Rapid Learning with Stochastic Focus of Attention

Raphael A. Pelossof

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2011
ABSTRACT

Rapid Learning with Stochastic Focus of Attention

Raphael A. Pelossof

We present a method to determine when to stop the evaluation of a decision-making process. The method determines to stop the evaluation process when the result of the full evaluation is obvious. This trait is highly desirable for margin-based Machine Learning algorithms where a classifier traditionally evaluates all the features for every example. However, some examples are easier to classify than others, a phenomenon which is characterized by the event when most of the features agree on the class of an example. By stopping the feature evaluation when encountering an easy to classify example, a margin-based Machine Learning algorithm can achieve substantial reduction in running times.

To determine when to stop the feature evaluation, we develop a set of novel sequential tests, the Sequential Thresholded Sum Tests (STST). These tests stop the partial evaluation of the sum when the result of the full summation is guaranteed with high probability. By making different assumptions on the data and the features different tests arise. In general we look at the feature evaluation process as a random walk and apply different Brownian motion early stopping inequalities to determine when to stop the walk. From these inequalities we derive a family of stopping thresholds for sequential feature evaluations under different assumptions.

We demonstrate the effectiveness of the different STST by speeding up several Online Learning algorithms on synthetic and real data.
# Table of Contents

1 Introduction 1
   1.1 Main Contributions .......................... 6
   1.2 Outline ...................................... 6

I Introduction to Margin-Based Learning on a Feature Budget 8

2 Margin-based Machine Learning 9
   2.1 Related Work .................................. 10
   2.2 Notation ...................................... 12
   2.3 Structural Risk Minimization .................. 13
   2.4 The VC Dimension ............................ 14
   2.5 Generalization Capabilities of Large Margin Classifiers 16

3 Learning on a Feature Budget 19
   3.1 Related Work .................................. 20
   3.2 Generalization ................................ 23

II Sequential Thresholded Sum Tests 25

4 Sequential Statistical Tests for Thresholded Sums 26
   4.1 Introduction .................................. 26
   4.2 Sequential Thresholded Sum Tests .......... 29
      4.2.1 Mathematical Roadmap .................. 29
List of Figures

1.1 First connection between sequential margin-based learning and clinical testing. 3
1.2 Meetings with Professor Zhiliang Ying at 1:30. 5

2.1 A two dimensional line can shatter three points. Equivalently, for any configuration of labels, the points can be separated by a line. 14
2.2 The SVM achieves a larger margin than the perceptron. According to theorem 4 it generalizes better. 17

3.1 Two examples are classified. The first is hard to classify, the second easy. The budgeted learning approach would evaluate the same number of features for both examples, whereas the stochastic would evaluate features according to how hard is the example to classify, while maintaining an average budget. 22

4.1 Margin evaluation as a random walk. The margin evaluation is a summation of $n$ summands. At each partial sum, the $i$th feature can add $+X_i$ or $-X_i$ to the partial sum. We are interested in calculating the probability that the random walk after $n$ steps will end below the threshold $\theta$ given that the current position $S_i$ passed a stopping thresholds (in red). This probability will give us the stopping boundary shown in red. 30
4.2 Illustration of the two types of decision errors. 31
4.3 A comparison between (upper) stopping boundaries. Lower boundaries produce faster decision-making. All our methods outperform the trivial case. The boundary found by using the central limit theorem is more aggressive than the one found by Hoeffding’s inequality. The flat boundary is more aggressive than both curved boundaries. Parameter setting: \( \delta = 0.1, \theta = 0, EX_i = 0, var(X_i) = 1 \).

4.4 The resulting constants from the application of Hoeffding’s inequality versus the Central Limit Theorem. Although the central limit theorem’s smaller constants are preferable, Hoeffding’s inequality requires less assumptions on the random variables.

4.5 A comparison of the error rate observed when using CLT constants vs. Hoeffding constants. Results plotted for simulation of 10000 random walks \( X_i \sim N(0, 1) \). Plotted are the required decision error rate \( \delta \), and the actual error rate obtained by simulation. CLT gives lower constants which result in a sharper and more aggressive stopping thresholds than the ones obtained by Hoeffding’s inequality.

4.6 CLT outperforms Hoeffding in terms of speed. Computational savings as a function on the decision error. The savings in computation is the fraction of features that were not evaluated from the entire set of features. CLT constants are smaller, and therefore the resulting stopping thresholds are smaller, the random walk stops more quickly, and savings are larger. Data obtained by simulation \( X_i \sim N(0, 1) \).

4.7 This figure demonstrates the reflection principle. The principle creates a reflection of all paths from the stopping boundary to the importance threshold. Thereby equating the probability of making a decision error to the probability of passing the new lower decision threshold.

4.8 Comparison of the flat boundary for the joint decision stopping thresholds, vs. the adaptive flat boundary and the conditional decision stopping thresholds.

4.9 Performance of the Brownian bridge boundary.
6.1 **Stochastic focus-of-attention.** Online Boosting, which is modified to have stochastic focus-of-attention, places the most computational effort into processing examples near the decision boundary. The training set is pictured in the left subfigure. In the right subfigure, the size and color of each point are proportional to the number of features evaluated throughout the curtailed online learning process. Most examples with a large margin are rejected early without a full evaluation of their features, whereas hard examples by the decision boundary get allocated high computational effort.

6.2 Generalization Error on synthetic and MNIST data sets.

6.3 Curved CLT-STST, $x_i \sim U(-1,1)$ and $w_i \sim U(-1,1)$, decision error rate is set to $\delta = 10\%$. We can see that the decision boundary is very tight. The distribution of stopping times when the weights are sorted (bottom right subfigure) shows that most of the random walks are terminated when only $20\%$ of the computation is done.

6.4 Results for Attentive Pegasos, MNIST 2 vs 3, $\delta = 10\%$. Our Brownian bridge decision boundary (blue) processes only 49 feature on average (15 times faster than full computation), while achieving similar generalization as the fully trained classifier (red, middle subfigure). On the right subfigure, when the boundary is applied to prediction, Attentive Pegasus achieves a lower error rate than the full computation, and less than half the error of the Budgeted Boundary (green).

6.5 Results for Attentive Pegasos, MNIST 3 vs 10, $\delta = 10\%$. Our Brownian bridge decision boundary (blue) processes only 72 feature on average, while achieving similar generalization as the fully trained classifier. On the right, when the boundary is applied to classification, Attentive Pegasus gets a lower error rate than the full computation, and over a 2\% advantage over the Budgeted Boundary.
List of Tables

6.1 MNIST test error in % for each classifier, and curtailment efficiency. Curtailed Online Boosting (COB) is compared to Online Boosting (OB) which is trained with the same average number of features that COB used throughout learning. COB always performs better than OB. COB achieves a speedup of 8x with only 0.01% loss in classification accuracy! 64
Acknowledgments

There are many people that took part in this research and my life throughout this PhD. First of all, I’d like to thank my advisor David Waltz who always supported me, believed in me, let me follow my instincts and enriched my knowledge in many fields. I’d like to thank Prof. Zhiliang Ying for his enlightening conversations, his many teachings, his openness to explore creative ideas outside the realm of traditional Statistics. Also, I would like to deeply thank Mike Jones, who taught me the basics of problem solving, from problem statement, to the fundamentals of simple solutions by the discriminative school of thought. Although we never made it to the Bahamas, our collaboration was an incredible journey for me, and yielded the problem that this thesis solves. Thank you Prof. Rocco Servedio for a very careful and thorough review of the thesis. I also thank Prof. Yoav Freund, my first advisor, who exposed me to the discriminative approach and helped lay the foundation for later on research.

Throughout the PhD I was a part of CCLS (Center of Computational Learning Systems). I made many friends there, and had a several constructive discussions. I’d like to thank Claire Monteleoni for her insightful conversations with me, Ilia Vovsha for his debates and ideas, Manoj Pooleery for his creative advice on writing, and German Cramer for lending an ear and listening when everybody else stopped a long time ago.

From the Computer Science department I’d like to thank Paul Blaer, Matei Ciocarlie, Andrew Howard, Risi Kondor, Darrin Lewis, Andrew Miller, and Atanas Georgiev for supporting me and being good friends throughout my early days in the department. Also, special thanks to Pat Harvey, Twinkle Edwards, Jessica Rosa, and Remiko Moss from the Computer Science department for making complicated procedures easy.

Finally, and most importantly, I’d like to thank my loving family, my dad Avi for letting me know that I can dream, and to my mom Sylvie for showing me how to. They always
support me and value education and family as a building block for life. Special thanks to my
sister Karen who is always there to support me and debate any idea I have with love. Special
thanks to my girlfriend Kasia for supporting me through hard times and easy times, always
listening and debating and lighting my day with just a simple smile. Thanks to my close
friends that indirectly earned this PhD with me by listening, debating, reviewing, editing
and supporting, and countless other ways which I cannot mention: Jonathan Cohen, Itamar
Raz, Iddo Gruengard, Tal Karny, Oded Padan, Hila Shmilovich, and Lauren Bibeau.
“Theories should be as simple as possible, but not simpler”

– Albert Einstein
Chapter 1

Introduction

I first got acquainted with the problem of speeding up decision-making for linear classifiers while interning at Mitsubishi Electric Research Lab under the supervision of Michael Jones. We were working on adapting the famous Viola and Jones (2004) face detector to new scenes. The face detector which is trained to be robust to many environments, is not as accurate on a specific environment as a detector which was trained solely on that environment. Our approach to improving the trained general detector was to adapt it to the specific environment by training it with an online algorithm which learns in real time. Since the detector was trained with a variant of AdaBoost, the natural pick for an online algorithm was Oza and Russell’s Online Boosting algorithm (Oza and Russell 2001).

