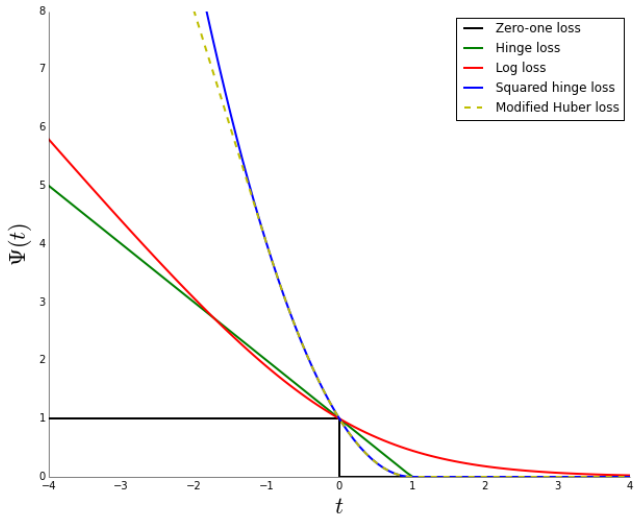


Figure 1.1: Various convex loss functions against $0-1$ loss.

elastic net, i.e. $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + \|\mathbf{w}\|_1$. For example, letting $\Omega(\mathbf{w}) = \|\mathbf{w}\|_1$ penalizes the empirical loss for adding large weights to $\mathbf{w}$. This leads to Lasso-type estimation algorithms [63]. A summary of common regularizers is given in Table 1.2.

By selecting various loss functions and regularizers, we have standard methods that can be used to solve a given machine learning problem. For example, the simplest method is the use of support vector machines (SVM). The empirical risk R_{SVM} minimized by SVM is the average of the hinge loss:

$$R_{\text{SVM}} = \frac{1}{T}\sum_{t=1}^{T} \max(0, 1 - y_t \mathbf{w}^T \mathbf{x}_t). \tag{1.3}$$

This method, originally developed by [103], is based on the idea of finding a hyperplane that separates the data such the distance between the hyperplane and the nearest data point is maximized. This is shown graphically in Figure 1.2.

Regularizer	$\Omega(\mathbf{w})$
L_2	$\frac{1}{2}\|\mathbf{w}\|_2^2$
L_1	$\|\mathbf{w}\|_1$
Elastic Net	$\frac{1}{2}\|\mathbf{w}\|_2^2 + \|\mathbf{w}\|_1$

Table 1.2: Various Regularizers

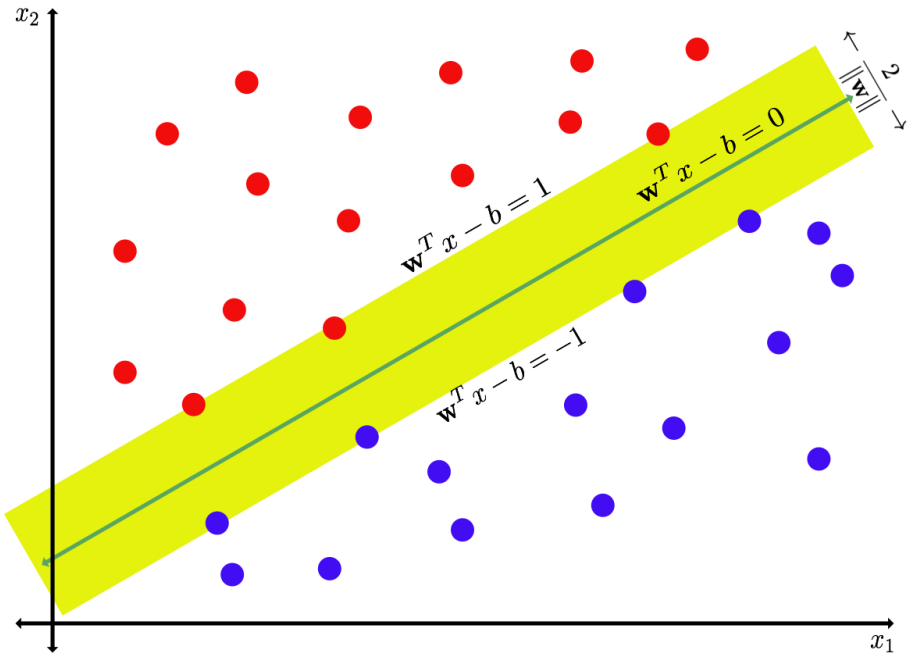


Figure 1.2: A graphical explanation of SVM

From the formulation in (1.1), we can also recover different standard problems by selecting various loss functions. Another popular choice is to use the logistic loss function with the L_2 regularizer, called logistic regression. This type of model is widely used in the medical community to predict mortality in injured patients. However, the focus of this thesis is on classification, and we will be using the least square loss function as well as various regularizers.

Finding the minimizer of $J(\mathbf{w})$ has two different approaches. The first way to obtain the solution $\mathbf{w}^*$ of problem (1.1) is by way of batch learning methods. These methods require storing of the data and learn by using either all available samples or a subset of the available samples. An important consideration for which algorithm is applicable is whether or not the objective function is smooth. In the case that $J(\mathbf{w})$ is not differentiable, two popular approaches are cutting plane methods (CPM) [45] and bundle methods [101]. CPM uses a generalization of gradients appropriate for convex functions, the *subgradient*. This includes for functions which are not necessarily smooth. By the definition of a subgradient, we can bound $J(\mathbf{w})$ from below by its first-order Taylor expansion. This lower bound forms the basis for cutting plane methods. Although CPMs can converge to the minimum, this

method is often very slow when iterates are too far away from previous ones (i.e. CPMs can have an unstable "zig-zag" behavior in the iterates). To stabilize CPMs, bundle methods have been proposed by introducing a proximity control function. This prevents overly large steps in the iterates [46]. There are 3 popular types of bundle methods: *proximal* [46], *trust region* [90], and *level set* [52]. These three variants each choose the L_2 norm, however, they all differ in how they compute the new iterate. SVM^{perf} has been shown to converge with a rate of $\mathcal{O}(1/\sqrt{T})$ and has several advantages. Using a modified version of the primal SVM formulation, the algorithm has a very easy implementation and a meaningful stopping criterion that avoids wasted time on solving the given problem [42].

Now when $J(\mathbf{w})$ is twice differentiable and convex, $\mathbf{w}^*$ is optimal when $\nabla J(\mathbf{w}^*) = \mathbf{0}$ (and when $X = \mathbb{R}^d$). In this situation there are numerous available methods for solving this type of problem. The most famous of all methods is Newton's method. Other standard techniques that can be applied are quasi-Newton methods. Unlike Newton's method, quasi-Newton methods can be used if computing the Jacobian or Hessian is too difficult or expensive for each iteration. These methods are generally used to find zeroes or local maxima and minima of functions. The most popular of which is the BFGS algorithm [108]. The method only requires that the gradient of the objective function needs to be available at each iterate. More importantly, since second derivatives are not required, quasi-Newton methods can be more efficient than Newton's method. Other state-of-the-art batch learning algorithms include OCAS [31] and LIBLINEAR [26]. OCAS, again based on cutting plane methods, has been shown to outperform SVM^{perf} while maintaining the same precision as the support vector solution. LIBLINEAR is an open source library for solving linear classification problems on a large scale. The package includes support for logistic regression and linear SVM. For the case of hinge loss, the package uses a coordinate descent method to solve for the optimal parameter [40]. More recently, NESVM [119], based on Nesterov's method [71] has a convergence rate of $\mathcal{O}(1/T^2)$. The algorithm smoothes the non-differentiable parts of the objective function, i.e., hinge loss and the ℓ_1-norm, and then uses gradient methods to solve the smoothed optimization.

Overall, batch algorithms have some very lucrative advantages but some disadvantages as well. Since either a subset or the entire data set is required, they are able to completely optimize the cost function for a set of training samples. This makes them very accurate.

However, such algorithms cannot evaluate as many samples because it must iterate several times over the training set to obtain the optimum [5]. Because many samples are considered at the same time, the computational complexity of these algorithms can be high. Even more so, if the data is of high dimension, that will only add to the complexity of the algorithm. In some cases, this could make the algorithm useless. Also, since these algorithms require access to most of the data, it also needs to be stored on the system and then loaded into memory [54].

The other approach to finding the minimizer is by using online learning algorithms. In the online setting, each instance of data is used to update the model iteratively. The efficiency and scalability of online learning makes the topic desirable for sequential data and of interest to the machine learning community since the data is not required to be stored. The most well known method for online learning is the *Perceptron Algorithm* [87]. The main idea behind predicting a sample $\mathbf{x}_t$ is to use a weight vector, and determine if $\mathbf{w}_t \cdot \mathbf{x}_t > 0$ for a positive sample; otherwise it would be classified as a negative sample. More recent work includes the first-order *Passive-Aggressive Algorithm* (PA) [14]. The PA algorithm introduces the maximum margin principle for classification. The idea is similar to the Perceptron Algorithm with the idea that we desire $\text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t) = y_t$. The method can be interpreted as a projection onto the half-space of vectors that gives a hinge-loss of zero. Therefore, the algorithm is *passive* when there is no change between iterations. However, when the hinge-loss is positive, the algorithm then forces the new solution to satisfy the hinge-loss constraint. This step has a closed form solution by using Lagrangian multipliers and is considered *aggressive* because there are no restrictions on the necessary step size. Hence, the algorithm is called the *Passive-Aggressive* Algorithm. However, a major problem of the PA algorithm is it fails to control the direction and scale of the parameter updates. Pegasos [92], a state-of-the-art algorithm, is a first order online method that introduces a chosen step size for each iteration. The computational cost of the algorithm can be dramatically reduced if the feature space is sparse, however, if the space is dense, such as in computer vision tasks, it will hardly outperform other algorithms such as SVM^{perf} [41].

To address these issues, some second-order online learning algorithms have been introduced and apply parameter confidence to improve the online learning performance. The first of which is the Adaptive Regularization of Weights (AROW) algorithm which uses con-

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

Table 1.3: The structure of a contingency table.

fidence weighted (CW) learning [22]. The method gives confidence in a linear classifier by maintaining a Gaussian distribution over the weights with mean $\mu \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$. The classifier is then trained by the passive-aggressive rule [14]. Further work has been done in this direction [15, 73].

The main draw of online algorithms is that they work by drawing a fresh random sample and adjust the parameters based on this new single sample only with very few statistical assumptions. In contrast to batch learning algorithms, this makes online learning algorithms very lucrative for when handling streaming data. Additionally, if new data arrives quickly, a large number of samples can be processed quickly because of the simple and low computational cost of the algorithms. On the other hand, however, because of the use of a single sample for updates, the cost function is usually not fully optimized. Online algorithms may seem hopelessly slow, however, it has been shown that they can converge just as fast as batch algorithms [5].

1.2 The Case Against Accuracy and other Measures

A crucial consideration of any algorithm is to measure its performance on predicting future data. So how does one measure the performance of a classifier? For binary classification, the decisions made by a classifier can have four possible outcomes. If the instance is positive and labeled as positive, then it is a *true positive* (TP); if it is labeled as negative, then it is a *false positive* (FP). If the instance is negative and is classified as negative, then it is a *true negative* (TN); if it is labeled positive, then it is a *false negative* (FN). These four possible decisions made by a classifier can be made into a structure called a *confusion matrix*, denoted by C. A confusion matrix, or as it is sometimes called a *contingency table*, is pictured in Table 1.3.

It is important to note that correct decisions made by the classifier are recorded along the major diagonal. Likewise, incorrect decisions are recorded along the minor diagonal.

	Actual Positive	Actual Negative
Predicted Positive	1	11
Predicted Negative	9	979

Table 1.4: Example of a confusion matrix.

Two important measures that can be calculated from the confusion matrix are the *true positive rate* (also called the *hit rate* or *recall*) and the *false positive rate* (also called the *false alarm rate*). To determine these values, let P denote the number of samples that are of the positive class and let N denote the number of samples that are of the negative class. These measures can then be calculated below:

$$\text{TPR} = \frac{TP}{P}, \text{FPR} = \frac{FP}{P}. \tag{1.4}$$

Using the confusion matrix as given above, there are many different types of measures that can be constructed from it, the most natural of course is accuracy (or misclassification error). Accuracy is then defined in terms of the confusion matrix as:

$$Accuracy = \frac{TP + TN}{P + N}. \tag{1.5}$$

Many different areas are interested in optimizing accuracy, however, an important question would be to consider is accuracy a robust metric for measuring perforamnce? It is has been determined that classification accuracy is a poor performance metric [80, 81]. An important distinction should be made, in that the statistical tests that are used are not to be questioned [20, 89], but it is important to question whether accuracy is an appropriate measure. As a motivating example, consider the confusion matrix given in Table 1.4. The classifier above has 98% accuracy, however, it was only able to successfully predict the majority class.

To precisely explain why this measurement is poor, consider two different accuracy measures: the accuracy of correctly identifying an input (correct acceptance) and the accuracy of correctly identifying an input as a nonmember of the class (correct rejection). Between these two individual accuracies, the classifier described in the previous example does not sufficiently predict the samples [65]. This reveals an underlying issue with accuracy in that it assumes the class distribution is known for the target [81]. In many cases for

benchmark data sets, we often do not know whether the existing distribution is the natural distribution. For example, the splice junction data set has 50% donor sites, 25% acceptor sites, and 25% nonboundary sites, however, no more than 6% of DNA actually codes for human genes [88]. An argument by some researchers is that large class skews and large changes in distribution may be seen as unrealistic. However, in many real world domains, such as fraud detection [27], vision [62], medicine [60], and language [8], class skews of 10^1 and 10^2 are very common. Distributions with skews up to 10^6 have been observed [9, 28, 49, 88, 53]. Changes in distribution are also not uncommon and occur in many different application domains. For example, epidemics will cause medical decision making to change as the number of cases of the disease increases over time. Fraud detection is another such example as well. Proportions of fraud can very from month to month and between different locations [27]. Lastly, a change in a manufacturing procedure could drastically change the number of defective units that are produced. In each of the cases, the prevalence of a particular class may change without any changes or alterations to the characteristics of a class. In these cases, predicting the minority class is the primary objective and this critical issue makes accuracy useless for model comparison for real world examples.

Another problematic assumption of accuracy is that it assumes the misclassification costs are equal [81]. This assumption does not translate well into real-world scenarios. There are many examples in medical diagnosis, decision theory, and pattern recognition where there is huge cost difference between positive and negative classes. For example, the cost of missing a case of fraud is very different from the cost of a false alarm [27]. From these issues, using accuracy to measure the performance of an algorithm is not a robust method. However, much of the existing methods focus on optimizing the classification error [5, 97, 64, 112]. Therefore, it is of the utmost importance to design algorithms that optimize another performance measure that is more robust than accuracy.

Prevailing over the shortcomings of accuracy when class distributions are unbalanced and the misclassification costs are unequal is best done by using a more appropriate performance measure [61]. Just as we derived accuracy from the confusion matrix, different measures can also be derived. Many of these are reserved for specific problems. In the area of pattern recognition and information retrieval, precision and recall is a performance

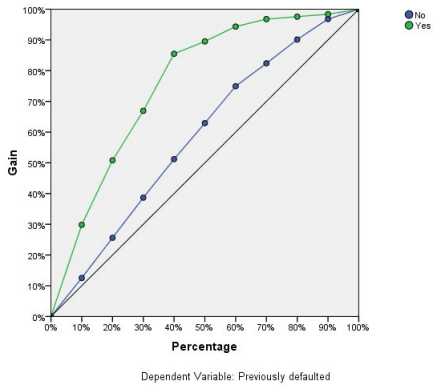


Figure 1.3: Example of a lift chart.

measure that is commonly used. We have that precision and recall are defined as:

$$\text{Prec}(C) = \frac{TP}{TP + FP} \text{ and } \text{Rec}(C) = \frac{TP}{P}, \tag{1.6}$$

where the values above are from the confusion matrix, C. Precision is defined as the fraction of relevant instances among the retrieved instances while recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. This method is popular to use for web search engines since the user typically only scans the first few results that are presented. A second type of measurement from this is called the *precision - recall breakeven point* (PRBEP). This requires that a classifier make a decision such that the precision and recall are equal. This gives that at any time, the following must hold:

$$TP + FN = FP + TN. \tag{1.7}$$

A problem with precision-recall is that it requires optimizing two values, instead of a single number. A way to combine the precision and recall into a single measurement is to use a weighted harmonic average between the two, called the F_β-Score. It is defined by:

$$F_\beta(C) = \frac{(1 + \beta^2)TP}{(1 + \beta^2)TP + FP + FN}. \tag{1.8}$$

The typical value of β is usually 1. This measure is used to evaluate binary classifiers for natural language applications such as text classification.

Visualization tools are also very popular and beneficial to gain insight into the perfor-

mance of a classifier. A lift chart is a well known visualization tool in data mining for binary classifiers in marketing and sales applications [4]. *Lift* is a measure of the performance of a model at predicting samples that will result in a response, with respect to the population as a whole, measured against a model that selects randomly. For example, consider a marketing company that wants to plan a strategy to advertise to households with specific attributes. Even though the ad has a small cost, the company wishes to advertise to consumers that have the highest chance of purchasing the product. Therefore, it is of great interest to design a classifier to determine whether or not a household is a potential customer or not. To examine the relationship between advertisement cost and customer acquisition, a lift chart is used to evaluate the performance of such a classifier. A critical assumption that is made is that the number of customers, P, is generally unknown and therefore the true positive rate cannot be calculated. A lift chart consists of the expression of the positive response rate on the vertical axis while the horizontal would be the percent of customers contacted with each point in the lift chart represents a binary classifier. By varying the threshold of a probabilistic classifier, we obtain a set of classifiers, and therefore a set of points. The curve given by this collection of points gives a *lift chart* [111, 106]. The area under a lift chart has a well defined statistical definition given by:

$$A_{\text{lift}} = P \cdot \mathbb{E}[\mathbb{I}_{\mathbf{w}^T(\mathbf{x}-\mathbf{x}')\geq 0} | y = 1], \tag{1.9}$$

where P is the number of positive samples. Equation (1.9) represents the probability of a random positive sample ranking higher than a random sample (either positive or negative). The optimal value for the area under the lift chart is given by P.

1.3 ROC Curves and AUC

In this section, we will introduce the *Receiver Operating Characteristic* (ROC) curve and the area under the ROC curve (AUC) [38]. This visualization tool and metric is considered by many as a robust method for model comparison. To obtain a better understanding of ROC analysis, we first use the following motivating example. In the area of marketing, advertisers wish to only promote their ad to customers who are most likely to buy the product or service. If accuracy was used, we would only be able to classify customers either as buyers or non-buyers, however, it is common practice to use different market strategies to customers

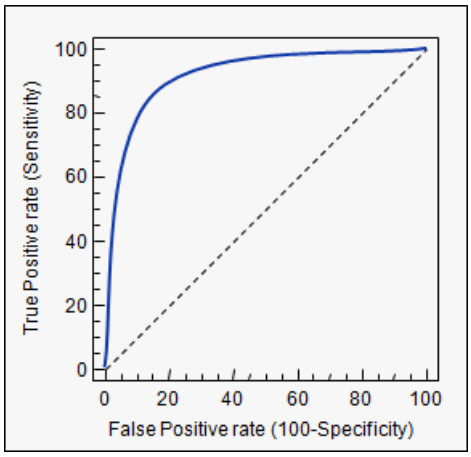


Figure 1.4: Example of an ROC curve.

with different probabilities of buying the product. It would be best to then have a ranking of customers in terms of their buying potential. Therefore, solving a ranking problem is more desirable than just a classification problem [57]. An important consideration follows from this. It is assumed that to that develop a better classifier, the true ranking of the training samples would be needed to be known, however, in practice this information is almost always unknown with only the training and testing labels are given [11]. So this naturally leads to the question: are there better methods than accuracy to evaluate the performance of a classifier that also produces rankings? The answer to this is, of course, the area under the ROC curve (AUC).

The ROC curve was originally developed in signal detection theory in connection with radio signals during WWII by engineers for detecting enemy objects in battlefields [25]. Since then, they have been used prominently in signal detection theory to measure the concessions between hit rates and false alarm rates of classifiers [25, 99]. Their use was then introduced into psychology to account for perceptual detection of stimuli, and then was further expanded into radiology, medicine, biometrics, forecasting natural hazards, meteorology, model performance assessment, and other areas [76, 67, 77, 122]. The work in [98] enlarged ROC analysis for use in visualizing and analyzing the behavior of diagnostic systems. Recently, ROC analysis has been introduced into machine learning and data mining research. The earliest use of ROC graphs in machine learning was by [96] who showed the robustness of evaluating and comparing algorithms using ROC curves since classification accuracy is often a poor metric for measuring performance [81]. The use of ROC analysis has only continued

Algorithm 1 Conceptual Method for Calculating an ROC Curve

Input: L - list of test examples, f - the probabilistic classifier's estimate that instance i is positive, min and max - the smallest and largest values returned by f, *increment* - the smallest difference between any two f values

```
for t = min to max by increment do
  FP ← 0
  TP ← 0
  for i ∈ L do
    if f(i) ≥ t then
      if i is a postive sample then
        TP ← TP + 1
      else
        FP ← FP + 1
      end if
    end if
  end for
  Add point (FP/N, TP/P) to ROC curve
end for
```

to expand in the machine learning community because ROC graphs have properties that make them desirable for datasets with unequal class distributions and cost-sensitive learning. ROC curves also have the added benefit by using the *area under the ROC curve* (AUC) as a performance metric for comparison.

The definition of an ROC curve is based on values from a confusion matrix. Recall that the *true positive rate* (also called the *hit rate*) of a classifier is TP/P. The *false positive rate* (also called the *false alarm rate*) is FN/N. ROC graphs are then two-dimensional graphs with the true positive rate on the y-axis and the false positive rate on the x-axis using changing thresholds. For a given discrete classifier, such as a decision tree, each decision by the classifier outputs a (fp rate, tp rate) pair for a point on the ROC curve. Once a classifier has made decisions for a particular dataset, this leads to only a single confusion matrix and thus only to a single point in ROC space. Some classifiers, such as a Naive Bayes classifier, produces a prediction based upon a numeric value, or probability, that an instance belongs to a particular class. For a higher probability, the greater the chance that the example will belong to a particular class. So on a very basic level, by varying a particular threshold (say from $-\infty$ to ∞), we can obtain a curve through ROC space. This procedure is not very precise nor computationally efficient, but explains the main observations of how an ROC curve is constructed.

Algorithm 2 Practical Method for Calculating an ROC Curve from Test Set

Input: L - list of test examples, f - scoring function (probability that an example is positive), P - number of positive sample, N - number of negative samples
Output: R - list of ROC points
$L_{sorted} \leftarrow L$ sorted by decreasing f scores
$FP \leftarrow TP \leftarrow 0$
$R \leftarrow \{\}$
$f_{prev} \leftarrow -\infty$
$i \leftarrow 1$
while $i \leq |L_{sorted}|$ **do**
  $f(i) \neq f_{prev}$ then
  Push $(\frac{FP}{N}, \frac{TP}{P})$ onto R
  $f_{prev} \leftarrow f(i)$
  end if
  if $L_{sorted}[i]$ is a positive example **then**
  $TP \leftarrow TP + 1$
  else
  $FP \leftarrow FP + 1$
  end if
  $i \leftarrow i + 1$
  end while
  Push $(\frac{FP}{N}, \frac{TP}{P})$ onto R (i.e. $(1, 1)$)
end while

An efficient approach to ROC construction begins with a scoring function $f(i)$ that assigns a probability to a particular instance, i. This allows the samples to be ordered with respect to the assigned probability. By assigning a particular threshold, we can then easily classify the given instances. A property that is of great interest from this is the monotonicity of thresholds, i.e., if an instance is classified as positive, then it will remain classified as positive for any lower threshold. So for a fixed threshold, we can order the samples by decreasing f scores, then one can move down the list to update the true positive and false positive rates. In this manner, an ROC curve can be created by a linear scan. The conceptual method is summarized in Algorithm 1. The practical method is summarized in Algorithm 2 that maintains a stack of ROC points. A new point is pushed onto the stack after each point is processed. The algorithm has the typical complexity of $\mathcal{O}(n \log n)$ for sorting and then $\mathcal{O}(n)$ for the linear scan for a complexity of $\mathcal{O}(n \log n)$. An important consideration to note for Algorithm 2 is that any point with the same f score as a previous sample will be omitted since we are only looking for the expected performance of the classifier.

The analysis of a constructed ROC curve has several areas of interest. The point $(0, 0)$ represents a classifier that never makes a classification where as the point $(1, 1)$ represents unconditionally making decisions. The upper left corner, i.e. the point $(0, 1)$, represents a perfect classifier with a false positive rate of 0. More generally, a point in the ROC space is better than another where the true positive rate is greater or the false positive rate is lower. A point in the left hand side of the ROC curve near the x-axis describes a classifier as being *conservative*. In other words, a classification is determined to be positive only if there is strong evidence that this will make a small error. Similarly, the upper right region of the ROC graph describes classifiers as *liberal* since positive classifications will be made with weak supporting evidence. From these descriptions, it is clear that liberal classifiers tend to have a high false positive rate while conservative classifiers have low true positive rates. The left hand-side of the ROC graph is of great importance since many application domains have a large number of negative instances while only a few positive instances. Additionally, the line $y = x$ represents the strategy of randomly guessing. For example, if half the samples are classified as positive half the time, then it is expected that only half the classifications will be correct for both the positive and negative classes. This is represented by the point $(0.5, 0.5)$. Any classifier that is in the lower right triangle, i.e. below the diagonal, is considered worse than random guess. However, if the classifier is negated, then it will produce a classifier in the desired upper left of the ROC space.

The widespread adoption of ROC curves for measuring classifier performance in many application areas can be attributed to the many benefits that they offer. First and foremost, plotting ROC curves on the same graph allow for a comparison between classifiers. For simplicity, consider two classifiers A and B. If classifier A lies entirely above classifier B, then A is said to *dominate* B. In other words, the classifier that lies above the others is considered the most *robust classifier* [78, 79]. However, an issue that could arise is that classifier A may not entirely dominate classifier B. To better determine which classifier is better, we desire a single scalar value, like accuracy, to represent the expected performance. A common method is to calculate the area under the ROC curve, denoted as AUC [7, 38]. Since AUC comes from the ROC curve, which is defined in the unit square, values are from 0 to 1, however, since the random guess line has an area of 0.5, no classifier should have an AUC score less than 0.5.

Analysis of ROC curves necessitates some basic considerations. First and foremost, the distributions for both the positive and negative classes are assumed to be normal (called the *binormal assumption*). Under this assumption, the decision threshold can be set anywhere, but it is commonly selected to minimize the Bayes error rate. However, if the error costs are unequal and known, then the decision threshold can be selected to minimize the overall cost of the errors. Also, it is important to note that the correspondence between the prior probability of a class and its error cost. If the class distribution of examples is consistent with the cost of errors, then it is straightforward to build a classifier coherent with those errors. However, if the data set is skewed in respect to the true cost of the errors, it may be difficult to build a classifier that is coherent with those errors. This remains true even if we know the cost of the errors in advance. Finally, we usually have very little evidence about the relationship between the class distribution and the class error. For example, consider the issue of predicting if a patient has cancer. It is obvious that the difference in cost is quite significant, but we don't have a way of conducting a cost analysis. In a situation like this, the decision threshold is varied to maximize the cost on the negative class rather than the positive class.

Inst. #	**Class (True)**	**Class (Hyp)**	**Score**
1	P	Y	0.999
2	P	Y	0.999
3	P	Y	0.992
4	P	Y	0.988
5	N	Y	0.974
6	N	Y	0.955
7	N	N	0.682
8	N	N	0.531
9	N	N	0.480
10	N	N	0.441

Table 1.5: Scores and classifications of 10 instances.

Even with careful considerations, ROC curves do have disadvantages. A particular point on a ROC graph measures *relative* sample scores meaning that the results only need to be able to differentiate between positive and negative samples. These results, however, are not required to be accurate. To better illustrate this point, it is best to consider an example. As from Table 1.5, the classifier used mislabeled examples 5 and 6, resulting in an 80% accuracy. However, the ROC curve illustrates a perfect classifier because it was able

to rank the positive samples over the negative samples. This discrepancy is because each metric is measuring something different. Accuracy is measuring the determined classification with respect to a given threshold (score > 0.5). If the values are probabilities, the accuracy would not be appropriate, but in this example they are not. This example shows when scores are not *properly calibrated* and there are methods to overcome this issue. Lastly, relative scoring presents an issue because they generally cannot be compared across model classes. For example, one model could produce scores between 0 and 1 while another might produce scores in the interval of $[1, 100]$. Thus, comparing model performance using a common threshold would not be possible.

Even with the existing issues and assumptions of ROC curves, they have a very attractive quality: they are not sensitive to changes in class distributions. If the proportion of positive to negative samples changes, or the number of negative instances greatly outnumbers the positive instances, ROC graphs are not affected by this. To understand why this is the case, recall the confusion matrix as defined in Table 1.3. The proportion of positive instances to negatives instances is based on information from both the left column (positive class) and the right column (negative class). If a metric uses information from both columns, then it will be sensitive to changes in class skew. Metrics such as accuracy, precision, recall, lift, and the F_β-score all use values from both columns and will be impressionable to changes in the class distribution even if the classifier performance does not. ROC graphs, on the other hand, are based on the true positive rate and false positive rate. Both of these measures are restricted to a single column calculation.

Finally, since AUC is derived from an ROC curve, then it is also insensitive to changes in class distributions. Even more so, AUC is of interest to researchers because it has a meaningful statistical property: the AUC score is equivalent to the probability that the classifier will rank a randomly chosen positive sample higher than a randomly chosen negative sample. Specifically, we can define AUC as follows:

Definition 1.1. *For a linear scoring function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, AUC is*

$$\begin{aligned} AUC(\mathbf{w}) &= \Pr(\mathbf{w}^T\mathbf{x} \geq \mathbf{w}^T\mathbf{x}' | y = 1, y' = -1) \\ &= 1 - \mathbb{E}[\mathbb{I}_{[\mathbf{w}^T(\mathbf{x}-\mathbf{x}')<0]} | y = 1, y' = -1] \end{aligned}$$

Algorithm 3 Calculating the area under an ROC curve

Input: L - list of test examples, f - scoring function (probability that an example is positive), P - number of positive sample, N - number of negative samples
Output: A - the area under the ROC curve
Require: $P > 0$ and $N > 0$
$L_{sorted} \leftarrow L$ sorted by decreasing f scores
$FP \leftarrow TP \leftarrow 0$
$FP_{\text{prev}} \leftarrow TP_{\text{prev}} \leftarrow 0$
$A \leftarrow 0$
$f_{prev} \leftarrow -\infty$
$i \leftarrow 1$
while $i \leq |L_{sorted}|$ **do**
  if $f(i) \neq f_{\text{prev}}$ **then**
    $A \leftarrow A + \text{Trapezoid}_{\text{Area}}(FP, FP_{\text{prev}}, TP, TP_{\text{prev}})$
    $f_{\text{prev}} \leftarrow f(i)$
    $TP_{\text{prev}} \leftarrow TP$
    $FP_{\text{prev}} \leftarrow FP$
  end if
  if i is a postive sample **then**
    $TP \leftarrow TP + 1$
  else
    $FP \leftarrow FP + 1$
  end if
  $i \leftarrow i + 1$
end while
$A \leftarrow A + +\text{Trapezoid}_{\text{Area}}(N, FP_{\text{prev}}, P, TP_{\text{prev}})$
$A \leftarrow A/(P \times N)$ /* Scale from $P \times N$ onto the unit square */

where $(\mathbf{x}, y)$, $(\mathbf{x}', y') \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ *are independent.*

This definition is equivalent to the Wilcoxon Test of Ranks [38]. In areas where ROC analysis is used, the AUC is also examined for precise model comparison. Computing the AUC score can be easily done by using trapezoids under the graph [7]. The method is summarized in Algorithm 3.

1.4 Summary of Thesis

In this thesis, we focus on the task of binary classification for AUC optimization and briefly describe the main contributions of this work below. In the next chapter, we explain the advantages and challenges of training algorithms to maximize the AUC score and review

the related work. In chapter 3, we formulate a modified version of the algorithm previously published in [113] that includes the use of a regularization term to be used as a benchmark for comparison. In chapter 4, we propose the online learning algorithm called Stochastic Proximal AUC Maximization (SPAM) with a theoretical analysis of its convergence of $\mathcal{O}(\log T/T)$. We also then demonstrate its effectiveness on standard benchmark data sets. In chapter 5, we propose the batch learning algorithm called Stochastic Primal-Dual AUC Maximization (SPDAM) that achieves a linear convergence rate. We confirm our results by demonstrating the algorithm on standard benchmark data sets. In chapter 6, using the methods discussed in chapters 3, 4, and 5, we test the algorithms on real world data sets for anomaly detection. Finally, the last chapter concludes with a summary of this thesis as well as some possible directions for future work.

Chapter 2

Optimizing AUC and Related Work

Recently, there has been increased interest in developing optimization algorithms for AUC maximization. Since the performance metric is used in so many different fields [89], it is natural to desire to develop algorithms that optimize AUC over accuracy. Progress has been made in both batch and online algorithms, however, the various approaches have severe limitations that hinders their usage in regards to high dimensional data as well as processing streaming data.

In the first section, we will review the advantages and challenges for using AUC to measure classifier performance. In the second section, we will review the related work. We will consider both batch and online algorithms and examine the strengths and weaknesses of the various methods. Specifically, we will examine how the proposed methods overcome the challenges of AUC optimization as well as the theoretical convergence rate.

2.1 Advantages and Challenges of AUC for Measuring Performance

In many application domains, AUC is used as the primary metric for evaluation instead of accuracy. The motivating reason behind this is that, just like with ROC curves, AUC is not susceptible to unequal class distributions. This is a compelling reason to use AUC over accuracy in many different application domains. Even more so, an additional benefit of AUC is that a classification problem can be reconsidered as a ranking problem. By the definition of AUC, the samples are ordered according to positive samples being ranked over negative samples. So not only will the samples be classified, but the samples in the minority class with greatest certainty can be determined. This can be done without an initial ordering of the samples. Only the initial class labels need to be known.