The original (static) face detector is trained in three stages. Training loops through these stages until the test error is sufficiently low. First, a discriminative feature is selected and a weight is assigned to it. Second, a new training set is sampled according to ease of classification of each example, creating a progressively harder set to classify. Finally, after having selected all the features, early stopping thresholds are set in such a way that most of the non-faces will be rejected after the evaluation of only a few features. Since most of the examples are negative, this yields a 200x speedup in classification when the detector is deployed over a classifier which is not thersholded.

To adapt the detector we needed to continue its training for a new environment. The adaptation was done using the Online Boosting algorithm. Our first task involved improving the accuracy of Oza and Russell’s Online Boosting algorithm. We noticed three problems:
CHAPTER 1. INTRODUCTION

The first, it turned out that in order to improve the accuracy of Online Boosting more computation was needed to compute second order statistics in the order of the number of features squared. Second, unlike the sampling scheme in the batch training, Online Boosting computes all the features for all the examples. This is very wasteful since most of the examples are uninformative. Finally, as the online algorithm adapts to a new data stream, the weights of the features change, and the stopping thresholds which were set during batch training are not as efficient anymore in rejecting negative examples.

Interestingly all these problems coincided. We didn’t want to compute all the features for every example since the computation was squared in the number of features. Also, since most of the examples were uninformative (e.g. white background) we wanted to find a way to quickly filter them without fully evaluating all the models’ features. Finally, the stopping thresholds that were set while training in batch typically quickly stop the computation of easy to classify negative examples. Therefore, these three problems all involved evaluating the least number of features possible when making the decision about whether an example is uninformative and should therefore be filtered. With this problem in mind I proceeded back to Columbia to propose the thesis to my PhD committee. I proposed that the thesis would find a mathematical basis for choosing early stopping and filtering thresholds.

For a few weeks after the proposal I was reading and searching for new relevant articles with little success. I then attended a “Great Presentations” seminar under Prof. Janet Kayfetz, which was a visiting professor from UC Santa Barbara. The seminar’s goal was to teach engineering students how to convey and present their ideas in a way that most people would understand. For one of the exercises we were asked to prepare a couple of slides about our research topic but make them with as little text and math as possible. The exercise’s task was to improve our expression-ability in a more natural “human” form. I prepared a slide showing my heart in the middle and surrounding it, different tests that are used to test if it’s healthy (see figure 1.1.) I went on explaining how when I turned 30 my mom told me “Happy birthday! you’re old now, go and check your heart!” I listened to my mom, and indeed I went and got different tests prescribed. According to the results of the first couple of tests others were prescribed (blood test, blood pressure, stress test, etc..). I thought to myself this diagnostic process is exactly the process which I need for my filtering
mechanism. I wondered “When is the interim point when the doctor has enough certainty about the outcome of the entire sequence of tests that he stops testing and predicts my condition? How does the doctor know when to stop testing and diagnose?” Equivalently for a classifier, when is the point where enough features were evaluated to stop the evaluation process and predict the importance (or class) of the example?

I wondered about that quite a bit and decided to search online for “clinical testing” methods. As it turns out, when pharmaceuticals test new drugs they don’t want to poke too many patients since they might kill them and the test costs are high. Therefore the pharmaceuticals turn to statisticians to design sequential tests which stop the testing procedure when there is enough evidence of the efficacy of the new drug. Tracing back this enormous body of work revealed one of the early works in sequential statistic, the work of Abraham Wald (Wald 1945).

In his paper Wald describes how back in the time of World War II he was a professor at Columbia University, and he was approached by the US Army to design a sequential test to decide when a batch of bombs had less than 20% failures. The test was desirable since the US military was sending England bombs to drop over the Nazis, and they wanted to save time testing the bombs for failures. After three days of work Wald came up with the SPRT (Sequential Probability Ratio Test). The SPRT is a sequential likelihood test, similar to
the Neyman Pearson statistical test (Neyman and Pearson 1933), however it can stop at any interim point if enough confidence is established at rejecting the null hypothesis. Not only did he solve the problem quickly, he also found the optimal solution that worked so well in practice that the military decided to make it a military secret. The test remained a military secret until after the end of WWII when the US won the war and it was released to the public. After its release to the public the test was adopted by the clinical testing community, and became one of the building blocks in clinical testing.

The SPRT is a very clean and simple sequential statistical test. The test however requires the practitioner to have two competing simple hypotheses and sample generating distributions. Both condition are typically not assumed by the discriminative Machine Learning community. I have heard a lot about the tension between the Discriminative school of thought and the Statistics community. I asked Vladimir Vapnik about the nature of this discord, he explained that the gap between the discriminative approach and the statistical approach has its root way back in philosophical debates between instrumentalists and realists. The members of the discriminative school of thought are Instrumentalists, who use observations to make predictions without knowledge of the underlying data generating distribution. The statisticians are realists, who model the underlying process that generated the observations. The gap between the two schools of thought is sometimes bridged, but most of the time researchers tend to stay within their school of thought. One of the nice side effects of this thesis is that it bridges between these two worlds. The bridge lets us solve discriminative machine learning problems with non-parametric statistical methods in ways that are not natural to either discipline.

I noticed that Abraham Wald was at Columbia University at the time, and I wondered whether there has been a continuation of his work at Columbia. After some investigation I found Professor Zhiliang Ying in the Statistics Department. We met and decided to work on connecting sequential testing with discriminative margin-based learning. The approach that was taken was neither typical for the statistical community, nor was it typical for the machine learning community. Nevertheless, our long meetings every Thursday at 1:30pm yielded a common language that bridged the two worlds (see Figure 1.2). After about a year since Mike and I first stumbled upon this problem, our first mathematically justified
CHAPTER 1. INTRODUCTION

The next six months of intensive meetings yielded three more solutions for the stopping boundary under varying assumptions.

The first solution came by defining a decision error, an error which occurs when an example is filtered but it should not have and bounding the probability of making such a decision error with Hoeffding’s inequality. The second and superior solution was formed by bounding the probability of the decision error event by applying the central limit theorem. The third looked at a different definition of a decision error event and used the reflection principle for random walks together with the Central Limit Theorem. The fourth and final approach assumed that making a decision error is a rare event, in which case a different bound can be applied by using a Brownian bridge to describe the event.

This variety of statistical approaches was always developed under two main constraints: the resulting boundary should be non-parametric as assumed by the discriminative learning community, and it should be simple to apply. Indeed most of our boundaries perform exceptionally well on real data and require a few lines of code to implement.

The journey from MERL to Columbia has given me an exciting opportunity to connect
two worlds that are not that close to each other, a bridge which has a lot of gaps waiting to be filled. I’d like to thank my advisors Dave Waltz, Zhiliang Ying and Mike Jones for their tremendous teachings, deeply enlightening conversations, infinite patience, and strong belief in me and this exciting research topic.

1.1 Main Contributions

By connecting sequential analysis with discriminative machine learning we found novel decision-making thresholds to speed up margin-based learning and prediction. We derived four different boundaries which have different characteristics in terms of their accuracy and speed tradeoff. Our sequential tests (STST) are very simple to implement, and are typically tight under the statistical assumptions made.

We applied the boundaries to different learning algorithms such as Online Boosting, Pegasos, and the Perceptron, and have witnessed substantial speedup of the original algorithms without significant loss in accuracy. Our equations can be applied anywhere a thresholded sum is evaluated sequentially, and will result in the order of square root amount of computation. For example our sped up Attentive Pegasos is about 10 times faster than then original Pegasus, and is even more accurate as well on MNIST classification tasks.

Finally, the STST can be viewed as a general attention mechanism for margin-based learning algorithms. Where the attentive algorithm, which is the sped up algorithm by the STST, focuses most of its computation on hard to classify examples. Therefore the attentive algorithm will quickly filter examples that are not important for learning, and focus only on the important ones with high probability.

1.2 Outline

This thesis is structured as follows. In Part I, we present relevant generalization properties of margin-based algorithms. First we look at the generalization capabilities of any learning algorithm, and then margin-based learning algorithms. Once we have established the generalization capabilities of margin-based classifiers, we take a look at the generalization capabilities of their budgeted counterparts. These are margin-based algorithm that are lim-
ited to evaluate a fixed number of features per example. We note that these generalization results do not hold to our stochastic method since we do not have a fixed budget. In Part II we describe the Sequential Thresholded Sum Tests which are the center of the thesis. We develop four different tests which have different characteristics. The first two STSTs result in curved threshold boundaries and are the most conservative. Then we develop the linear STST which is more aggressive, and finally the constant STST which is less aggressive but is very easy to compute. We also look at the expected stopping time of these boundaries. In Part III we apply the STST to different margin-based algorithms and test them on synthetic and real datasets. Finally, in Part IV we conclude the work.
Part I

Introduction to Margin-Based Learning on a Feature Budget
Chapter 2

Margin-based Machine Learning

Discriminative margin-based learning has become one of the corner stones of the Machine learning community. The perceptron of Rosenblatt (1958) is one of the earliest algorithms that incorporates the margin in its training process. The perceptron updates its model every time an example is misclassified. The misclassification is computed as the disagreement between the prediction of the perceptron model, and the label of the example, which is essentially the definition of the margin. The emphasis of the margin’s importance was later developed by Vapnik and Chervonenkis (1971) through their ground work on Structural Risk Minimization. This concept related the generalization error rate of a classifier to its empirical error independently of the data-generating distribution. This was accomplished through the definition of the VC dimension, a quantity which measures the expressiveness of the learning function. Based on this research, the connection between the margin attained by a learning algorithm and its generalization error was proved. As a result of the new theory the ubiquitous SVM (Support Vector Machine) algorithm was developed by Boser et al. (1992); Cortes and Vapnik (1995). These central developments put the margin as one of the central building blocks of discriminative machine learning.