To make a more precise argument as to why AUC is better than accuracy, we first need

some important concepts as to how to compare two learning algorithms. Specifically, when we compare two measures f and g for evaluating learning algorithms A and B, we desire that both f and g be *consistent* with each other, i.e., if f concludes that algorithm A is strictly better than algorithm B then measure g will not contradict this conclusion. Furthermore, if f is more *discriminating* than g, then measure f should be able to tell the difference between algorithms A and B, while measure g can not. We now make the following topics more precise [55] with the following definitions:

Definition 2.1. *(**Consistency**) For two measures f and g on domain Ω, f and g are (strictly) consistent if there exists no $a, b \in \Omega$ such that $f(a) > f(b)$ and $g(a) < g(b)$.*

Definition 2.2. *(**Discriminancy**) For two measures f and g on domain Ω, f is (strictly) more discriminating than g if there exists $a, b \in \Omega$ such that $f(a) > f(b)$ and $g(a) = g(b)$, and there exist no $a, b \in \Omega$ such that $g(a) > g(b)$ and $f(a) = f(b)$.*

A good way to gain a better understanding of this is to compare numerical marks and letter marks for grading. Numerical marks are values $0, 1, ..., 100$ while letter marks are A, B, C, D, and F. Clearly numerical marks are consistent with letter marks (and vice versa), but more importantly numerical marks are more discriminating than letter marks, i.e. two students who receive 91 and 93 will both receive an A, but with letter marks it is impossible to determine which student was better. We will now give the probabilistic versions of the previous two definitions:

Definition 2.3. *(**Degree of Consistency**) For two measures f and g on domain Ω, let $R = \{(a, b)|a, b \in \Omega, f(a) > f(b), g(a) > g(b)\}$ and $S = \{(a, b) : \Omega, f(a) > f(b), g(a) < g(b)\}$. The degree of consistency of f and g is C ($0 \leq C \leq 1$), where $C = \frac{|R|}{|R|+|S|}$.*

Definition 2.4. *(**Degree of Discriminancy**) For two measures f and g on domain Ω, let $P = \{(a, b)|a, b \in \Omega, f(a) > f(b), g(a) = g(b)\}$ and $Q = \{(a, b) : \Omega, g(a) > g(b), f(a) = f(b)\}$. The degree of discriminancy for f over g is D, where $D = \frac{|P|}{|Q|}$.*

The connection between these two definitions is very important when evaluating two machine learning algorithms. Consider that if f and g are consistent with some degree C, then if f determines that A is better than B then there will also be a probability that g will determine that A is also better than B. Also, if f is D times more discriminating than g, then it is D

times more likely that f can determine the difference between algorithms A and B where as g cannot tell the algorithms apart. From this, it should be clear that if we want f to be a better measure than g, then we require that $C > 0.5$ and $D > 1$. This leads to the following definition:

Definition 2.5. *The measure f is statistically consistent and more discriminating than g if and only if $C > 0.5$ and $D > 1$. In this case, we say that f is a better measure than g.*

Using the previous definitions we desire to show that AUC is statistically more consistent and more discriminating than accuracy, by replacing the measure f by AUC and g by accuracy. The domain Ω is the ranked lists of testing samples (with n_- negative samples and n_+ positive samples). To show that AUC is better than accuracy it suffices to show that $C > 0.5$ and $D > 1$. This is summarized in the following theorems:

Theorem 2.1 (Theorem 1 in [55])**.** *Given a domain Ω, let $R = \{(a, b)|AUC(a) > AUC(b), acc(a) > acc(b), a, b \in \Omega\}$ and $S = \{(a, b)|AUC(a) < AUC(b), acc(a) > acc(b), a, b \in \Omega\}$. Then $\frac{|R|}{|R|+|S|} > 0.5$ or $|R| > |S|$.*

Theorem 2.2 (Theorem 2 in [55])**.** *Given a domain Ω, let $P = \{(a, b)|AUC(a) > AUC(b), acc(a) = acc(b), a, b \in \Omega\}$ and $Q = \{(a, b)|acc(a) < acc(b), AUC(a) = AUC(b), a, b \in \Omega\}$. Then $|P| > |Q|$.*

From Theorem 2.1 and Theorem 2.2 it is clear that AUC gives more information about the performance of a classifier than accuracy. These arguments make a strong justification that AUC is a preferred metric over accuracy.

Even with the benefits of using the AUC score for measuring the performance of a classifier, the measure still has some drawbacks. Recall that for a scoring function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, the AUC is defined as

$$\text{AUC}(f) = \text{P}(\mathbf{w}^T\mathbf{x} \geq \mathbf{w}^T\mathbf{x}'|y = 1.y' = -1),$$

where $(\mathbf{x}, y)$ and $(\mathbf{x}', y')$ are independent. It is clear from the definition that computing the AUC score is quadratic in nature, meaning that it relies on pairs of samples. Worse yet, those pairs need to be of opposite classes. In a real world scenario, data arrives sequentially,

not in pairs of different classes, making AUC difficult to adopt for streaming data. The computational cost of the AUC score is of critical importance to overcome.

Lastly, an important question to consider is how do AUC and accuracy (or error rate) relate. In [13], such a question was considered by examining the expectation and variance of AUC. For a classification task with n_+ positive samples and n_- negative samples, the expected value of AUC is the same as accuracy when $n_+ = n_-$. Otherwise, AUC is a monotonic function of accuracy. This makes it seem that there is nothing to be gained from designing algorithms that optimize AUC, i.e., a classification algorithm that minimizes the error will also optimize the AUC. However, the variance tells a different story entirely. An uneven class distribution will result in having a higher variance. The variance will further be increased by any errors that have been made by the classifier. This contradicts the claim in [109] that the error rate is zero for any error rate with even distributions ($n_+ = n_-$). Overall, optimizing error rate will not lead to an optimized AUC score. Therefore, it is best to optimize the AUC score directly.

2.2 Related Work

Designing algorithms that maximize the AUC score of a classifier is of much importance to researchers. It has been statistically shown that optimizing the error rate does not lead to optimized AUC values [13] and it is best to optimize the AUC score directly, however, doing so has significant challenges that have been addressed in different ways.

One of the first cases of optimizing the AUC score was with decision trees [30]. Typically, accuracy was used to measure the performance of a classifier. However, as argued in Section 1.2, accuracy is not appropriate when the misclassification costs are unequal or unknown. For a binary classification problem with n samples, there are 2^n possible classifiers that can be formed. Using ROC analysis, the classification from these classifiers can be viewed as an ordering of the data. This viewpoint has special characteristics such as for a given classifier, there exists a classifier that makes opposite predictions. The authors proposed a AUC-based splitting criterion, instead of an accuracy based approach. To eliminate some of the 2^n possibilities, it was determined that the convex hull of the 2^n possible outcomes is determined by the ROC points that have optimal labelings of the data. This led to better AUC values without loosing the same degree of accuracy using an accuracy-splitting

approach.

Optimizing the AUC score directly, however, presents major challenges due to the pairwise nature of the method. A common approach was to design algorithms that maximized an approximation of the AUC value [65, 109]. One of the earliest proposed strategies, the RankOpt algorithm, used gradient descent to optimize the AUC score directly by using the AUC statistic as its objective function [39]. AUC can be defined as:

$$\mathrm{AUC}(f) = \frac{\sum_{i=1}^{n_+} \sum_{j=1}^{n_-} \mathbb{I}_{f(\mathbf{x}_i) > f(\mathbf{x}_j)}}{n_+ n_-}. \tag{2.1}$$

Clearly from this formulation, AUC is not differentiable. To account for this, the authors replaced the indicator function with the heavy-side function:

$$g(x) = \begin{cases} 0 & x < 0 \\ 0.5 & x = 0 \\ 1 & x > 0. \end{cases}$$

AUC can then be reformulated as:

$$\hat{\mathrm{AUC}}(f) = \frac{1}{n_+ n_-} \sum_{i=1}^{n_+} \sum_{j=1}^{n_-} g(f(\mathbf{x}_i) - f(\mathbf{x}_j)), \tag{2.2}$$

where (2.2) is an unbiased estimator for AUC. However, the above formulation is again not differentiable. The authors replaced $g(x)$ by the sigmoid function, $s(x) = 1/(1 + e^{-x})$, since $\lim_{|x| \to \infty} s(x) = g(x)$ implies that for large $\|x\|$ the sigmoid function is a good approximation for AUC [109]. The first and second derivatives of the sigmoid function can also be easily determined. The computational complexity of the formulation is still $\mathcal{O}(n^2)$ in the number of observations. To overcome this, the authors proposed the following relationships for two positive class observations $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$ and two negative class observations $\mathbf{x}_{j_1}$ and $\mathbf{x}_{j_2}$:

$$(\mathbf{x}_{i_1} - \mathbf{x}_{j_1}) + (\mathbf{x}_{i_2} - \mathbf{x}_{j_2}) = (\mathbf{x}_{i_1} - \mathbf{x}_{j_2}) + (\mathbf{x}_{i_2} - \mathbf{x}_{j_1}). \tag{2.3}$$

Note that for these four pairs, the argument to the sigmoid function for any of these is completely determined by the other three pairs. Thus, using all $n_+ n_-$ pairs is not necessary.

The authors suggest to use only the $(k \mod P)$-th observation. This gives the following ranking statistic, which is of $\mathcal{O}(n)$ complexity,

$$R_l(\mathbf{w}) = \frac{1}{n_-} \sum_{j=1}^{n_-} s(\mathbf{w}^T(\mathbf{x}_{(k \mod P)} - \mathbf{x}_k) \tag{2.4}$$

Performing gradient descent on $R_l(\mathbf{w})$ will lead to an optimized AUC score, however, this is based on an approximation of the AUC score. This does not directly optimize the true AUC.

Another early method for optimizing the AUC score was to use support vector machines [84]. First, the authors considered the setup of trying to learn a linear classifier of the form $\text{sign}(f(\mathbf{x}))$ where $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$ and $\mathbf{w} \in \mathbb{R}^d$. AUC optimization is then equivalent to:

$$\text{AUC} = \frac{\sum_{i=1}^{n_+} \sum_{j=1}^{n_-} \mathbb{I}_{\xi_{ij}>0}}{n_+ n_-}$$
$$\xi_{ij} = f(\mathbf{x}_i^+) - f(\mathbf{x}_j^-).$$

Hence the problem of maximizing AUC becomes:

$$\max_w \frac{1}{n_+ n_-} \sum_{i=1}^{n_+} \sum_{j=1}^{n_-} \mathbb{I}_{\xi_{ij}>0}$$
$$\text{with } \xi_{ij} = f(\mathbf{x}_i^+) - f(\mathbf{x}_j^-), 1 \leq i \leq n_+, 1 \leq j \leq n_-.$$

Again, this formulation has several issues that need to be addressed. The objective function is not differentiable over the range of ξ_{ij}. Another issue is that the solution might not be unique. To overcome these two issues, an approximated differentiable function is needed along with a regularizer to avoid overfitting. Therefore, AUC optimization is equivalent to the following minimization problem:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n_+} \sum_{j=1}^{n_-} \xi_{ij}$$
$$\text{with } f(\mathbf{x}_i^+) \geq f(\mathbf{x}_j^-) + \rho - \xi_{ij}, 1 \leq i \leq n_+, 1 \leq j \leq n_-$$
$$\xi_{ij} \geq 0, 1 \leq i \leq n_+, 1 \leq j \leq n_-.$$

The parameter C allows for trade-off between the regularization term and the constraint violation ξ_{ij}. It is important to note that the authors chose a linear function of ξ_{ij}. Other works have done this in a similar manner [109]. The above optimization problem can now be easily solved via Lagrange multipliers. The Lagrangian function of the above optimization is:

$$L(\mathbf{w}, \xi_{i,j}) = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i,j=1}^{n_+,n_-} \xi_{ij} - \sum_{i,j=1}^{n_+,n_-} \alpha_{ij}(\langle \mathbf{w}, \mathbf{x}_i^+ - \mathbf{x}_j^- \rangle) - \rho + \xi_{ij}) - \sum_{i,j=1}^{n_+,n_-} \gamma_{ij}\xi_{ij}.$$

The Lagrangian derivatives are:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i,j=1}^{n_+,n_-} \alpha(\mathbf{x}_i^* - \mathbf{x}_j^*),$$
$$\frac{\partial L}{\partial \xi_{u,v}} = C - \alpha_{u,v} - \gamma_{u,v}.$$

Using the above derivatives leads to the following dual optimization problem:

$$\max_{\alpha} -\frac{1}{2} \sum_{i,j=1}^{n_+,n_-} \sum_{u,v=1}^{n_+,n_-} \alpha_{i,j}\alpha_{u,v}\langle \mathbf{x}_i^* - \mathbf{x}_j^-, \mathbf{x}_u^* - \mathbf{x}_v^- \rangle + \rho \sum_{i,j=1}^{n_+,n_-} \alpha_{i,j}$$
$$\text{with } C \geq \alpha_{i,j} \geq 0.$$

This quadratic optimization problem can then be solved using classical algorithms such as interior point or active constraint methods. Similarly to classical SVM, $\mathbf{w}$ shows that the decision function depends only on the data points. For this specific case, it only depends on the difference between negative and positive pairings:

$$f(\mathbf{x}) = \sum_{i,j=1}^{n_+,n_-} \alpha_{i,j}^* \langle \mathbf{x}_i^+ - \mathbf{x}_j^-, \mathbf{x} \rangle + b,$$

where $\alpha_{i,j}^*$ are the solutions to the dual problem.

Recently, there has been work in creating online algorithms for maximizing the performance of a classifier by maximizing the AUC score. The first of which, Online AUC Maximization (OAM) [118], attempts to overcome the issue of the quadratic nature of computing the AUC score. The algorithm begins with the following objective function using

Algorithm 4 Online AUC Maximization (OAM)

Input: the penalty parameter C, the maximum buffer size N_+ and N_-
Initialize: $\mathbf{w}_1 = \mathbf{0}$, $B_+^1 = B_-^1 = \emptyset$, $N_+^1 = N_-^1 = 0$
for $t = 1, 2, ..., T$ **do**
  Receive training instance $(\mathbf{x}_t, y_t)$
  if $y_t = +1$ **then**
    $N_+^{t+1} = N_+^t + 1$, $N_-^{t+1} = N_-^t$, $B_-^{t+1} = B_-^t$
    $C_t = C \max(1, N_-^t, N_-)$
    $B_+^{t+1} =$**UpdateBuffer**$(B_+^t, \mathbf{x}_t, N_+, N_+^{t+1})$
    $\mathbf{w}_{t+1} =$**UpdateClassifier**$(\mathbf{w}_t, y_t, C_t, B_-^{t+1}$
  else
    $N_-^{t+1} = N_-^t + 1$, $N + -^{t+1} = N_+^t$, $B_+^{t+1} = B_+^t$
    $C_t = C \max(1, N_+^t, N_+)$
    $B_-^{t+1} =$**UpdateBuffer**$(B_-^t, \mathbf{x}_t, N_-, N_-^{t+1})$
    $\mathbf{w}_{t+1} =$**UpdateClassifier**$(\mathbf{w}_t, y_t, C_t, B_+^{t+1}$
  end if
end for

hinge loss:

$$L(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n_+}\sum_{j=1}^{n_-} \max\{0, 1 - \mathbf{w}^T(x_i^+ - x_j^-)\}. \tag{2.5}$$

To overcome this challenge, the authors rewrite the loss function as a sum of losses for individual instances, i.e. $L(\mathbf{w}) = \sum_{t=1}^{T} L_t(\mathbf{w})$ where:

$$L_t(\mathbf{w}) = \mathbb{I}_{y_t=1} h_+^t(\mathbf{w}) + \mathbb{I}_{y_t=-1} h_-^t(\mathbf{w}). \tag{2.6}$$

In (2.6), $h_\pm^t(\mathbf{w})$ are defined as:

$$h_+^t(\mathbf{w}) = \sum_{t'=1}^{t-1} \mathbb{I}_{y_{t'}=-1} \ell(\mathbf{w}, x_t - x_{t'}) \text{ and } h_-^t(\mathbf{w}) = \sum_{t'=1}^{t-1} \mathbb{I}_{y_{t'}=+1} \ell(\mathbf{w}, x_{t'} - x_t). \tag{2.7}$$

From this, the main idea behind the algorithm is to use reservoir sampling [105]. Specifically, for a given training sample $(\mathbf{x}_t, y_t)$, it will be added to the buffer $B_{y_t}^t$ if $\mid B_{y_t}^t \mid < N_{y_t}$. Otherwise, with probability $\frac{N_{y_t}}{N_{y_t}^{t+1}}$, the buffer will be updated by randomly replacing one instance in $B_{y_t}^t$ with $\mathbf{x}_t$. This is summarized in Algorithm 5. Although not all samples need to be stored, a buffer of sufficient size still needs to be maintained. Different variations of the algorithm were proposed. These methods are summarized in Algorithm 6 and Algorithm 7. A drawback of the algorithm is that some samples still need to be stored. This could prove

Algorithm 5 A **Reservoir Sampling** Approach for UpdateBuffer

Input:

- B^t: the current buffer
- $\mathbf{x}_t$: a training instance
- N: the buffer size
- N_{t+1}: the number of instances received till trial t

Output: Updated buffer B^{t+1}
if $|B^t| < N$ **then**
  $B^{t+1} = B^t \cup \{\mathbf{x}_t\}$
else
  Sample Z from a Bernoulli distribution with $\Pr(Z = 1) = N/N_{t+1}$
  if $Z = 1$ **then**
    Randomly delete an instance from B^t
    $B^{t+1} = B^t \cup \{\mathbf{x}_t\}$
  end if
end if
Return B^{t+1}

challenging with high dimensional data, however, the novelty of the algorithm at the time provided a path forward to design algorithms that directly optimize the AUC score. The various versions all converge with the same rate of $\mathcal{O}(1/\sqrt{T})$.

One-Pass AUC Maximization (OPAUC) [34], again began with a similar objective function, but instead used the least square loss:

$$L(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n_+}\sum_{j=1}^{n_-}\frac{(1 - \mathbf{w}^T(\mathbf{x}_i^+ - \mathbf{x}_j^-))^2}{2n^+n^-}. \tag{2.9}$$

To address the challenge of $L(\mathbf{w})$, the authors modified the loss function, similarly like OAM, to a sum of losses over individual samples, i.e. $L(\mathbf{w}) = \sum_{t=1}^{T} L_t(\mathbf{w})$ where:

$$L_t(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{\sum_{i=1}^{t-1}\mathbb{I}[y_i \neq y_t](1 - y_t(\mathbf{x}_t - \mathbf{x}_i)^T\mathbf{w})^2}{2|\{i \in [t-1] : y_i y_t = -1\}|}. \tag{2.10}$$

Algorithm 6 A **Sequential Updating** Approach for UpdateClassifier

Input:

- $\mathbf{w}_t$: the current classifier
- $(\mathbf{x}_t, y_t)$: a training example
- B: the buffer to which the training example will be compared
- C_t: a parameter that weights the comparison between $(\mathbf{x}_t, y_t)$ and B

Output: Updated classifier $\mathbf{w}_{t+1}$
Initialize: $\mathbf{w}^1 = \mathbf{w}_t$ and $i = 1$
for $\mathbf{x} \in B$ **do**
Update classifier $\mathbf{w}^i$ by

$$\mathbf{w}^i = \underset{\mathbf{w}}{\operatorname{argmin}} |\mathbf{w} - \mathbf{w}^i|_2^2 + C_t \ell(\mathbf{w}, y_t(\mathbf{x}_t - \mathbf{w})) \tag{2.8}$$

$i = i + 1$
end for
Return $\mathbf{w}_{t+1} = \mathbf{w}^{|B|+1}$

Then taking the gradient of $L_t(\mathbf{w})$ when $y_t = 1$ gives

$$\begin{aligned}\nabla L_t(\mathbf{w}) = \lambda \mathbf{w} + \mathbf{x}_t \mathbf{x}_t^T - \mathbf{x}_t \\ + \sum_{i:y_i=-1} (\mathbf{x}_i + (\mathbf{x}_i \mathbf{x}_i^T - \mathbf{x}_i \mathbf{x}_t^T - \mathbf{x}_t \mathbf{x}_i^T)\mathbf{w})/T_t^-.\end{aligned}$$

The authors noticed that is is easy to observe that: $c_t^- = \sum_{i:y_i=-1} \mathbf{x}_i / T_t^-$ and $S_t^- = \sum_{i:y_i=-1} (\mathbf{x}_i \mathbf{x}_i^T - c_t^-[c_t^-]^T)/T_t^-$ correspond to the mean and covariance matrix of the negative class. Thus we have:

$$\nabla L_t(\mathbf{w}) = \lambda \mathbf{w} - \mathbf{x}_t + c_t^- + (\mathbf{x}_t - c_t^-)(\mathbf{x}_t - c_t^-)^T \mathbf{w} + S_t^- \mathbf{w}. \tag{2.11}$$

We can repeat this process for when $y_t = 1$. This gives that:

$$\nabla L_t(\mathbf{w}) = \lambda \mathbf{w} + \mathbf{x}_t - c_t^+ + (\mathbf{x}_t - c_t^+)(\mathbf{x}_t - c_t^+)^T \mathbf{w} + S_t^+ \mathbf{w}, \tag{2.12}$$

where $c_t^+ = \sum_{i:y_i=1} \mathbf{x}_i / T_t^+$ and $S_t^+ = \sum_{i:y_i=1} (\mathbf{x}_i \mathbf{x}_i^T - c_t^+[c_t^+]^T)/T_t^+$ correspond to the mean and covariance matrix of the positive class. We can then update the mean for both the

Algorithm 7 A **Gradient Updating** Approach for UpdateClassifier

Input:

- $\mathbf{w}_t$: the current classifier
- $(\mathbf{x}_t, y_t)$: a training example
- B: the buffer to which the training example will be compared
- C_t: a parameter that weights the comparison between $(\mathbf{x}_t, y_t)$ and B

Output: Updated classifier $\mathbf{w}_{t+1}$
Initialize: $\mathbf{w}_{t+1} = \mathbf{w}_t$ and $i = 1$
for $\mathbf{x} \in B$ **do**
  if $y_t \mathbf{w}_t \cdot (\mathbf{x}_t - \mathbf{x}) \leq 1$ **then**
    $\mathbf{w}_{t+1} = \mathbf{w}_{t+1} + C_t y_t (\mathbf{x}_t - \mathbf{x})/2$
  end if
end for
Return $\mathbf{w}_{t+1}$

negative and positive class as follows:

$$c_t^- = c_{t-1}^- + \frac{1}{T_t^-}(\mathbf{x}_t - c_{t-1}^-), \quad c_t^+ = c_{t-1}^+ + \frac{1}{T_t^+}(\mathbf{x}_t - c_{t-1}^+). \tag{2.13}$$

To update the covariance matrices, we can define $\Gamma_0^- = \Gamma_0^+ = [\mathbf{0}]_{d \times d}$ to be the covariance matrix for the positive and negative classes. Then at each iteration the covariance matrices for the positive and negative classes can be updated as follows:

$$\Gamma_t^- = \Gamma_{t-1}^- + \frac{\mathbf{x}_t \mathbf{x}_t^T - \Gamma_{t-1}^-}{T_t^-} + c_{t-1}^-[c_{t-1}]^T - c_t^-[c_t^-]^T, \tag{2.14}$$

$$\Gamma_t^+ = \Gamma_{t-1}^+ + \frac{\mathbf{x}_t \mathbf{x}_t^T - \Gamma_{t-1}^+}{T_t^+} + c_{t-1}^+[c_{t-1}]^T - c_t^+[c_t^+]^T. \tag{2.15}$$

Once the gradient has been calculated, then the solution of $\mathbf{w}_{t+1}$ is updated by $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla L(\mathbf{w}_t)$. A summary of the algorithm is in Algorithm 8. However, an issue is that the mean and covariance matrices need to be stored. In particular, the storage of the covariance matrix is of the order $\mathcal{O}(d^2)$. For high dimensional data, this could become problematic. The authors anticipated this and reasoned that the covariance matrix could be approximated by a low rank matrix, however, the results are not as strong. Just as with OAM, OPAUC has a convergence rate of $\mathcal{O}(1/\sqrt{T})$, however, the algorithm may be hindered

Algorithm 8 One-Pass AUC Maximization (OPAUC)

Input: The regularization parameter $\lambda > 0$ and step size $\{\eta_t\}_{t=1}^{T}$.
Initialize: Set $T_0^+ = T_0^- = 0$, $\mathbf{c}_0^+ = \mathbf{c}_0^- = \mathbf{0}$, $\mathbf{w}_0 = \mathbf{0}$, and $\Gamma_0^+ = \Gamma_0^- = [\mathbf{0}]_{d\times u}$ for some $u > 0$.
for $t = 1, 2, ..., T$ **do**
  Receive training instance $(\mathbf{x}_t, y_t)$
  if $y_t = +1$ **then**
    $T_t^+ = T_{t-1}^+ + 1$ and $T_t^- = T_{t-1}^-$
    $\mathbf{c}_t^+ - = \mathbf{c}_{t-1}^+ + \frac{1}{T_t^+}(\mathbf{x}_t - \mathbf{c}_{t-1}^+$ and $\mathbf{c}_t^- = \mathbf{c}_{t-1}^-$
    Update Γ_t^+ and $\Gamma_t^- = \Gamma_{t-1}^-$
    Calculate the gradient $\hat{g}_t(\mathbf{w}_{t-1})$
  else
    $T_t^- = T_{t-1}^- + 1$ and $T_t^+ = T_{t-1}^+$
    $\mathbf{c}_t^- = \mathbf{c}_{t-1}^- + \frac{1}{T_t^-}(\mathbf{x}_t - \mathbf{c}_{t-1}^-$ and $\mathbf{c}_t^+ = \mathbf{c}_{t-1}^+$
    Update Γ_t^- and $\Gamma_t^+ = \Gamma_{t-1}^+$
    Calculate the gradient $\hat{g}_t(\mathbf{w}_{t-1})$
  end if
  $\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \hat{g}_t(\mathbf{w}_{t-1})$
end for

by the storage of the covariance matrix.

Similarly, to OAM and OPAUC, a sub-gradient method for AUC maximization has also been proposed [21]. The Adaptive Online AUC Maximization (AdaOAM) algorithm is based on the same ideas and assumptions as OPAUC [34]. Taking the same loss function as in (2.10), the mean and covariance matrix of the positive and negative classes can be calculated as before. Once the gradient $\mathbf{g}_t = \nabla L_t(\mathbf{w}_t)$ has been calculated, the critical step is to move the current solution $\mathbf{w}_t$ in the opposite direction of the gradient $\mathbf{g}_t$ while maintaining the constraint $\|\mathbf{w}_{t+1}\| \leq 1/\sqrt{\lambda}$ for the projected gradient update [120]:

$$\mathbf{w}_{t+1} = \Pi_{\|\mathbf{w}\|\leq 1/\sqrt{\lambda}}(\mathbf{w}_t - \eta \mathbf{g}_t) = \arg\min_{\|\mathbf{w}\|\leq 1/\sqrt{\lambda}} \|\mathbf{w} - (\mathbf{w}_t - \eta \mathbf{g}_t)\|_2^2, \tag{2.16}$$

since $\|\mathbf{w}_*\| \leq 1/\sqrt{\lambda}$. A critical issue though is that the above assigns different features with the same learning rate. To perform feature-wise gradient updating, the authors proposed a second-order gradient optimization method, Adaptive Gradient Updating inspired by [23]. The algorithm is summarized in Algorithm 9. The notation $g_{1:t} = [\mathbf{g}_1 ... \mathbf{g}_t]$ is the matrix obtained by concatenating the gradient sequences together. Additionally, the i-th row of this matrix given by $\mathbf{g}_{1:t,i}$ which, accidentally, is also a concatenation of the i-th component

Algorithm 9 Adaptive Online AUC Maximization (AdaOAM)

Input: The regularization parameter λ, the learning rate $\{\eta\}_{t=1}^{T}$, the smooth parameter $\delta \geq 0$.
Output: Updated classifier $\mathbf{w}_t$.
Variables: $s \in \mathbb{R}^d, H \in \mathbb{R}^{d\times d}, g_{1:t,i} \in \mathbb{R}^t$ for $i \in \{1, ..., d\}$.
Initialize: $\mathbf{w}_0 = \mathbf{0}$, $\mathbf{c}_0^+ = \mathbf{c}_0^- = \mathbf{0}, \Gamma_0^+ = \Gamma_0^- = [0]_{d\times d}, T_0^+ = T_0^- = 0, g_{1:0} = []$.
for $t = 1, 2, ..., T$ **do**
  Receive an incoming instance $(\mathbf{x}_t, y_t)$
  if $y_t = 1$ **then**
    $T_t^+ = T_{t-1}^+ + 1$, $T_t^- = T_{t-1}^-$
    $\mathbf{c}_t^+ = \mathbf{c}_{t-1}^+ + \frac{1}{T_t^+}(\mathbf{x}_t - \mathbf{c}_{t-1}^+)$ and $\mathbf{c}_t^- = \mathbf{c}_{t-1}^-$
    Update Γ_t^+ and $\Gamma_t^- = \Gamma_{t-1}^-$
    Receive gradient $\mathbf{g}_t = \nabla L_t(\mathbf{w})$
    Update $g_{1:t} = [g_{1:t-1}\mathbf{g}_t]$, $s_{t,i} = \|g_{1:t,i}\|_2$
    $H_t = \delta I + \text{diag}(s_t)$
    $\hat{\hat{g}}_t = H_t^{-1}\mathbf{g}_t$
  else
    $T_t^+ = T_{t-1}^+$, $T_t^- = T_{t-1}^- + 1$
    $\mathbf{c}_t^- = \mathbf{c}_{t-1}^- + \frac{1}{T_t^-}(\mathbf{x}_t - \mathbf{c}_{t-1}^-)$ and $\mathbf{c}_t^+ = \mathbf{c}_{t-1}^+$
    Update Γ_t^- and $\Gamma_t^+ = \Gamma_{t-1}^+$
    Receive gradient $\mathbf{g}_t = \nabla L_t(\mathbf{w})$
    Update $g_{1:t} = [g_{1:t-1}\mathbf{g}_t]$, $s_{t,i} = \|g_{1:t,i}\|_2$
    $H_t = \delta I + \text{diag}(s_t)$
    $\hat{\hat{g}}_t = H_t^{-1}\mathbf{g}_t$
  end if
  $\mathbf{w}_t = \Pi_{1/\sqrt{\lambda}}^{H_t}(\mathbf{w}_{t-1} - \eta_t\hat{\hat{\mathbf{g}}}_t)$
end for

of each gradient. Lastly, the outer matrix product is denoted by $G_t = \sum_{\tau=1}^{t}\mathbf{g}_\tau\mathbf{g}_\tau^T$. Using these previous notations, the generalization of the standard adaptive gradient descent leads to the following weight update:

$$\begin{aligned}\mathbf{w}_{t+1} &= \Pi_{\|\mathbf{w}\|\leq 1/\sqrt{\lambda}}^{G_t^{1/2}}(\mathbf{w}_t - \eta G_t^{-1/2}\mathbf{g}_t) = \arg\min_{\|\mathbf{w}\|\leq 1/\sqrt{\lambda}}\|\mathbf{w} - (\mathbf{w}_t - \eta G_t^{-1/2}\mathbf{g}_t)\|_{G_t^{1/2}} \\ &= \arg\min_{\|\mathbf{w}\|\leq 1/\sqrt{\lambda}}\langle\mathbf{w} - (\mathbf{w}_t - \eta G_t^{-1/2}\mathbf{g}_t), G_t^{1/2}(\mathbf{w} - (\mathbf{w}_t - \eta G_t^{-1/2}\mathbf{g}_t))\rangle.\end{aligned}$$

The above equation is just the Mahalanobis norm for the projection of a point onto the set $\{\mathbf{w}|\|\mathbf{w}\| \leq 1/\sqrt{\lambda}\}$. A major drawback of this algorithm is that a large amount of time will

be needed to calculate the root and inverse root of the outer matrix G_t for high dimensional data. To enable the algorithm to handle high dimensional data, a diagonal *proxy* of G_t is used to make the update:

$$\mathbf{w}_{t+1} = \Pi_{\|\mathbf{w}\| \leq 1/\sqrt{\lambda}}(\mathbf{w}_t - \eta \text{diag}(G_t)^{-1/2}\mathbf{g}_t). \tag{2.17}$$

This will enable the root and inverse root of $\text{diag}(G_t)$ to be computed in linear time. However, this results in one last issue to consider that $\text{diag}(G_t)$ might not be invertible. To handle this issue, take $H_t = \delta I + \text{diag}(G_t)^{1/2}$, where $\delta > 0$ is very small. This will ensure that the diagonal matrix is invertible and the algorithm is robust. So given H_t, the update of the feature-wise adaptive update then becomes:

$$\mathbf{w}_{t+1} = \Pi^{H-t}_{\|\mathbf{w}\| \leq 1/\sqrt{\lambda}}(\mathbf{w}_t - \eta H_t \mathbf{g}_t) \tag{2.18}$$

AdaOAM should have a lower regret bound than non-adaptive algorithms due to its using the geometry of the data space. The gradient terms should be smaller than $\sqrt{T}$. If the feature space is dense, then the convergence rate will be $\mathcal{O}(1/\sqrt{T})$ just as in OPAUC and OAM.

Recently, the work done by [113] considers the least square loss function and takes an entirely different approach. The authors assume that the training data $\mathbf{z} = \{(\mathbf{x}_i, y_i), i = 1, ..., n\}$ is *i.i.d.*, and drawn from an unknown distribution ρ on $Z = X \times Y$. Recall that for a given scoring function $f : X \to \mathbb{R}$, the area under the ROC curve (AUC) is the probability of a positive sample ranking higher than a negative sample [10]. Therefore, the AUC is:

$$\text{AUC}(f) = \Pr(f(\mathbf{x}) > f(\mathbf{x}')|y = +1, y' = -1), \tag{2.19}$$

where $(\mathbf{x}, y)$ and $(\mathbf{x}', y')$ are independently drawn from ρ. As previously discussed, we want the classifier with the ROC curve that dominates all others, i.e., has the greatest AUC score. So the target of AUC maximization is to find the optimal decision function f:

$$\begin{aligned} \arg\max_f \text{AUC}(f) &= \arg\min_f \Pr(f(\mathbf{x}) < f(\mathbf{x}')|y = 1, y' = -1) \\ &= \arg\min_f \mathbb{E}\Big[\mathbb{I}_{[f(\mathbf{x}')-f(\mathbf{x})>0]}\big|y = 1, y' = -1\Big], \end{aligned} \tag{2.20}$$

where $\mathbb{I}(\cdot)$ is the indicator function. Define $p = \Pr(y = 1)$ and recall that for any random variable $\xi(z)$, its conditional expectation is defined by $\mathbb{E}[\xi(z)|y = 1] = \frac{1}{p}\int\int \xi(z)\mathbb{I}_{y=1}d\rho(z)$. Since $\mathbb{I}(\cdot)$ is not continuous, it is often replaced by a convex surrogates. The authors used the ℓ_2 loss, as it has been shown to be statistically consistent with AUC while the hinge loss is not [35]. AUC maximization can be formulated by

$$\begin{aligned} & \operatorname*{argmin}_{\mathbf{w}} \mathbb{E}\Big[(1 - \mathbf{w}^\top(\mathbf{x} - \mathbf{x}'))^2 | y = 1, y' = -1\Big] + \frac{\lambda}{2}\|\mathbf{w}\|^2 \\ = & \operatorname*{argmin}_{\mathbf{w}} \frac{1}{p(1-p)} \iint_{\mathcal{Z}\times\mathcal{Z}} (1 - \mathbf{w}^\top(\mathbf{x} - \mathbf{x}'))^2 \mathbb{I}_{[y=1]}\mathbb{I}_{[y'=-1]} d\rho(z)d\rho(z'). \end{aligned} \tag{2.21}$$

This objective function can be exploited to reduce this problem from a double integral to a single integral. Equation (3.2) can be reformulated as a stochastic saddle point problem (SPP) [70].

Definition 2.6. *A stochastic SPP is generally in the form of*

$$\min_{u\in\Omega_1}\max_{\alpha\in\Omega_2}\big\{f(u,\alpha) := \mathbb{E}[F(u,\alpha,\xi)]\big\}, \tag{2.22}$$

where $\Omega_1 \subseteq \mathbb{R}^d$ and $\Omega_2 \subseteq \mathbb{R}^m$ are nonempty closed convex sets, ξ is a random vector with non-empty measurable set $\Xi \subseteq \mathbb{R}^p$, and $F : \Omega_1 \times \Omega_2 \times \Xi \to \mathbb{R}$. Here $\mathbb{E}[F(u,\alpha,\xi)] = \int_\Xi F(u,\alpha,\xi)d\Pr(\xi)$, and function $f(u,\alpha)$ is convex in $u \in \Omega_1$ and concave in $\alpha \in \Omega_2$. In general, u and α are referred to as the primal variable and the dual variable, respectively.

In order to state the main result, we first define $F : \mathbb{R}^d \times \mathbb{R}^3 \times \mathcal{Z} \to \mathbb{R}$, for any $\mathbf{w} \in \mathbb{R}^d$, $a, b, \alpha \in \mathbb{R}$ and $z = (\mathbf{x}, y) \in \mathcal{Z}$, by

$$\begin{aligned} F(\mathbf{w}, a, b, \alpha; z) = {} & (1-p)(\mathbf{w}^\top\mathbf{x} - a)^2\mathbb{I}_{[y=1]} + p(\mathbf{w}^\top\mathbf{x} - b)^2\mathbb{I}_{[y=-1]} \\ & + 2(1+\alpha)(p\mathbf{w}^\top\mathbf{x}\mathbb{I}_{[y=-1]} - (1-p)\mathbf{w}^\top x\mathbb{I}_{[y=1]}) - p(1-p)\alpha^2. \end{aligned} \tag{2.23}$$

Theorem 2.3 (Theorem 1 in [113])**.** *The AUC optimization* (3.3) *is equivalent to*

$$\min_{\substack{\mathbf{w}\in\mathbb{R}^d \\ (a,b)\in\mathbb{R}^2}} \max_{\alpha\in\mathbb{R}} \Big\{ f(\mathbf{w}, a, b, \alpha) := \int_{\mathcal{Z}} F(\mathbf{w}, a, b, \alpha; z)d\rho(z) \Big\}. \tag{2.24}$$

The authors applied projected gradient descent to solve the above formulation as in [70].

Algorithm 10 Stochastic Online AUC Maximization (SOLAM)

Input: Step size $\{\gamma_t\}_{t=1}^T$.
Initialize: $v_1 \in \Omega_1$, $\alpha_1 \in \Omega_2$, and let $\hat{p}_0 = 0$, $\bar{v}_0 = 0$, $\bar{\alpha}_0 = 0$, $\bar{\gamma}_0 = 0$.
for $t = 1, 2, ..., T$ **do**
  Receive training instance $(\mathbf{x}_t, y_t)$ and compute $\hat{p}_t = \frac{(t-1)\hat{p}_{t-1} + \mathbb{I}_{[y_t=1]}}{t}$
  Update $v_{t+1} = P_{\Omega_1}(v_t - \gamma_t \partial_v \hat{F}_t(v_t, \alpha_t, z_t))$
  Update $\alpha_{t+1} = P_{\Omega_2}(\alpha_t - \gamma_t \partial_\alpha \hat{F}_t(v_t, \alpha_t, z_t))$
  Update $\bar{\gamma}_t = \bar{\gamma}_{t-1} + \gamma_t$
  Update $\bar{v}_t = \frac{1}{\bar{\gamma}}(\bar{\gamma}_{t-1}\bar{v}_{t-1} + \gamma_t v_t)$, and $\bar{\alpha}_t = \frac{1}{\bar{\gamma}_t}(\bar{\gamma}_{t-1}\bar{\alpha}_{t-1} + \gamma_t \alpha_t)$
end for

Since the value of p is generally unknown, an *unbiased stochastic estimator* of the true gradient to perform was used instead:

$$\begin{aligned}\hat{F}_t(v, \alpha, z) = (1 - \hat{p}_t)(\mathbf{w}^\top \mathbf{x} - a)^2 \mathbb{I}_{[y=1]} + \hat{p}_t(\mathbf{w}^\top \mathbf{x} - b)^2 \mathbb{I}_{[y=-1]} \\ + 2(1 + \alpha)(\hat{p}_t \mathbf{w}^\top \mathbf{x} \mathbb{I}_{[y=-1]} - (1 - \hat{p}_t)\mathbf{w}^\top \mathbf{x} \mathbb{I}_{[y=1]}) - \hat{p}_t(1 - \hat{p}_t)\alpha^2. \end{aligned} \tag{2.25}$$

where $\hat{p}_t = \frac{\sum_{i=1}^t \mathbb{I}_{[y_i=1]}}{t}$ at iteration t and $v = (\mathbf{w}, a, b)$. For each iteration t, the method uses the stochastic estimator

$$\hat{G}_t(v, \alpha, z) = (\partial_v \hat{F}_t(v, \alpha, z), -\partial_\alpha \hat{F}_t(v, \alpha, z)) \tag{2.26}$$

to replace the unbiased, but practically inaccessible, stochastic estimator $G(v, \alpha, z)$. Assume $\kappa = \sup_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x}\| < \infty$, and recall that $\|\mathbf{w}\| \leq R$. It is important to note that $(\mathbf{w}, a, b)$ and α can be restricted to the following bounded domains:

$$\begin{aligned}\Omega_1 &= \big\{(\mathbf{w}, a, b) \in \mathbb{R}^{d+2} : \|\mathbf{w}\| \leq R, |a| \leq R\kappa, |b| \leq R\kappa\big\} \\ \Omega_2 &= \big\{\alpha \in \mathbb{R} : |\alpha| \leq 2R\kappa\big\}.\end{aligned}$$

In this case, the projection steps (e.g. steps 4 and 5) in SOLAM can be easily computed. The pseudo-code of the online AUC optimization algorithm is described in Algorithm 10.

As for the convergence analysis, the authors used the duality gap to measure the performance of an approximate solution:

$$\varepsilon_f(\bar{v}_t, \bar{\alpha}_t) = \max_{\alpha \in \Omega_2} f(\bar{v}_t, \alpha) - \min_{v \in \Omega_1} f(v, \bar{\alpha}_t), \tag{2.27}$$

Algorithm 11 Fast Stochastic AUC Maximization (FSAUC)

Set: $m = \lfloor \frac{1}{2} \log_2 \frac{2n}{\log_2 n} \rfloor - 1$, $n_0 = \lfloor n/m \rfloor$, $R_0 = 2\sqrt{1+2\kappa^2}$, $G > \max((1+4\kappa)\kappa(R+1), 2\kappa(2R+1+2R\kappa), 2\kappa(4\kappa R+11R+1))$, $\beta_0 = (1+8\kappa)^2$, and $D_0 = 2\sqrt{2}\kappa R_0$
Initialize: $\hat{\mathbf{v}}_0 = 0 \in \mathbb{R}^{d+2}, \hat{\alpha}_0 = 0$
for $t = 1, 2, ..., T$ **do**
Set $\eta_k = \frac{\sqrt{\beta_{k-1}}}{G\sqrt{2n_0}} R_{k-1}$
Call PDSG to obtain

$$(\hat{\mathbf{v}}_k, \hat{\alpha}_k, \beta_k, R_k, D_k) = \text{PDSG}(\hat{\mathbf{v}}_{k-1}, \hat{\alpha}_{k-1}, \beta_{k-1}, R_{k-1}, D_{k-1}, n_0, \eta_k) \tag{2.28}$$

end for
Return $\hat{\mathbf{v}}_m$

which results in a convergence rate of $\mathcal{O}(1/\sqrt{T})$. While the convergence rate is obtained by choosing decaying step sizes, one can establish a similar result when a constant step size is appropriately chosen.

Continuing with the work based on [113] was expanded upon by [59]. The authors proposed a similar primal-dual style algorithm, but with a constant step size. The primal variables $\mathbf{w}$, a, and b and the dual variable α are projected into an intersection of their constrained domains and an ℓ_2 balled centered at the initial solution. These modifications allow the algorithm to achieve a $\mathcal{O}(1/T)$ convergence rate by way of a geometrically decreasing radius r. The method is summarized in Algorithms 11 and 12.

A completely different approach to optimizing AUC, or rather any other metric that can be computed from the contingency table, was based on the sparse approximation algorithm for structural SVMs [100, 102]. This new method can be interpreted as a generalization of

Algorithm	**Loss**	**Penalty**	**Storage**	**Iteration**	**Rate**
OAM	General	L^2	$\mathcal{O}(sd)$	$\mathcal{O}(sd)$	$\mathcal{O}(1/\sqrt{T})$
AdaOAM	General	L^2	$\mathcal{O}(td)$	$\mathcal{O}(td)$	$\mathcal{O}(1/\sqrt{T})$
OPAUC	Least-Square	L^2	$\mathcal{O}(d^2)$	$\mathcal{O}(d^2)$	$\mathcal{O}(1/\sqrt{T})$
SOLAM	Least-Square	L^2	$\mathcal{O}(d)$	$\mathcal{O}(d)$	$\mathcal{O}(1/\sqrt{T})$

Table 2.1: A comparison of existing online algorithms for AUC maximization. Note that s is the buffer size and t is the current iteration.

Algorithm 12 PDSG($\mathbf{v}_1, \alpha_1, r, D, T, \eta$)

Initialize: $\hat{A}_+ \in \mathbb{R}^{d+2}$, $\hat{A}_- \in \mathbb{R}^{d+2}$, T_+, T_-, $\hat{p} \in \mathbb{R}$ as zeros
for $t = 1, 2, ..., T$ **do**
  Receive a sample $\mathbf{z}_t = (\mathbf{x}_t, y_t)$
  Update $\hat{A}_\pm$, $\hat{T}_\pm$, $\hat{p}$ using data $\mathbf{z}_t$
  $\mathbf{v}_{t+1} = \Pi_{\Omega_1 \cap \mathbb{B}(\mathbf{v}_1, r)}(\mathbf{v}_t - \eta \partial_{\mathbf{v}} \hat{F}_t(\mathbf{v}_t, \alpha_t, \mathbf{z_t}))$
  $\alpha_{t+1} = \Pi_{\Omega_1 \cap \mathbb{B}(\alpha_1, r)}(\alpha_t + \eta \partial_{\alpha} \hat{F}_t(\mathbf{v}_t, \alpha_t, \mathbf{z_t}))$
end for
Compute $\bar{\mathbf{v}}_T = \frac{\sum_{t=1}^{T} \mathbf{v}_t}{T}$ and $\hat{\alpha}_T = \left(\frac{\hat{A}_-}{T_-} - \frac{\hat{A}_+}{T_+}\right)^T \bar{\mathbf{v}}_T$
Let $r = r/2$
Update β, D according to lemma 1
Return $(\bar{\mathbf{v}}_T, \hat{\alpha}, \beta, r, D)$

Algorithm 13 Algorithm for Solving Quadratic Problem of Multivariate $\text{SVM}^{\Delta}_{multi}$

Input: $\bar{\mathbf{x}} = (\mathbf{x}_1, ..., \mathbf{x}_n)$, $\bar{y} = (y_1, ..., y_n)$, C, ϵ, $\bar{\mathcal{Y}}$
$\mathcal{C} \leftarrow \emptyset$
while $\mathcal{C}$ is changing **do**
  $\bar{y}' \leftarrow \text{argmax}_{\bar{y} \in \bar{\mathcal{Y}}} \{\Delta(\bar{y}, \bar{y}') + \mathbf{w}^T \Psi(\bar{\mathbf{x}}, \bar{y}')\}$
  $\xi \leftarrow \max_{\bar{y}' \in \mathcal{C}} \left\{\max\{0, \Delta(\bar{y}', \bar{y}) - \mathbf{w}^T[\Psi(\bar{\mathbf{w}}, \bar{y}) - \Psi(\bar{\mathbf{x}}), \bar{y}')]\}\right\}$
  if $\Delta(\bar{y}, \bar{y}') - \mathbf{w}^T[\Psi(\bar{\mathbf{w}}, \bar{y}) - \Psi(\bar{\mathbf{x}}, \bar{y}')] > \xi + \epsilon$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\bar{y}'\}$
    $\mathbf{w} \leftarrow$ optimize $\text{SVM}^{\Delta}_{multi}$ objective over $\mathcal{C}$
  end if
end while
Return $\mathbf{w}$

SVMs [41]. To describe this method, we first begin with the following optimization problem:

$$\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{i=1}^{n} \xi_i$$
$$s.t. : \forall_{i=1}^{n} : y_i[\mathbf{w} \cdot \mathbf{x}_i] \geq 1 - \xi_i. \tag{2.29}$$

The symbols in the above formulation have standard meanings. Essentially, if a training sample is not on the correct side of the hyperplane, the slack variable ξ_i will be greater than 1 giving that $\sum_{i=1}^{n} \xi_i$ is an upper bound on the number of errors.

To use this problem to allow SVMs to optimize different non-linear classifiers, the main idea is to consider the learning problem as a multivariate prediction problem. Many algorithms define the function f to be learned to only consider a single feature vector $\mathbf{x}$

Algorithm 14 Algorithm for computing argmax with loss functions that can be computed with the contingency table

Input: $\bar{\mathbf{x}} = (\mathbf{x}_1, ..., \mathbf{x}_n)$, $\bar{y} = (y_1, ..., y_n)$, $\bar{\mathcal{Y}}$
$(i^p_1, ..., p^p_{\#pos}) \leftarrow$ sort $\{i : y_i = 1\}$ by $\mathbf{w}^T\mathbf{x}_i$
$(i^n_1, ..., p^n_{\#neg}) \leftarrow$ sort $\{i : y_i = -1\}$ by $\mathbf{w}^T\mathbf{x}_i$
for $a \in [0, ..., \#pos]$ **do**
  $c \leftarrow \#pos - a$
  Set $y'_{i^p_1}, ..., y'_{i^p_a}$ to 1 AND set $y'_{i^p_{a+1}}, ..., y'_{i^p_{\#pos}}$ to -1
  for $d \in [0, ..., \#neg]$ **do**
    $b \leftarrow \#neg - d$
    Set $y'_{i^n_1}, ..., y'_{i^n_b}$ to 1 AND set $y'_{i^n_{b+1}}, ..., y'_{i^n_{\#neg}}$ to -1
    $v \leftarrow \Delta(a, b, c, d) + \mathbf{w}^T \sum_{i=1}^n y'_i\mathbf{x}_i$
    if v is the largest so far **then**
      $\bar{y}^* \leftarrow (y'_1, ..., y'_n)$
    end if
  end for
end for
Return $\bar{y}^*$

and single label $y \in \{\pm 1\}$. Instead the authors consider a function $\bar{f} : \bar{\mathcal{X}} \to \bar{\mathcal{Y}}$ where $\bar{\mathcal{X}} = \mathcal{X} \times ... \times \mathcal{X}$ and $\bar{\mathcal{Y}} = \{\pm 1\}^n$ are the set of all possible vectors. To determine the mapping, the authors consider the function of the following form:

$$\bar{h}_{\mathbf{w}}(\bar{x}) = \underset{\bar{y}' \in \bar{\mathcal{Y}}}{\operatorname{argmax}} \left\{ \mathbf{w}^T \Psi(\bar{\mathbf{x}}, \bar{y}') \right\} \tag{2.30}$$

For a fixed parameter $\mathbf{w}$ and Ψ, a function that returns a vector that describes the relationship between $\bar{x}$ and $\bar{y}'$, the function gives the tuple $\bar{y}'$ of labels that gives the highest score according to a linear function. The authors restricted their results to Ψ of the form:

$$\Psi(\bar{\mathbf{x}}, \bar{y}') = \sum_{i=1}^{n} y'_i \mathbf{x}_i. \tag{2.31}$$

In other words, the argmax is achieved when y'_i is assigned to $h(\mathbf{x}_i)$. A benefit of this formulation is that this is equivalent to the original SVM problem, however, we can reformulate

Algorithm 15 Algorithm for computing argmax with ROCArea-loss

Input: $\bar{\mathbf{x}} = (\mathbf{x}_1, ..., \mathbf{x}_n)$, $\bar{y} = (y_1, ..., y_n)$, $\bar{\mathcal{Y}}$
for $i \in \{i : y_i = 1\}$ **do**
  $s_i \leftarrow -0.25 + \mathbf{w}^T \mathbf{x}_i$
end for
for $i \in \{i : y_i = -1\}$ **do**
  $s_i \leftarrow 0.25 + \mathbf{w}^T \mathbf{x}_i$
end for
$(r_1, ..., r_n) \leftarrow$ sort $\{1, ..., n\}$ by s_i
$s_p = \#pos$, $s_n = 0$
for $\in \{1, ..., n\}$ **do**
  if $y_{r_i} > 0$ **then**
    $c_{r_i} \leftarrow (\#neg - 2s_n)$
    $s_p \leftarrow s_p - 1$
  else
    $c_{r_i} \leftarrow (-\#pos + 2s_p)$
    $s_n \leftarrow s_n + 1$
  end if
end for
Return $(c_1, ..., c_n)$

it as below:

$$\begin{aligned} &\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ &s.t. \forall \bar{y}' \in \bar{\mathcal{Y}} / \bar{y} : \mathbf{w}^T[\Psi(\bar{\mathbf{x}}, \bar{y}) - \Psi(\bar{\mathbf{x}}, \bar{y}')] \geq \Delta(\bar{y}', \bar{y}) - \xi, \end{aligned} \tag{2.32}$$

where $\Delta(\bar{y}', \bar{y})$ is a sample based loss function.

Solving the above problem may seem complicated due to the exponential size of $\bar{\mathcal{Y}}$, however, by adapting the the sparse approximation algorithm, this problem can be solved in polynomial time for many types of loss function $\Delta(a, b, c, d)$ [102]. Therefore, the solution of

$$\underset{\bar{y}' \in \bar{\mathcal{Y}}}{\operatorname{argmax}} \{\Delta(a, b, c, d) + \mathbf{w}^T \Psi(\bar{\mathbf{x}}, \bar{y}')\} \tag{2.33}$$

can be solved by Algorithm 14 in polynomial time for any loss function that can be computed in polynomial time. More importantly, is the following result for the relationship between the SVM formulations.

Theorem 2.4 (Theorem 1 in [41]). *At the solution* $\mathbf{w}^*$*,* ξ^* *of the* SVM^{Δ}_{multi} *optimization*

Algorithm 16 Mini-Batch AUC Optimization (MBA)

Require: B, T, λ_1, λ_2
Input: X^+, X^-
Output: $\mathbf{w}^*$
$\mu_S = \mathbf{0}$ and $\mathbf{\Sigma}_S = \mathbf{0}$
for $t = 1, \ldots, T$ **do**
  Construct index set S_t^+ of size B sampling positive examples uniformly with replacement.
  Construct index set S_t^- of size B sampling negative examples uniformly with replacement.
  Construct $S_t(i_=(S_t^+(i), S_t^-(i))$, for $i = 1, \ldots, B$.
  $\mu_S \leftarrow \mu_S + \frac{1}{BT}\sum_{(i,j)\in S_t}(\mathbf{x}_i^+ - \mathbf{x}_j^-)$
  $\mathbf{\Sigma}_S \leftarrow \mathbf{\Sigma}_S + \frac{1}{BT}\sum_{(i,j)\in S_t}(\mathbf{x}_i^+ - \mathbf{x}_j^-)(\mathbf{x}_i^+ - \mathbf{x}_j^-)^T$
end for
$\mathbf{w}^* = \text{argmin}_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{\Sigma}_S\mathbf{w}^T - \mathbf{w}^T\mu_S + \lambda_1\|\mathbf{w}\|_1 + \frac{\lambda_2}{2}\|\mathbf{w}\|_2^2$

problem on the training data $\bar{\mathbf{x}}$ *with labels* $\bar{y}$*, the value of* ξ^* *is an upper bound on the training loss* $\Delta(\bar{h}_{\mathbf{w}^*}(\bar{\mathbf{x}}, \bar{y})$.

The above theorem shows that Problem 2.32 is similar to Problem 2.29. The method outlined in Algorithm 13 is based on a sparse approximation algorithm to give the solution to Problem 2.32. The importance of solving Problem 2.32 is that it can be used to design algorithms to optimize performance metrics that are obtained from the confusion matrix. The ideas here can be modified to optimize the AUC score as well. The critical observation is to use the idea of *swapped pairs*:

$$\text{Swapped Pairs} =\mid \{(i,j) : (y_i > y_j) \text{ and } (\mathbf{w}^T\mathbf{x}_i < \mathbf{w}^T\mathbf{x}_j)\} \mid .$$

AUC can be defined as:

$$AUC = 1 - \frac{\text{Swapped Pairs}}{n_+ n_-}.$$

With this definition of AUC, the AUC score can be optimized in $\mathcal{O}(n \log n)$ time.

The method proposed in [36] is based on a convex relaxation of the AUC function, but instead of using stochastic gradients, the algorithm, called Mini-Batch AUC Optimization (MBA), uses the first and second order U-statistics of pairwise distances. The authors use a

slightly similar method to [113] where they use the definition of AUC as an expectation:

$$\begin{aligned} R(f) &= \mathbb{E}[(1 - \mathbf{w}^T(\mathbf{x}_i^+ - \mathbf{x}_j^-))^2] \\ &= 1 - 2\mathbf{w}^T\mathbb{E}[(\mathbf{x}_i^+ - \mathbf{x}_j^-)] + \mathbf{w}^T\mathbb{E}[(\mathbf{x}_i^+ - \mathbf{x}_j^-)(\mathbf{x}_i^+ - \mathbf{x}_j^-)^T]\mathbf{w} \\ &= 1 - 2\mathbf{w}^T\mu + \mathbf{w}^T\mathbf{\Sigma}\mathbf{w}, \end{aligned}$$

where $\mathbf{x}_{ij} = (\mathbf{x}_i^+ - \mathbf{x}_j^-)$, $\mu = \mathbb{E}[\mathbf{x}_{ij}]$, and $\mathbf{\Sigma} = \mathbb{E}[\mathbf{x}_{ij}\mathbf{x}_{ij}^T]$. Note that μ and $\mathbf{\Sigma}$ are the mean and variance of $\mathbf{x}_{ij}$. The challenge is then to solve the following minimization problem:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}\mathbf{w}^T\mathbf{\Sigma}\mathbf{w} - 2\mathbf{w}^T\mu.$$

Determining μ and $\mathbf{\Sigma}$ is difficult since the number of pairs grows quadratically as the number of samples increases. The authors suggest to overcome these issues by sampling at each round t. Selecting B positive samples and B negative samples with replacement to create sets S_t^+ and S_t^-, respectively, with $S_t = (S_t^+, S_t^-)$. Therefore, the first and second statistics can be computed as follows:

$$\begin{aligned} \mu_t &= \frac{1}{B}\sum_{(i,j)\in S_t}(\mathbf{x}_i^+ - \mathbf{x}_j^-) \\ \mathbf{\Sigma}_t &= \frac{1}{B}\sum_{(i,j)\in S_t}(\mathbf{x}_i^+ - \mathbf{x}_j^-)(\mathbf{x}_i^+ - \mathbf{x}_j^-)^T. \end{aligned}$$

Letting $S = BT$ be number of pairs sampled by the algorithm and using the notation of $S_{1:T}$ for all pairs of samples, we have the following approximations:

$$\begin{aligned} \mu_S &= \frac{1}{BT}\sum_{(i,j)\in S_{1:T}}(\mathbf{x}_i^+ - \mathbf{x}_j^-) \\ \mathbf{\Sigma}_S &= \frac{1}{BT}\sum_{(i,j)\in S_{1:T}}(\mathbf{x}_i^+ - \mathbf{x}_j^-)(\mathbf{x}_i^+ - \mathbf{x}_j^-)^T. \end{aligned}$$

Using the above constructions, we can state the objective function to optimize the AUC score. Note that the authors used elastic net to avoid over fitting:

$$\mathbf{w}_S^* = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}\mathbf{w}^T\mathbf{\Sigma}_S\mathbf{w} - 2\mathbf{w}^T\mu_S + \lambda_1\|\mathbf{w}\|_1 + \frac{\lambda_2}{2}\|\mathbf{w}\|_2^2.$$

Algorithm 17 Proximal SVRG for AUC Maximization

Input: Constant step size η and update frequency m
Initialize: $\bar{\mathbf{w}}_0$
for $s = 1, 2, \ldots$ **do**
  $\bar{\mathbf{w}}_{s-1}$
  $\mu = \frac{1}{n}\sum_{i=1}^{n} G(\bar{\mathbf{w}}, z_t)$
  $\mathbf{w}_0 = \bar{\mathbf{w}}$
  for $t = 1, \ldots, m$ **do**
    Randomly pick $i_t \in \{1, \ldots, n\}$ and update weight
    $\mathbf{w} = \mathbf{w}_{t-1} - \eta(G(\mathbf{w}_{t-1}, z_{i_t}) - G(\bar{\mathbf{w}}, z_{i_t}) + \bar{\mu})$
    $\mathbf{w}_t = \text{prox}_{\eta\Omega}(\mathbf{w})$
  end for
end for
$\bar{\mathbf{w}}_s = \mathbf{w}_m$

A critical feature of this approach is that it is learning rate free as training the step size is a time consuming task. The method is outline in Algorithm 16.

Finally, another consideration of SGD is the high variance that can result from the process of the step size decaying to zero. Variance reduction methods [43] have been developed to resolve such an issue and have been applied recently to AUC optimization [18]. The work done by the authors is a direct extension of the work in chapter 4. The main idea of the method is to only update $\bar{\mathbf{w}}$ after m iterations and to calculate the full gradient given by:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} G(\bar{\mathbf{w}}, z_i). \tag{2.34}$$

This information is then used to update the next m gradients. For each step t, $i_i \in \{1, \ldots, t\}$, we compute $\mathbf{w}_t = \mathbf{w}_{t-1} = \eta\mathbf{v}_t$ where:

$$\mathbf{v}_t = G(\mathbf{w}_{t-1}, z_{i_t}) - G(\bar{\mathbf{w}}, z_{i_t}) + \bar{\mu}.$$

The method is summarized in Algorithm 17 and is shown to have a faster rate of convergence than the method in chapter 4, however, this method has a higher per iteration complexity than the method in chapter 4.

Chapter 3

Regularized AUC Maximization

This chapter is based on the work as done in [113] of reformulating AUC optimization as a saddle point problem, but with the inclusion of a regularization term. The theorems and proofs are all done in a similar spirit. The algorithm developed in this section forms a baseline for comparison with the algorithms developed in chapters 4 and 5.

3.1 Problem Formulation

Recall from Chapter 1 that we previously defined the input space to be $\mathcal{X} \subseteq \mathbb{R}^d$ and the output space to be $\mathcal{Y} = \{-1, +1\}$. The training data, $\mathbf{z} = \{(x_i, y_i),\ i = 1, \ldots, n\}$, is assumed to be an *i.i.d.* sample drawn from an unknown distribution ρ on $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Using these notations, the area under the ROC curve (AUC) for any scoring function $f : \mathcal{X} \to \mathbb{R}$ is equivalent to the probability of a positive sample ranking higher than a negative sample [10, 37],

$$\mathrm{AUC}(f) = \Pr(f(\mathbf{x}) \geq f(\mathbf{x}')|y = +1, y' = -1), \tag{3.1}$$

where $(\mathbf{x}, y)$ and $(\mathbf{x}', y')$ are independently chosen from ρ. The scoring function f is restricted to the family of linear functions, *i.e.*, $f(x) = \mathbf{w}^\top \mathbf{x}$. Therefore, optimizing the AUC score means to find the optimal decision function f:

$$\begin{aligned} \arg\max_f \mathrm{AUC}(f) &= \arg\min_f \Pr(f(\mathbf{x}) < f(\mathbf{x}')|y = 1, y' = -1) \\ &= \arg\min_f \mathbb{E}\Big[\mathbb{I}_{[f(\mathbf{x}')-f(\mathbf{x})>0]}\big|y = 1, y' = -1\Big], \end{aligned} \tag{3.2}$$

where $\mathbb{I}(\cdot)$ is the indicator function. From the above formulation (3.2), the indicator function is not continuous nor convex. It is common practice to replace it by a convex surrogate loss function. In this work, we will use the ℓ_2 loss which is $(1-(f(\mathbf{x})-f(\mathbf{x}')))^2$. The logic behind the choice of the ℓ_2 loss is that it has been shown to be statistically consistent with AUC

while the hinge loss is not [35].

A modification that can be made to the work in [113] is the inclusion of a regularization term $\Omega(\mathbf{w})$ with parameter λ. Recall that the conditional expectation for a random variable $\xi(z)$ is defined by

$$\mathbb{E}[\xi(z)|y=1] = \frac{1}{p}\iint \xi(z)\mathbb{I}_{y=1}d\rho(z),$$

where $p = \Pr(y=1)$. By applying the previous definition to 3.2, and including a regularization term to avoid overfitting, AUC maximization can then be formulated as

$$\begin{aligned}
&\operatorname*{argmin}_{\mathbf{w}} \mathbb{E}\Big[(1-\mathbf{w}^\top(\mathbf{x}-\mathbf{x}'))^2|y=1,y'=-1\Big] + \frac{\lambda}{2}\|\mathbf{w}\|^2 \\
&= \operatorname*{argmin}_{\mathbf{w}} \frac{1}{p(1-p)}\iint_{\mathcal{Z}\times\mathcal{Z}} (1-\mathbf{w}^\top(\mathbf{x}-\mathbf{x}'))^2\mathbb{I}_{[y=1]}\mathbb{I}_{[y'=-1]}d\rho(z)d\rho(z') + \frac{\lambda}{2}\|\mathbf{w}\|^2. \qquad (3.3)
\end{aligned}$$

It is evident from the formulation in 3.3 that optimizing AUC presents a significant challenge since the metric is based on two samples of opposing classes. Therefore, it is of interest to overcome the quadratic nature of the objective function.

3.2 AUC Optimization as a Saddle Point Problem

A novel approach to overcome the quadratic objective function of AUC optimization is to reformulate AUC optimization as a stochastic saddle point problem (SPP) [70]. First, we give a definition of a stochastic SPP.

Definition 3.1. *A stochastic SPP is generally in the form of*

$$\min_{u\in\Omega_1}\max_{\alpha\in\Omega_2}\big\{f(u,\alpha) := \mathbb{E}[F(u,\alpha,\xi)]\big\}, \qquad (3.4)$$

where $\Omega_1 \subseteq \mathbb{R}^d$ and $\Omega_2 \subseteq \mathbb{R}^m$ are nonempty closed convex sets, ξ is a random vector with non-empty measurable set $\Xi \subseteq \mathbb{R}^p$, and $F : \Omega_1 \times \Omega_2 \times \Xi \to \mathbb{R}$. Here $\mathbb{E}[F(u,\alpha,\xi)] = \int_\Xi F(u,\alpha,\xi)d\Pr(\xi)$, and function $f(u,\alpha)$ is convex in $u \in \Omega_1$ and concave in $\alpha \in \Omega_2$.

The variables u and α are referred to as the primal and the dual variables, respectively. The following work, based on [113], gives a modified version of the result that includes a ℓ_2 regularization term. Before we state the theorem, we first need to define $F : \mathbb{R}^d\times\mathbb{R}^3\times\mathcal{Z} \to \mathbb{R}$,

for any $\mathbf{w} \in \mathbb{R}^d$, $a, b, \alpha \in \mathbb{R}$ and $z = (\mathbf{x}, y) \in \mathcal{Z}$, by

$$F(\mathbf{w}, a, b, \alpha; z) = (1-p)(\mathbf{w}^\top \mathbf{x} - a)^2 \mathbb{I}_{[y=1]} + p(\mathbf{w}^\top \mathbf{x} - b)^2 \mathbb{I}_{[y=-1]} + 2(1+\alpha)(p\mathbf{w}^\top x \mathbb{I}_{[y=-1]} - (1-p)\mathbf{w}^\top \mathbf{x} \mathbb{I}_{[y=1]}) - p(1-p)\alpha^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2. \tag{3.5}$$

Equation (3.5) is similar as in the previous work [113] where the inclusion of a regularization term is the only difference. The main result still holds in a similar manner.