With the explosion of the amount of available data and its high dimensionality, recent research has focused on paradigms where the learner is not interested in evaluating the full margin, thereby making decisions while only evaluating a subset of the features that are available to the learning algorithm. This limit on the access to the number of features that are evaluated changes the generalization ability of the classifier. In this chapter, a
theoretical connection between the generalization capabilities of a full linear classifier and its empirical error is made. The following chapter looks at the effects that a feature budget would have on the generalization abilities of a linear classifier.

2.1 Related Work

Margin-based learning has spurred countless algorithms in many different disciplines and domains. Typically a margin-based learning algorithm evaluates the sign of the margin of each example and performs a decision. Our work provides early stopping rules for the margin evaluation when the result of the full evaluation is obvious. This approach lowers the average number of features evaluated for each example according to its importance. Our stopping thresholds apply to the majority of margin-based learning algorithms.

The most easily applicable machine learning algorithms are margin-based online learning algorithms. Many margin-based Online Algorithms base their model update on the margin of each example in the stream. Online algorithms such as Kivinen and Warmuth’s Exponentiated Gradient [Kivinen and Warmuth 1997] and Oza and Russell’s Online Boosting [Oza and Russell 2001] update their respective models by using a margin-based potential function. Passive online algorithms, such as Rosenblatt’s perceptron [Rosenblatt 1958] and Crammer et al.’s online passive-aggressive algorithms [Crammer et al. 2006], define a margin-based filtering criterion for update, which only updates the algorithm’s model if the value of the margin falls below a defined threshold. All these algorithms fully evaluate the margin for each example, which means that they evaluate all their features for every example. Recently Shalev-Shwartz and Srebro (2008); Shalev-Shwartz et al. (2010) invented Pegasos, an online stochastic gradient descent based SVM solver. The solver is a stochastic gradient descent solver that produces a maximum margin classifier at the end of the training process.

The above mentioned algorithms evaluate all the features for each given example in the stream. However, if there is a cost assigned to a feature evaluation we would like to design an efficient learner that actively chooses which features it would like to evaluate. As an example, if we want to design a classifier for cancer, we would want to evaluate a few
features before the algorithms diagnoses the illness. One of the first works which addressed this notion of cost sensitive active attribute selection is of [Greiner et al. (1996)]. In their work they modified the PAC (Valiant 1984) framework by making an assumption that the learner observes partially specified instances; however the learner has access to complete instances, but must actively request the attribute values at a cost. The authors analyzed the expected cost of obtaining attributes as well as classification accuracy. Similar work on the idea of learning with a feature budget was first introduced to the machine learning community by [Ben-David and Dichterman (1998)]. The authors introduced a formal framework for the analysis of learning algorithms with restrictions on the amount of information they can extract, specifically allowing the learner to access only a fixed number of attributes that is smaller than the entire set of attributes. They presented a framework that is a natural refinement of the PAC learning model; but traditional PAC characteristics do not hold in this framework. Later [Greiner et al. (2002)] extended their work to deal with learning under a constant feature budget. They showed that indeed PAC-learnability is possible even without access the full feature set. Very recently, both [Cesa-Bianchi et al. (2010a)] and [Reyzin (2010)] studied how to efficiently learn a linear predictor under a feature budget.

Similar active learning algorithms were developed in the context of when to pay for a label (as opposed to an attribute). Such active learning algorithms are presented with a set of unlabeled examples and decide which examples labels to query at a cost. The algorithm’s task is to pay for labels as little as possible while achieving specified accuracy and reliability rates (Dasgupta et al. 2005; Cesa-Bianchi et al. 2006; Settles 2009). Typically, for selective sampling active learning algorithms the algorithm would ignore examples that are easy-to-classify, and pay for labels for harder-to-classify examples that are close to the decision boundary.

Our work stems from connecting the underlying ideas between these two active learning domains, attribute querying and label querying. The main idea is that typically an algorithm should not query many attributes for examples that are easy to classify. The labels for such examples, in the label query active learning setting, are typically not queried. For such examples most of the attributes would agree on the classification of the example, and therefore the algorithm need not evaluate too many before deciding the importance of such
examples. As a simple example to the approach, assume we have a budgeted algorithm that can only query 10 features out of a set of 20 features. The algorithm will query these 10 features regardless of how easy the example is to classify. Our approach on an easy-to-classify example might stop once six features vote the same way, saving four features, and using them later on. If the second example is hard to classify we might then use 14 features to classify it. Our allocation of computation resources might be better, we might improve the accuracy of the algorithm, and on average we would perform the same amount of computation as the budgeted algorithm.

2.2 Notation

Throughout the thesis we assume that the data is generated from a fixed and unknown distribution $D$. Each data vector is sampled independently, and has an associated “truth” label $(x, y)$. The sample is $n$ dimensional $x \in \mathbb{R}^n$, and the label belongs to the set $y \in \{-1, +1\}$. Therefore the environment is modeled as a joint distribution over the instance and label space $\mathbb{R}^n \times \mathbb{R}$. We assume that the training $S$ set is composed of $m$ labeled training examples $S = \{(x_i, y_i)\}_{i=1}^m$. Let $H$ denote the space from which base classifiers are chosen. And let $h \in H$ be a mapping from instance space to label space $D \mapsto \{-1, 1\}$. We would like the learning machine to find the mapping $x_i \mapsto y_i$. The learning machine has access to a set of functions that map $x \mapsto f(x, w)$, where $w$ are adjustable parameters of the mapping function. In the case of a linear classifier, the $w$ parameters correspond to the linear model weights, and $f$ is a sum of weak hypotheses $f(x, w) = \sum_{i=1}^{n} w_i x_i = \langle x, w \rangle$. The performance of the learning algorithm is measured by a loss function $l(\langle x, w \rangle, y)$. An example for such a function is the squared loss function $l(\langle x, w \rangle, y) = l(\langle x, w \rangle - y)^2$. Another is the margin $yf(x, w)$, where for prediction mistakes we get a negative margin $yf(x, w) < 0$. We denote $P$ as a cumulative distribution, and $p = dP$ as the corresponding density. Let the probability of event $A$ when an example $(x, y)$ is chosen uniformly at random from the training set be $P_{(x, y) \sim S}(A)$. And let the probability of event $A$ when $(x, y)$ is chosen according to $D$ be $P_{(x, y) \sim D}(A)$. For brevity we abbreviate these by $P_D[A]$ and $P_S[A]$. Let $E_D[A]$ and $E_S[A]$ denote the corresponding expected value. We denote the natural logarithm as $\log = \ln$, and
specify the base otherwise.

2.3 Structural Risk Minimization

The margin has played a central role in discriminative machine learning. The attention to the importance of the margin came as a result of Vapnik’s work in the late 70’s on the uniform convergence of empirical quantities to their actual value (Vapnik and Kotz 1982; Vapnik 1998). These bounds established the relationship between the empirical error of a classifier to the real error. Vapnik proved that this relationship is a function of the number of training examples and the VC dimension of the classifier, which is essentially the capacity of the learned model (see Definition 1).

To solve a learning task, the learning machine is given a finite set of training data that it can use for learning a predictive model that will have good performance on new unseen data. The model trades off two quantities, the accuracy attained on the training data and the capacity of the learning model. The capacity measures the ability of the machine to learn any training set without error. If the capacity is too low the machine would not be able to generalize well since it will have the same prediction on all the data. On the other hand if the capacity is too high, the machine will be over specific and will not be robust to variations within the class. A good tutorial by Burges can be found at (Burges 1998).

The follow theorem proven by Vapnik shows the relationship between the empirical error of the learned model and its generalization capability.

**Theorem 1.** [Vapnik 1999]. If the generalization error of a trained machine is

\[
R(w) = \int \frac{1}{2} |y - f(x, w)| dP(x, y),
\]

and the empirical risk is

\[
R_{emp}(w) = \frac{1}{2m} \sum_{i=1}^{m} |y_i - f(x_i, w)|.
\]

Then for any \( \delta \in [0, 1] \) with probability \( 1 - \delta \) the following bound holds

\[
R(w) \leq R_{emp}(w) + \sqrt{\frac{h(\log(2m/h) + 1) - \log(\delta/4)}{m}}
\]

\[
= R_{emp}(w) + \tilde{O} \left( \sqrt{\frac{h}{m}} \right),
\]
Figure 2.1: A two dimensional line can shatter three points. Equivalently, for any configuration of labels, the points can be separated by a line.

where \( h \) is the VC dimension, and is a measure the capacity of the learning machine.

The VC dimension will be discussed in the next section. However, as an example, a learning machine with high capacity would over-specialize, and with low capacity oversimplify concepts. The tradeoff is between the \( R_{emp} \) which is low for an over specialized learning machine, and the VC dimension \( h \) which would be high. A similar tradeoff exists between simple low VC dimension learners and their high empirical error.

The bound is not dependent on the distribution of the data \( P(x,y) \), and therefore for a choice of \( \eta \) and \( m \), will give us a way to find the best learned model for any distribution. This bound essentially gives us a mechanism to choose between different learning machines. Learning machines that over specialize will have typically low empirical error but high VC dimension \( h \), whereas machines that over generalize will have high empirical error but low VC dimension. By comparing the different machines’ performance through Vapnik’s inequality we can choose the machine that with high probability will have the lowest true error. Although the bound is not tight, it does typically give a good empirical indication of better predictive models.

2.4 The VC Dimension

The VC dimension is a measure of capacity of a function class. The capacity is the number of examples that a function can shatter. Shattering is the ability to produce the correct
label assignment to any labeling permutation of the points. For example if we have $m$
points, then we have $2^m$ possible labelings. If a function is found within the function class
that can produce perfect classification on all the label assignments, we say that its VC
dimension is $m$. More strictly, the VC dimension is the largest number of points that can
be shattered by the function class.