Theorem 3.1. *AUC optimization* (3.3) *is equivalent to*

$$\min_{\substack{\mathbf{w}\in\mathbb{R}^d \\ (a,b)\in\mathbb{R}^2}} \max_{\alpha\in\mathbb{R}} \Big\{ f(\mathbf{w}, a, b, \alpha) := \int_{\mathcal{Z}} F(\mathbf{w}, a, b, \alpha; z) d\rho(z) \Big\}. \tag{3.6}$$

Proof. It is enough to prove the claim that the objective function of (3.3) is equivalent to

$$1 + \min_{(a,b)\in\mathbb{R}^2} \max_{\alpha\in\mathbb{R}} \int_{\mathcal{Z}} \frac{F(\mathbf{w}, a, b, \alpha; z)}{p(1-p)} d\rho(z). \tag{3.7}$$

As from our initial assumptions, recall that $z = (\mathbf{x}, y)$ and $z' = (\mathbf{x}', y')$ are samples independently drawn from an unknown distribution ρ. It is critical to note that the double integral comes from the multiplication of two single integrals:

$$\begin{aligned}
&\mathbb{E}[(1 - \mathbf{w}^\top(\mathbf{x} - \mathbf{x}'))^2 | y = 1, y' = -1] + \frac{1}{2}\|\mathbf{w}\|^2 \\
&= 1 - 2\mathbb{E}[\mathbf{w}^\top \mathbf{x} | y = 1] + 2\mathbb{E}[\mathbf{w}^\top \mathbf{x}' | y' = -1] + (\mathbb{E}[\mathbf{w}^\top \mathbf{x} | y = 1] - \mathbb{E}[\mathbf{w}^\top \mathbf{x}' | y' = -1])^2 \\
&+ \mathrm{Var}[\mathbf{w}^\top \mathbf{x} | y = 1]) + \mathrm{Var}[\mathbf{w}^\top \mathbf{x}' | y' = -1]) + \frac{1}{2}\|\mathbf{w}\|^2.
\end{aligned} \tag{3.8}$$

Observe that

$$\begin{aligned}
\mathrm{Var}[\mathbf{w}^\top \mathbf{x} | y = 1] &= \mathbb{E}[(\mathbf{w}^\top \mathbf{x})^2 | y = 1] - \big(\mathbb{E}[\mathbf{w}^\top \mathbf{x} | y = 1]\big)^2 \\
&= \frac{1}{p}\int_{\mathcal{Z}} (\mathbf{w}^\top \mathbf{x})^2 \mathbb{I}_{[y=1]} d\rho(z) - \left(\frac{1}{p}\int_{\mathcal{Z}} \mathbf{w}^\top \mathbf{x} \mathbb{I}_{[y=1]} d\rho(z)\right)^2 \\
&= \min_{a\in\mathbb{R}} \frac{1}{p}\int_{\mathcal{Z}} (\mathbf{w}^\top \mathbf{x} - a)^2 \mathbb{I}_{[y=1]} d\rho(z) \\
&= \min_{a\in\mathbb{R}} \mathbb{E}[(\mathbf{w}^\top \mathbf{x} - a)^2 | y = 1],
\end{aligned}$$

where the minimization is achieved by

$$a = \mathbb{E}[\mathbf{w}^\top \mathbf{x} | y = 1]. \tag{3.9}$$

Similarly, the same holds true for b:

$$\mathrm{Var}[\mathbf{w}^\top \mathbf{x}' | y' = -1] = \min_b \mathbb{E}[(\mathbf{w}^\top \mathbf{x}' - b)^2 | y' = -1], \tag{3.10}$$

where the minimization is obtained when

$$b = \mathbb{E}[\mathbf{w}^T \mathbf{x} | y = -1]. \tag{3.11}$$

Recognize the fact that

$$\begin{aligned}(\mathbb{E}[\mathbf{w}^\top \mathbf{x} | y = 1] - \mathbb{E}[\mathbf{w}^\top \mathbf{x}' | y' = -1])^2 \\ = \max_\alpha \{-\alpha^2 + 2\alpha(\mathbb{E}[\mathbf{w}^\top \mathbf{x}' | y' = -1] - \mathbb{E}[\mathbf{w}^\top \mathbf{x} | y = 1])\}.\end{aligned} \tag{3.12}$$

The minimization for α can then be written as:

$$\alpha = \mathbb{E}[\mathbf{w}^T \mathbf{x} | y = -1] - \mathbb{E}[\mathbf{w}^T \mathbf{x} | y = 1] = b - a. \tag{3.13}$$

Combining the above equalities into (3.8) implies that

$$\begin{aligned}&\mathbb{E}\Big[(1 - \mathbf{w}^\top(\mathbf{x} - \mathbf{x}'))^2 | y = 1, y' = -1\Big] + \frac{\lambda}{2}\|\mathbf{w}\|^2 \\ &\quad = 1 + \mathbb{E}[(\mathbf{w}^\top x - a)^2 | y = 1] + \mathbb{E}[(\mathbf{w}^\top x - b)^2 | y = -1] \\ &\quad + 2(1 + \alpha)\left(\mathbb{E}[\mathbf{w}^\top x | y = -1] - \mathbb{E}[\mathbf{w}^\top x | y = 1]\right) - \alpha^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \\ &\quad = 1 + \min_{(a,b) \in \mathbb{R}^2} \max_{\alpha \in \mathbb{R}} \frac{\int_{\mathcal{Z}} F(\mathbf{w}, a, b; z) d\rho(z)}{p(1-p)}.\end{aligned}$$

This completes the proof of the claim and the theorem. □

An important consideration is whether or not the objective function remains convex. Replacing the indicator function by the least square loss made the objective convex, however, does this hold true for the stochastic SPP with respect to the primal or dual variables? The

answer to this question is proved in the following result.

Proposition 3.2.1. *Function $f(\mathbf{w}, a, b, \alpha)$ is convex in $(\mathbf{w}, a, b) \in \mathbb{R}^{d+2}$ and concave in $\alpha \in \mathbb{R}$.*

Proof. For fixed $(\mathbf{w}, a, b)$, the concavity of $f(\mathbf{w}, a, b, \alpha)$ in α is obvious, since it is a quadratic function of α. For fixed α and $z = (\mathbf{x}, y)$, the Hessian of $F(\cdot, \cdot, \cdot, \alpha; z)$ is given by

$$\begin{pmatrix} 2(1-p)\mathbf{x}\mathbf{x}^\top + \lambda & -2(1-p)\mathbf{x} & 0 \\ -2(1-p)\mathbf{x}^\top & 2(1-p) & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbb{I}_{[y=1]} + \begin{pmatrix} 2p\mathbf{x}\mathbf{x}^\top + \lambda & 0 & -2p\mathbf{x} \\ 0 & 0 & 0 \\ -2p\mathbf{x}^\top & 0 & 2p \end{pmatrix} \mathbb{I}_{[y=-1]}.$$

It is easy to see that the above matrices are positive semi-definite. Therefore, $F(\mathbf{w}, a, b, \alpha; z)$ is jointly convex with respect to $(\mathbf{w}, a, b)$ for any fixed α and z. Notice that the function $f(\mathbf{w}, a, b, \alpha) = \iint_{\mathcal{Z}} F(\mathbf{w}, a, b, \alpha; z) d\rho(z)$. Hence, $f(\mathbf{w}, a, b, \alpha)$ is convex with respect to $(\mathbf{w}, a, b)$ for any fixed α. This completes the proof of the proposition. □

Lastly, we show that the solution to (3.6) is the same optimizer for the original AUC optimization problem.

Proposition 3.2.2. *For any saddle point $(w^*, a^*, b^*, \alpha^*)$ of the SPP formulation (3.6), w^* is a minimizer of the original AUC optimization problem (3.3).*

Proof. Define $\bar{f}(\mathbf{w}, a, b, \alpha) = 1 + \frac{\int_{\mathcal{Z}} F(\mathbf{w}, a, b, \alpha; z) d\rho(z)}{p(1-p)} + \frac{\lambda}{2}\|\mathbf{w}\|^2$ and let $(\mathbf{w}^*, a^*, b^*, \alpha^*)$ be a saddle point of the problem

$$\min_{\substack{\mathbf{w} \in \mathbb{R}^d \\ (a,b) \in \mathbb{R}^2}} \max_{\alpha \in \mathbb{R}} \bar{f}(\mathbf{w}, a, b, \alpha).$$

Since it does not matter which minimization occurs first (i.e. minimizing with respect to $\mathbf{w}$ and minimizing with respect to (a, b)), this does not affect the result. We can conclude for every fixed $\mathbf{w}$, (a^*, b^*, α^*) is a saddle point of the sub-problem

$$\min_{(a,b) \in \mathbb{R}^2} \max_{\alpha \in \mathbb{R}} \bar{f}(\mathbf{w}, a, b, \alpha).$$

Furthermore, from the proof for Theorem 3.1 we have

$$\mathbb{E}\Big[(1 - \mathbf{w}^\top(\mathbf{x} - \mathbf{x}'))^2 | y = 1, y' = -1\Big] + \frac{\lambda}{2}\|\mathbf{w}\|^2 = \min_{(a,b) \in \mathbb{R}^2} \max_{\alpha \in \mathbb{R}} \bar{f}(\mathbf{w}, a, b, \alpha). \tag{3.14}$$

Therefore,

$$\mathbb{E}\Big[(1-\mathbf{w}^\top(\mathbf{x}-\mathbf{x}'))^2|y=1,y'=-1\Big]+\frac{\lambda}{2}\|\mathbf{w}\|^2=\bar{f}(\mathbf{w},a^*,b^*,\alpha^*).$$

This implies that

$$\min_{\mathbf{w}}\mathbb{E}\Big[(1-\mathbf{w}^\top(\mathbf{x}-\mathbf{x}'))^2|y=1,y'=-1\Big]+\frac{\lambda}{2}\|\mathbf{w}\|^2=\min_{\mathbf{w}}f(\mathbf{w},a^*,b^*,\alpha^*). \tag{3.15}$$

Since $\mathbf{w}^*$ is assumed to be the minimizer of the righthand side of (3.15), $\mathbf{w}^*$ is also a minimizer of the lefthand side of the equation.

□

3.3 Algorithm

The objective for AUC maximization as detailed in the previous section is to determine the saddle point $(\mathbf{w}^*,a^*,b^*,\alpha^*)$ in (3.6). Stochastic first-order methods are a very common approach to obtain such a solution. A key observation of such algorithms (e.g.[51, 70]) is to use an *unbiased stochastic estimator* of the true gradient to perform the gradient update. For each iteration, gradient descent in the primal variable and gradient ascent in the dual variable are performed to obtain the saddle point.

As from [113], we can use the stochastic SPP formulation (3.6) for AUC optimization and develop a regularized stochastic online learning algorithm that only needs to pass the data once. We will use similar notation as in [113]: let vector $v=(\mathbf{w}^\top,a,b)^\top\in\mathbb{R}^{d+2}$, and for any $w\in\mathbb{R}^d$, $a,b,\alpha\in\mathbb{R}$ and $z=(\mathbf{x},y)\in\mathcal{Z}$, we write $f(\mathbf{w},a,b,\alpha)$ as $f(v,\alpha)$, and $F(\mathbf{w},a,b,\alpha,z)$ as $F(v,\alpha,z)$. It is important to note that the gradient of the stochastic SPP problem (3.6) is a $(d+3)$-dimensional column vector, specifally $g(v,\alpha)=(\partial_v f(v,\alpha),-\partial_\alpha f(v,\alpha))$. For any $z\in\mathcal{Z}$, its unbiased stochastic estimator is given by

$$G(v,\alpha,z)=(\partial_u F(v,\alpha,z),-\partial_\alpha F(v,\alpha,z)).$$

Solving the stochastic SPP formulation (3.6) could be done by directly applying the first-order method in [70], however, from the definition of F in (3.5), this would require

knowing the probability $p = \Pr(y = 1)$ *a priori*. Generally, this is unknown. To this end, for any $v^\top = (\mathbf{w}^\top, a, b) \in \mathbb{R}^{d+2}$, $\alpha \in \mathbb{R}$ and $z \in \mathcal{Z}$, let

$$\begin{aligned}\hat{F}_t(v, \alpha, z) = (1 - \hat{p}_t)(\mathbf{w}^\top \mathbf{x} - a)^2 \mathbb{I}_{[y=1]} + \hat{p}_t(\mathbf{w}^\top \mathbf{x} - b)^2 \mathbb{I}_{[y=-1]} \\ + 2(1 + \alpha)(\hat{p}_t \mathbf{w}^\top x \mathbb{I}_{[y=-1]} - (1 - \hat{p}_t)\mathbf{w}^\top x \mathbb{I}_{[y=1]}) \\ - \hat{p}_t(1 - \hat{p}_t)\alpha^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2. \end{aligned} \tag{3.16}$$

where $\hat{p}_t = \dfrac{\sum_{i=1}^{t} \mathbb{I}_{[y_i=1]}}{t}$ at iteration t. At each iteration t, use the stochastic estimator

$$\hat{G}_t(v, \alpha, z) = (\partial_v \hat{F}_t(v, \alpha, z), -\partial_\alpha \hat{F}_t(v, \alpha, z)) \tag{3.17}$$

in place of the inaccessible stochastic estimator $G(v, \alpha, z)$.

For a fair comparison with the algorithms discussed in chapters 4 and 5, we can reformulate the bound R in terms of the regularization parameter λ. First, assume $\kappa = \sup_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x}\| < \infty$, and recall that $\|\mathbf{w}\| \leq R$. As from before, for the optimal solution $\mathbf{w}^*$, we have the following:

$$\frac{\lambda}{2}\|\mathbf{w}^*\|^2 \leq \mathbb{E}\Big[(1 - \mathbf{w}^\top(\mathbf{x} - \mathbf{x}'))^2 | y = 1, y' = -1\Big] + \frac{\lambda}{2}\|\mathbf{w}\|^2.$$

Letting $\mathbf{w} = \mathbf{0}$ and recalling that $\|\mathbf{w}\| \leq R$, it is easy to see that $R \leq \sqrt{\frac{2}{\lambda}}$.

Now, given an optimal solution $(\mathbf{w}^*, a^*, b^*)$ of the stochastic SPP (3.6) for AUC optimization, by (3.9), (3.11) and (3.13) we know that

$$|a^*| = \frac{1}{p}\Big| \int_{\mathcal{Z}} \langle \mathbf{w}^*, \mathbf{x} \rangle \mathbb{I}_{[y=1]} d\rho(z) \Big| \leq R\kappa,$$

$$|b^*| = \frac{1}{1-p}\Big| \int_{\mathcal{Z}} \langle \mathbf{w}^*, \mathbf{x}' \rangle \mathbb{I}_{[y'=-1]} d\rho(z') \Big| \leq R\kappa,$$

$$|\alpha^*| = \Big| \frac{1}{1-p} \int_{\mathcal{Z}} \langle \mathbf{w}^*, \mathbf{x}' \rangle \mathbb{I}_{[y'=-1]} d\rho(z') - \frac{1}{p} \int_{\mathcal{Z}} \langle \mathbf{w}^*, \mathbf{x} \rangle \mathbb{I}_{[y=1]} d\rho(z) \Big| \leq 2R\kappa.$$

Algorithm 18 Regularized Stochastic Online AUC Maximization (regSOLAM)

Input: Step sizes $\{\eta_t > 0 : t \in \mathbb{N}\}$
Initialize $t = 1$, $v_1 \in \Omega_1$, $\alpha_1 \in \Omega_2$ and let $\hat{p}_0 = 0, \bar{v}_0 = 0$, $\bar{\alpha}_0 = 0$ and $\bar{\gamma}_0 = 0$.
for $t = 1$ **to** T **do**
  $z_t = (x_t, y_t)$ and compute $\hat{p}_t = \frac{(t-1)\hat{p}_{t-1} + \mathbb{I}_{[y_t=1]}}{t}$
  Update $v_{t+1} = P_{\Omega_1}(v_t - \gamma_t \partial_v \hat{F}_t(v_t, \alpha_t, z_t))$
  Update $\alpha_{t+1} = P_{\Omega_2}(\alpha_t + \gamma_t \partial_\alpha \hat{F}_t(v_t, \alpha_t, z_t))$
  Update $\bar{\gamma}_t = \bar{\gamma}_{t-1} + \gamma_t$
  Update $\bar{v}_t = \frac{1}{\bar{\gamma}_t}(\bar{\gamma}_{t-1}\bar{v}_{t-1} + \gamma_t v_t)$, and $\bar{\alpha}_t = \frac{1}{\bar{\gamma}_t}(\bar{\gamma}_{t-1}\bar{\alpha}_{t-1} + \gamma_t \alpha_t)$
end for

Finally, we can restrict $(\mathbf{w}, a, b)$ and α to the following bounded domains:

$$\Omega_1 = \big\{(\mathbf{w}, a, b) \in \mathbb{R}^{d+2} : \|\mathbf{w}\| \leq R, |a| \leq R\kappa, |b| \leq R\kappa\big\},$$
$$\Omega_2 = \big\{\alpha \in \mathbb{R} : |\alpha| \leq 2R\kappa\big\}. \tag{3.18}$$

The pseudo-code of the online AUC optimization algorithm is described in Algorithm 18, which we will refer to as *regSOLAM*. It is important to note that $P_{\Omega_1}(\cdot)$ and $P_{\Omega_2}(\cdot)$ denote the projection to the convex sets Ω_1 and Ω_2, respectively, and can be easily computed.

3.4 Convergence Analysis

Finally, we will detail the convergence results of the proposed regSOLAM algorithm. The results for the convergence rate is similar as in [113]. We begin by letting $u = (v, \alpha) = (\mathbf{w}, a, b, \alpha)$. The quality of the approximation solution $(\bar{v}_t, \bar{\alpha}_t)$ is given by the duality gap at iteration t:

$$\varepsilon_f(\bar{v}_t, \bar{\alpha}_t) = \max_{\alpha \in \Omega_2} f(\bar{v}_t, \alpha) - \min_{v \in \Omega_1} f(v, \bar{\alpha}_t). \tag{3.19}$$

The following theorem gives the measure of the duality gap.

Theorem 3.2. *Assume that samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T)\}$ are i.i.d. drawn from a distribution ρ over $\mathcal{X} \times \mathcal{Y}$, let Ω_1 and Ω_2 be given by* (3.18) *and the step sizes given by $\{\gamma_t > 0 : t \in \mathbb{N}\}$. For sequence $\{(\bar{v}_t, \bar{\alpha}_t) : t \in [1, T]\}$ generated by regSOLAM (Algorithm 18), and any $0 < \delta < 1$, with probability $1 - \delta$, the following holds*

$$\varepsilon_f(\bar{v}_T, \bar{\alpha}_T) \leq C_\kappa \max(R^2, 1)\sqrt{\ln\frac{4T}{\delta}}\Big(\sum_{j=1}^{T}\gamma_j\Big)^{-1}\Big[1 + \sum_{j=1}^{T}\gamma_j^2 + \Big(\sum_{j=1}^{T}\gamma_j^2\Big)^{\frac{1}{2}} + \sum_{j=1}^{T}\frac{\gamma_j}{\sqrt{j}}\Big],$$

where C_κ is an absolute constant independent of R and T (see the detailed constant in the proof).

Let the optimum of (3.6) be f^*, and by Theorem 3.1, is the same as the optimal value of AUC optimization (3.3). Using Theorem 3.2, we can state the convergence rate of regSOLAM.

Corollary 3.4.1. *Under the same assumptions as in Theorem 3.2, and $\{\gamma_j = \zeta j^{-\frac{1}{2}} : j \in \mathbb{N}\}$ with constant $\zeta > 0$, with probability $1-\delta$, it holds*

$$|f(\bar{v}_T, \bar{\alpha}_T) - f^*| \leq \varepsilon_f(\bar{u}_T) = \mathcal{O}\left(\frac{\ln T \sqrt{\ln\left(\frac{4T}{\delta}\right)}}{\sqrt{T}}\right).$$

Unlike the above convergence rate where a decaying step size was chosen, a similar result can be obtained using an appropriately chosen constant step size as done in [59].

To prove Theorem 3.2, we first require several lemmas. Many of these results are standard from convex online learning [120].

Lemma 3.1. *For any $T \in \mathbb{N}$, let $\{\xi_j : j \in [1,T]\}$ be a sequence of vectors in $\mathbb{R}^m$, and $\tilde{u}_1 \in \Omega$ where Ω is a convex set. For any $t \in [1,T]$ define $\tilde{u}_{t+1} = P_\Omega(\tilde{u}_t - \xi_t)$. Then, for any $u \in \Omega$, there holds $\sum_{t=1}^T (\tilde{u}_t - u)^\top \xi_t \leq \frac{\|\tilde{u}_1 - u\|^2}{2} + \frac{1}{2}\sum_{t=1}^T \|\xi_t\|^2$.*

The next lemma is the Pinelis-Bernstein inequality for martingale difference sequence in a Hilbert space from [74, Theorem 3.4]

Lemma 3.2. *Let $\{S_k : k \in \mathbb{N}\}$ be a martingale difference sequence in a Hilbert space. Suppose that almost surely $\|S_k\| \leq B$ and $\sum_{k=1}^T \mathbb{E}[\|S_k\|^2 | S_1, \ldots, S_{k-1}] \leq \sigma_T^2$. Then, for any $0 < \delta < 1$, there holds, with probability at least $1-\delta$, $\sup_{1\leq j\leq T} \left\|\sum_{k=1}^j S_k\right\| \leq 2\left(\frac{B}{3} + \sigma_T\right)\log\frac{2}{\delta}$.*

The last lemma states that the approximate stochastic estimator $\hat{G}_j(u,z)$ defined by (3.17), is close to the unbiased $G(u,z)$.

Lemma 3.3. *Let Ω_1 and Ω_2 be given by (3.18) and denote by $\Omega = \Omega_1 \times \Omega_2$. For any $t \in \mathbb{N}$, with probability $1-\delta$, there holds $\sup_{u\in\Omega, z\in\mathcal{Z}} \|\hat{G}_t(u,z) - G(u,z)\| \leq (4R\kappa^2 + \kappa + R(\lambda + 18\kappa)\big(\ln(\frac{2}{\delta})/t\big)^{\frac{1}{2}}$.*

Using the above results, we will now prove the main convergence theorem. It is important to note that the proof is similar as to the one presented in [113].

Proof of Theorem 3.2. First, using the convexity of $f(\cdot, \alpha)$ and the concavity of $f(v, \cdot)$, for any $u = (v, \alpha) \in \Omega_1 \times \Omega_2$, we have

$$\begin{aligned} f(v_t, \alpha) - f(v, \alpha_t) &= (f(v_t, \alpha_t) - f(v, \alpha_t)) + (f(v_t, \alpha) - f(v_t, \alpha_t)) \\ &\leq (v_t - v)^\top \partial_v f(v_t, \alpha_t) - (\alpha_t - \alpha)\partial_\alpha f(v_t, \alpha_t) \\ &= (u_t - u)^\top g(u_t) \end{aligned}$$

Hence, it holds that

$$\begin{aligned} \max_{\alpha \in \Omega_2} f(\bar{v}_T, \alpha) - \min_{v \in \Omega_1} f(v, \bar{\alpha}_T) &\leq \left(\sum_{t=1}^T \gamma_t \right)^{-1} \left(\max_{\alpha \in \Omega_2} \sum_{t=1}^T \gamma_t f(v_t, \alpha) - \min_{v \in \Omega_1} \sum_{t=1}^T \gamma_t f(v, \alpha_t) \right) \\ &\leq \left(\sum_{t=1}^T \gamma_t \right)^{-1} \max_{u \in \Omega_1 \times \Omega_2} \sum_{t=1}^T \gamma_t (u_t - u)^\top g(u_t) \end{aligned} \tag{3.20}$$

Recall that by definition $\Omega = \Omega_1 \times \Omega_2$. Applying the notation that $u = (v, \alpha)$, we can rewrite steps 4 and 5 in Algorithm 18 as

$$u_{t+1} = (v_{t+1}, \alpha_{t+1}) = P_\Omega(u_t - \gamma_t \hat{G}_t(u_t, z_t))$$

Using Lemma 3.1 with $\xi_t = \gamma_t \hat{G}_t(u_t, z_t)$, it holds, for any $u \in \Omega$, that

$$\sum_{t=1}^T \gamma_t (u_t - u)^\top \hat{G}_t(u_t, z_t) \leq \frac{\|u_1 - u\|^2}{2} + \frac{1}{2} \sum_{t=1}^T \gamma_t^2 \|\hat{G}_t(u_t, z_t)\|^2,$$

which gives

$$\begin{aligned}
\sup_{u\in\Omega}\sum_{t=1}^{T}\gamma_t(u_t-u)^\top g(u_t) &\le \sup_{u\in\Omega}\frac{\|u_1-u\|^2}{2}+\frac{1}{2}\sum_{t=1}^{T}\gamma_t^2\|\hat{G}_t(u_t,z_t)\|^2\\
&\quad+\sup_{u\in\Omega}\sum_{t=1}^{T}\gamma_t(u_t-u)^\top(g(u_t)-\hat{G}_t(u_t,z_t))\\
&\le \sup_{u\in\Omega}\frac{\|u_1-u\|^2}{2}+\frac{1}{2}\sum_{t=1}^{T}\gamma_t^2\|\hat{G}_t(u_t,z_t)\|^2\\
&\quad+\sup_{u\in\Omega}\sum_{t=1}^{T}\gamma_t(u_t-u)^\top(g(u_t)-G(u_t,z_t))\\
&\quad+\sup_{u\in\Omega}\sum_{t=1}^{T}\gamma_t(u_t-u)^\top(G(u_t,z_t)-\hat{G}_t(u_t,z_t)) \qquad (3.21)
\end{aligned}$$

To complete the proof it is necessary to estimate the four terms on the right hand side of (3.21). For the first term, it is easy to see that

$$\frac{1}{2}\sup_{u\in\Omega}\|u_1-u\|^2 \le 2\sup_{v\in\Omega_1,\alpha\in\Omega_2}(\|v\|^2+|\alpha|^2) \le 2\sup_{u\in\Omega}\|u\|^2 \le 2R^2(1+6\kappa^2). \qquad (3.22)$$

For the second term of (3.21), recall that $\sup_{\mathbf{x}\in\mathcal{X}}\|\mathbf{x}\|\le\kappa$ and $u_t=(\mathbf{w}_t,a_t,b_t,\alpha_t)\in\Omega=\big\{(\mathbf{w},a,b,\alpha):\|\mathbf{w}\|\le R,|a|\le\kappa R,|b|\le\kappa R,|\alpha|\le 2\kappa R\big\}$. Using this with the definition of $\hat{G}_t(u_t,z_t)$ given by (3.17), one can obtain that $\|\hat{G}_t(u_t,z_t)\|\le\|\partial_{\mathbf{w}}\hat{F}_t(u_t,z_t)\|+|\partial_a\hat{F}_t(u_t,z_t)|+|\partial_b\hat{F}_t(u_t,z_t)|+|\partial_\alpha\hat{F}_t(u_t,z_t)|\le 4R\kappa^2+\kappa+R(\lambda+5\kappa)$. Therefore, we can bound the second term as follows:

$$\frac{1}{2}\sum_{t=1}^{T}\gamma_t^2\|\hat{G}_t(u_t,z_t)\|^2 \le (4R\kappa^2+\kappa+R(\lambda+5\kappa))^2\left(\sum_{t=1}^{T}\gamma_t^2\right). \qquad (3.23)$$

It is important to note that this bound is different from the proof in [113] because of the inclusion of the regularization term. Bounding the third term on the right hand side of

(3.21) can be done by

$$\sup_{u\in\Omega}\sum_{t=1}^{T}\gamma_t(u_t-u)^\top(g(u_t)-G(u_t,z_t)) \le \sup_{u\in\Omega}[\sum_{t=1}^{T}\gamma_t(\tilde{u}_t-u)^\top(g(u_t)-G(u_t,z_t))] + \sum_{t=1}^{T}\gamma_t(u_t-\tilde{u}_t)^\top(g(u_t)-G(u_t,z_t)),$$

where $\tilde{u}_1=0\in\Omega$ and $\tilde{u}_{t+1}=P_\Omega(\tilde{u}_t-\gamma_t(g(u_t)-G(u_t,z_t)))$ for any $t\in[1,T]$. Using Lemma 3.1 with $\xi_t=\gamma_t(g(u_t)-G(u_t,z_t))$ gives that

$$\begin{aligned}\sup_{u\in\Omega}\sum_{t=1}^{T}&\gamma_t(\tilde{u}_t-u)^\top(g(u_t)-G(u_t,z_t))\\ &\le \sup_{u\in\Omega}\frac{\|u\|^2}{2}+\frac{1}{2}\sum_{t=1}^{T}\gamma_t^2\|g(u_t)-G(u_t,z_t)\|^2\\ &\le \frac{1}{2}R^2(1+6\kappa^2)+(4R\kappa^2+\kappa+R(\lambda+5\kappa))^2\sum_{t=1}^{T}\gamma_t^2. \end{aligned} \tag{3.24}$$

In the above, we used that $\|G(u_t,z_t)\|$ and $\|g(u_t)\|$ are uniformly bounded by $4R\kappa+\kappa+R(\lambda+5\kappa)$. It is important to note that u_t and $\tilde{u}_t$ are only dependent on $\{z_1,z_2,\ldots,z_{t-1}\}$ and $\{S_t=\gamma_t(u_t-\tilde{u}_t)^\top(g(u_t)-G(u_t,z_t)):t=1,\ldots,t\}$ is a martingale difference sequence. Therefore, we have that

$$\begin{aligned}\mathbb{E}[\|S_t\|^2|z_1,\ldots,z_{t-1}] &= \gamma_t^2\iint_{\mathcal{Z}}((u_t-\tilde{u}_t)^\top(g(u_t)-G(u_t,z)))^2d\rho(z)\\ &\le \gamma_t^2\sup_{u\in\Omega,z\in\mathcal{Z}}[\|u_t-\tilde{u}_t\|^2\|g(u_t)-G(u_t,z_t)\|^2]\\ &\le \gamma_t^2[R\sqrt{1+6\kappa^2}(4R\kappa^2+\kappa+R(\lambda+5\kappa))]^2.\end{aligned}$$

Implementing Lemma 3.2 with $\sigma_T^2=[R\sqrt{1+6\kappa^2}(4R\kappa^2+\kappa+R(\lambda+5\kappa))]^2\sum_{t=1}^{T}\gamma_t^2$, $B=\sup_{t=1}^{T}\gamma_t|(u_t-\tilde{u}_t)^\top(g(u_t)-G(u_t,z_t))|\le\sigma_T$ gives, with probability $1-\frac{\delta}{2}$, that

$$\begin{aligned}\sum_{t=1}^{T}&\gamma_t(u_t-\tilde{u}_t)^\top(g(u_t)-G(u_t,z_t))\\ &\le \frac{8R\sqrt{1+6\kappa^2}(4R\kappa^2+\kappa+R(\lambda+5\kappa))}{3}\sqrt{\sum_{t=1}^{T}\gamma_t^2}.\end{aligned} \tag{3.25}$$

Putting together (3.24) with (3.25) implies, with probability $1-\frac{\delta}{2}$,

$$\begin{aligned}\sup_{u\in\Omega}\sum_{t=1}^{T}\gamma_t(u_t-u)^\top(g(u_t)-G(u_t,z_t))&\\ \leq \frac{R^2(1+6\kappa^2)}{2}+(4R\kappa^2+\kappa+R(\lambda+5\kappa))^2\sum_{t=1}^{T}\gamma_t^2&\\ +\frac{8R\sqrt{1+6\kappa^2}(4R\kappa^2+\kappa+R(\lambda+5\kappa))}{3}\big(\sum_{t=1}^{T}\gamma_t^2\big)^{1/2}.&\end{aligned}\tag{3.26}$$

Applying Lemma 3.3, for any $t\in[1,T]$, gives

$$\sup_{u\in\Omega,z\in\mathcal{Z}}\|\hat{G}_t(u,z)-G(u,z)\|\leq(4R\kappa^2+\kappa+R(\lambda+18\kappa))\sqrt{\ln(\frac{4T}{\delta})/t}.$$

Finally, the fourth term on the righthand side of (3.21) can be estimated with probability $1-\frac{\delta}{2}$:

$$\begin{aligned}\sup_{u\in\Omega}\sum_{t=1}^{T}\gamma_t(u_t-u)^\top(G(u_t,z_t)-\hat{G}_t(u_t,z_t))&\\ \leq 2\sup_{u\Omega}\|u\|\big(\sum_{t=1}^{T}\gamma_t\sup_{u\in\Omega,z\in\mathcal{Z}}\|\hat{G}_t(u,z)-G(u,z)\|\big)&\\ \leq 4R(4R\kappa^2+\kappa+R(\lambda+18\kappa))\sqrt{6\kappa^2+1}\sum_{t=1}^{T}\frac{\gamma_t}{\sqrt{t}}.&\end{aligned}\tag{3.27}$$

Applying the estimations (3.22), (3.23), (3.26), (3.27) and (3.21) to (3.20) implies that

$$\varepsilon_f(\bar{u}_T)\leq C_\kappa\max(R^2,1)\sqrt{\ln\frac{4T}{\delta}}\big(\sum_{t=1}^{T}\gamma_t\big)^{-1}\Big[1+\sum_{t=1}^{T}\gamma_t^2+\big(\sum_{t=1}^{T}\gamma_t^2\big)^{\frac{1}{2}}+\sum_{t=1}^{T}\frac{\gamma_t}{\sqrt{t}}\Big],$$

where $C_\kappa=\frac{5}{2}(1+6\kappa^2)+2(4\kappa^2+6\kappa+\lambda)^2+(\frac{80}{3}\kappa^2+88\kappa+\frac{20}{3}\lambda)\sqrt{1+6\kappa^2}$ □

3.5 Experiments

In this section, we will validate the theoretical results of the previous section by examining the experimental evaluations of regSOLAM with existing state-of-the-art learning algorithms for AUC optimization. Specifically, these experiments will demonstrate that reg-

Datasets	♯ Inst	♯ Feat	Datasets	♯ Inst	♯ Feat
diabetes	768	8	fourclass	862	2
german	1,000	24	splice	3,175	60
usps	9,298	256	a9a	32,561	123
mnist	60,000	780	acoustic	78,823	50
ijcnn1	141,691	22	covtype	581,012	54
sector	9,619	55,197	news20	15,935	62,061

Table 3.1: Basic information about the benchmark datasets.