**Definition 1.** *(Vapnik 1999)* The VC dimension of a set of indicator functions $Q(x, w)$ is
the maximum number $h$ of vectors $x_1, ..., x_h$ that can be separated into two classes in all $2^h$
possible ways using functions of the set. If for any $h$ there exists a set of $h$ vectors that can
be shattered by the set $Q(x, w)$, then the VC dimension is equal to infinity.

**Example 1.** The VC dimension of the set of linear indicator functions

$$Q(x, w) = \text{sign} \left\{ \sum_{p=1}^{n} w_p x_p + w_0 \right\}$$

in n-dimensional coordinate space $X = (x_1, ..., x_n)$ is equal to $h = n + 1$, since by using
functions of this set one can shatter at most $n + 1$ vectors.

**Example 2.** The VC dimension of the set of linear functions

$$Q(x, w) = \sum_{p=1}^{n} w_p x_p + w_0$$

in n-dimensional coordinate space $X = (x_1, ..., x_n)$ is equal to $h = n + 1$, since by using
functions of this set one can shatter at most $n + 1$ vectors, and the VC dimension of the
corresponding linear indicator functions is equal to $n + 1$. For a full proof of the VC
dimension of linear classifiers see Burges (1998) theorem 1.

**Theorem 2.** *(Burges 1998) theorem 1.)* Consider some set of $m$ points in $\mathbb{R}^n$. Choose
any one of the points as origin. Then the $m$ points can be shattered by oriented hyperplanes
if and only if the position vectors of the remaining points are linearly independent.

**Corollary 1.** The VC dimension of the set of oriented hyperplanes in $\mathbb{R}^d$ is $d + 1$, since
we can always choose $d + 1$ points, and then choose one of the points as origin, such that
the position vectors of the remaining $d$ points are linearly independent, but can never choose
$d + 2$ such points (since no $d + 1$ vectors in $\mathbb{R}^h$ can be linearly independent).
Consequently, in the linear case, Vapnik argued that the linear separator that achieves the largest margin on the data tends to give the best generalization error. See figure 2.2 for an intuitive realization of the margin. Following Vapnik’s original work, Cortes and Vapnik proposed the first linear Support Vector Machine, which finds the hyperplane that maximizes the margin of separation between the two classes of data. Their work extended the old work by adding slack variables that allowed the algorithm to give up on classifying some of the data correctly, in hope to find a hyperplane that obtains a larger margin, and thereby achieves lower generalization error.

2.5 Generalization Capabilities of Large Margin Classifiers

In the linear case all the separating hyperplanes of the same dimension will have the same VC dimension, and therefore the same generalization capabilities according to the basic VC theory. It is well known that hyperplanes that achieve large margin separation of the data, tend to generalize better than other separating hyperplanes. The following theorem establishes relationship between large margin classifiers and their VC dimension.

**Theorem 3.** The VC dimension of the set of $\rho$–margin separating hyperplanes. Let the vectors $x \in \mathcal{R}^d$ be bounded by a sphere of radius $R$. Then the VC dimension $h$ of the set of hyperplanes with margin $\rho = (\langle w, w \rangle)^{-1}$ has the bound $h \leq \{\langle w, w \rangle, d\} + 1$.

**Proof.** A geometrical proof of the bound can be found in Chapter 10, section 5 of Vapnik (1982). Also, an algebraic proof can be found in Gurvits (1997).

While theorem 3 relates the margin to the VC dimension, the relationship looks at the maximum margin and does not look at the entire distribution of margins on the data. Following the success of AdaBoost, Schapire et al. (1998) followed proof techniques from Shawe-Taylor et al. (1998) and Bartlett (1998) to show the generalization capabilities of a learning machine to be a function of the margin distribution. In particular the authors prove that achieving a large margin on the training set results in an improved bound on generalization. Nevertheless, in their bound there is a tradeoff between the probability of achieving a large margin and the margin achieved.
Figure 2.2: The SVM achieves a larger margin than the perceptron. According to theorem 4, it generalizes better.

The following theorem from Schapire et al. (1998) states that with high probability, the generalization error of any majority-vote classifier can be bounded in terms of the number of training examples with margin below a threshold plus an additional term that depends on the number of training examples, the VC dimension of the function class, and the margin threshold. The theorem below applies to the case where the base classifier space is finite, which covers the algorithms that are discussed throughout the rest of this thesis.

**Theorem 4.** (Theorem 1 of Schapire et al. (1998)) Let \( D \) be a distribution over \( X \times \{−1, +1\} \), and let \( S \) be a sample of \( m \) examples chosen independently at random according to \( D \). Assume that the base-classifier space \( \mathcal{H} \) is finite, and let \( \delta > 0 \). Then with probability at least \( 1 − \delta \) over the random choice of the training set \( S \), every weighted average function
f \in C satisfies the following bound for all \( \theta > 0 \):

\[
P_D[yf(x) \leq 0] \leq P_S[yf(x) \leq \theta] + O\left(\frac{1}{\sqrt{m}} \left(\frac{\log m \log_2 |\mathcal{H}|}{\theta^2} + \log (1/\delta)\right)^{1/2}\right). \tag{2.1}
\]

We can simplify the above theorem by the following

\[
P_D[yf(x) \leq 0] \leq P_S[yf(x) \leq \theta] + \tilde{O}\left(\frac{\sqrt{n}}{m \theta^2}\right),
\]

where we set \( n = \log_2 |\mathcal{H}| \). The theorem provides a good bound for large datasets. If the training achieves a large enough margin on all samples, the first quantity on the right hand side is equal to zero. For the second quantity, if for example \( \theta = 1/2 \) and the number of features is \( n = 1000 \), then we would need in the order of \( m = 100,000 \) examples to obtain an upper limit of 10% on the test error, or equivalently 90% accuracy on test data.

In the next chapter we look at these bounds in a slightly different setting. When features are expensive, and the practitioner wants to limit the number of features he/she will query, the number of features that need to be evaluated is smaller than the size of the entire set of features. This field of research is typically referred to as Learning on a Feature Budget, where the learner has a fixed budget on the number of features it can evaluate for any example.
Chapter 3

Learning on a Feature Budget

The linear classifiers described in the previous chapter are passive. They evaluate all the features for every example in the dataset. In contrast, an active classifier can at some cost decide which attributes it wants to see. In many cases an active classifier can be used when it is desirable to evaluate fewer features. Such is the case in medical testing, where a doctor would like to find out the patient’s condition while administering as few tests as possible. Medical tests tend to be very costly, and might even harm the patient. When a doctor tries to diagnose whether a patient has a certain condition (or not) he has a set of diagnostic tests at his disposal. From this entire set of tests the doctor will typically only administer a subset after which the diagnosis can be performed with high accuracy. Typically if a sequence of these tests returns negative, the doctor would stop the testing procedure and determine that the patient is healthy.

In the case of diagnosing lung cancer, the doctor would test the patient sequentially. First the doctor would test for respiratory symptoms, then, if the test is positive, the doctor would test for abnormalities in an x-ray. If that test comes back positive the doctor may perform a CT scan or MRI to get a better picture, and continue the testing procedure. At any point in the testing procedure, the doctor can stop the test sequence and diagnose the patient.

Similarly to the doctor, the feature active classifier evaluates a sequence of features from a set of possible features. At each point in the sequential feature evaluation procedure the feature active classifier can choose which features it wants to evaluate next (with replace-
ment), and consequently make the decision whether to stop evaluating features and predict
the class of the sample, or to continue evaluating more features. The active classifier may
stop the evaluation of features at any time, and does not have to fully evaluate the entire
feature set for each example to predict its class.

3.1 Related Work

The topic of learning active classifiers where the evaluation of the features carries a cost has
been studied in the machine learning literature. A good review of active learning both in
terms of feature selection and example selection was done by Blum and Langley (1997). In
their work they stress the importance of connecting the work on membership query models,
where the learning algorithm needs to pay to see a label, with the work on filtering unlabeled
instances. We indeed also stress this connection throughout our work.

We concentrate on the development of feature active learning in this section. Some of
the original works in the field included learning with incomplete instances, where only a
part of the feature vector was observed. The most famous work was done by Dempster et al.
(1977) which resulted in the EM algorithm. Other works such as Little and Rubin (1987);
Quinlan (1989); Schuurmans and Greiner (1994) dealt with learning with incomplete data.
However the methods presented in these works were not active, and assumed that some of
the features simply could not be observed.

Feature active learning has two main paradigms of research. Both stem from the PAC
learning paradigm, and state the problem of looking for a feature active classifier which
best approaches the error-rate of the equivalent full feature classifier. The first paradigm
Global Budget Constraint limits the total cost that is allocated for feature querying over all
examples. In contrast, the second paradigm Local Budget Constraints limits the cost allowed
for each example. Both paradigms and works thereafter also typically make a distinction
between learning and prediction.

Both paradigms allow the classifier to query feature values at a cost from a set of
available features. In the global budget constraint setting, the learner is allowed to query
a limited number of \( k \) attributes throughout training. The learner has the choice of how
to spend the feature budget between all the features and all the examples. The budget is hard, which means that once the learner paid $k$ on the feature queries the learning process stops.

One of the earliest works on learning with global budget constraints was done by Greiner et al. (1996) and extended later in Greiner et al. (2002). The authors proposed a modification of the PAC learning framework, where unlike the PAC setting, the active learner is presented with examples with hidden feature values. The learner has to pay a cost to reveal or query a feature to obtain its value. At the end of the learning process the learner returns the features that are important for learning the target concept, and only these features are later used for prediction. Greiner et al. show that traditional PAC learning efficiency holds in this new setting, although they prove that finding the optimal minimal cost classifier is NP-hard. Nevertheless, they also prove that an arbitrarily close to optimal classifier can be learned efficiently.

Based on this framework Lizotte and Madani (2003) and Kapoor and Greiner (2005) propose efficient learning algorithms for finding the classifier that is close to optimal. To achieve this task they assume that the features are Naive Bayes, which means that they are conditionally independent given the class label. They propose a few different search algorithms that trade off exploration exploitation of querying features within the same example or continue to other examples. Extending the idea of exploration vs. exploitation, Deng et al. (2007) propose to conduct the feature search across examples with the multi-armed bandit approaches of Auer et al. (1995) and Kalai and Vempala (2003).

A slightly different learning framework, Restricted Focus-of-attention (RFA), was proposed by Ben-David and Dichterman (1993, 1998); Birkendorf et al. (1998). Unlike Greiner et al. (2002) this work puts a budget $k$ on the number of features that can be evaluated by the classifier for each example, and therefore imposes a local budget constraint. Essentially, if we set the local budget constraint to $k/m$ we get a specific training method which will ensure a local budget learning algorithm maintains a global budget of $k$ with $m$ examples.

The learner under the k-RFA (Restricted Focus-of-attention) framework evaluates $k$ attributes out of the $n$ attributes existing in each instance. Ben-David and Dichterman also extend the k-RFA model by making weaker restrictions on the learner. The extension,
Figure 3.1: Two examples are classified. The first is hard to classify, the second easy. The budgeted learning approach would evaluate the same number of features for both examples, whereas the stochastic would evaluate features according to how hard is the example to classify, while maintaining an average budget.

k-weakly restricted focus-of-attention (k-wRFA), allows the learner to evaluate up to $k$ bits, where each bit can be a function of the input. This allowed k-wRFA learners to learn concepts which are not learnable by k-RFA learners.

Our work stems from the k-RFA framework where we query feature values sequentially for each example. We try to query as few features as possible per example. In fact our decision of how many features to query adapts according to the feature values. The k-RFA framework is a natural setting in an online learning paradigm, where the learner and classifier are presented with examples sequentially, and need to make decisions per example. Conversely, the global budget constraint is more suitable for batch algorithms, where all the examples are presented to the algorithm before the training begins.

The restricted focus-of-attention framework has been studied by Cesa-Bianchi et al. (2010a) in the context of regression, and worst case scenarios were analyzed as well by Cesa-Bianchi et al. (2010b). Related ideas and generalization bounds were proposed by Reyzin (2010); however, Reyzin’s work only deals with prediction and not learning.

Throughout the next section we establish theoretical results that are applicable to our stochastic focus-of-attention algorithm. Our algorithm was developed using statistical methods on general sums of features, and it can be directly applied in the k-RFA framework. Because of the stochastic nature of our algorithm, we cannot use a hard budget, but rather look at the performance in expectation as shown in Figure 3.1.
CHAPTER 3. LEARNING ON A FEATURE BUDGET

Prediction on a budget, rather than learning on a budget, allows the predictor to query at most \( k \) attributes. Such a setting was studied by Greiner et al. (1996, 2002) where all the features were available to the learner, and a partial set of features was available during testing. The next section shows that theoretically these two problems are different.

3.2 Generalization

The generalization inequalities from Section 2.5 hold in the case where all the base-classifiers are evaluated, which might be costly if there are thousands of base classifiers to evaluate. In such a scenario, one would want to find a learning algorithm that has access to all the base classifiers, but requires fewer base-classifier evaluations, while achieving the same generalization capabilities. The largest number of base-classifiers that the algorithm is allowed to evaluate is called the budget.

Theoretically, interesting questions arise when we consider learning on a feature budget. How would evaluating less base-classifiers instead of the entire set affect generalization? Is there a minimal number of base-classifiers which need to be evaluated to guarantee that the budgeted learner maintains similar generalization accuracy to the full learner? Is there a difference between the number of features needed for learning and prediction? In the following section we answer these questions.

We start by looking at the expected generalization power of a budgeted classifier. The theoretical guarantee was derived by Reyzin (2010) in the case where the learner has access to \( k \) base learners at test time, and was trained using \( n \) base classifiers. In his work the generalization of the budgeted classifier is related to the budget. To tie the two quantities together Reyzin applied Hoeffding’s inequality (Theorem 5) to Schpire et al.’s margin-based generalization (Theorem 4.)

**Theorem 5.** (Reyzin (2010)) Under conditions of Theorem 4, if algorithm AdaBoostRS (a budgeted AdaBoost) is run, taking \( k \) samples of hypotheses, where the hypothesis space has VC-dimension \( h \), then with probability at least \( 1 - \delta \),

\[
P_D[yf(x) \leq 0] \leq P_S[yf(x) \leq \theta] + \tilde{O} \left( \sqrt{\frac{h}{m\theta^2}} \right) + e^{-k\theta^2/2}.
\]
This theorem states the generalization error of the budgeted version of AdaBoost converges exponentially fast to that of AdaBoost as the budget grows.

We have established the relationship between the generalization of a linear classifier, and its empirical error rate. Furthermore, a relationship between a budgeted linear predictor and its generalization error was also established. However, there is a minimal budget that is required for learnability. In their work Cesa-Bianchi et al. (2010b) show that there is a limit to how small the budget can be, while allowing the learner to learn.

**Theorem 6.** (Cesa-Bianchi et al. (2010b)) For any $\epsilon \in (0, 1/16)$, there exists a distribution over examples and a weight vector $w^* \in \mathbb{R}^n$, with $||w^*||_0 = 1$ and $||w^*||_2 = ||w^*||_1 = 2\sqrt{\epsilon}$, such that any learning algorithm must see $\Omega\left(\frac{n}{\epsilon}\right)$ attributes in expectation in order to learn a linear predictor $w$ with $L_D(w) - L_D(w^*) < \epsilon$.

Cesa Bianchi et al. also prove the minimal number of features that are required to learn a concept with $\epsilon$ accuracy, and a budget of $k$ features.

**Theorem 7.** (Cesa-Bianchi et al. (2010a)) For any $\epsilon \in (0, 1/16)$, $k$, and $n > 4k$, there exists a distribution over examples and a weight vector $w^*$, with $||w^*||_0 = 1$ and $||w^*||_2 = ||w^*||_1 = 2\sqrt{\epsilon}$, such that any attribute efficient regression algorithm accessing at most $k$ attributes per training example must see at least $\Omega\left(\frac{n}{k\epsilon}\right)$ attributes in expectation in order to learn a linear predictor $w$ with $w$ with $L_D(w) - L_D(w^*) < \epsilon$.

We have gone over the generalization capabilities of general learning algorithms. Then, we analyzed the generalization abilities of arbitrary linear classifiers and in particular large margin classifiers. Subsequently, we discussed what happens to the generalization capabilities of large margin classifiers when put under a strict budget. Now that we established the generalization capabilities of budgeted classifiers, Part II of the thesis will propose the use of stochastic processes to create such classifiers.
Part II

Sequential Thresholded Sum Tests
Chapter 4

Sequential Statistical Tests for Thresholded Sums

4.1 Introduction

The running time of margin-based online algorithms is a function of the number of features, or the dimensionality of the input space. Since models today may have thousands of features, running time seems daunting, and depending on the task, one may wish to speed-up these online algorithms, by pruning uninformative examples. We propose to early stop the computation of feature evaluations for uninformative examples by connecting Sequential Analysis (Wald (1945); Lan et al. (1982); Siegmund (1985); O’Brian and Fleming (1979); Pocock (1977); Jennison and Turnbull (1999)) to margin-based learning algorithms.

Many decision-making algorithms make a decision by comparing a sum of observations to a threshold. If the sum is smaller than a pre-defined threshold, then a certain action is taken, otherwise a different action is taken or no action is taken. This type of reasoning is prevalent in the margin-based Machine Learning community, where typically an additive model is compared to a threshold, and a subsequent action is taken depending on the result of the comparison. Margin-based learning algorithms average multiple weak hypotheses to form one strong combined hypothesis - the majority-vote. For training, the combined hypothesis is usually compared to a threshold to make a decision about when to update the algorithm. For testing, the combined hypothesis is compared to a threshold to make a
predictive decision about the class of the evaluated example.

For example, there has been extensive research in the Computer Vision community on early stopping the computation of the margin or the score of examples. One example of this research is the well known Viola-Jones face detector (Viola and Jones 2001; Jones and Viola 2003) which uses a cascade of classifiers to quickly reject non-face image patches. The Viola-Jones cascaded face detector is a discriminative classifier that is trained using batch AdaBoost (Freund and Schapire 1997). Following this work, Bourdev and Brandt (2005) replaced the cascaded classifier with a soft cascade that uses a set of rejection thresholds, one after each weak classifier, to determine whether to continue or to reject the example as a non-face. Attempting to make the tradeoff between early stopping and accuracy mathematically optimal, Matas and Sochman (2007) propose WaldBoost, which puts Wald’s sequential test into a boosting framework. Our work differs by means of development of a new theory that does not require a likelihood function over the votes of weak hypotheses, and is therefore purely discriminative.