SOLAM has a lower computational complexity while maintaining a similar performance in comparison to existing methods.

We conduct comprehensive studies by comparing the proposed algorithm with other AUC optimization algorithms for both online and batch scenarios. The algorithms considered in our experiments include:

- **regSOLAM**: The regularized online projected gradient descent algorithm for AUC maximization.
- **OPAUC**: The one-pass AUC optimization algorithm with square loss function [34].
- **OAMseq**: The OAM algorithm with reservoir sampling and sequential updating method [118].
- **OAMgra**: The OAM algorithm with reservoir sampling and online gradient updating method [118].
- **Online Uni-Exp**: Online learning algorithm which optimizes the (weighted) univariate exponential loss [48].
- **B-SVM-OR**: A batch learning algorithm which optimizes the pairwise hinge loss [42].
- **B-LS-SVM**: A batch learning algorithm which optimizes the pairwise square loss.

To examine the performance of the proposed regSOLAM algorithm in comparison to state-of-the-art methods, we conduct experiments on 12 benchmark datasets. Table 3.1 shows the details of each of the datasets. All of these datasets are available for download from the LIBSVM and UCI machine learning repository. Note that some of the datasets (*mnist*, *covtype*, etc.) are multi-class, which we converted to binary data by randomly partitioning the data into two groups, where each group includes the same number of classes.

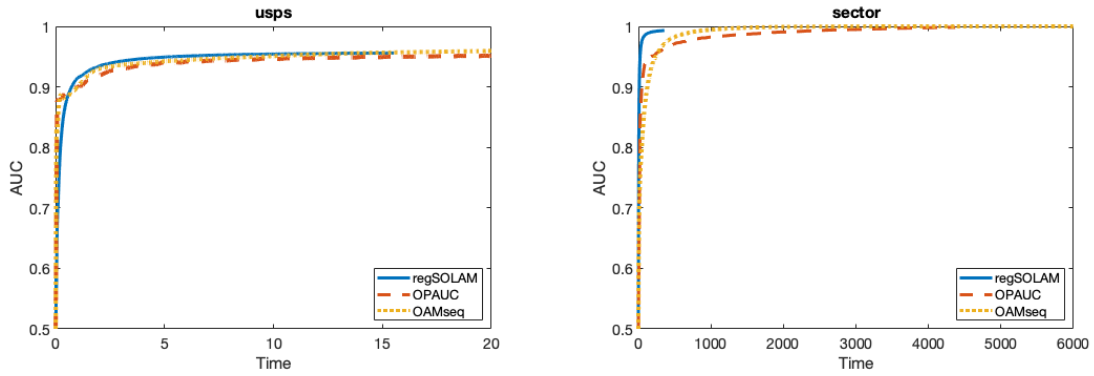
Datasets	regSOLAM	OPAUC	OAM_{seq}	OAM_{gra}	B-SVM-OR	B-LS-SVM
diabetes	.8140±.0330	.8309±.0350	.8264±.0367	.8262±.0338	.8326±.0328	.8325±.0329
fourclass	.8222±.0276	.8310±.0251	.8306±.0247	.8295±.0251	.8305±.0311	.8309±.0309
german	.7830±.0247	.7978±.0347	.7747±.0411	.7723±.0358	.7935±.0348	.7994±.0343
splice	.9237±.0090	.9232±.0099	.8594±.0194	.8864±.0166	.9239±.0089	.9245±.0092
usps	.9848±.0021	.9620±.0040	.9310±.0159	.9348±.0122	.9630±.0047	.9634±.0045
a9a	.8970±.0048	.9002±.0047	.8420±.0174	.8571±.0173	.9009±.0036	.8982±.0028
mnist	.9599±.0014	.9242±.0021	.8615±.0087	.8643±.0112	.9340±.0020	.9336±.0025
acoustic	.8114±.0035	.8192±.0032	.7113±.0590	.7711±.0217	.8262±.0032	.8210±.0033
ijcnn1	.9108±.0030	.9269±.0021	.9209±.0079	.9100±.0092	.9337±.0024	.9320±.0037
covtype	.9332±.0020	.8244±.0014	.7361±.0317	.7403±.0289	.8248±.0013	.8222±.0014
sector	.9734±.0036	.9292±.0081	.9163±.0087	.9043±.0100	-	-
news20	.9399±.0038	.8871±.0083	.8543±.0099	.8346±.0094	-	-

Table 3.2: Comparison of the testing AUC values (mean±std.) on the evaluated datasets. To accelerate the experiments, the value for *sector* was determined after five runs instead of 25 for the other data sets. The performances of OPAUC, OAMseq, OAMgra, online Uni-Exp, B-SVM-OR and B-LS-SVM were taken from [34].

For the experiments, the features were normalized by taking $\mathbf{x}_i \leftarrow \frac{\mathbf{x}_i - \text{mean}(\mathbf{x}_i)}{\|\mathbf{x}_i\|}$ for the large datasets and $\mathbf{x}_i \leftarrow \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}$ for the small datasets (*diabetes*, *fourclass*, and *german*). For each dataset, the data is randomly partitioned into 5 folds (4 are for training and 1 is for testing). We generate this partition for each dataset 5 times. This results in 25 runs for each dataset for which we use to calculate the average AUC score and standard deviation. To determine the proper parameter for each dataset, we conduct 5-fold cross validation on the training sets to determine the learning rate $\zeta \in [1 : 9 : 100]$ and the regularization parameter $\lambda \in 10^{[-5:5]}$ by a grid search. The buffer size for OAMseq and OAMgra was 100 as suggested [118]. All experiments for regSOLAM were conducted with MATLAB.

A summary of the classification performances on the testing dataset of all methods is given in Table 3.2. These results give evidence that regSOLAM achieves a similar performance as other state-of-the-art online and offline methods based on AUC maximization. In some cases, regSOLAM performs better than some of the other online learning algorithms. There is a significant improvement in the text classification dataset *sector* and *covtype*. The difference in performance of regSOLAM could be due to the fact that since the data is randomly partitioned into two classes, the value of p could be resulting in a higher AUC score.

However, the main advantage of regSOLAM is the running time performance while maintaining low per iteration costs. The theoretical convergence rate of $\mathcal{O}(1/\sqrt{T})$ is demonstrated in Figure 3.1. The theory tells us that regSOLAM should have a similar rate of



(a) usps (b) sector

Figure 3.1: AUC vs. Time curves of regSOLAM against other state of the art learning algorithms.

convergence as the other existing algorithms and the convergence plots confirm that assumption.

Chapter 4

Stochastic Proximal AUC Maximization

The previous chapter established the saddle point formulation for AUC optimization, however, we can exploit the primal and dual variables to express the formulation in Chapter 3 to purely be a minimization problem. First, we begin with the AUC formulation from the previous chapter:

$$\min_{\mathbf{w},a,b} \max_{\alpha \in \mathbb{R}} \big\{ \mathbb{E}[F(\mathbf{w}, a, b, \alpha; z)] + \Omega(\mathbf{w}) \big\}, \tag{4.1}$$

where the expectation is with respect to $z = (\mathbf{x}, y)$ and $F(\mathbf{w}, a, b, \alpha; z)$ is given as before. The algorithm regSOLAM determined the saddle point by using gradient decent on the primal variables and gradient ascent on the dual variable. An alternative, but novel formulation can be derived for AUC maximization by observing the definitions of the variables a, b, and α. Observe the fact that

$$\mathrm{Var}[\mathbf{w}^\top \mathbf{x} | y = 1] = \min_a \mathbb{E}[(\mathbf{w}^\top \mathbf{x} - a)^2 | y = 1], \tag{4.2}$$

and

$$\mathrm{Var}[\mathbf{w}^\top \mathbf{x}' | y' = -1] = \min_b \mathbb{E}[(\mathbf{w}^\top \mathbf{x}' - b)^2 | y' = -1]. \tag{4.3}$$

In addition,

$$\begin{aligned} (\mathbb{E}[\mathbf{w}^\top \mathbf{x} | y = 1] - \mathbb{E}[\mathbf{w}^\top \mathbf{x}' | y' = -1])^2 = \max_\alpha \{ -\alpha^2 \\ + 2\alpha(\mathbb{E}[\mathbf{w}^\top \mathbf{x}' | y' = -1] - \mathbb{E}[\mathbf{w}^\top \mathbf{x} | y = 1]) \}. \end{aligned} \tag{4.4}$$

It is easy to see that the optima for (4.2), (4.3), and (4.4) are respectively achieved at

$$a(\mathbf{w}) = \mathbf{w}^\top \mathbb{E}[\mathbf{x} | y = 1], \quad b(\mathbf{w}) = \mathbf{w}^\top \mathbb{E}[\mathbf{x} | y = -1], \tag{4.5}$$

$$\alpha(\mathbf{w}) = \mathbf{w}^\top (\mathbb{E}[\mathbf{x} | y' = -1] - \mathbb{E}[\mathbf{x} | y = 1]). \tag{4.6}$$

Algorithm 19 Stochastic Forward-Backward Splitting

Input: Step sizes $\{\eta_t > 0 : t \in \mathbb{N}\}$
Initialize $\mathbf{w}_1 \in R^d$.
for $t = 1$ **to** T **do**
  Receive sample $z_t = (x_t, y_t)$
  $\hat{\mathbf{w}}_{t+1} = \mathbf{w}_t - \eta_t \nabla \ell$
  $\mathbf{w}_{t+1} = \text{prox}_{\eta_t} \hat{\mathbf{w}}_t$
end for

This allows us to reformulate AUC optimization to be purely just a minimization problem with respect to $\mathbf{w}$:

$$\min_{\mathbf{w}} \big\{ \mathbb{E}[F(\mathbf{w}, a(\mathbf{w}), b(\mathbf{w}), \alpha(\mathbf{w}); z)] + \Omega(\mathbf{w}) \big\}, \tag{4.7}$$

where a, b, and α are updated by (4.5) and (4.6). We can now apply a different approach to solving this problem that is more novel than the method discussed in chapter 3.

4.1 Proximal Methods and Algorithm Formulation

The general structure of (4.7) is a common problem in machine learning that takes the following form:

$$\min_{\mathbf{w}} \ell(\mathbf{w}) + \Omega(\mathbf{w}), \tag{4.8}$$

where ℓ is a loss function while Ω represents some regularizer. A solution to this problem can be obtained by solving the problem in *two* steps: first, determine a solution for $\mathbf{w}$ by minimizing ℓ, then find a solution that minimizes $\Omega(\mathbf{w})$ that is close to the solution obtained in the first step. This idea is more precisely known as a "forward-backward" splitting algorithm [86]. The first step is to simply perform gradient descent on ℓ. The critical step is determining a solution that minimizes $\Omega(\mathbf{w})$ while also minimizing $\ell(\mathbf{w})$. By design, the proximal operator finds a solution that minimizes $\Omega(\mathbf{w})$ while the solution is restricted to being close to a solution obtained from gradient descent. The proximal step is given by:

$$\text{prox}_{\eta_t \Omega}(u) = \arg\min \left\{ \frac{1}{2} \|u - \mathbf{w}\|_2^2 + \eta \Omega(\mathbf{w}) \right\}, \tag{4.9}$$

where η is a chosen step size. The motivating algorithm is summarized in Algorithm 19.

Proximal algorithms are an entire class of methods that are used to solve convex optimization problems. Like Newton's method, proximal algorithms are a standard tool

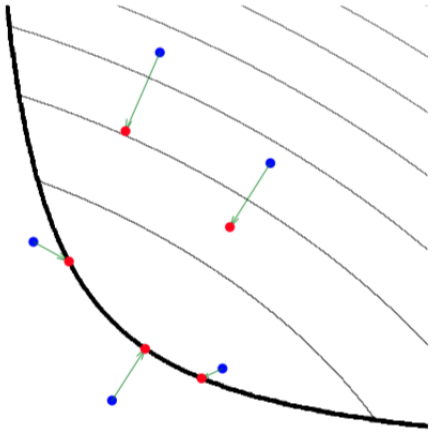


Figure 4.1: Interpretation of a proximal operator at selected points.

for non-smooth problems. They have been applied in many areas and are of high interest for high-dimensional data sets [115]. The key idea behind this operator can be derived from the optimality condition such that

$$\begin{aligned} 0 \in \nabla \ell(\mathbf{w}) + \partial \Omega(\mathbf{w}) &\iff \mathbf{w} - \eta \nabla \ell(\mathbf{w}) \in \mathbf{w} + \eta \partial \Omega(\mathbf{w}) \\ &\iff \mathbf{w} = P_f^{\eta}(\mathbf{w} - \eta \nabla \ell(\mathbf{w})). \end{aligned} \tag{4.10}$$

Another way of interpreting proximal operators is to consider when $\Omega(\mathbf{w})$ is the indicator function for a convex set W, i.e. $\Omega(\mathbf{w}) = \infty$ when $\mathbf{w} \notin W$ and 0 otherwise. Therefore, the proximal operator can simply be interpreted as the Euclidean projection onto W:

$$\Pi_X(y) = \underset{\mathbf{w} \in W}{\operatorname{argmin}} \|\mathbf{w} - y\|.$$

Proximal operators can be viewed as just a generalization of projections. However, a critical reason as to why proximal operators are used is because of their fixed point property: $\mathbf{w}^*$ is a fixed point of the proximal operator (i.e. $\text{prox}_{\Omega}(\mathbf{w}^*) = \mathbf{w}^*$) if and only if $\mathbf{w}^*$ minimizes $\Omega(\mathbf{w})$. Therefore, solving a minimization problem is equivalent to finding the fixed point of the proximal operator.

We can apply the previous ideas and motivations to solve problem (4.7) by conducting stochastic gradient descent only on $\mathbf{w}$, while a, b, and α are then updated using equations (4.5) and (4.6), rather than doing stochastic gradient updates. Specifically, for each new

Algorithm 20 Stochastic Proximal AUC Maximization (SPAM)

Input: Step sizes $\{\eta_t > 0 : t \in \mathbb{N}\}$
Initialize $\mathbf{w}_1 \in R^d$.
for $t = 1$ **to** T **do**
  Receive sample $z_t = (x_t, y_t)$
  Compute $a(\mathbf{w}_t)$, $b(\mathbf{w}_t)$, and $\alpha(\mathbf{w}_t)$ according to (4.5) and (4.6).
  $\hat{\mathbf{w}}_{t+1} = \mathbf{w}_t - \eta_t \partial_1 F(\mathbf{w}_t, a(w_t), b(\mathbf{w}_t), \alpha(\mathbf{w}_t); z_t)$
  $\mathbf{w}_{t+1} = \text{prox}_{\eta_t \Omega}(\hat{\mathbf{w}}_{t+1})$
end for

data z_t, we update $\mathbf{w}$ by

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \partial_1 F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}_t); z_t), \tag{4.11}$$

where $\partial_1 F$ denotes the gradient with respect to the first argument and η_t denotes the step size. The proximal map for a convex function $\Omega : \mathbb{R}^d \to \mathbb{R}$ is defined as

$$\text{prox}_{\eta_t \Omega}(u) = \arg\min\{\frac{1}{2}\|u - \mathbf{w}\|^2 + \eta_t \Omega(\mathbf{w})\}. \tag{4.12}$$

The novelty of this approach is that $\Omega(\cdot)$ could be a non-smooth penalty function such as ℓ_1. The proposed algorithm is similar to forward-backward splitting [24, 86] with the pseudo-code summarized in Algorithm 20. Our algorithm differs significantly since the forward-backward methods focused on accuracy. The storage and per-iteration cost is one datum, however, the algorithm has an assumption that the class means are known, i.e. $\mathbb{E}(\mathbf{x}|y = 1)$ and $\mathbb{E}(\mathbf{x}|y = -1)$ as well as the probability of class 1. This assumption can be alleviated by using a portion of the training data to estimate the probability of class 1 as well as the class means can be estimated by sample means.

By reformulating AUC in such a way, it is expected that SPAM will have a faster convergence rate than regSOLAM. To illustrate why this is true, let $f(\mathbf{w}) = p(1-p)\mathbb{E}\big[(1 - \mathbf{w}^\top(x - x'))^2 \big| y = 1, y' = -1\big]$ which is the same as $\min_{a,b}\max_\alpha \mathbb{E}[F(\mathbf{w}, a, b, \alpha; z)]$. The following critical lemma determines why this can be done.

Lemma 4.1. *Let $\mathbf{w}_t$ be given by SPAM described in Algorithm 20. Then, we have that*

$$\partial f(\mathbf{w}_t) = \mathbb{E}_{z_t}[\partial_1 F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}_t); z_t)],$$

where $\mathbb{E}_{z_t}[\cdot]$ *denotes the expectation with respect to* $z_t = (x_t, y_t)$.

Proof. First, the notation $\partial_i F$ denotes the partial derivative of F with respect to the ith argument. The application of the chain rule gives

$$\begin{aligned}
\partial_{\mathbf{w}} f(\mathbf{w}_t) &= \partial_{\mathbf{w}} \mathbb{E}_{z_t}[F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}_t); z_t)] \\
&= \mathbb{E}_{z_t}\Big[\partial_{\mathbf{w}} F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}_t); z_t)\Big] \\
&= \mathbb{E}_{z_t}\Big[\partial_1 F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}_t); z_t)\Big] \\
&+ \mathbb{E}_{z_t}\Big[\partial_2 F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}); z_t)\, \partial_{\mathbf{w}} a(\mathbf{w}_t)\Big] \\
&+ \mathbb{E}_{z_t}\Big[\partial_3 F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}_t); z_t)\, \partial_{\mathbf{w}} b(\mathbf{w}_t)\Big] \\
&+ \mathbb{E}_{z_t}\Big[\partial_4 F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}_t); z_t) \partial_{\mathbf{w}} \alpha(\mathbf{w}_t)\Big]. \qquad (4.13)
\end{aligned}$$

The interchanging between differentiation and integration in the second inequality follows from Leibniz's Integral rule since $F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}); z_t)$ is quadratic and the input space $\mathcal{X}$ is a bounded domain. For the last part, we have that

$$\begin{aligned}
&\mathbb{E}_{z_t}\Big[\partial_2 F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}); z_t)\, \partial_{\mathbf{w}} a(\mathbf{w}_t)\Big] \\
&\qquad = \partial_2 \mathbb{E}_{z_t}\Big[F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}); z_t)\Big][\mathbb{E}(x|y=1)], \qquad (4.14)
\end{aligned}$$

since $\mathbf{w}_t$ depends on $\{z_1, z_2, \ldots, z_{t-1}\}$. To prove the lemma, it just requires the use of the first order optimality condition applied on a, b, and α. For

$$\min_{a,b} \max_{\alpha} \mathbb{E}_{z_t}[F(\mathbf{w}_t, a, b, \alpha; z_t)],$$

$a(\mathbf{w}_t)$ is the minimizer which gives that $\partial_2 \mathbb{E}_{z_t}\Big[F(\mathbf{w}_t, a(\mathbf{w}_t), b, \alpha; z_t)\Big] = 0$. It then follows that:

$$\begin{aligned}
&\partial_2 \mathbb{E}_{z_t}\Big[F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}_t); z_t)\Big] \\
&\qquad = \mathbb{E}_{z_t}\Big[\partial_2 F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}); z_t)\, \partial_{\mathbf{w}} a(\mathbf{w}_t)\Big] = 0.
\end{aligned}$$

A similar argument can be made for the third and fourth terms on the righthand side of (4.13). This completes the proof of the lemma. □

The significance of the above lemma is that on $\{z_1, \ldots, z_{t-1}\}$,

$$\partial_1 F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}_t); z_t)$$

is an unbiased estimator of the true gradient $\partial_{\mathbf{w}} f(\mathbf{w}_t)$. This strongly indicates that SPAM will have a faster convergence rate than regSOLAM in Chapter 3. Therefore, SPAM should have a convergence rate similar to SGD methods for a strongly convex objective function [82, 91]. Using this intuition, we will prove the convergence rate in the next section.

4.2 Convergence Analysis

Before we present our convergence results, we first discuss some critical differences in assumptions about SPAM and the proofs in [24, 86]. The convergence analysis of those methods critically depends on that the iterates are uniformly bounded. The variance of the stochastic gradient is bounded as well, however, verifying this is difficult in practice. The techniques used here do not use any of these assumptions. Additionally, a primal-dual method for saddle point problems that can achieve a convergence rate of $\mathcal{O}(1/T)$ could be applied to AUC maximization, however, the saddle point construction differs from (4.7) [74]. Also, the method assumes strong convexity on both the primal and dual variables unlike the work presented in this chapter.

Before we introduce the convergence rate of SPAM, first we need to define some important notations and assumptions. First, we will let $f(\mathbf{w}) = p(1-p)\mathbb{E}\big[(1 - \mathbf{w}^\top(\mathbf{x} - \mathbf{x}'))^2 \big| y = 1, y' = -1\big]$. The solution of (4.7) will be given by $\mathbf{w}^*$. Next, we will define the following constant for notational simplicity:

$$\mathbb{E}[\|G(\mathbf{w}^*; z) - \partial f(\mathbf{w}^*)\|^2] = \sigma_*^2, \tag{4.15}$$

where $G(\mathbf{w}; z) = \partial_1 F(\mathbf{w}, a(\mathbf{w}), b(\mathbf{w}), \alpha(\mathbf{w}); z)$. We will base the convergence results on the following assumptions:

Assumption 4.1. *Assume that $\Omega(\cdot)$ is β-strongly convex.*

Assumption 4.2. *There exists an $M > 0$ such that $\|\mathbf{x}\| \leq M$ for any $x \in \mathcal{X}$.*

Finally, the convergence results are based on some constants. We denote them as

follows: $C_{\beta,M} := \frac{\beta}{128M^4}$, $\widetilde{C}_{\beta,M} = \frac{\beta}{(1+\frac{\beta^2}{128M^4})^2}$, and $\bar{C}_{\beta,M} = \widetilde{C}_{\beta,M} C_{\beta,M} = \frac{128M^4\beta^2}{(128M^4+\beta^2)^2}$. As from the previous chapter, we use the conventional notation that for any $T \in \mathbb{N}$, $\mathbb{N}_T = \{1, \ldots, T\}$.. To prove the convergence rates of SPAM, we require the following critical lemma that establishes how $\|\mathbf{w}_t - \mathbf{w}^*\|$ changes with t.

Lemma 4.2. *Under Assumptions 4.1 and 4.2, let $\{\mathbf{w}_t : t \in \mathbb{N}_{T+1}\}$ be generated by SPAM. Then, the following statements hold true.*

(i) For any $t \in \mathbb{N}$ there holds

$$\mathbb{E}[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2] \leq \frac{1 + 128M^4\eta_t^2}{(1+\eta_t\beta)^2}\mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2] + 2\sigma_*^2\eta_t^2. \tag{4.16}$$

(ii) If, furthermore, $0 < \eta_t \leq C_{\beta,M} := \dfrac{\beta}{128M^4}$ for any $t \in \mathbb{N}_T$, then we have , for any $t \in \mathbb{N}_T$,

$$\mathbb{E}[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2] \leq \big(1 - \widetilde{C}_{\beta,M}\,\eta_t\big)\mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2] + 2\sigma_*^2\eta_t^2. \tag{4.17}$$

Proof. Recall that we previously stated that $\mathbf{w}^*$ is the solution of (4.7). Now using the first order optimality conditions and (4.10), we have, for any $\eta_t > 0$,

$$\mathbf{w}^* = \mathrm{prox}_{\eta_t\Omega}(\mathbf{w}^* - \eta_t\partial f(\mathbf{w}^*)).$$

Applying the definition of $\mathbf{w}_{t+1}$ in Algorithm 20 as the above oberservation gives that

$$\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 = \|\mathrm{prox}_{\eta_t\Omega}(\hat{\mathbf{w}}_{t+1}) - \mathrm{prox}_{\eta_t\Omega}(\mathbf{w}^* - \eta_t\partial f(\mathbf{w}^*)\|^2. \tag{4.18}$$

By Assumption 4.1, $\eta_t\Omega(\mathbf{w})$ is $\eta_t\beta$-strongly convex and furthermore, by Proposition 23.11 in [2], $\mathrm{prox}_{\eta_t\Omega}(\cdot)$ is $(1+\eta_t\beta)$-cocoercive, i.e., for any $\mathbf{u}$ and $\mathbf{w}$, there holds $\langle \mathbf{u} - \mathbf{w}, \mathrm{prox}_{\eta_t\Omega}(\mathbf{u}) - \mathrm{prox}_{\eta_t\Omega}(\mathbf{w})\rangle \geq (1+\eta_t\beta)\|\mathrm{prox}_{\eta_t\Omega}(\mathbf{u}) - \mathrm{prox}_{\eta_t\Omega}(\mathbf{w})\|^2$. Applying the Cauchy-Schwartz inequality yields that

$$\|\mathrm{prox}_{\eta_t\Omega}(\mathbf{u}) - \mathrm{prox}_{\eta_t\Omega}(\mathbf{w})\| \leq \frac{1}{1+\eta_t\beta}\|\mathbf{u} - \mathbf{w}\|.$$

Substituting this back into (4.18), we obtain

$$\begin{aligned}\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 &= \|\text{prox}_{\eta_t\Omega}(\hat{\mathbf{w}}_{t+1}) - \text{prox}_{\eta_t\Omega}(\mathbf{w}^* - \eta_t\partial f(\mathbf{w}^*)\|^2 \\ &\leq \frac{1}{(1+\eta_t\beta)^2}\|\hat{\mathbf{w}}_{t+1} - (\mathbf{w}^* - \eta_t\partial f(\mathbf{w}^*))\|^2 \\ &= \frac{1}{(1+\eta_t\beta)^2}\|(\mathbf{w}_t - \mathbf{w}^*) - \eta_t(G(\mathbf{w}_t, z_t) - \partial f(\mathbf{w}^*))\|^2. \end{aligned} \tag{4.19}$$

Recall that the last equality uses the notation

$$G(\mathbf{w}_t; z_t) = \partial_1 F(\mathbf{w}_t, a(\mathbf{w}_t), b(\mathbf{w}_t), \alpha(\mathbf{w}_t); z_t).$$

The expectation of both sides of (4.19) and expanding out the righthand side yields

$$\begin{aligned}\mathbb{E}[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2] \leq \frac{1}{(1+\eta_t\beta)^2}\Big(&\mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2] \\ &- 2\eta_t\mathbb{E}[\langle \mathbf{w}_t - \mathbf{w}^*, G(\mathbf{w}_t; z_t) - \partial f(\mathbf{w}^*)\rangle] \\ &+ \eta_t^2\mathbb{E}[\|G(\mathbf{w}_t; z_t) - \partial f(\mathbf{w}^*)\|^2\Big). \end{aligned} \tag{4.20}$$

The first term in on the right hand side of (4.20) gives the desired term. It now suffices to bound the remaining terms. To bound the middle term, we can apply Lemma 4.1 to obtain:

$$\begin{aligned}\mathbb{E}[\langle \mathbf{w}_t - \mathbf{w}^*, G(\mathbf{w}_t; z_t) - \partial f(\mathbf{w}^*)\rangle] &= \mathbb{E}[\langle \mathbf{w}_t - \mathbf{w}^*, \mathbb{E}_{z_t}[G(\mathbf{w}_t; z_t)] - \partial f(\mathbf{w}^*)\rangle] \\ &= \mathbb{E}[\langle \mathbf{w}_t - \mathbf{w}^*, \partial f(\mathbf{w}_t) - \partial f(\mathbf{w}^*)\rangle] \geq 0, \end{aligned} \tag{4.21}$$

where the last inequality follows from the convexity of f. To bound the last term of (4.20), we can do the following:

$$\mathbb{E}[\|G(\mathbf{w}_t; z_t) - \partial f(\mathbf{w}^*)\|^2] \leq 2\mathbb{E}[\|G(\mathbf{w}_t; z_t) - G(\mathbf{w}^*; z_t)\|^2] + 2\mathbb{E}[\|G(\mathbf{w}^*; z_t) - \partial f(\mathbf{w}^*)\|^2].$$

A critical observation is that G is a linear function of $\mathbf{w}_t$. Since by Assumption 4.2 that

$\|\mathbf{x}_t\| \leq M$, we have

$$\begin{aligned}\|G(\mathbf{w}_t; z_t) - G(\mathbf{w}^*; z_t)\| &\leq 4M^2(1-p)\|\mathbf{w}_t - \mathbf{w}^*\|\mathbb{I}_{[y_t=1]} \\ &\quad + 4M^2 p\|\mathbf{w}_t - \mathbf{w}^*\|\mathbb{I}_{[y_t=-1]} \\ &\quad + 4M^2|p - \mathbb{I}_{[y_t=1]}|\|\mathbf{w}_t - \mathbf{w}^*\| \\ &\leq 8M^2\|\mathbf{w}_t - \mathbf{w}^*\|. \end{aligned} \tag{4.22}$$

Even more so, from (4.15), we have $\mathbb{E}[\|G(\mathbf{w}^*; z_t) - \partial f(\mathbf{w}^*)\|^2] = \mathbb{E}_{z_t}[\|G(\mathbf{w}^*; z_t) - \partial f(\mathbf{w}^*)\|^2] = \sigma_*^2$. Therefore,

$$\mathbb{E}[\|G(\mathbf{w}_t; z_t) - \partial f(\mathbf{w}^*)\|^2] \leq 2(8M^2)^2\mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2] + 2\sigma_*^2. \tag{4.23}$$

To finish part (i) of the lemma, we need to just combine together (4.20), (4.21) and (4.23):

$$\begin{aligned}\mathbb{E}[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2] &\leq \frac{1}{(1+\eta_t\beta)^2}\Big(\mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2] \\ &\quad + 2(8M^2)^2\eta_t^2\mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2] + 2\sigma_*^2\eta_t^2\Big) \\ &\leq \frac{1 + 128M^4\eta_t^2}{(1+\eta_t\beta)^2}\mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2] + 2\sigma_*^2\eta_t^2. \end{aligned} \tag{4.24}$$

This finishes part (i) of the lemma.

To prove the second part of the lemma, the coefficient in (4.24) can be reformulated as

$$\frac{1 + 128M^4\eta_t^2}{(1+\eta_t\beta)^2} = 1 - \Big(1 - \frac{1 + 128M^4\eta_t^2}{(1+\eta_t\beta)^2}\Big) = 1 - \frac{[2\beta + \beta^2\eta_t - 128M^4\eta_t]\eta_t}{(1+\eta_t\beta)^2}. \tag{4.25}$$

Applying the hypothesis in (ii) of the lemma that $\eta_t \leq \dfrac{\beta}{128M^4}$ yields

$$\frac{[2\beta + \beta^2\eta_t - 128M^4\eta_t]}{(1+\eta_t\beta)^2}\eta_t \geq \frac{\beta}{\big(1 + \frac{\beta^2}{128M^4}\big)^2}\eta_t. \tag{4.26}$$

Even more so, observe that $\frac{\beta}{128M^4} \leq \frac{\big(1+\frac{\beta^2}{128M^4}\big)^2}{\beta}$. This implies the assumption $\eta_t \leq \frac{\beta}{128M^4}$ guarantees that $1 - \frac{\beta}{\big(1+\frac{\beta^2}{128M^4}\big)^2}\eta_t \geq 0$. To complete the proof, combine together (4.25) and (4.26) to obtain the desired result. This completes the proof of the lemma. □

Before we state the convergence results, the following lemma will be helpful in the proofs [95].

Lemma 4.3. *For any* $0 < \nu \leq 1$, $0 < \alpha < 1$, $t < T$, *and* $0 < \theta \leq 1$, *the following estimations hold true.*

(i) $\sum_{j=t+1}^{T} j^{-\alpha} \geq \frac{1}{1-\alpha}[(T+1)^{1-\alpha} - (t+1)^{1-\alpha}]$,

(ii) $\sum_{t=1}^{T-1} \frac{1}{t^{2\alpha}} \exp\Big\{ -\nu \sum_{j=t+1}^{T} j^{-\alpha} \Big\} \leq \frac{18}{\nu T^{\alpha}} + \frac{9T^{1-\alpha}}{(1-\alpha)2^{1-\alpha}} \exp\{-\frac{\nu(1-2^{\alpha-1})}{1-\alpha}(T+1)^{1-\alpha}\}$,

(iii) $e^{-cx} \leq \big(\frac{b}{ce}\big)^{b} x^{-b}$ *for* $x > 0, c > 0$ *and* $b > 0$.

With the above lemma, we can present and prove the convergence results.

An important observation in Lemma 4.2 is that there are no assumptions about the step size. By including an additional assumption on η_t, SPAM obtains different rates of convergence. The first is stated and proved below.