With the rapid growth of the size of data sets, both in terms of the number of samples, and in terms of the dimensionality, margin-based learning algorithms can average thousands of hypotheses to create a single highly accurate combined hypothesis. Evaluating all the hypotheses for each example becomes a daunting task since the size of the data set can be very large, in terms of number of examples and the dimensionality of each example. In terms of number of examples, we can speed up processing by filtering out un-informative examples for the learning process from the data set. The measure of the importance of an example is typically a function of the majority-vote. In terms of dimensionality, we would like to compute the least number of dimensions before we decide whether the majority-vote will end below or above the decision threshold. Filtering out un-informative examples, and trying to compute as few hypotheses as possible are closely related problems (Blum and Langley 1997). The decision whether an example is informative or not depends on the magnitude of the majority-vote that is the amount of disagreement between the hypotheses. Therefore, to find which example to filter, the algorithm needs to evaluate its majority-vote, and we would like it to evaluate the least number of weak hypotheses before coming to a decision about the example’s importance.
Majority-vote based decision-making can be generalized to comparing a weighted sum of random variables to a given threshold. If the majority-vote falls below the pre-specified threshold a decision is made, typically the model is updated, otherwise the example is ignored. Since, the majority-vote is a summation of weighted random variables, or averaged weak hypotheses, it can be computed sequentially. Sequential Analysis allows us to develop the statistical tools needed to speed up this evaluation process when its result is evident. We use the terms margin and full margin to describe the summation of all the feature evaluations, and partial margin as the summation of a part of the feature evaluations. The calculation of the margin is broken up for each example in the stream. This break-up allows the algorithm to make a decision after the evaluation of each feature whether the next feature should also be evaluated or the feature evaluation should be stopped, and the example should be rejected for lack of importance in training. By making a decision after each evaluation we are able to early stop the evaluation of features on examples with a large partial margin after having evaluated only a few features. Examples with a large partial margin are unlikely to have a full margin below the required threshold. Therefore, by rejecting these examples early, large savings in computation are achieved.

This thesis proposes several simple novel methods based on Sequential Analysis (Wald [1945], Lan et al. [1982]) and stopping methods for Brownian Motion to drastically improve the computational efficiency of margin-based learning algorithms. Our methods accurately stop the evaluation of the margin when the result of the entire summation is evident.

To find the early stopping thresholds we use ideas from the field of Sequential Analysis which has been an active research field for over 60 years. It is mostly used and developed by the Biomedical community for designing clinical trials. In a clinical trial the researcher would like to design a test that requires the smallest number of patients to prove the efficacy of a drug or treatment. The cost of many tests is monetarily high and patients may die throughout the test. Therefore the tests are designed sequentially, so that the minimum number of patients will be tested. The connection between the clinical tests and margin evaluation is done by looking at each feature as a patient, and the margin evaluation as the patient population’s response to the drug. Therefore, a sequential test can be designed to stop the margin evaluation when the interim evaluation of the margin has high probability
that full evaluation will have the same outcome. If at any interim point we have enough confidence that the full evaluation will determine that an example is uninformative, we will early stop the test, reject the example, and save on computational costs while maintaining accuracy (see Figure 6.1.)

A tradeoff between accuracy and speed is established. Instead of looking at the traditional classification error we look at decision error. Decision errors are errors that occur when the algorithm rejects an example that should be accepted for training. Given a desired decision error rate we would like the test to decide when to stop the computation. This test is adaptive, and changes according to the partial computation of the margin. The generalization error of the classifier can be calculated through Theorem 5. We demonstrate that this simple test can speed-up Online Boosting by an order of magnitude while maintaining generalization accuracy.

4.2 Sequential Thresholded Sum Tests

The novel Sequential Thresholded Sum Test is a test which is designed to control the rate of decision errors a margin-based learning algorithm makes. Although some of these tests are known in statistics, they have never been applied to learning algorithms before. The connection with discriminative learning has forced us to manipulate the traditional random walk statistics into a discriminative world.

The following sections describe different Sequential Thresholded Sum Tests with different characteristics to control the decision error rate. These different tests would be favorable under different assumptions, and therefore the practitioner would need to choose the appropriate test for speeding up their own margin-based algorithm.

4.2.1 Mathematical Roadmap

Our task is to find a filtering framework that would speed-up margin-based learning algorithms by quickly rejecting examples of little importance. Quick rejection is done by creating a test that stops the margin evaluation process given the partial computation of the margin. We measure the importance of an example by the size of its margin which is the distance
Chapter 4. Sequential Statistical Tests for Thresholded Sums

Figure 4.1: Margin evaluation as a random walk. The margin evaluation is a summation of $n$ summands. At each partial sum, the $i$th feature can add $+X_i$ or $-X_i$ to the partial sum. We are interested in calculating the probability that the random walk after $n$ steps will end below the threshold $\theta$ given that the current position $S_i$ passed a stopping threshold (in red). This probability will give us the stopping boundary shown in red.

from the decision boundary geometrically. We define $\theta$ as the importance threshold, where examples that are important to learning have a margin smaller than $\theta$. Statistically, this problem can be generalized to finding a test for early stopping the computation of a partial sum of independent random variables when the result of the full summation is guaranteed with a high probability (see Figure 4.1).

To design such a sequential test we would first upper bound the probability of making decision errors. There are two types of decision errors. The first occurs when the sum is predicted to end above the importance thresholds, but actually ends below. The second occurs when the sum is predicted to end below the importance thresholds, but actually ends above. In both cases, the summation is stopped at an interim point and the prediction is opposite the result that would have been realized if the entire summation was performed. The two types of errors are depicted in Figures 4.2(a) and 4.2(b).

We look at decision errors of a sum of weighted independent random variables. Then
given a required decision error rate we will derive the Sequential Thresholded Sum Test (STST) which will provide adaptive early stopping thresholds that maintain the required confidence.

Let the sum of weighted independent random variables \((w_i, X_i)\), \(i = 1, ..., n\) be defined by \(S_n = \sum_{i=1}^{n} w_i X_i\), where \(w_i\) is the weight assigned to the random variable \(X_i\). We require that \(w_i \in \mathbb{R}, X_i \in [-1, 1]\). We define \(S_n\) as the full sum, \(S_i\) as the partial sum, and \(S_{in} = S_n - S_i = \sum_{j=i+1}^{n} w_i X_i\) as the remaining sum. Once we computed the partial sum up to the \(i\)th random variable we know its value \(S_i\). Let the stopping threshold at coordinate \(i\) be defined by \(\tau_i\). We use the notation \(ES_{in}\) to denote the expected value the remaining sum.

### 4.2.2 Decision Errors

There are four basic events that we are interested in when designing sequential tests that involve controlling decision error rates. Each of these events is important for different applications under different assumptions. The sequential method looks at events that involve
the entire random walks, whereas the curtailed method looks at events that accumulate information as the random walk progresses. We establish the basic relationship between the sequential method, on the left hand side of the following equations, and the curtailed method on the right hand side by Bayes rule

\[ P(\text{stop before } n|S_n < \theta)P(S_n < \theta) = P(S_n < \theta|\text{stop before } n)P(\text{stop before } n), \quad (4.1) \]

where \( \text{stop} \) is the event which occurs when the partial sum crosses a stopping boundary.

Our first proposed curtailed test, the Curved STST (Curved Sequential Thresholded Sum Test) is derived by looking at the following “curtailed” conditional probability

\[ P(S_n < \theta|\text{stop before } n) = \frac{P(S_n < \theta, \text{stop before } n)}{P(\text{stop before } n)}. \quad (4.2) \]

As we will witness, the simplicity of deriving the curtailed method stems from the fact that neither the joint probability nor the stopping time probability need to be explicitly calculated to upper bound this conditional probability. Our resulting first stopping boundary, the Curved STST, gives us a constant conditional error probability throughout the curve, which means that it is a rather conservative boundary.

We then develop a more aggressive boundary that allows higher decision error rates at the beginning of the random walk and lower decision error rates at the end. Such a boundary would essentially stop more random walks early on, and less later on. In machine learning, this approach has the natural interpretation that we want to shorten the feature evaluation for obvious un-important samples, but we want to prolong the evaluations for samples we are not sure about. A linear boundary achieves this exact “error spending” characteristic and is derived by controlling the joint decision error event \( P(S_n < \theta, \text{stop before } n) \).

Finally, we develop the Brownian bridge STST, a constant boundary by looking at the following decision error \( P(\text{stop before } n|S_n < \theta) \). This boundary is desirable if the event \( \{S_n < \theta\} \) is rare, therefore controlling the joint decision error rate \( P(\text{stop before } n, S_n < \theta) \) is not effective.

See figure 4.3 for a graphic representation of the resulting boundaries under a simple setting.
4.3 General Probability Inequalities for Thresholded Sums

We are interested in bounding the probability of making a decision error. First, we define and analyze a decision error as the event when the full sum of weighted independent random variables is smaller than a given threshold but the partial sum passed above an early stopping threshold. See subfigure 4.2(a). Second, we define and analyze a decision error as the event when the full sum of weighted independent random variables is larger than a given threshold but the partial sum passed below an early stopping threshold. See subfigure 4.2(b).

Before we analyze the probability of making decision errors, we state Hoeffding’s fundamental inequality. We will use it shortly to bound the probability of making a decision error. Hoeffding’s inequality upper bounds the probability of a sum $S_n$ of independent random variables $\{X_i\}_{i=1}^n$ deviating from its expectation by more than $t$ where the summands are bounded $X_i \in [a_i, b_i]$. 
**Theorem 8.** (Hoeffding (1963)) Let $X_1, \ldots, X_n$ be independent real-valued random variables, such that each $X_i \in [a_i, b_i]$, and let $S_n = \sum_{i=1}^{n} X_i$. Then for every $t > 0$,

\[
P[S_n - \mathbb{E}[S_n] < -t] \leq \exp \left( -\frac{2t^2}{\sum_{i=1}^{n} (b_i - a_i)^2} \right)
\]

and also

\[
P[S_n - \mathbb{E}[S_n] > t] \leq \exp \left( -\frac{2t^2}{\sum_{i=1}^{n} (b_i - a_i)^2} \right).
\]

### 4.4 The Curved Sequential Thresholded Sum Test

We derive the Curved boundary STST by requiring that the joint distribution of stopping time and the sum ending below the required threshold is less than a required error rate $\delta$

\[
P(S_n < \theta, \text{stop before n}) < \delta.
\]

We can express the joint event as a summation of conditional decision error events at each coordinate (summand) $i$

\[
P(S_n < \theta; \text{stop before n}) = \sum_{i=1}^{n-1} P(S_n < \theta, \text{stop at i}) \quad (4.3)
\]

\[
= \sum_{i=1}^{n-1} P(S_n < \theta | \text{stop at i}) P(\text{stop at i}) \quad (4.4)
\]

By requiring that each conditional probability satisfies $P(S_n < \theta | \text{stop at i}) \leq \delta$ we get that the joint probability will also be upper bounded by $\delta$

\[
P(S_n < \theta, \text{stop before n}) \leq \delta \sum_{i=1}^{n-1} P(\text{stop at i}) \leq \delta, \quad (4.5)
\]

where $\sum_{i=1} P(\text{stop at i}) < 1$ since we are summing over all the stopping times. Therefore, it suffices to control the conditional probability $P(S_n < \theta | \text{stop at i})$ in order to control the joint decision error rate.

Let an upper decision error be defined by

\[
\text{Upper decision error: } \varepsilon^u_i = \{S_n \leq \theta | \text{stop at i}\}.
\]

To simplify notation we will not differentiate between upper stopping thresholds $\tau^u_i$ and lower ones $\tau^l_i$. It is clear from the context which stopping thresholds are we referring to.
Lemma 1. The probability of making an upper decision error \( \varepsilon_i^u \) for thresholded sums of weighted independent random variables is bounded by

\[
P(\varepsilon_i^u) \leq \exp \left( -\frac{(\theta - \tau_i - ES_{in})^2}{2\sum_{j=i+1}^n w_j^2} \right).
\]

Proof. A decision error is an error that occurs when the computation of a sum is stopped early, when in fact it should have continued. This means that the partial sum passed above a given early stopping threshold \( \tau_i \), when in fact it should not have since the full sum satisfies the importance requirement \( S_n \leq \theta \). Our task is to bound the following probability (see Figure 4.1):

\[
P(\varepsilon_i^u) = P(S_n \leq \theta | \text{stop at } i) \quad (4.6)
\]

\[
\leq P(S_n \leq \theta | S_i = \tau_i) \quad (4.7)
\]

\[
= P(S_n - S_i \leq \theta - \tau_i | S_i = \tau_i) \quad (4.8)
\]

\[
= P(S_{in} \leq \theta - \tau_i). \quad (4.9)
\]

The inequality in equation 4.7 is valid since \( \tau_i \) is closer than \( S_i \) to \( \theta \) by construction, and tight, since once the random walk passes the stopping threshold it is stopped, and its final value is close to the value of the corresponding stopping threshold. The condition in equation 4.9 is dropped since \( S_{in} \) does not depend on \( S_i \).

We can upper bound equation 4.9 by applying Hoeffding’s inequality from Theorem 8. An interesting outcome of this connection is that the resulting threshold would change according to the random walk uncovered until that point \( S_i \).

To apply this bound to thresholded sums we need to convert equation 4.9 to the form used by the Hoeffding bound:

\[
P(S_{in} \leq \theta - \tau_i) = P(S_{in} - ES_{in} \leq \theta - \tau_i - ES_{in}) \quad (4.10)
\]

Equation 4.10 gives an adaptive threshold \( \theta - \tau_i - ES_{in} \) which changes each time a random variable is observed and added to the sum. Let \( t_i = \theta - \tau_i - ES_{in} \), \( a_i = -w_i \), \( b_i = w_i \).

Combining equations 4.9 - 4.10 we get the following inequality

\[
P(S_n \leq \theta | \text{stop at } i) \leq \exp \left( -\frac{(\theta - \tau_i - ES_{in})^2}{2\sum_{j=i+1}^n w_j^2} \right). \quad (4.11)
\]
The threshold $\theta$, the partial sum $S_i$, and the weights $w_j$ are known (see figure 4.1). What is left to calculate the upper bound is to compute the expectation of the remaining sum $ES_m$ which is discussed in Section 5.4.

Lemma 1 can be modified to handle the opposite case, where the decision error is defined as the event where the random walk crosses the stopping threshold from below causing the prediction $S_n < \theta$ where in fact $S_n \geq \theta$. We define

$$\text{Lower decision error: } \varepsilon^l = \{S_n \geq \theta|\text{stopped at } i\}.$$ See figure 4.2(b).

Modifying this lemma 1 to handle decision errors we get the following equivalent lemma

**Lemma 2.** The probability of making a lower decision error $\varepsilon^l$ for thresholded sums of random variables is bounded by

$$P(\varepsilon^l) \leq \exp\left(-\frac{(\theta - ES_m - \tau_i)^2}{2\sum_{j=i+1}^{n} w_j^2}\right).$$

**Proof.** To bound the probability of a decision error, we look at the summation process as a random sum (see Figure 4.1). Where as we compute partial sums we re-evaluate the probability of making a decision error

$$P(\varepsilon^l) = P(S_n \geq \theta|\text{stop at } i)$$

$$\lesssim P(S_n \geq \theta|S_i = \tau_i)$$

$$= P(S_n - S_i \geq \theta - \tau_i|S_i = \tau_i)$$

$$= P(S_{in} \geq \theta - \tau_i|S_i = \tau_i)$$

$$= P(S_{in} - ES_{in} \geq \theta - \tau_i - ES_{in}|S_i = \tau_i)$$

$$= P(-S_{in} - E(-S_{in}) \leq -\theta + \tau_i + ES_{in}|S_i = \tau_i).$$

Again, equation 4.13 is a good approximation since if the random walk ever crosses the decision boundary it is stopped. Therefore, if the random walk is stopped, which is the condition for a decision error, the partial sum is approximately equivalent to the stopping threshold.
\[ P(-S_{in} + E S_{in} \leq -\theta + \tau_i + E S_{in} | S_i = \tau_i) \leq \exp\left( \frac{-(-\theta + \tau_i + E S_{in})^2}{2 \sum_{j=i+1}^{n} w_j^2} \right) = \exp\left( \frac{-(\theta - E S_{in} - \tau_i)^2}{2 \sum_{j=i+1}^{n} w_j^2} \right). \] (4.18)

Lemmas \#1 and \#2 provide us methods to control the decision error made by a stochastic decision-making algorithm. However, we would like to extract the corresponding stopping thresholds \( \tau_i \) given the allowed decision error rate \( \delta \). The following theorem establishes this relationship.

**Theorem 9.** The Sequential Thresholded Sum Test. Lemma \#1 holds with probability at least \( 1 - \delta \) if the stopping boundary is specified by

\[ \tau_i \leq \theta - E S_{in} + ||w_{in}|| \sqrt{\ln \frac{1}{\delta^2}}, \]

where \( ||w_{in}|| = \sqrt{\sum_{j=i+1}^{n} w_j^2} \) is the norm of the maximum variation of the residual random variables. I.e. the random walk stops when the above inequality is violated.

**Proof.** Lemma \#1 gives us an upper bound on the decision error rate, as more information is gathered sequentially. Let us explicitly calculate the early stopping thresholds for a required error rate of at most \( \delta \)

\[ \exp\left\{ \frac{-(\tau_i + E S_{in} - \theta)^2}{2 ||w_{in}||^2} \right\} \leq \delta. \] (4.19)

Solving for \( \tau_i \)

\[ (\tau_i + E S_{in} - \theta)^2 \geq ||w_{in}||^2 \log \frac{1}{\delta^2} \] (4.20)

\[ \tau_i \geq \theta - E S_{in} + ||w_{in}|| \sqrt{\log \frac{1}{\delta^2}}. \] (4.21)

We pick the smallest \( \tau_i \) which holds for the required error rate, and we get the stopping boundary

\[ \tau_i = \theta - E S_{in} + ||w_{in}|| \sqrt{\log \frac{1}{\delta^2}}. \] (4.22)
As long as $S_i$ satisfies $S_i \leq \tau_i$ we continue the summation by adding the next summand. This fact gives us the Curved Sequential Thresholded Sum Test (Curved STST)

$$\text{Stop if } S_i > \theta - ES_{in} + ||w_{in}||\sqrt{\frac{1}{\delta^2}}.$$  \hspace{1cm} (4.23)

This is pictured as the red upper stopping thresholds in Figure 4.1.

If the expected value of the random variables is zero $EX_i = 0$, the curved STST can be simplified to the following test

$$\text{Stop if } S_i > \theta + ||w_{in}||\sqrt{\frac{1}{\delta^2}}.$$  \hspace{1cm} (4.24)

This assumption is not so bad, since if the classifier tends to be mostly correct, most of its weak hypotheses will vote correctly, and hence $ES_{in}$, which is the remaining margin would be positive. Therefore this simpler test will typically be more conservative and still maintain the required error rate.

Similarly to the upper Curved STST we can develop the lower Curved STST. We state the lower Curved STST which is a direct consequence of lemma 2, and is provable similarly to theorem 9.

**Theorem 10.** The lower Curved Sequential Thresholded Sum Test. Lemma 2 holds with probability at least $1 - \delta$ if the stopping boundary is specified by

$$S_i > \theta - ES_{in} - ||w_{in}||\sqrt{\frac{1}{\delta^2}},$$

where $||w_{in}|| = \sqrt{\sum_{j=i+1}^n w_j^2}$ is the norm of the maximum variation of the residual random variables.