Theorem 4.1. *Under Assumptions 4.1 and 4.2, and choosing step sizes with some* $\theta \in (0,1)$ *in the form of* $\big\{\eta_t = \frac{C_{\beta,M}}{t^{\theta}} : t \in \mathbb{N}\big\}$, *the algorithm SPAM achieves the following:*

$$\begin{aligned}\mathbb{E}[\|\mathbf{w}_{T+1} - \mathbf{w}^*\|^2] \leq \Big[\exp\Big(\frac{\bar{C}_{\beta,M}}{1-\theta}\Big)\Big(\frac{\theta}{\bar{C}_{\beta,M} e}\Big)^{\frac{\theta}{1-\theta}} \mathbb{E}[\|\mathbf{w}_1 - \mathbf{w}^*\|^2] \\ + 2\sigma_*^2 C_{\beta,M}^2 \Big(\frac{9}{(1-\theta)2^{1-\theta}} \Big(\frac{1}{\bar{C}_{\beta,M}(1-2^{\theta-1})e} \Big)^{\frac{1}{1-\theta}} \\ + \frac{18}{\bar{C}_{\beta,M}} + 1 \Big) \Big] T^{-\theta}.\end{aligned}$$

Proof. For notational simplicity, let $r_t = \mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2]$. The choice of the step sizes $\eta_t = \frac{C_{\beta,M}}{t^{\theta}}$ satisfies the hypothesis in Lemma 4.2, i.e. $\eta_t \leq C_{\beta,M}$. Recall the previously stated constants $C_{\beta,M} = \frac{\beta}{128M^4}$, $\widetilde{C}_{\beta,M} = \frac{\beta}{(1+\frac{\beta^2}{128M^4})^2}$, and $\bar{C}_{\beta,M} = \widetilde{C}_{\beta,M} C_{\beta,M}$ which guarantees that $1 - \widetilde{C}_{\beta,M}\eta_t \geq 1 - \widetilde{C}_{\beta,M} C_{\beta,M} = 1 - \bar{C}_{\beta,M} \geq 0$ for any $t \in \mathbb{N}_T$. From (4.17), after T iterations, it is should be clear that

$$r_{T+1} \leq r_1 \prod_{k=1}^{T} \Big(1 - \widetilde{C}_{\beta,M}\eta_k\Big) + 2\sigma_*^2 \sum_{k=1}^{T-1} \prod_{i=k+1}^{T} \Big(1 - \widetilde{C}_{\beta,M}\,\eta_i\Big)\,\eta_k^2 + 2\sigma_*^2 \eta_T^2. \tag{4.27}$$

It now suffices to bound the above two terms. The first term can be bounded by the fact

that $1 - x \leq \exp(-x)$ for all $x \in \mathbb{R}$. This yields that

$$r_1 \prod_{k=1}^{T} \Big(1 - \widetilde{C}_{\beta,M}\eta_k\Big) = r_1 \prod_{k=1}^{T} \Big(1 - \widetilde{C}_{\beta,M}C_{\beta,M}/k^{\theta}\Big) \leq r_1 \exp\Big(-\bar{C}_{\beta,M}\sum_{k=1}^{T}\frac{1}{k^{\theta}}\Big), \tag{4.28}$$

where $\bar{C}_{\beta,M} = \widetilde{C}_{\beta,M}C_{\beta,M} = \dfrac{128M^4\beta^2}{(128M^4+\beta^2)^2}$. Using part (i) in Lemma 4.3 gives that

$$\begin{aligned} r_1 \exp\Big(-\bar{C}_{\beta,M}\sum_{k=1}^{T}\frac{1}{k^{\theta}}\Big) &\leq r_1 \exp\left(\frac{\bar{C}_{\beta,M}}{1-\theta}\big[1-(T+1)^{1-\theta}\big]\right) \\ &= r_1 \exp\Big(\frac{\bar{C}_{\beta,M}}{1-\theta}\Big)\exp\Big(-\frac{\bar{C}_{\beta,M}}{1-\theta}(T+1)^{1-\theta}\Big). \end{aligned}$$

Furthermore, we can apply part (iii) in Lemma 4.3. Note that $b = \frac{\theta}{1-\theta}$, $x = (T+1)^{1-\theta}$and $c = \frac{\bar{C}_{\beta,M}}{1-\theta}$. This gives that

$$\exp\left(-\frac{\bar{C}_{\beta,M}}{1-\theta}(T+1)^{1-\theta}\right) \leq \left(\frac{\theta}{\bar{C}_{\beta,M}e}\right)^{\frac{\theta}{1-\theta}}(T+1)^{-\theta}.$$

Combining the above two inequalities back into (4.28), we have

$$r_1 \prod_{k=1}^{T} \Big(1 - \widetilde{C}_{\beta,M}\eta_k\Big) \leq r_1 \exp\Big(\frac{\bar{C}_{\beta,M}}{1-\theta}\Big)\Big(\frac{\theta}{\bar{C}_{\beta,M}e}\Big)^{\frac{\theta}{1-\theta}}T^{-\theta}. \tag{4.29}$$

We can proceed in a similar manner to bound the second term on the right hand side of (4.27):

$$\begin{aligned} \sum_{k=1}^{T-1}\prod_{i=k+1}^{T}\Big(1-\widetilde{C}_{\beta,M}\eta_i\Big)\eta_k^2 &= C_{\beta,M}^2\sum_{k=1}^{T-1}\frac{1}{k^{2\theta}}\prod_{i=k+1}^{T}\left(1-\frac{\bar{C}_{\beta,M}}{i^{\theta}}\right) \\ &\leq C_{\beta,M}^2\sum_{k=1}^{T-1}\frac{1}{k^{2\theta}}\exp\Big(-\bar{C}_{\beta,M}\sum_{i=k+1}^{T}\frac{1}{i^{\theta}}\Big). \end{aligned} \tag{4.30}$$

Applying Lemma 4.3 (ii) with $\nu = \bar{C}_{\beta,M}$ and $\alpha = \theta$ gives that the above is bounded by

$$\sum_{k=1}^{T-1} \frac{1}{k^{2\theta}} \exp\Big(- \bar{C}_{\beta,M} \sum_{i=k+1}^{T} \frac{1}{i^{\theta}}\Big) \leq \frac{9T^{1-\theta}}{(1-\theta)(2^{1-\theta})} \exp\Big(- \frac{\bar{C}_{\beta,M}(1-2^{\theta-1})}{1-\theta}(T+1)^{1-\theta}\Big) + \frac{18}{\bar{C}_{\beta,M}T^{\theta}}. \tag{4.31}$$

We can apply Lemma 4.3 (iii) with $b = \frac{1}{1-\theta}, x = (T+1)^{1-\theta}$ and $c = \frac{\bar{C}_{\beta,M}(1-2^{\theta-1})}{1-\theta}$ to (4.31) yields that

$$\exp\Big(- \frac{\bar{C}_{\beta,M}(1-2^{\theta-1})}{1-\theta}(T+1)^{1-\theta}\Big) \leq \Big(\frac{1}{\bar{C}_{\beta,M}(1-2^{\theta-1})e}\Big)^{\frac{1}{1-\theta}}(T+1)^{-1}. \tag{4.32}$$

Substituting (4.31) and (4.32) back into (4.30), we have

$$2\sigma_*^2 \sum_{k=1}^{T-1} \prod_{i=k+1}^{T} \Big(1 - \widetilde{C}_{\beta,M}\,\eta_i\Big)\,\eta_k^2 \leq 2\sigma_*^2 C_{\beta,M}^2 \Big[\frac{9}{(1-\theta)2^{1-\theta}}\Big(\frac{1}{\bar{C}_{\beta,M}(1-2^{\theta-1})e}\Big)^{\frac{1}{1-\theta}} + \frac{18}{\bar{C}_{\beta,M}}\Big]T^{-\theta}. \tag{4.33}$$

The last term on the righthand side of (4.27) can be bounded by: $2\sigma_*^2\eta_T^2 \leq 2\sigma_*^2 C_{\beta,M}^2 T^{-\theta}$. The combination of (4.29) and (4.33), gives the desired result. □

The theorem above shows that with polynomial decaying step sizes in the form of $\eta_t = \mathcal{O}(t^{-\theta})$ for $\theta \in (0,1)$, SPAM achieves the convergence rate of $\mathcal{O}(T^{-\theta})$. We can improve the above result when $\theta = 1$.

Theorem 4.2. *Under the Assumptions of 4.1 and 4.2, and choosing step sizes* $\{\eta_t = [\widetilde{C}_{\beta,M}(t+1)]^{-1} : t \in \mathbb{N}\}$, *the algorithm SPAM achieves the following:*

$$\mathbb{E}[\|\mathbf{w}_{T+1} - \mathbf{w}^*\|^2] \leq \big(t_0 \mathbb{E}[\|\mathbf{w}_{t_0} - \mathbf{w}^*\|^2]\big)\frac{1}{T} + \frac{4\sigma_*^2}{\widetilde{C}_{\beta,M}^2}\frac{\log T}{T}.$$

where $t_0 = \max\Big(2, \Big\lceil 1 + \frac{(128M^4+\beta^2)^2}{128M^4\beta^2}\Big\rceil\Big)$.

Proof. The condition that $t \geq t_0$ is imposed so that the assumption in part (ii) of Lemma 4.2 that $\eta_t = \Big[\widetilde{C}_{\beta,M}(t+1)\Big]^{-1} \leq C_{\beta,M}$ is guaranteed. As from before, letting $r_t = \mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2]$

we have

$$r_{t+1} \leq \Big(1 - \widetilde{C}_{\beta,M}\,\eta_t\Big) r_t + 2\sigma_*^2 \eta_t^2. \tag{4.34}$$

Then, this gives

$$r_{T+1} \leq r_{t_0} \prod_{k=t_0}^{T} \Big(1 - \widetilde{C}_{\beta,M}\eta_k\Big) + 2\sigma_*^2\eta_T^2 + 2\sigma_*^2 \sum_{k=t_0}^{T-1} \prod_{i=k+1}^{T} \Big(1 - \widetilde{C}_{\beta,M}\eta_i\Big)\,\eta_k^2. \tag{4.35}$$

It now suffices to bound the above two terms. We can estimate the first term as follows:

$$r_{t_0} \prod_{k=t_0}^{T} (1 - \widetilde{C}_{\beta,M}\eta_k) = r_{t_0} \prod_{k=t_0}^{T} \frac{k}{k+1} = \frac{t_0 r_{t_0}}{T+1} \leq \frac{t_0 r_{t_0}}{T}.$$

The second term of (4.35) can be bounded as $2\sigma_*^2\eta_T = \frac{2\sigma_*^2}{\widetilde{C}_{\beta,M}^2(T+1)^2} \leq \frac{2\sigma_*^2}{\widetilde{C}_{\beta,M}^2 T}$. Finally, to bound the third term, we can do the following:

$$\begin{aligned}
\sum_{k=t_0}^{T-1} \prod_{i=k+1}^{T} (1 - \widetilde{C}_{\beta,M}\eta_i)\eta_k^2 &= \widetilde{C}_{\beta,M}^{-2} \sum_{k=t_0}^{T-1} \prod_{i=k+1}^{T-1} \Big(1 - \frac{1}{i+1}\Big) \frac{1}{(k+1)^2} \\
&= \widetilde{C}_{\beta,M}^{-2} \frac{1}{T} \sum_{k=t_0}^{T-1} \frac{1}{k+1} \\
&\leq \widetilde{C}_{\beta,M}^{-2} \frac{\log(T-1) - \log t_0}{T} \leq \widetilde{C}_{\beta,M}^{-2} \frac{\log T}{T}.
\end{aligned}$$

Combining all the above estimations gives the desired result. □

The theorem above shows that SPAM can achieve a convergence of $\mathcal{O}(1/T)$ up to a logarithmic term. This matches the optimal rate of standard stochastic gradient descent [50, 83, 93]. However, it is important to consider a bound on the term $\mathbb{E}[\|\mathbf{w}_{t_0} - \mathbf{w}^*\|^2]$. For $\eta_t = [\widetilde{C}_{\beta,M}(t+1)]^{-1}$ from part (i) of Lemma 4.2, we have, for any $t \in \mathbb{N}$, we can estimate it as follows:

$$\begin{aligned}
\mathbb{E}[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2] &\leq \frac{1 + 128M^4\eta_t^2}{(1+\eta_t\beta)^2} \mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2] + 2\sigma_*^2\eta_t^2 \\
&\leq (1 + 128M^4\eta_t^2)\mathbb{E}[\|\mathbf{w}_t - \mathbf{w}^*\|^2] + 2\sigma_*^2\eta_t^2.
\end{aligned}$$

Data	Name	# Inst	# Feat	Data	Name	# Inst	# Feat
1	diabetes	768	8	7	mnist	60,000	780
2	fourclass	862	2	8	acoustic	78,823	50
3	german	1000	24	9	ijcnn1	141,691	22
4	splice	3175	60	10	covtype	581,012	54
5	usps	9,298	256	11	sector	9,619	55,197
6	a9a	32,561	123	12	news20	15,935	62,061

Table 4.1: Basic information about the datasets.

Therefore,

$$\begin{aligned}\mathbb{E}[\|\mathbf{w}_{t_0} - \mathbf{w}^*\|^2] &\leq \prod_{k=1}^{t_0-1}(1 + 128M^4\eta_k^2) + 2\sigma_*^2 \sum_{k=1}^{t_0-1}\prod_{j=k}^{t_0-1}(1 + 128M^4\eta_j^2)\eta_k^2 \\ &\leq \Big(\prod_{k=1}^{t_0-1}(1 + 128M^4\eta_k^2)\Big)\Big(1 + 2\sigma_*^2\sum_{k=1}^{t_0-1}\eta_k^2\Big).\end{aligned}$$

Observe that

$$\prod_{k=1}^{t_0-1}(1 + 128M^4\eta_k^2) \leq \exp\Big(\frac{128M^4}{\widetilde{C}_{\beta,M}^2}\sum_{k=1}^{t_0-1}(k+1)^{-2}\Big) \leq \exp\left(\frac{128M^4}{\widetilde{C}_{\beta,M}^2}\right),$$

and $2\sigma_*^2\sum_{k=1}^{t_0-1}\eta_k^2 = \frac{2\sigma_*^2}{\widetilde{C}_{\beta,M}^2}\sum_{k=1}^{t_0-1}(k+1)^{-2} \leq \frac{2\sigma_*^2}{\widetilde{C}_{\beta,M}^2}$. This results in the following bound based on β and M:

$$\mathbb{E}[\|\mathbf{w}_{t_0} - \mathbf{w}^*\|^2] \leq \frac{2\sigma_*^2}{\widetilde{C}_{\beta,M}^2} + \exp\left(\frac{128M^4}{\widetilde{C}_{\beta,M}^2}\right).$$

4.3 Experiments

In this section, we will verify the theoretical results of SPAM by comparing it with existing algorithms for AUC optimization. In particular, we will use two variations of SPAM with different regularizers: SPAM-L^2 denotes SPAM with the Frobenius norm, i.e., $\Omega(\mathbf{w}) = \frac{\beta}{2}\|\mathbf{w}\|^2$, and SPAM-NET denotes SPAM with the elastic net norm [121], i.e., $\Omega(\mathbf{w}) = \frac{\beta}{2}\|\mathbf{w}\|^2 + \beta_1\|\mathbf{w}\|_1$. In the first case, the solution to the proximal step is straight forward with the Frobenius norm. In the second version, the proximal step can be formulated as

$$\arg\min_{\mathbf{w}}\left\{\frac{1}{2}\left\|\mathbf{w} - \frac{\hat{\mathbf{w}}_{t+1}}{\eta_t\beta + 1}\right\|^2 + \frac{\eta_t\beta_1}{\eta_t\beta+1}\|\mathbf{w}\|_1\right\}.$$

Data	SPAM-L^2	SPAM-NET	regSOLAM	OPAUC	OAM_{seq}	OAM_{gra}	B-LS-SVM
1	.8272±.0277	.8085±.0431	.8128±.0304	.8309±.0350	.8264±.0367	.8262±.0338	.8325±.0329
2	.8210±.0203	.8211±.0205	.8213±.0209	.8310±.0251	.8306±.0247	.8295±.0251	.8309±.0309
3	.7942±.0388	.7937±.0386	.7778±.0373	.7978±.0347	.7747±.0411	.7723±.0358	.7994±.0343
4	.9263±.0091	.9267±.0090	.9246±.0087	.9232±.0099	.8594±.0194	.8864±.0166	.9245±.0092
5	.9868±.0032	.9855±.0029	.9822±.0036	.9620±.0040	.9310±.0159	.9348±.0122	.9634±.0045
6	.8998±.0046	.8980±.0047	.8966±.0043	.9002±.0047	.8420±.0174	.8571±.0173	.8982±.0028
7	.9254±.0025	.9132±.0026	.9118±.0029	.9242±.0021	.8615±.0087	.8643±.0112	.9336±.0025
8	.8120±.0030	.8109±.0028	.8099±.0036	.8192±.0032	.7113±.0590	.7711±.0217	.8210±.0033
9	.9174±.0024	.9155±.0024	.9129±.0030	.9269±.0021	.9209±.0079	.9100±.0092	.9320±.0037
10	.9504±.0011	.9508±.0011	.9503±.0012	.8244±.0014	.7361±.0317	.7403±.0289	.8222±.0014
11	.8768±.0126	.9077±.0104	.8767±.0129	.9292±.0081	.9163±.0087	.9043±.0100	-
12	.8708±.0069	.8704± .0070	.8712±.0073	.8871±.0083	.8543±.0099	.8346±.0094	-

Table 4.2: Comparison of the testing AUC values (mean±std.). To accelerate the experiments, the values for OPAUC, OAMseq, OAMgra, and B-LS-SVM were taken from [34].

The solution to this step is the soft-thresholding operator [75].

We will compare SPAM to the same algorithms that were compared against regSOLAM. Table 4.3 summarizes the details of each of the data sets as from Chapter 3. We used 80% of the data for training and the remaining 20% for testing. The average AUC score and standard deviation results are based on 20 runs for each dataset. The proper parameters for each data set was determined by 5-fold cross validation on the training sets to determine the parameter $\beta \in 10^{[-5:5]}$ for SPAM-L^2 and $\beta_1 \in 10^{[-5:5]}$ for SPAM-NET. The experiments were conducted with MATLAB.

The classification performance on each data set is summarized in Table 4.3. SPAM-L^2 and SPAM-NET both achieve a comparable performance as the other state of the art AUC maximization algorithms in both the online and batch settings validating the proposed methods. It is important to note that the data set `sector` shows the advantage of using elastic net with the ℓ_1 term since the ℓ_1 norm can be used to detect noise in the data. The CPU running time of SPAM-L^2 versus regSOLAM is shown in Figure 4.2 on various data sets demonstrating the main advantage of SPAM's running time efficiency. The faster convergence rate of SPAM over regSOLAM is validated by these figures along with the per-iteration running time is linear in data dimension. The results shown here demonstrates that SPAM maintains a competitive performance while achieving a faster rate of performance.

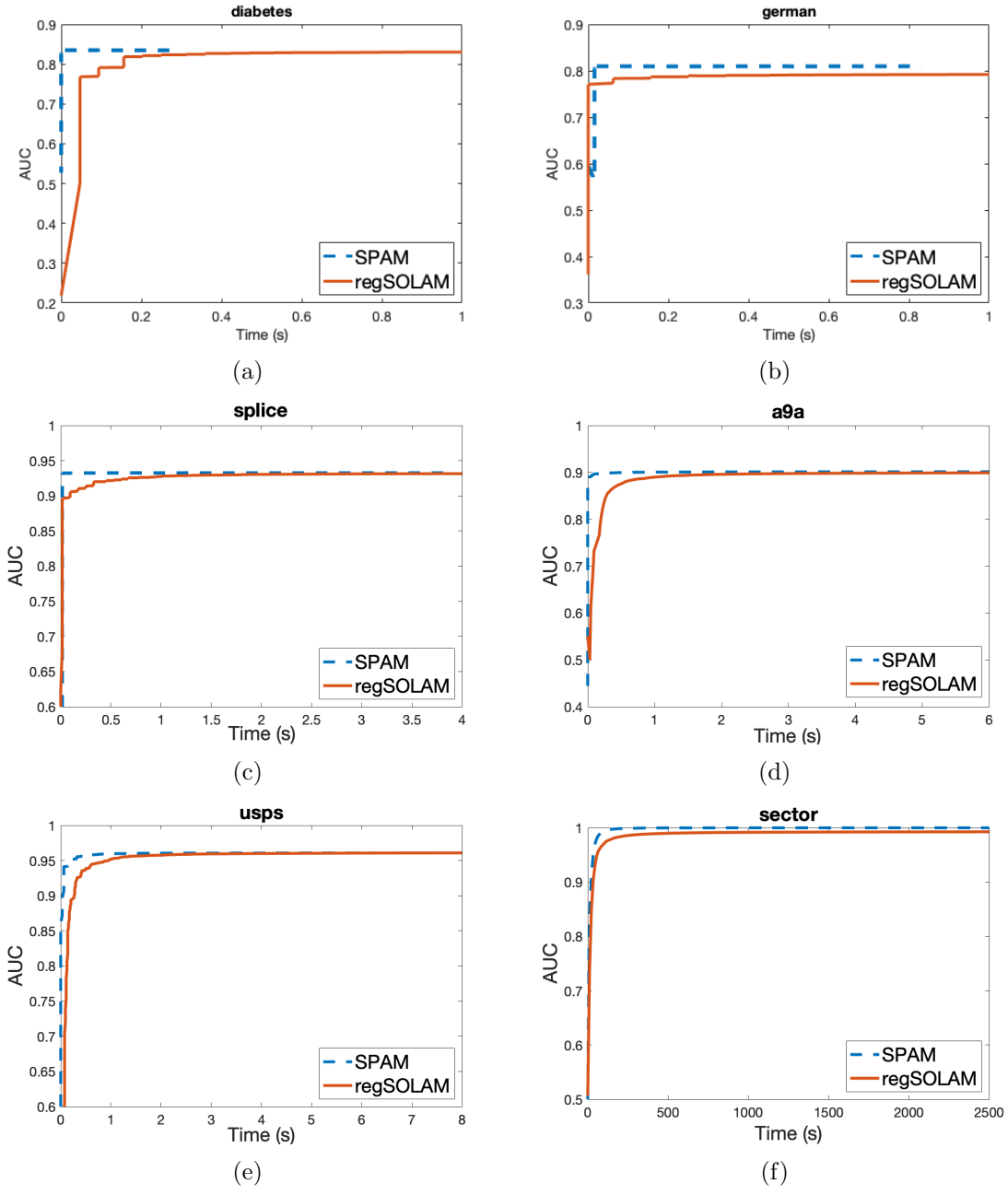


Figure 4.2: Comparison of SPAM vs. regSOLAM for AUC vs. iteration count.

Chapter 5

Stochastic Primal-Dual AUC Maximization

In the previous two chapters we introduced two novel online algorithms for AUC maximization. In this chapter, we will introduce a unique batch learning algorithm with a linear convergence rate. To accomplish this, we will compromise by increasing the per-iteration cost to solve the AUC optimization problem in (3.6). Recall from chapter 2 the empirical minimization (ERM) problem for AUC optimization studied in [34, 118]:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{n_+ n_-} \sum_{i=1}^{n} \sum_{j=1}^{n} (1 - \mathbf{w}^\top (\mathbf{x}_i - \mathbf{x}_j))^2 \mathbb{I}_{[y_i=1 \wedge y_j=-1]} + \frac{\lambda}{2} \|\mathbf{w}\|^2, \tag{5.1}$$

where n_+ and n_- denote the numbers of instances in the positive and negative classes, respectively. In this chapter, we will focus on the ERM problem and design a batch learning algorithm with a linear convergence rate.

5.1 Method Formulation

To obtain a better understanding of the algorithm to be outlined, consider the general optimization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x}) + g(\mathbf{x}), \tag{5.2}$$

where f_i is a convex loss function and g is a regularization term. The formulation in (5.2) has many examples of well established classification and regression problems based upon the choice of the convex loss function and regularizer. More background can be found in [33].

An innovative approach to problem (5.2) is to reformulate it as a convex-concave saddle point problem. This can done by making use of the fenchel conjugate [29] to reformulate the loss function:

$$f_i(\mathbf{x}) = \sup_{y_i \in \mathbb{R}} \{y_i \mathbf{x} - f_i^*(y_i)\}, \tag{5.3}$$

where $f_i^*(y_i) = \sup_{\alpha\in\mathbb{R}}\{\alpha y_i - f_i(\alpha)\}$. Note that $f_i^{**} = f_i$ when f_i is both closed and convex [6]. By applying equation (5.3) to problem (5.2) this results in a convex-concave saddle point problem as desired:

$$\min_{\mathbf{x}\in\mathbb{R}^d}\max_{y\in\mathbb{R}^n}\frac{1}{n}\sum_{i=1}^{n}(y_i\langle a_i,\mathbf{x}\rangle - f_i^*(y_i)) + g(\mathbf{x}), \tag{5.4}$$

where $a_i \in \mathbb{R}^d$ are the feature vectors. Solving the above problem can be done by alternating between maximizing with respect to y and minimizing with respect to $\mathbf{x}$. However, the dual variable y has n coordinates, so the maximization step would be of order $\mathcal{O}(nd)$ resulting in this step being very computationally expensive. To overcome this issue, a randomly selected coordinate of y is only used to maximize f. This results in a computational cost of order $\mathcal{O}(d)$ for each iteration. A natural extension of this idea is to use a mini-batch of coordinates of y.

The same innovative approach as explained above can be applied to the problem of AUC maximization. Assuming that ρ is now a uniform distribution and by denoting $\mathbb{N}_n = \{1, 2, \ldots, n\}$ for any $n \in \mathbb{N}$, we can reformulate (5.1) as:

$$\min_{\substack{\mathbf{w}\in\mathbb{R}^d\\(a,b)\in\mathbb{R}^2}}\max_{\alpha\in\mathbb{R}}\frac{1}{n}\sum_{i\in\mathbb{N}_n}F(\mathbf{w},a,b,\alpha,z_i). \tag{5.5}$$

Before the previous ideas can be considered, a few modifications need to be applied. By using the definition of F, we obtain the following convex-concave problem:

$$\begin{aligned}\min_{\mathbf{w},a,b}\max_{\alpha}\Big\{&\frac{1}{n_+}\sum_{i\in\mathbb{N}_n}(\mathbf{w}^\top\mathbf{x}_i - a)^2\mathbb{I}_{y_i=1} + \frac{1}{n_-}\sum_{i\in\mathbb{N}_n}(\mathbf{w}^\top\mathbf{x}_i - b)^2\mathbb{I}_{y_i=-1}\\ &+ 2(1+\alpha)\mathbf{w}^\top\Big[\frac{1}{n_-}\sum_{i\in\mathbb{N}_n}\mathbf{x}_i\mathbb{I}_{y_i=-1} - \frac{1}{n_+}\sum_{i\in\mathbb{N}_n}\mathbf{x}_i\mathbb{I}_{y_i=1}\Big] - \alpha^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2\Big\}.\end{aligned} \tag{5.6}$$

If we replace the ℓ_2 regularizer by the restriction of $\mathbf{w}$ to a ball of radius R, the above formulation is equivalent to the saddle point formulation (5.5). For this special case, we will develop in this section a stochastic primal-dual algorithm for AUC optimization (5.5) which is able to converge with a linear convergence rate.

Before we modify the above formulation, it is beneficial to introduce some helpful notation. For the discrete formulation, let $p = \frac{n_+}{n}$. We will define the class means as $\mathbf{m}_+$

and $\mathbf{m}_-$ where $\mathbf{m}_+ = \frac{1}{n_+}\sum_{i\in\mathbb{N}_n}\mathbf{x}_i\mathbb{I}_{y_i=1}$ and $\mathbf{m}_- = \frac{1}{n_-}\sum_{i\in\mathbb{N}_n}\mathbf{x}_i\mathbb{I}_{y_i=-1}$, and let $\mathbf{b} = \mathbf{m}_- - \mathbf{m}_+$. It will also be helpful for any $i \in \mathbb{N}_n$ to apply the following transformation to the data:

$$\bar{x}_i = \frac{\mathbf{x}_i - m_+}{\sqrt{2p}} \quad \text{if } y_i = 1, \qquad \bar{x}_i = \frac{\mathbf{x}_i - m_-}{\sqrt{2(1-p)}} \quad \text{if } y_i = -1. \tag{5.7}$$

Letting $g(\mathbf{w}) = \frac{1}{2}|\mathbf{b}^\top\mathbf{w}|^2 + \mathbf{b}^\top\mathbf{w} + \frac{\lambda}{2}\|\mathbf{w}\|^2$ satisfies the assumption that g is a λ-strongly convex function. Now we can present the following reformulation of (5.6) of which will be the basis for the stochastic primal-dual algorithm for AUC maximization.

Proposition 5.1.1. *Formulation* (5.6) *is equivalent to*

$$\min_{\mathbf{w}}\max_{\beta}\Big\{\frac{1}{n}\sum_{i\in\mathbb{N}_n}\beta_i\mathbf{w}^\top\bar{x}_i - \frac{\|\beta\|^2}{2} + g(\mathbf{w})\Big\}, \tag{5.8}$$

where $g : \mathbb{R}^d \to \mathbb{R}$ *is defined, for any* $\mathbf{w} \in \mathbb{R}^d$*, by* $g(\mathbf{w}) = \frac{|\mathbf{b}^\top\mathbf{w}|^2}{2} + \mathbf{b}^\top\mathbf{w} + \frac{1}{2}\|\mathbf{w}\|^2$.

Proof. The minimization with respect to a, b and α gives that formulation (5.6) is equivalent to

$$\min_{\mathbf{w}}\max_{\alpha}\Big\{\frac{1}{n_+}\sum_{i\in\mathbb{N}_n}(\mathbf{w}^\top(\mathbf{x}_i - \mathbf{m}_+))^2\mathbb{I}_{y_i=1} + \frac{1}{n_-}\sum_{i\in\mathbb{N}_n}(\mathbf{w}^\top(\mathbf{x}_i - \mathbf{m}_-))^2\mathbb{I}_{y_i=-1} + 2\mathbf{b}^\top\mathbf{w} + |\mathbf{b}^\top\mathbf{w}|^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2\Big\}.$$

Substituting the transformation of the data (5.7) into the above equation gives the desired result. □

Using the new formulation of (5.8) this exactly matches the formulation of (5.4) and we can efficiently solve this problem by using the previously discussed method by first maximizing with respect to the dual variable β and then minimizing with respect to the primal variable $\mathbf{w}$. The method is summarized in Algorithm 21. The proposed method in Algorithm 21 is strongly motivated by the stochastic primal-dual algorithm proposed in [116, 114] which focused on Support Vector Machine (SVM) and logistic regression. It is important to discuss the key steps in the algorithm. Maximizing with respect to the primal variable is equivalent to solving for the convex conjugate. A regularization term with parameter σ is introduced to ensure that the new solution of β is closer to the previous solution. The last step is an

Algorithm 21 Stochastic Primal-Dual Algorithm for AUC Maximization (SPDAM)

Choose parameters $\sigma > 0$ and $\tau > 0$
Initialize $\beta^{(0)}$ and $\mathbf{w}^{(0)}$. Let $\bar{\mathbf{w}}^{(0)} = \mathbf{w}^{(0)}$ and $u^{(0)} = \frac{1}{n}\sum_{i\in\mathbb{N}_n} \beta_i^{(0)}\bar{x}_i$.
for $t = 0$ **to** $T-1$ **do**
Uniformly and randomly choose $I \subseteq \mathbb{N}_n$ of size m and execute the following updates:

$$\beta_i^{(t+1)} = \begin{cases} \operatorname{argmax}_{\beta_i\in\mathbb{R}}\{\beta_i\langle\bar{\mathbf{w}}^{(t)}, \mathbf{x}_i\rangle - \frac{|\beta_i|^2}{2} - \frac{|\beta_i-\beta_i^{(t)}|^2}{2\sigma}\} & \text{if } i \in I \\ \beta_i^{(t)} & \text{otherwise.} \end{cases}$$

$$u^{(t+1)} = u^{(t)} + \frac{1}{n}\sum_{i\in I}(\beta_i^{(t+1)} - \beta_i^{(t)})\mathbf{x}_i.$$

$$\bar{u}^{(t+1)} = u^{(t)} + \frac{n}{m}(u^{(t+1)} - u^{(t)}).$$

$$\mathbf{w}^{(t+1)} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d}\{\langle\bar{u}^{(t+1)}, \mathbf{w}\rangle + g(\mathbf{w}) + \frac{\|\mathbf{w}-\mathbf{w}^{(t)}\|^2}{2\tau}\}.$$

$$\bar{\mathbf{w}}^{(t+1)} = \mathbf{w}^{(t+1)} + \theta(\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}).$$

end for

extrapolation step based on Nesterov's acceleration technique to ensure a fast convergence rate [72].

Before we detail the convergence analysis, it is appropriate to make some important comments. The algorithm SPDAM has a faster convergence rate than the methods in the previous two chapters, however, a disadvantage of SPDAM is that it cannot process streaming data. The algorithm requires a sufficient batchsize, the probability of the positive class, as well as the means of both the both positive and negative samples.

5.2 Convergence Analysis

The main novelty of Algorithm 21 is its linear convergence rate. Using the notation that $\kappa = \max\{\|\mathbf{x}_i\| : i \in \mathbb{N}_n\}$, establishing the convergence rate of SPDAM requires the use of the following critical lemma.

Lemma 5.1. *After $t+1$ updates in SPDAM, we have*

$$\begin{aligned}\Big(\frac{1}{m} + \frac{1}{2\sigma m}\Big)\mathbb{E}\Big[\|\beta^{(t+1)} - \beta^*\|^2\Big] &= \Big(\frac{1}{2\sigma m} + \frac{n-m}{nm}\Big)\mathbb{E}\Big[\|\beta^{(t)} - \beta^*\|^2\Big] \\ &\quad - \frac{1}{2\sigma m}\mathbb{E}\big[\|\beta^{(t+1)} - \beta^{(t)}\|^2\big] \\ &\quad + \mathbb{E}\big[\langle\bar{u}^{(t+1)}, \bar{\mathbf{w}}^{(t)} - \mathbf{w}^*\rangle\big], \end{aligned} \tag{5.9}$$

and

$$\Big(\lambda+\frac{1}{2\tau}\Big)\mathbb{E}\big[\|\mathbf{w}^{(t+1)}-\mathbf{w}^*\|^2\big]\leq\frac{1}{2\tau}\mathbb{E}\big[\|\mathbf{w}^{(t)}-\mathbf{w}^*\|^2\big]-\frac{1}{2\tau}\mathbb{E}\big[\|\mathbf{w}^{(t+1)}-\mathbf{w}^{(t)}\|^2\big]$$
$$-\mathbb{E}\big[\langle\bar{u}^{(t+1)},\mathbf{w}^{(t+1)}-\mathbf{w}^*\rangle\big]. \tag{5.10}$$

Proof. We begin by first proving equation (5.9). For any $i\in\mathbb{N}_n$, let $\tilde{\beta}_i$ be defined as

$$\tilde{\beta}_i=\operatorname*{argmax}_{\beta_i\in\mathbb{R}}\big\{\beta_i\langle\bar{\mathbf{w}}^{(t)},\mathbf{x}_i\rangle-\frac{|\beta_i|^2}{2}-\frac{|\beta_i-\beta_i^{(t)}|^2}{2\sigma}\big\}.$$

Hence,

$$\frac{|\beta_i^{(t)}-\beta_i^*|^2}{2\sigma}+\frac{|\beta_i^*|^2}{2}-\beta_i^*\langle\bar{\mathbf{w}}^{(t)},\mathbf{x}_i\rangle=\frac{|\beta_i^{(t)}-\tilde{\beta}_i|^2}{2\sigma}+\frac{|\tilde{\beta}_i|^2}{2}-\tilde{\beta}_i\langle\bar{\mathbf{w}}^{(t)},\mathbf{x}_i\rangle$$
$$+\big(\frac{1}{2}+\frac{1}{2\sigma}\big)|\tilde{\beta}_i-\beta_i^*|^2. \tag{5.11}$$

The saddle point $(\mathbf{w}^*,\beta^*)$ gives by definition that

$$\beta^*=\operatorname*{argmax}_{\beta_i}\left\{\beta_i\langle\mathbf{w}^*,\mathbf{x}_i\rangle-\frac{|\beta_i|^2}{2}\right\}.$$

As a consequence of this, $\tilde{\beta}_i\langle\mathbf{w}^*,x_i\rangle-\frac{|\tilde{\beta}_i|^2}{2}=\beta_i^*\langle\mathbf{w}^*,x_i\rangle-\frac{|\beta_i^*|^2}{2}-\frac{1}{2}|\tilde{\beta}_i-\beta_i^*|^2$ gives that $\frac{|\tilde{\beta}_i|^2}{2}-\frac{|\beta_i^*|^2}{2}=(\tilde{\beta}_i-\beta_i^*)\langle\mathbf{w}^*,x_i\rangle+\frac{1}{2}|\tilde{\beta}_i-\beta_i^*|^2$. Substituting this back into (5.11) gives

$$\frac{|\beta_i^{(t)}-\beta_i^*|^2}{2\sigma}+(\tilde{\beta}_i-\beta_i^*)\langle\bar{\mathbf{w}}^{(t)}-\mathbf{w}^*,x_i\rangle=\frac{|\beta_i^{(t)}-\tilde{\beta}_i|^2}{2\sigma}+\big(1+\frac{1}{2\sigma}\big)|\tilde{\beta}_i-\beta_i^*|^2. \tag{5.12}$$

Now define $\mathcal{F}_t$ to be the sigma field generated by all random variables defined before round t. The expectation conditioned over $\mathcal{F}_t$ gives that

$$\mathbb{E}\big(|\beta_i^{(t)}-\beta_i^{(t+1)}|^2|\mathcal{F}_t\big)=\frac{m}{n}|\tilde{\beta}_i-\beta_i^{(t)}|^2$$
$$\mathbb{E}\big(|\beta_i^{(t+1)}-\beta_i^*|^2|\mathcal{F}_t\big)=\frac{m}{n}|\tilde{\beta}_i-\beta_i^*|^2+\frac{n-m}{n}|\beta_i^{(t)}-\beta_i^*|^2$$
$$\mathbb{E}\big(|\beta_i^{(t+1)}|^2|\mathcal{F}_t\big)=\frac{m}{n}|\tilde{\beta}_i|^2+\frac{n-m}{n}|\beta_i^{(t)}|^2,\quad\mathbb{E}\big(\beta_i^{(t+1)}|\mathcal{F}_t\big)=\frac{m}{n}\tilde{\beta}_i+\frac{n-m}{n}\beta_i^{(t)}.$$

The above equalities can be used to represent terms involving $\tilde{\beta}_i$ by $\beta_i^{(t+1)}$ on the righthand

side of (5.12). Therefore, we have

$$\begin{aligned}\big(\frac{1}{m}+\frac{1}{2\sigma m}\big)\mathbb{E}\big[|\beta_i^{(t+1)}-\beta_i^*|^2|\mathcal{F}_t\big] &= \big(\frac{1}{2\sigma m}+\frac{n-m}{nm}\big)|\beta_i^{(t)}-\beta_i^*|^2-\frac{1}{2\sigma m}\mathbb{E}[\|\beta^{(t+1)}-\beta^{(t)}\|^2]\\ &\quad+\mathbb{E}\big[\langle\bar{\mathbf{w}}^{(t)}-\mathbf{w}^*,\frac{1}{m}(\beta_i^{t+1}-\beta_i^*)+\frac{1}{n}(\beta_i^{(t)}-\beta_i^*)x_i\rangle|\mathcal{F}_t\big]\end{aligned}$$

The summation over $i\in\mathbb{N}_n$ and the fact that $\bar{u}^{(t+1)}=\frac{1}{m}\sum_{i\in\mathbb{N}_n}(\beta_i^{t+1}-\beta_i^*)\mathbf{x}_i+\frac{1}{n}\sum_{i\in\mathbb{N}_n}(\beta^{(t)}-\beta_i^*)\mathbf{x}_i$ yields

$$\begin{aligned}\big(\frac{1}{m}+\frac{1}{2\sigma m}\big)\mathbb{E}\big[|\beta^{(t+1)}-\beta^*|^2|\big] &= \big(\frac{1}{2\sigma m}+\frac{n-m}{nm}\big)\mathbb{E}[\|\beta^{(t)}-\beta^*\|^2]\\ &\quad-\frac{1}{2\sigma m}\mathbb{E}[\|\beta^{(t+1)}-\beta^{(t)}\|^2]\\ &\quad+\mathbb{E}\big[\langle\bar{\mathbf{w}}^{(t)}-\mathbf{w}^*,\bar{u}^{(t+1)}\big].\end{aligned}$$

This concludes the proof of the equality in (5.9).

To prove the inequality (5.10) in Lemma 5.1, we begin by using the definition of $\mathbf{w}^{(t+1)}$ and the λ-strong convexity of g. This yields

$$\begin{aligned}\langle\bar{u}^{(t+1)},\mathbf{w}^*\rangle+g(\mathbf{w}^*)+\frac{\|\mathbf{w}^{(t)}-\mathbf{w}^*\|^2}{2\tau} &\geq \langle\bar{u}^{(t+1)},\mathbf{w}^{(t+1)}\rangle+g(\mathbf{w}^{(t+1)})\\ &\quad+\frac{\|\mathbf{w}^{(t+1)}-\mathbf{w}^{(t)}\|^2}{2\tau}\\ &\quad+\big(\frac{\lambda}{2}+\frac{1}{2\tau}\big)\|\mathbf{w}^{(t+1)}-\mathbf{w}^*\|^2. \qquad (5.13)\end{aligned}$$

For the optimal β_i^*, define $u^*=\frac{1}{n}\sum_{i\in\mathbb{N}_n}\beta_i^*\mathbf{x}_i$. The saddle point $(\mathbf{w}^*,\beta^*)$ gives by definition that

$$\langle u^*,\mathbf{w}^{(t+1)}\rangle+g(\mathbf{w}^{(t+1)})\geq\langle u^*,\mathbf{w}^*\rangle+g(\mathbf{w}^*)+\frac{\lambda}{2}\|\mathbf{w}^{(t+1)}-\mathbf{w}^*\|^2.$$

Combining the above inequality with (5.13) yields that

$$\begin{aligned}\big(\lambda+\frac{1}{2\tau}\big)\|\mathbf{w}^{(t+1)}-\mathbf{w}^*\|^2 &\leq \frac{\|\mathbf{w}^{(t)}-\mathbf{w}^*\|^2}{2\tau}-\frac{\|\mathbf{w}^{(t+1)}-\mathbf{w}^{(t)}\|^2}{2\tau}\\ &\quad-\langle\mathbf{w}^{(t+1)}-\mathbf{w}^*,\bar{u}^{(t+1)}-u^*\rangle.\end{aligned}$$

This completes the proof of the lemma. □

We will now state and proof the main theorem regarding the convergence rate of SPDAM. The use of Lemma 5.1 will be critical in its proof. It is important to note that the method assumes that the number of samples is known, a pre-selected batch size m, the probability of a positive class, and the class means.

Theorem 5.1. *Assume that g is λ-strongly convex. Let $(\mathbf{w}^*, \beta^*)$ be the saddle point of (5.8). If the parameter σ, τ and θ are chosen such that*

$$\sigma = \frac{(n-m) + \sqrt{(n-m)^2 + 4n\kappa^2 m/\lambda}}{8m\kappa^2}, \tau = \frac{1}{4\sigma\kappa^2} \text{ and } \theta = 1 - \frac{\lambda}{\lambda + 2\sigma\kappa^2},$$

then, for any $t \geq 1$, the SPDAM algorithm achieves

$$\begin{aligned}
&\big(\frac{1}{m} + \frac{1}{4\sigma m}\big)\mathbb{E}\big[\|\beta^{(t+1)} - \beta^*\|^2\big] + \big(\lambda + \frac{1}{2\tau}\big)\mathbb{E}\big[\|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|^2\big] + \frac{1}{4\tau}\mathbb{E}\big[\|\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}\|^2\big] \\
&\leq \theta^t\Big[\big(\frac{1}{m} + \frac{1}{2\sigma m}\big)\|\beta^{(0)} - \beta^*\| + \big(\lambda + \frac{1}{2\tau}\big)\|\mathbf{w}^{(0)} - \mathbf{w}^*\|^2\Big]. \qquad (5.14)
\end{aligned}$$

Proof. Combining (5.9) and (5.10) together, this yields

$$\begin{aligned}
&\big(\frac{1}{m} + \frac{1}{2\sigma m}\big)\mathbb{E}\big[\|\beta^{(t+1)} - \beta^*\|^2\big] + \big(\lambda + \frac{1}{2\tau}\big)\mathbb{E}\big[\|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|^2\big] \\
&\qquad \leq \Big(\frac{1}{2\sigma m} + \frac{1}{m} - \frac{1}{n}\Big)\mathbb{E}\big[\|\beta^{(t)} - \beta^*\|\big] + \frac{1}{2\tau}\mathbb{E}\big[\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2\big] \\
&\qquad - \frac{1}{2\sigma m}\mathbb{E}\big[\|\beta^{(t+1)} - \beta^{(t)}\|^2\big] - \frac{1}{2\tau}\mathbb{E}\big[\|\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}\|^2\big] \\
&\qquad + \mathbb{E}\big[\langle u^{(t)} - u^* + \frac{n}{m}(u^{(t+1)} - u^{(t)}), \bar{\mathbf{w}}^{(t)} - \mathbf{w}^{(t+1)}\rangle\big]. \qquad (5.15)
\end{aligned}$$

From the definitions of $u^{(t)}, u^{(t+1)}$ and $\bar{\mathbf{w}}^{(t)}$, gives that

$$\begin{aligned}
&\langle u^{(t)} - u^* + \frac{n}{m}(u^{(t+1)} - u^{(t)}), \bar{\mathbf{w}}^{(t)} - \mathbf{w}^{(t+1)}\rangle \\
&= \theta\langle u^{(t)} - u^*, \mathbf{w}^{(t)} - \mathbf{w}^{(t-1)}\rangle - \langle u^{(t+1)} - u^*, \mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}\rangle \\
&+ \frac{n\theta}{m}\langle u^{(t+1)} - u^{(t)}, \mathbf{w}^{(t)} - \mathbf{w}^{(t-1)}\rangle + \frac{n-m}{m}\langle u^{(t+1)} - u^{(t)}, \mathbf{w}^{(t)} - \mathbf{w}^{(t+1)}\rangle.
\end{aligned}$$

From the Cauchy-Schwartz inequality, letting $X = [x_1, x_2, \ldots, x_n]^\top$ and the fact that $\kappa^2\sigma =$

$\frac{1}{4\tau}$ gives

$$
\begin{aligned}
n\langle u^{(t+1)}-u^{(t)}, \mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\rangle &= \langle \sum_{i\in K}(\beta_i^{(t+1)}-\beta_i^{(t)})x_i, \mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\rangle \\
&\leq \frac{\|\beta^{(t+1)}-\beta^{(t)}\|\kappa^2 m}{4\sigma\kappa^2 m} + \frac{\|\mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\|^2 m}{4\tau} \\
&= \frac{\|\beta^{(t+1)}-\beta^{(t)}\|}{4\sigma} + \frac{\|\mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\|^2 m}{4\tau}. \qquad (5.16)
\end{aligned}
$$

Similarly, $n\langle u^{(t+1)}-u^{(t)}, \mathbf{w}^{(t)}-\mathbf{w}^{(t+1)}\rangle \leq \frac{\|\beta^{(t+1)}-\beta^{(t)}\|}{4\sigma} + \frac{\|\mathbf{w}^{(t+1)}-\mathbf{w}^{(t)}\|^2 m}{4\tau}$. Substituting these estimations into (5.17) gives that

$$
\begin{aligned}
&\big(\frac{1}{m}+\frac{1}{2\sigma m}\big)\mathbb{E}\big[\|\beta^{(t+1)}-\beta^*\|^2\big] + \big(\lambda+\frac{1}{2\tau}\big)\mathbb{E}\big[\|\mathbf{w}^{(t+1)}-\mathbf{w}^*\|^2\big] \\
&+ \frac{1}{2\tau}\mathbb{E}\big[\|\mathbf{w}^{(t+1)}-\mathbf{w}^{(t)}\|^2\big] + \mathbb{E}\big[\langle u^{t+1}-u^*, \mathbf{w}^{(t+1)}-\mathbf{w}^{(t)}\rangle\big] \\
&\leq \Big(\frac{1}{m}+\frac{1}{2\sigma m}-\frac{1}{n}\Big)\mathbb{E}\big[\|\beta^{(t)}-\beta^*\|\big] + \frac{1}{2\tau}\mathbb{E}\big[\|\mathbf{w}^{(t)}-\mathbf{w}^*\|^2\big] \\
&+ \theta\Big(\frac{1}{2\tau}\mathbb{E}\big[\|\mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\|^2\big] + \mathbb{E}\big[\langle u^t-u^*, \mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\rangle\big]\Big). \qquad (5.17)
\end{aligned}
$$

Assuming that $\sigma = \frac{(n-m)+\sqrt{(n-m)^2+4n\kappa^2 m/\lambda}}{8m\kappa^2}$, $\tau = \frac{1}{4\sigma\kappa^2}$ and $\theta = 1-\frac{\lambda}{\lambda+2\sigma\kappa^2}$, this yields

$$
\Big(\frac{1}{m}+\frac{1}{2\sigma m}-\frac{1}{n}\Big) = \theta(1+\frac{1}{2\sigma}) \quad \text{and} \quad \frac{1}{2\tau} = \theta(\lambda+\frac{1}{2\tau}). \qquad (5.18)
$$

Defining $\triangle_t = \big(\frac{1}{m}+\frac{1}{2\sigma m}\big)\mathbb{E}\big[\|\beta^{(t)}-\beta^*\|^2\big] + \big(\lambda+\frac{1}{2\tau}\big)\mathbb{E}\big[\|\mathbf{w}^{(t)}-\mathbf{w}^*\|^2\big] + \frac{1}{2\tau}\mathbb{E}\big[\|\mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\|^2\big] + \mathbb{E}\big[\langle u^t-u^*, \mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\rangle\big]$, from (5.18) and (5.17) we have that $\triangle_{t+1} \leq \theta\triangle_t$. Using the exact same argument as in (5.16), there holds

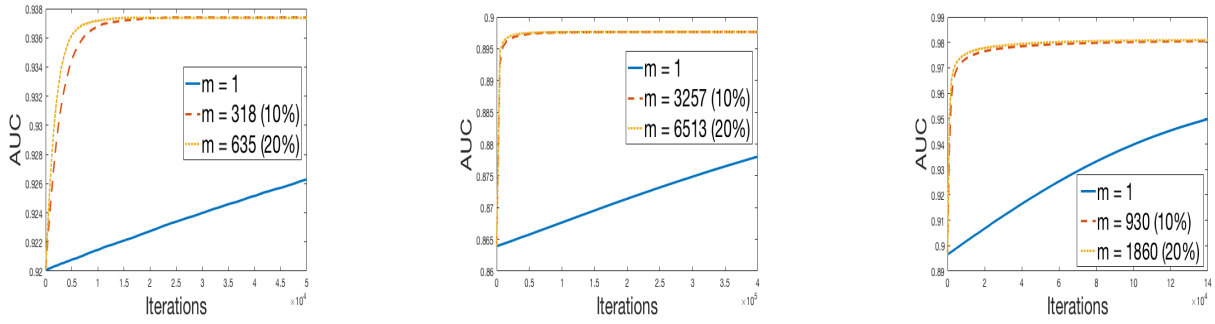
$$
\begin{aligned}
&|\langle u^t-u^*, \mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\rangle| \leq \frac{\|\mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\|^2}{4\tau} + \frac{\|(\beta^{(t)}-\beta^*)^\top X\|^2}{n^2/\tau} \\
&\leq \frac{\|\mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\|^2}{4\tau} + \frac{\|(\beta^{(t)}-\beta^*)^\top X\|}{4n\sigma\kappa^2} \leq \frac{\|\mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\|^2}{4\tau} + \frac{\|\beta^{(t)}-\beta^*\|}{4n\sigma}. \qquad (5.19)
\end{aligned}
$$

Now, for any t, we have that

$$
\begin{aligned}
\triangle_t \geq\ & \big(\frac{1}{m}+\frac{1}{4\sigma m}\big)\mathbb{E}\big[\|\beta^{(t)}-\beta^*\|^2\big] + \big(\lambda+\frac{1}{2\tau}\big)\mathbb{E}\big[\|\mathbf{w}^{(t)}-\mathbf{w}^*\|^2\big] \\
&+ \frac{1}{4\tau}\mathbb{E}\big[\|\mathbf{w}^{(t)}-\mathbf{w}^{(t-1)}\|^2\big] \geq 0. \qquad (5.20)
\end{aligned}
$$

Datasets	# Inst	# Feat	Datasets	# Inst	# Feat
diabetes	768	8	mnist	60,000	780
fourclass	862	2	acoustic	78,823	50
german	1000	24	ijcnn1	141,691	22
splice	3175	60	covtype	581,012	54
usps	9,298	256	sector	9,619	55,197
a9a	32,561	123			

Table 5.1: Basic information about the datasets.



(a) splice (b) a9a (c) usps

Figure 5.1: AUC vs. Iteration curves of SPDAM algorithm for various batch sizes. The batch size is a percentage of the number of samples.

Finally,

$$\triangle_{t+1} \leq \theta^t \triangle_0 = \theta^t \big((\frac{1}{m} + \frac{1}{2\sigma m}) \|\beta^{(0)} - \beta^*\| + (\lambda + \frac{1}{2\tau}) \|\mathbf{w}^{(0)} - \mathbf{w}^*\|^2 \big).$$

Combining this with the inequality (5.20) gives the desired result. □

5.3 Experiments

In this last section, we will demonstrate the effectiveness of SPDAM against regSOLAM along with the previous state-of-the-art AUC maximization algorithms from the previous chapters. We will use the same conditions as previously used. In other words, 80% of the data was used for training with the remaining used for testing. The parameter λ was determined to be tuned between $10^{[-5:1]}$. A batch size of 10% was used for m.

A summary of the performances of SPDAM against regSOLAM and other state-of-the-art AUC optimization methods is given in Table 5.2. Both SPDAM and regSOLAM achieve

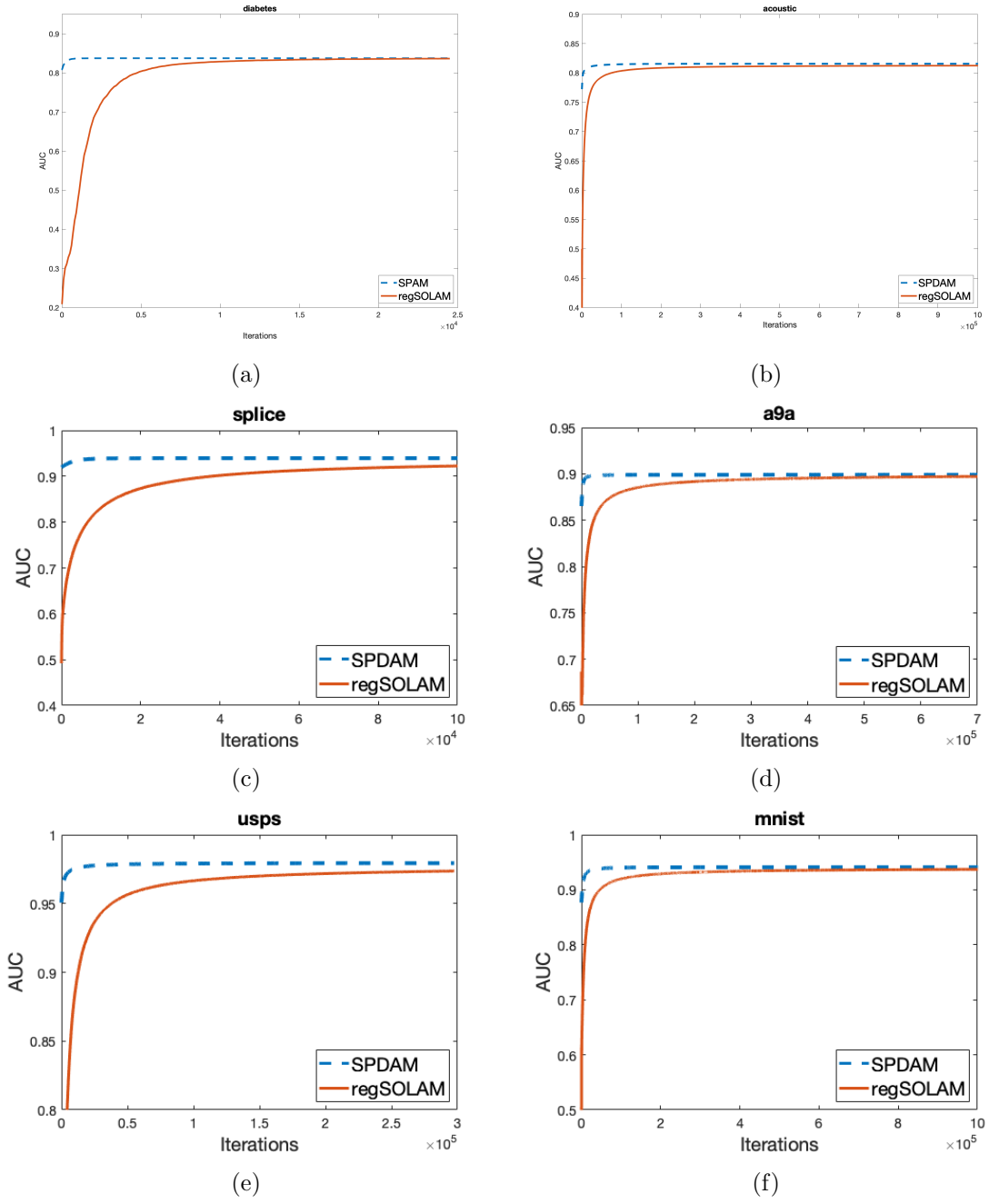


Figure 5.2: Comparison of SPDAM vs. regSOLAM for AUC vs. iteration count. For SPDAM, 10% of the data was chosen for a batch size.

Datasets	SPDAM	regSOLAM	OPAUC	OAM_{seq}	OAM_{gra}	B-SVM-OR	B-LS-SVM
diabetes	.8275±.0302	.8140±.0330	.8309±.0350	.8264±.0367	.8262±.0338	.8326±.0328	.8325±.0329
fourclass	.8223±.0275	.8222±.0276	.8310±.0251	.8306±.0247	.8295±.0251	.8305±.0311	.8309±.0309
german	.7959±.0265	.7830±.0247	.7978±.0347	.7747±.0411	.7723±.0358	.7935±.0348	.7994±.0343
splice	.9227±.0128	.9237±.0090	.9232±.0099	.8594±.0194	.8864±.0166	.9239±.0089	.9245±.0092
usps	.9854±.0019	.9848±.0021	.9620±.0040	.9310±.0159	.9348±.0122	.9630±.0047	.9634±.0045
a9a	.8967±.0032	.8970±.0048	.9002±.0047	.8420±.0174	.8571±.0173	.9009±.0036	.8982±.0028
mnist	.9552±.0011	.9599±.0014	.9242±.0021	.8615±.0087	.8643±.0112	.9340±.0020	.9336±.0025
acoustic	.8119±.0039	.8114±.0035	.8192±.0032	.7113±.0590	.7711±.0217	.8262±.0032	.8210±.0033
ijcnn1	.9132±.0016	.9108±.0030	.9269±.0021	.9209±.0079	.9100±.0092	.9337±.0024	.9320±.0037
covtype	.9409±.0011	.9332±.0020	.8244±.0014	.7361±.0317	.7403±.0289	.8248±.0013	.8222±.0014
sector	.9406±.0062	.9734±.0036	.9292±.0081	.9163±.0087	.9043±.0100	-	-

Table 5.2: Comparison of the testing AUC values (mean±std.) on the evaluated datasets. To accelerate the experiments, the value for *sector* was determined after five runs instead of 25 for the other data sets. The results of OPAUC, OAMseq, OAMgra, online Uni-Exp, B-SVM-OR and B-LS-SVM were taken from [34] as in Chapters 3 and 4.

similar performances as the other state-of-the-art online and offline methods. It should be noted that in some cases SPDAM and regSOLAM outperform the other learning algorithms. For example, there is is notable improvement in the data sets *mnist* and *covtype*. This performance increase could be due to the fact that the data sets are randomly partitioned into two classes making the value of p abnormally high resulting in a higher AUC score.

The main advantage of SPDAM, however, is the running time performance. The theoretical linear convergence rate of SPDAM shows that it should be faster than regSOLAM's $\mathcal{O}(1/\sqrt{T})$ convergence. In Figure 5.2, we present plots of AUC vs. Iterations for SPDAM against regSOLAM over 6 datasets. These figures confirm our hypothesis that SPDAM is faster than regSOLAM while maintaining a competitive performance. It is important to note a disadvantage of SPDAM. To obtain the desired convergence rate, a sufficiently large batch size (m) needs to be selected. The result in Theorem 5.1 shows that the value of θ needs to be set small enough to ensure a fast convergence. For this to occur, m needs to be large enough. Various batch sizes were selected in Figure 5.1 that demonstrates this. A batch size of 10% is sufficient so that SPDAM is faster than regSOLAM. It is important to note that for a batch size $m = 1$, SPDAM has very poor performance making the algorithm useless for online learning.

Chapter 6

Evaluation and Application

The algorithms SPAM and SPDAM were compared in the previous chapters by using regSOLAM as a benchmark for comparison, however, it is important to conduct an analysis of the three algorithms together. It is important to note that SPDAM is a batch learning algorithm while regSOLAM and SPAM are online learning algorithms. So it is expected that SPDAM should have a faster rate of convergence in comparison to regSOLAM and SPAM. We introduce the data sets in Section 6.1 and discuss the implementation in Section 6.2. A detailed discussion of the results is in the final section.

6.1 Data Set Descriptions

To examine the performance of the algorithms regSOLAM, SPAM, and SPDAM in comparison to each other, we conduct experiments on both benchmark datasets and data related to anomaly detection tasks. A summary of the 16 benchmark datasets is available in Table 6.1. All of these datasets are available for download from the LIBSVM and UCI machine learning repository. Note that some of the datasets (*mnist*, *covtype*, etc.) are multi-class, which we converted to binary data by randomly partitioning the data into two groups, where each group includes the same number of classes.

The methods were also tested on datasets related to anomaly detection tasks to demonstrate the methods effectiveness in many application domains. As from [21], our algorithms for AUC maximization could be used for such tasks. The data sets were obtained from [85]. Specifically, we evaluated our algorithms on data sets pertaining to the following application domains:

- **Malicious Websites**. We can apply the algorithms to determine if a website is malicious or not using the *webspam* dataset.

Datasets	♯ Inst	♯ Feat	Datasets	♯ Inst	♯ Feat
a9a	32,561	123	ijcnn1	141,691	22
acoustic	78,823	50	ionosphere	351	34
alpha	500,000	500	mnist	60,000	780
beta	500,000	500	news20	15,935	62,061
covtype	581,012	54	sector	9,619	55,197
diabetes	768	8	splice	3,175	60
fourclass	862	2	svmguide3	1243	21
german	1,000	24	usps	9,298	256

Table 6.1: Summary of standard benchmark datasets used in the experiments.

- **Bioinformatics** Detecting noncoding RNAs from sequenced genomes will be done using the *cod-rna* dataset.
- **Credit Card Fraud**. The *australian* dataset is used for predicting whether a credit card application is fraudulent or not.
- **Medical Diagnosis** We can apply the following datasets for the following medical diagnosis:
  - The *arrhythmia* dataset determines the presence of cardiac arrhythmia (irregular heartbeat).
  - The *breastw* and *mammography* datasets determines if a breast tumor is benign or malignant.
  - The *thyroid* datset is used for detecting if a patient is hypothyroid.
- **Spam Filter** The *spambase* dataset is used for determining whether an email is considered legitimate or not.

The datasets *arrhythmia*, *breastw*, *mammography*, *thyroid* were obtained from [85]. The datasets *australian*, *cod-rna*, *spambase*, and *webspam* were obtained from [32].

6.2 Implementation and Setup

The experiments were conducted in a similar manner as in the previous chapters. First, the features of each dataset were normalized. The data is randomly partitioned into 5 folds (4 are for training and 1 is for testing). To determine the proper parameter for each

Datasets	♯ Inst	♯ Feat	p (%)	Datasets	♯ Inst	♯ Feat	p (%)
arrhythmia	452	274	15.0	mammography	11183	6	2.32
australian	690	14	45.0	spambase	4601	57	40.0
breastw	683	9	35.0	thyroid	3772	6	2.5
cod-rna	59535	9	33.3	webspam	350,000	254	39.3

Table 6.2: Summary of datasets used for anomaly detection. The statistic p represents the occurrence of the minority class.

Datasets	regSOLAM	SPAM	SPDAM	Datasets	regSOLAM	SPAM	SPDAM
a9a	.8951±.0046	.8995±.0041	.8969±.0048	ijcnn1	.9161±.0024	.9285±.0019	.9145± .0019
acoustic	.7926±.0040	.8055±.0084	.8153±.0032	ionosphere	.8821±.0400	.9064±.0376	.9292±.0364
alpha	.8152±.0025	.8525±.0027	.8152±.0012	mnist	.9267±.0093	.9467±.0067	.9356±.0028
beta	.5011±.0019	.5037±.0011	.5033±.0006	news20	.9399±.0038	.8708±.0069	.8655±.0028
covtype	.7658±.0156	.7990±.0001	.8197±.0013	sector	.9734±.0036	.8768±.0126	.9406±.0062
diabetes	.8178±.0309	.8269±.0339	.8287 ±.0311	splice	.9100±.0155	.9173±.0143	.9243±.0125
fourclass	.8212±.0209	.8214±.0214	.8217±.0205	svmguide3	.6488±.0328	.6073±.0490	.6226±.0407
german	.7765±.0360	.7899±.0313	.7913±.0302	usps	.9690±.0033	.9775±.0032	.9791±.0033

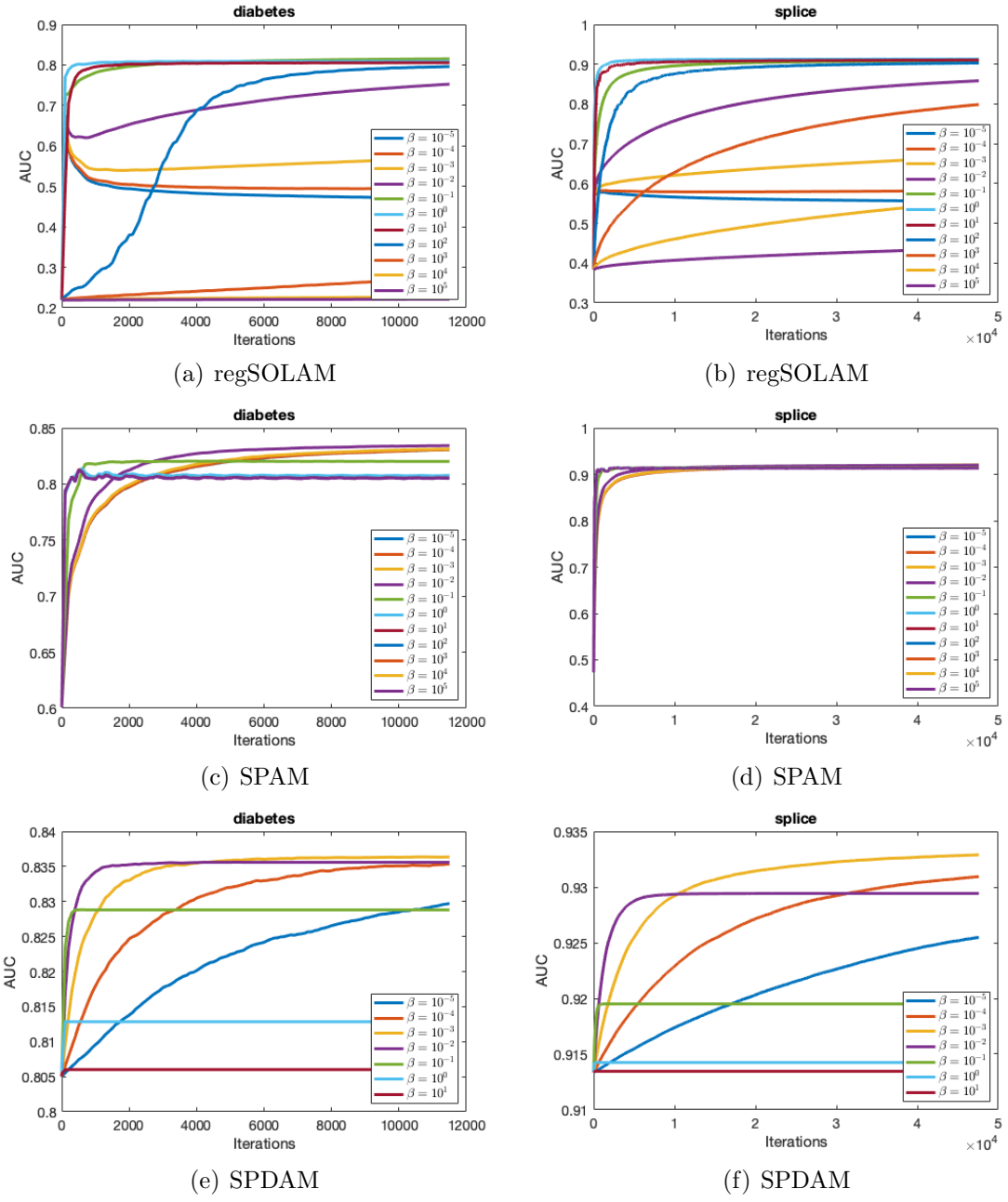
Table 6.3: Comparison of the testing AUC values (mean±std.) on the evaluated benchmark datasets.

dataset, we conducted 5-fold cross validation on the training sets to determine the parameter $\lambda \in 10^{[-5:1]}$ for SPDAM and SPAM. For regSOLAM, the learning rate $\zeta \in [1:9:100]$ and the regularization parameter $\lambda \in 10^{[-5:5]}$ were found by a grid search. The buffer size for OAMseq and OAMgra is 100 as suggested [118]. We generate this partition for each dataset 5 times. This results in 25 runs for each dataset for which we used to calculate the average AUC score and standard deviation. For large datasets, such as *alpha* and *beta*, the AUC score was determined after only 5 runs. All experiments for SPDAM and regSOLAM were conducted with MATLAB.

Selecting the optimal parameter for each algorithm is a necessary step as demonstrated in Figure 6.1. For regSOLAM and SPDAM, selecting the optimal parameter β can have a critical effect on both the achieved AUC performance as well as the convergence of the method. However, as from the same figure, it is shown that SPAM is less sensitive to changes in parameter selection.

6.3 Results

In this section, we present the results of regSOLAM, SPAM, and SPDAM on both standard benchmark and anomaly detection datasets as well as discuss the strengths and



(a) regSOLAM (b) regSOLAM

(c) SPAM (d) SPAM

(e) SPDAM (f) SPDAM

Figure 6.1: Comparison of regSOLAM, SPAM, and SPDAM for various parameter values.

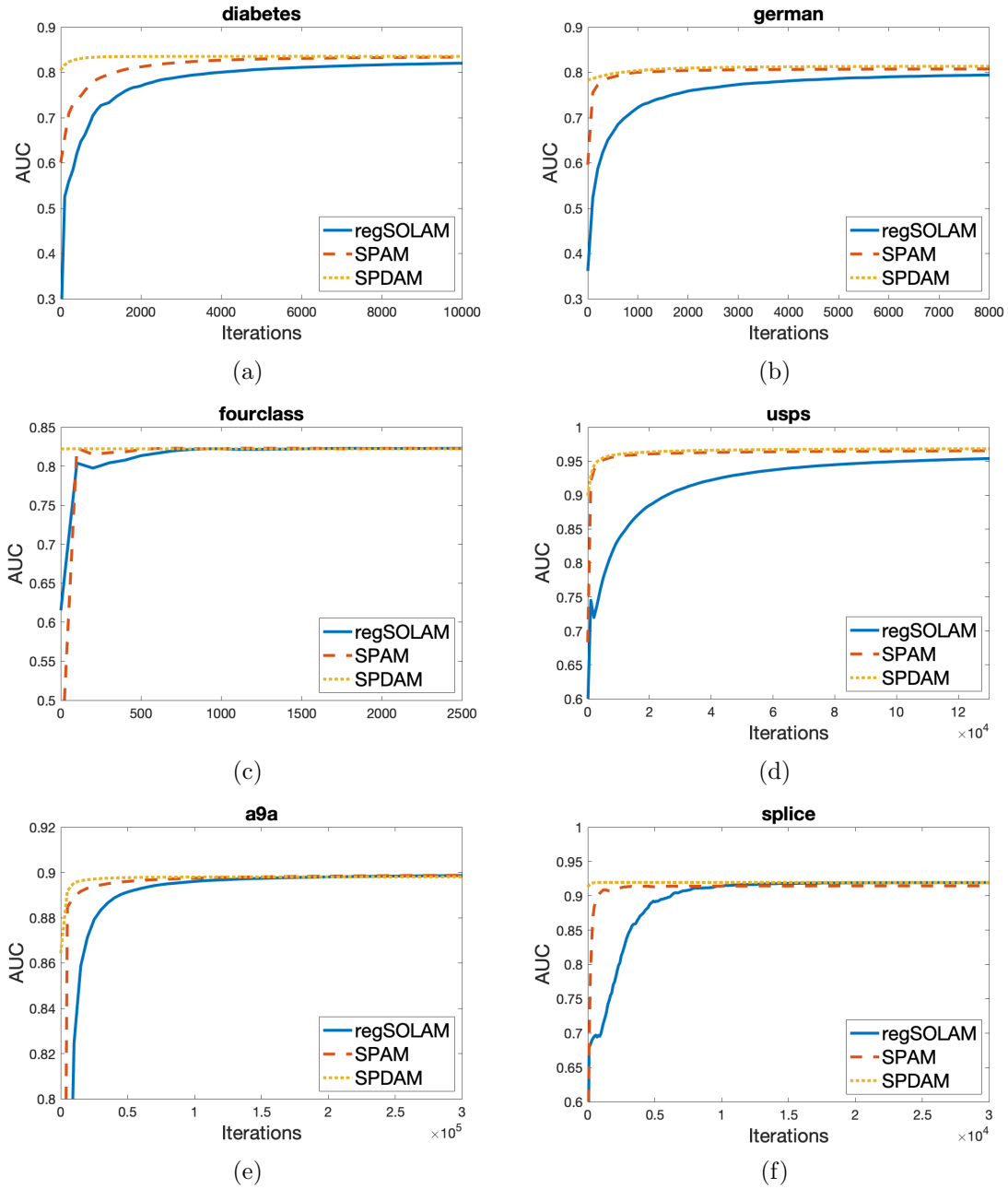


Figure 6.2: Comparison of regSOLAM against SPAM and SPDAM for AUC vs. # of iterations on benchmark datasets.

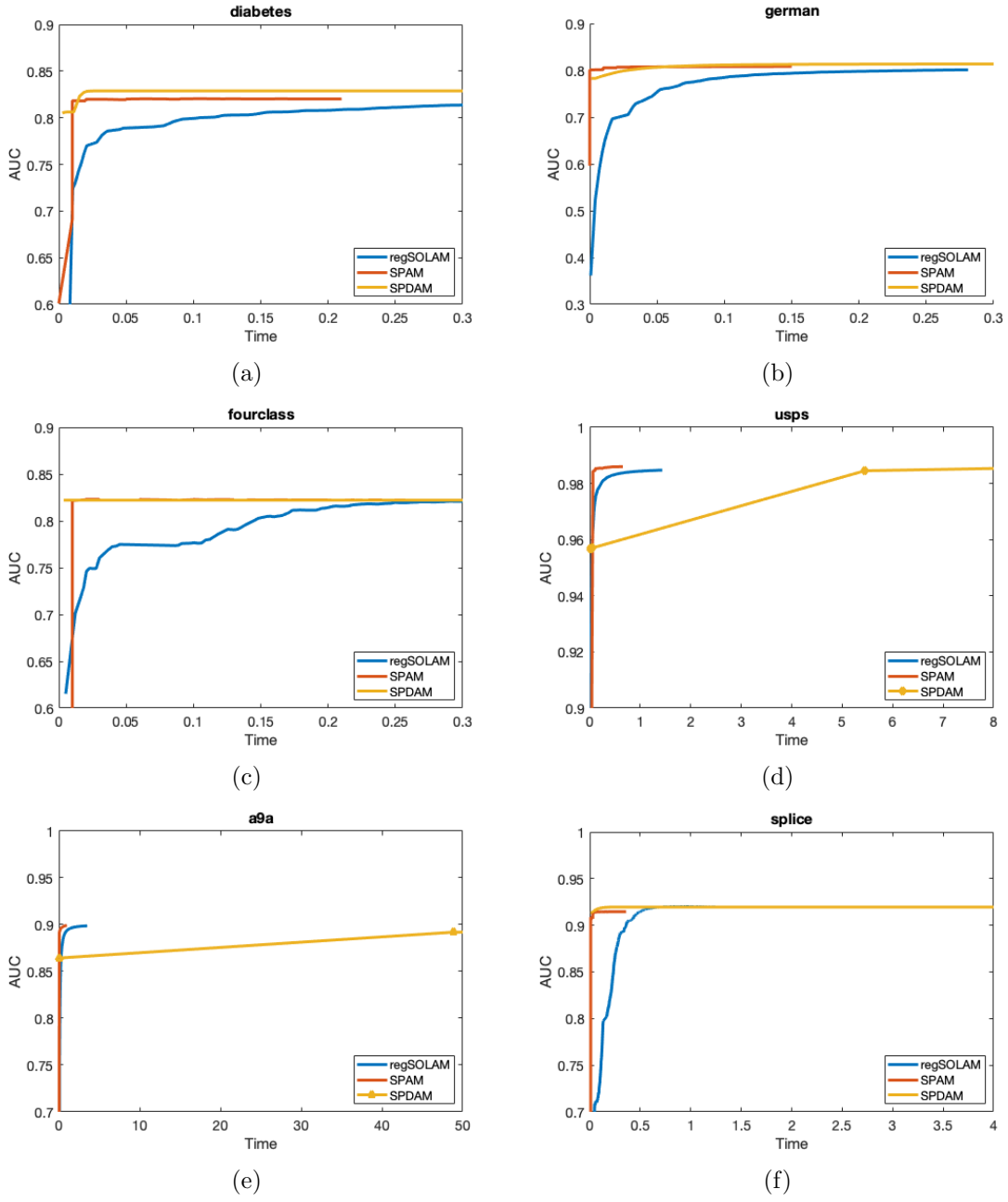


Figure 6.3: Comparison of regSOLAM against SPAM and SPDAM for AUC vs. time (seconds) on benchmark datasets. The points on the plot of SPDAM represents the first and second iterations of the algorithm.

Datasets	regSOLAM	SPAM	SPDAM
arrhythmia	.8284±.0775	.8523±.0672	.8738±.0576
australian	.7178±.0462	.7178±.0466	.7656±.0406
breastw	.9308±.0208	.9352±.0168	.9315±.0204
cod-rna	.9930±.0001	.9062±.0025	.9931 ±.0001
mammography	.8961±.0352	.9178±.0205	.9152±.0181
spambase	.7295±.0268	.7993±.0158	.7716±.0277
thyroid	.9972±.0023	.9976±.0014	.9976±.0012
webspam	.9609±.0022	.9660±.0005	.9527±.0006

Table 6.4: Comparison of the testing AUC values (mean±std.) on anomaly detection datasets.

weaknesses of the proposed methods. The performance of each method is demonstrated in Tables 6.3 and 6.4. Convergence plots of selected datasets are found in Figures 6.2, 6.3, and 6.4.

It is known that batch learning algorithms are able to better optimize the objective function over online learning since more data is being processed during each iteration. As from Table 6.3, SPDAM outperforms both regSOLAM and SPAM in many of the benchmark datasets. In the case of the anomaly detection, datasets in Table 6.4, regSOLAM and SPAM achieve a similar performance to SPDAM in many of the tasks, but overall SPDAM achieves a better AUC performance.

Even though SPDAM is able to outperform regSOLAM and SPAM, a critical analysis to consider is the convergence rate of each method. A major draw back of SPDAM, and other batch learning methods, is that these methods result in a higher computational cost for each iteration step. Specifically, SPDAM processes a mini batch of size m with each sample of dimension d, thus resulting in a complexity of $\mathcal{O}(md)$, while the per-iteration cost of regSOLAM and SPAM are only of the order $\mathcal{O}(d)$. This difference is insignificant for small datasets such as *diabetes* or *fourclass*. But for high dimensional data sets such as *alpha* and *beta*, the difference in computational time is significant and puts SPDAM at a major disadvantage in terms of computational time. This is illustrated in Figures 6.2, 6.3, and 6.4. Since SPDAM was shown in chapter 5 to have a linear convergence rate, it is expected to outperform the other methods when the rate of convergence is measured against the iteration count. However, when considering the computational time of each method as in Figure 6.3, SPDAM is considered extremely slow compared to regSOLAM and SPAM. Since each step of SPDAM is of order $\mathcal{O}(md)$, for low dimensional datasets such as *diabetes*

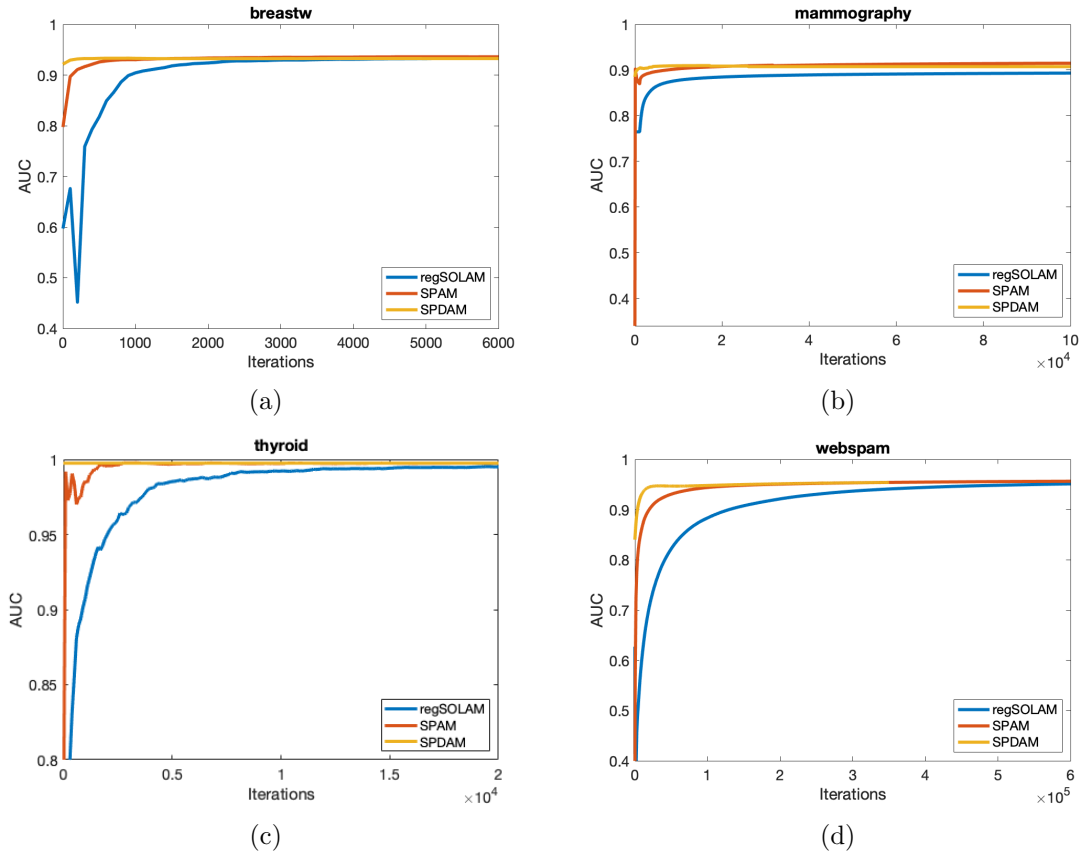


Figure 6.4: Comparison of regSOLAM against SPAM and SPDAM for AUC vs. # of iterations for anomaly detection tasks.

and *fourclass*, SPDAM is still able to outperform regSOLAM and SPAM, however, for high dimensional datasets such as *usps* and *a9a*, regSOLAM and SPAM are able to convergence even before SPAM completes its second iteration. Overall, even though SPDAM results in a higher optimized AUC score, the method would not be useful for big data.

A drawback of the proposed algorithms in this thesis is demonstrated by the poor peformance of the dataset *beta*. As from Theorem 3.1, a critical assumption of this thesis is that for each method, the goal is to determine a linear function for classification. In many datasets like *beta*, this cannot always be achieved. To overcome this problem, data is generally mapped to a higher dimension to where the data can be separated by a linear function. This is done by way of feature maps or kernel methods.

Chapter 7

Conclusion

In this thesis, we presented several novel contributions to the area of designing algorithms that overcome the challenges in optimizing AUC. Novel online and batch learning methods were developed for optimizing the AUC metric in chapters 3, 4, and 5. Experimental validation was performed on benchmark datasets for each of the proposed methods as well as a comparison between each of the methods. Furthermore, we applied the methods to anomaly detection tasks in numerous application domains. We will first outline the main contributions of this work as well as outline directions for future work.

7.1 Contributions

The main contributions of this thesis will be outlined in this section. In chapter 3, we formed the saddle point formulation for AUC optimization that overcomes the significant challenges of AUC optimization. Existing methods have attempted to overcome this, but with significant drawbacks for high dimensional data. Using chapter 3 as a foundation, we are able to further develop other algorithms that optimize the AUC score. To ensure a fair comparison with the methods developed in chapters 4 and 5, a regularization term was included in the saddle point formulation and the stochastic online method (regSOLAM) was appropriately modified. We also showed that the method still obtains a convergence rate of $\mathcal{O}(1/\sqrt{T})$.

From the formulation in chapter 3, it is of interest to develop algorithms with a faster rate of convergence. The first of which, was discussed in chapter 4. Exploiting the idea that the primal and dual variables can be determined in terms of the current iterate of $\mathbf{w}$, a novel stochastic proximal algorithm (SPAM) for AUC maximization with general penalty terms was developed. We proved that the algorithm could achieve a convergence rate of $\mathcal{O}(1/T)$ up to a logarithmic term for strongly convex regularizers. The L_2 and elastic net regularizers

were chosen for experimental validation with the latter showing increased performance. The resulting method has a space and per-iteration complexity of one datum.

For the last contribution of this thesis, we began with the empirical risk minimization problem for AUC that has been previously studied extensively. We developed a stochastic primal-dual algorithm for AUC optimization by compromising on the per-iteration costs and exploited the objective function by using the conjugate of the objective function. Unlike the previous methods, the proposed algorithm used a mini-batch of samples to update the model. We then showed that the method can achieve a linear convergence rate and validated it experimentally.

Finally, we compared all three methods using numerous standard benchmark datasets. Additionally, we applied the methods to anomaly detection problems such as spam detection, credit card fraud detection, and medical diagnosis. Overall, we showed that regSOLAM, SPAM, and SPDAM are capable of optimizing the AUC score. Furthermore, we showed that SPDAM can achieve better better performance in some cases, which is to be expected since it is a batch learning algorithm. However, if the batch size is not selected sufficiently large enough, the method has poor performance. An advantage of SPAM over SPDAM is that it is ideal for processing streaming data.

7.2 Future Work

There are several promising directions for future work based on this thesis.

- Stochastic gradient descent is preferred for large scale optimization problems, however, the method is based on the learning rate η. It is desirable for a large learning rate to ensure faster convergence, but this increases the variance of each solution. A popular algorithmic approach is designing stochastic variance reduction algorithms [43] and stochastic primal-dual algorithms [116] that have a linear convergence rate. It is of interest to develop variance reduction methods for AUC optimization with reduced per-iteration costs.

- For regSOLAM, it is of interest to determine the strong convergence of $\bar{\mathbf{w}}_T$ in comparison to the optimal solution of the AUC problem.

- The saddle point formulation was only shown for the least square loss. It is not known if the formulation holds true of other loss functions such as the hinge loss.

- SPAM achieves a convergence of $\mathcal{O}(1/T)$ under strong convexity. It remains unclear whether or not this assumption is necessary. The proof techniques in [1, 110] could be used to remove this assumption.

- A major assumption of this work was to find a function linear scoring function, i.e., $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})$. A novel extension of this work would be to extend these methods to nonlinear scoring functions using kernel methods. Some work has been done for stochastic methods to include kernel functions [17].

- In addition to the previous extension, these ideas could be used with AUC maximization in deep learning. Previous work on this topic has already been done, however, there still exists many open problems in this area [58].

- In addition to ROC analysis being used in many application domains, lift is another measurement that is used for imbalanced class distributions in areas of direct marketing [56] and fraud detection [94]. The ideas presented in this thesis could be used to design methods that optimize the area under the lift chart.

Bibliography

[1] F. Bach and E. Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate o (1/n). In *Advances in neural information processing systems*, pages 773–781, 2013.

[2] H H Bauschke and P L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

[3] Kristin P Bennett and Olvi L Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization methods and software*, 1(1):23–34, 1992.

[4] Michael J Berry and Gordon Linoff. *Data mining techniques: for marketing, sales, and customer support*. John Wiley & Sons, Inc., 1997.

[5] L. Bottou and Y. L. Cun. Large scale online learning. In *Advances in neural information processing systems*, 2004.

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

[7] A P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[8] Claire Cardie and Nicholas Howe. Improving minority class prediction using case-specific feature weights. In *ICML*, pages 57–65, 1997.

[9] SH Clearwater and EG Stern. A rule-learning program in high energy physics event classification. *Computer physics communications*, 67(2):159–182, 1991.

[10] S. Clémençon, G. Lugosi, and N. Vayatis. Ranking and empirical minimization of u-statistics. *The Annals of Statistics*, pages 844–874, 2008.

[11] William W Cohen, Robert E Schapire, and Yoram Singer. Learning to order things. In *Advances in Neural Information Processing Systems*, pages 451–457, 1998.

[12] Michael Collins, Robert E Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1-3):253–285, 2002.

[13] C. Cortes and M. Mohri. Auc optimization vs. error rate minimization. In *Advances in neural information processing systems*, pages 313–320, 2004.

[14] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[15] Koby Crammer, Mark Dredze, and Fernando Pereira. Exact convex confidence-weighted learning. In *Advances in Neural Information Processing Systems*, pages 345–352, 2009.

[16] Noel Cressie. Statistics for spatial data. *Terra Nova*, 4(5):613–617, 1992.

[17] Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 3041–3049, 2014.

[18] Soham Dan and Dushyant Sahoo. Variance reduced stochastic proximal algorithm for auc maximization. *arXiv preprint arXiv:1911.03548*, 2019.

[19] AP Dawid, RG Cowell, SL Lauritzen, and DJ Spiegelhalter. Probabilistic networks and expert systems. Springer-Verlag, 1999.

[20] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

[21] Yi Ding, Peilin Zhao, Steven CH Hoi, and Yew-Soon Ong. An adaptive gradient method for online auc maximization. In *AAAI*, pages 2568–2574, 2015.

[22] Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th international conference on Machine learning*, pages 264–271. ACM, 2008.

[23] J. Duchi, E. Hazan., and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[24] J Duchi and Y Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10(Dec):2899–2934, 2009.

[25] James P Egan. *Signal Detection Theory and ROC Analysis Academic Press Series in Cognition and Perception*. London, UK: Academic Press, 1975.

[26] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[27] Tom Fawcett and Foster Provost. Adaptive fraud detection. *Data mining and knowledge discovery*, 1(3):291–316, 1997.

[28] Tom Fawcett and Foster J Provost. Combining data mining and machine learning for effective user profiling. In *KDD*, pages 8–13, 1996.

[29] Werner Fenchel. On conjugate convex functions. *Canadian Journal of Mathematics*, 1(1):73–77, 1949.

[30] César Ferri, Peter Flach, and José Hernández-Orallo. Learning decision trees using the area under the roc curve. In *ICML*, volume 2, pages 139–146, 2002.

[31] Vojtěch Franc and Soeren Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 320–327, New York, NY, USA, 2008. ACM.

[32] Andrew Frank and Arthur Asuncion. Uci machine learning repository [http://archive. ics. uci. edu/ml]. irvine, ca: University of california. *School of information and computer science*, 213, 2010.

[33] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[34] W. Gao, R. Jin, S. Zhu, and Z.-H. Zhou. One-pass auc optimization. In *International Conference on Machine Learning*, pages 906–914, 2013.

[35] Wei Gao and Zhi-Hua Zhou. On the consistency of auc pairwise optimization. In *IJCAI*, pages 939–945, 2015.

[36] San Gultekin, Avishek Sana, Adwait Ratnaparkhi, and John Paisley. Mba: Mini-batch auc optimization. *arXiv preprint arXiv:1805.11221*, 2018.

[37] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[38] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[39] A Herschtal and B Raskutti. Optimising area under the roc curve using gradient descent. In *Proceedings of the twenty-first international conference on Machine learning*, page 49. ACM, 2004.

[40] 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.

[41] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384. ACM, 2005.

[42] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[43] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[44] S Sathiya Keerthi and Dennis DeCoste. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6(Mar):341–361, 2005.

[45] James E Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[46] Krzysztof C Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical programming*, 46(1-3):105–122, 1990.

[47] Roger Koenker and Kevin F Hallock. Quantile regression. *Journal of economic perspectives*, 15(4):143–156, 2001.

[48] W Kotlowski, K J Dembczynski, and E Huellermeier. Bipartite ranking through minimization of univariate loss. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1113–1120, 2011.

[49] Miroslav Kubat, Robert C Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2-3):195–215, 1998.

[50] Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an o (1/t) convergence rate for the projected stochastic subgradient method. *arXiv preprint arXiv:1212.2002*, 2012.

[51] G Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1):365–397, 2012.

[52] Claude Lemaréchal, Arkadii Nemirovskii, and Yurii Nesterov. New variants of bundle methods. *Mathematical programming*, 69(1-3):111–147, 1995.

[53] David D Lewis. Evaluating text categorization i. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*, 1991.

[54] Alexandra L'heureux, Katarina Grolinger, Hany F Elyamany, and Miriam AM Capretz. Machine learning with big data: Challenges and approaches. *IEEE Access*, 5:7776–7797, 2017.

[55] Charles X Ling, Jin Huang, and Harry Zhang. Auc: a better measure than accuracy in comparing learning algorithms. In *Conference of the canadian society for computational studies of intelligence*, pages 329–341. Springer, 2003.

[56] Charles X Ling and Chenghui Li. Data mining for direct marketing: Problems and solutions. In *Kdd*, volume 98, pages 73–79, 1998.

[57] Charles X Ling and Huajie Zhang. Toward bayesian classifiers with accurate probabilities. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 123–134. Springer, 2002.

[58] Mingrui Liu, Zhuoning Yuan, Yiming Ying, and Tianbao Yang. Stochastic auc maximization with deep neural networks. *arXiv preprint arXiv:1908.10831*, 2019.

[59] Mingrui Liu, Xiaoxuan Zhang, Zaiyi Chen, Xiaoyu Wang, and Tianbao Yang. Fast stochastic auc maximization with o (1/n)-convergence rate. In *International Conference on Machine Learning*, pages 3195–3203, 2018.

[60] Brian Mac Namee, Padraig Cunningham, Stephen Byrne, and Owen I Corrigan. The problem of bias in training data in regression problems in medical decision support. *Artificial intelligence in medicine*, 24(1):51–70, 2002.

[61] M A Maloof. Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 workshop on learning from imbalanced data sets II*, volume 2, pages 2–1, 2003.

[62] Marcus A Maloof, Pat Langley, Thomas O Binford, Ramakant Nevatia, and Stephanie Sage. Improved rooftop detection in aerial images with machine learning. *Machine Learning*, 53(1-2):157–191, 2003.

[63] Olvi L Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations research*, 13(3):444–452, 1965.

[64] Eric Moulines and Francis R Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459, 2011.

[65] Michael C Mozer, Robert Dodier, Michael D Colagrosso, César Guerra-Salcedo, and Richard Wolniewicz. Prodding the roc curve: Constrained optimization of classifier performance. In *Advances in Neural Information Processing Systems*, pages 1409–1415, 2002.

[66] K-R Müller, Alexander J Smola, Gunnar Rätsch, Bernhard Schölkopf, Jens Kohlmorgen, and Vladimir Vapnik. Predicting time series with support vector machines. In *International Conference on Artificial Neural Networks*, pages 999–1004. Springer, 1997.

[67] Allan H Murphy. The finley affair: A signal event in the history of forecast verification. *Weather and Forecasting*, 11(1):3–20, 1996.

[68] Michael Natole, Yiming Ying, and Siwei Lyu. Stochastic proximal algorithms for auc maximization. In *International Conference on Machine Learning*, pages 3707–3716, 2018.

[69] Michael Anthony Natole, Yiming Ying, and Siwei Lyu. Stochastic auc optimization algorithms with linear convergence. *Frontiers in Applied Mathematics and Statistics*, 5:30, 2019.

[70] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

[71] Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[72] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[73] Francesco Orabona and Koby Crammer. New adaptive algorithms for online classification. In *Advances in neural information processing systems*, pages 1840–1848, 2010.

[74] Balamurugan Palaniappan and Francis Bach. Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems*, pages 1416–1424, 2016.

[75] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[76] DJ Peres and A Cancelliere. Derivation and evaluation of landslide-triggering thresholds by a monte carlo approach. *Hydrology and Earth System Sciences*, 18(12):4913–4931, 2014.

[77] DJ Peres, C Iuppa, L Cavallaro, A Cancelliere, and E Foti. Significant wave height record extension by neural networks and reanalysis wind data. *Ocean Modelling*, 94:128–140, 2015.

[78] F Provost and T Fawcett. Robust classification for imprecise environments. *Machine learning*, 42(3):203–231, 2001.

[79] Foster Provost and Tom Fawcett. Robust classification systems for imprecise environments. In *AAAI/IAAI*, pages 706–713, 1998.

[80] Foster J Provost, Tom Fawcett, et al. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *KDD*, volume 97, pages 43–48, 1997.

[81] Foster J Provost, Tom Fawcett, Ron Kohavi, et al. The case against accuracy estimation for comparing induction algorithms. In *ICML*, volume 98, pages 445–453, 1998.

[82] A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 449–456, 2012.

[83] A Rakhlin, O Shamir, and K Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning*, pages 449–456, 2012.

[84] Alain Rakotomamonjy. Optimizing auc with support vector machine. In *European Conference on Artificial Intelligence Workshop on ROC Curve and AI*, 2004.

[85] Shebuti Rayana. Odds library, 2016.

[86] Lorenzo Rosasco, Silvia Villa, and Bang Công Vũ. Convergence of stochastic proximal gradient algorithm. *arXiv preprint arXiv:1403.5074*, 2014.

[87] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[88] Lorenza Saitta and Filippo Neri. Learning in the "real world". *Machine learning*, 30(2-3):133–163, 1998.

[89] Steven L Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data mining and knowledge discovery*, 1(3):317–328, 1997.

[90] Helga Schramm and Jochem Zowe. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM journal on optimization*, 2(1):121–152, 1992.

[91] S Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[92] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[93] O Shamir and T Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning*, pages 71–79, 2013.

[94] Aihua Shen, Rencheng Tong, and Yaochen Deng. Application of classification models on credit card fraud detection. In *2007 International conference on service systems and service management*, pages 1–4. IEEE, 2007.

[95] S. Smale and Y. Yao. Online learning algorithms. *Foundations of computational mathematics*, 6(2):145–170, 2006.

[96] Kent A. Spackman. Signal detection theory: Valuable tools for evaluating inductive learning. In Alberto Maria Segre, editor, *Proceedings of the Sixth International Workshop on Machine Learning*, pages 160 – 163. Morgan Kaufmann, San Francisco (CA), 1989.

[97] N. Srebro and A. Tewari. Stochastic optimization for machine learning. *ICML Tutorial*, 2010.

[98] John A Swets. Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293, 1988.

[99] John A Swets, Robyn M Dawes, and John Monahan. Better decisions through science. *Scientific American*, 283(4):82–87, 2000.

[100] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Advances in neural information processing systems*, pages 25–32, 2004.

[101] Choon Hui Teo, SVN Vishwanthan, Alex J Smola, and Quoc V Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11(Jan):311–365, 2010.

[102] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

[103] V Vapnik. Statistical learning theory new york. *NY: Wiley*, 1998.

[104] Vladimir Vapnik. Support vector method for function estimation, September 7 1999. US Patent 5,950,146.

[105] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[106] Miha Vuk and Tomaz Curk. Roc curve, lift chart and calibration plot. *Metodoloski zvezki*, 3(1):89, 2006.

[107] Christopher KI Williams. Prediction with gaussian processes: From linear regression to linear prediction and beyond. In *Learning in graphical models*, pages 599–621. Springer, 1998.

[108] Stephen Wright and Jorge Nocedal. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

[109] Lian Yan, Robert H Dodier, Michael Mozer, and Richard H Wolniewicz. Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 848–855, 2003.

[110] Tianbao Yang and Qihang Lin. Rsg: Beating subgradient method without smoothness and strong convexity. *arXiv preprint arXiv:1512.03107*, 2015.

[111] I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2):2473–2480, 2009.

[112] Y. Ying and M. Pontil. Online gradient descent learning algorithms. *Foundations of Computational Mathematics*, 8(5):561–596, 2008.

[113] Y. Ying, L. Wen, and S. Lyu. Stochastic online auc maximization. In *Advances in Neural Information Processing Systems*, 2016.

[114] Adams Wei Yu, Qihang Lin, and Tianbao Yang. Doubly stochastic primal-dual coordinate method for empirical risk minimization and bilinear saddle-point problem. *arXiv preprint arXiv:1508.03390*, 2015.

[115] Yao-Liang Yu. The proximity operator. 2014.

[116] Yuchen Zhang and Lin Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. *The Journal of Machine Learning Research*, 18(1):2939–2980, 2017.

[117] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017.

[118] P. Zhao, R. Jin, T. Yang, and S. C. Hoi. Online auc maximization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011.

[119] Tianyi Zhou, Dacheng Tao, and Xindong Wu. NESVM: a fast gradient method for support vector machines. *CoRR*, abs/1008.4000, 2010.

[120] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 928–936, 2003.

[121] H Zou and T Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[122] Kelly H Zou. Receiver operating characteristic (roc) literature research. *On-line bibliography available from:¡ http://splweb. bwh. harvard. edu*, 8000, 2002.