The stopping rule which is equivalent to theorem 10

$$\text{Stop if } S_i < \theta - ES_{in} - ||w_{in}||\sqrt{\frac{1}{\delta^2}}.$$  \hspace{1cm} (4.25)

This is pictured as the green lower stopping thresholds in figure 4.1.
4.4.1 Improved Curved STST by the Central Limit Theorem

The central limit theorem defines the limiting distribution of sums of random variables. Let \( X_1, X_2, \ldots, X_n \) be independent random variables which have finite mean and variance, and let \( S_n = X_1 + X_2 + \ldots + X_n \). The central limit theorem states that for large enough \( n \) the distribution of \( S_n \) is approximately normal. A similar statement is true for sums of remaining random variables \( S_{in} \).

The central limit theorem allows us to derive a slightly different, tighter, stopping rule. We standardize the distribution of the remaining sum and get the following probability equality

\[
P(S_n < \theta | \text{stop at } i) \leq P(S_{in} < \theta - \tau_i | S_i = \tau_i) \tag{4.26}
\]

\[
= P(S_{in} < \theta - \tau_i) \tag{4.27}
\]

\[
= P\left( \frac{S_{in} - ES_{in}}{\sqrt{\text{var}(S_{in})}} < \frac{\theta - \tau_i - ES_{in}}{\sqrt{\text{var}(S_{in})}} \right) \tag{4.28}
\]

where equation \(4.26\) follows from the Markovian property of \( S_n \), and the fact that when crossing the boundary \( S_i \) must be very close to the stopping boundary \( \tau_i \). We can remove the conditioning in equation \(4.26\), since the future is independent of the past.

**Lemma 3.** Let an upper decision error be defined by the event \( \{ S_n \leq \theta | \text{stop at } i \} \). If \( X_1, \ldots, X_n \) are independent then upper bounding the probability of making a decision error

\[
P(S_n < \theta | \text{stop at } i) \leq \delta,
\]

is equivalent to the following upper bound

\[
\frac{\theta - \tau_i - ES_{in}}{\sqrt{\text{var}(S_{in})}} \leq \Phi^{-1}(1 - \delta), \tag{4.29}
\]

where \( \Phi^{-1} \) is the inverse standard normal distribution function.

**Proof.** Since \( X_1, \ldots, X_n \) are independent, the sum \( S_n = X_1 + \ldots + X_n \) is approximately normally distributed by the Central Limit Theorem. By standardizing \( S_n \) we can upper bound the probability of making a decision error with the inverse of the standard normal distribution function \( \Phi^{-1} \). From equation \(4.28\) we have

\[
P(S_n < \theta | \text{stop at } i) \leq P\left( \frac{S_{in} - ES_{in}}{\sqrt{\text{var}(S_{in})}} < \frac{\theta - \tau_i - ES_{in}}{\sqrt{\text{var}(S_{in})}} \right).
\]
CHAPTER 4. SEQUENTIAL STATISTICAL TESTS FOR THRESHOLDED SUMS

By the independence assumption \((S_{in} - ES_{in})/\sqrt{\text{var}(S_{in})} \sim N(0, 1)\), which implies \(\frac{\theta - \tau_i - ES_{in}}{\sqrt{\text{var}(S_{in})}} = \Phi^{-1}(1 - \delta)\).

**Theorem 11.** The (upper) curved CLT Sequential Thresholded Sum Test. Lemma 3 holds with probability at least \(1 - \delta\) if the stopping boundary is specified by

\[ S_i < \theta - ES_{in} + \sqrt{\text{var}(S_{in})}\Phi^{-1}(1 - \delta). \]

**Proof.** Lemma 3 gives us an upper bound on the decision error rate, as more information is gathered sequentially. Solving equation 4.29 for the thresholds we get

\[ \tau_i \geq \theta - ES_{in} + \sqrt{\text{var}(S_{in})}\Phi^{-1}(1 - \delta). \]

We pick the smallest \(\tau_i\) which holds for the required decision error rate

\[ \tau_i = \theta - ES_{in} + \sqrt{\text{var}(S_{in})}\Phi^{-1}(1 - \delta). \]

As long as \(S_i\) satisfies \(S_i < \tau_i\) we continue the summation by adding the next summand. This gives the curved CLT-STST

\[ \text{Stop if } S_i > \theta - ES_{in} + \sqrt{\text{var}(S_{in})}\Phi^{-1}(1 - \delta). \] (4.30)

Similarly to the upper curved CLT STST we derive the lower stopping boundary.

**Theorem 12.** The lower curved CLT Sequential Thresholded Sum Test. The probability of making a lower decision error \(P(\varepsilon^l)\) is at most \(\delta\) if the stopping thresholds are set as

\[ \tau_i = \theta - ES_{in} - \sqrt{\text{var}(S_{in})}\Phi^{-1}(1 - \delta). \]

**Proof.** From equation 3 we can derive the following upper bound for a lower decision mistake

\[ P(\varepsilon^l) = P(S_n > \theta | \text{stop at } i) \leq P \left( \frac{S_{in} - ES_{in}}{\sqrt{\text{var}(S_{in})}} > \frac{\theta - \tau_i - ES_{in}}{\sqrt{\text{var}(S_{in})}} \right). \]

Since \(X_1, ..., X_n\) are independent, the sum \(S_n = X_1 + ... + X_n\) is approximately normally distributed by the Central Limit Theorem. By standardizing \(S_n\) we can upper bound the probability of making a decision error with the inverse normal cumulative distribution function \(\Phi^{-1}\). By the independence assumption \((S_{in} - ES_{in})/\sqrt{\text{var}(S_{in})} \sim N(0, 1)\), which implies \(\frac{\theta - \tau_i - ES_{in}}{\sqrt{\text{var}(S_{in})}} = -\Phi^{-1}(1 - \delta)\). Solving for \(\tau_i\) we get the stopping thresholds stated.
Figure 4.4: The resulting constants from the application of Hoeffding’s inequality versus the Central Limit Theorem. Although the central limit theorem’s smaller constants are preferable, Hoeffding’s inequality requires less assumptions on the random variables.

Alternatively to theorem 12 we can require that as long as $S_i$ satisfies $S_i > \tau_i$ we continue the summation by adding the next summand. This gives the stopping rule for the curved (lower) CLT-STST

$$\text{Stop if } S_i < \theta - ES_{in} - \sqrt{\text{var}(S_{in})\Phi^{-1}(1 - \delta)}. \quad (4.31)$$

4.4.2 A Comparison of Curved STST Tests

Both the tests that were derived using Hoeffding’s inequality, and the Central Limit Theorem share the same structure. However, when we look at the constants it turns out the the CLT-STST is much tighter and aggressive than the curved STST.

From theorem 9 we get the following stopping boundary

$$\tau_i = \theta - ES_{in} + ||w_{in}||\sqrt{\log(\delta^{-2})},$$

whereas theorem 11 gives us the following stopping boundary

$$\tau_i = \theta - ES_{in} + \sqrt{\text{var}(S_{in})\Phi^{-1}(1 - \delta)}.$$
CHAPTER 4. SEQUENTIAL STATISTICAL TESTS FOR THRESHOLDED SUMS 42

Figure 4.5: A comparison of the error rate observed when using CLT constants vs. Hoeffding constants. Results plotted for simulation of 10000 random walks $X_i \sim N(0, 1)$. Plotted are the required decision error rate $\delta$, and the actual error rate obtained by simulation. CLT gives lower constants which result in a sharper and more aggressive stopping thresholds then the ones obtained by Hoeffding’s inequality.

If the process under examination obeys $||w_{in}|| = \sqrt{\text{var}(S_{in})}$ (which is true for normally distributed $X_i$ for example), then the difference between the two bounds is a function of the required error rate delta.

According to the Central Limit Theorem we get the constant $\Phi^{-1}(1 - \delta)$, while Hoeffding’s inequality gives $\sqrt{\log \delta^{-2}}$. Figure 4.4 shows the two constants which arise from these two different approaches. We observe that the central limit theorem gives better (smaller) constants than Hoeffding’s inequality for all error rates $\delta$.

Moreover, if the variables $X_i$ are iid, we get that the variance of the residual sum is

$$\sqrt{\text{var}(S_{in})} = \sqrt{\sum_{j=i+1}^{n} w_j^2 \cdot \text{var}(X_i)} = ||w_{in}|| \sqrt{\text{var}(X_i)}.$$

If the variance of $X_i$ satisfies $\sqrt{\text{var}(X_i)} < 1$ the central limit theorem will give even sharper (smaller) stopping thresholds. This relationship between the constants obtained by Hoeffding’s worst case probability inequalities and the central limit theorem inequality was also
CHAPTER 4. SEQUENTIAL STATISTICAL TESTS FOR THRESHOLDED SUMS

Figure 4.6: CLT outperforms Hoeffding in terms of speed. Computational savings as a function on the decision error. The savings in computation is the fraction of features that were not evaluated from the entire set of features. CLT constants are smaller, and therefore the resulting stopping thresholds are smaller, the random walk stops more quickly, and savings are larger. Data obtained by simulation $X_i \sim N(0, 1)$.

observed by Watanabe (2000) in a slightly different sampling setting.

Figures 4.5 and 4.6 compare the two decision boundaries obtained by CLT and Hoeffding for $X_i \sim N(0, 1)$. Since the variance of the uniform distribution is less than one, the CLT-STST provides a significantly more accurate, and faster stopping rule than the Hoeffding-STST as expected. Notice that in figure 4.5 the curved CLT-STST produces decision error rates that are very tight to the wanted error rate, which results in up to a 3 times speedup portrayed in 4.6.

The curved STST is interesting. If we were to design a “trivial” test with 100% accuracy we would create the decision boundary

$$\tau_i = \theta + |w_{i,n}|,$$

(4.32)

since if the random walk passes above $\tau_i$, it would be impossible for it to end below $\theta$. The STST uses the $L2$ norm which is smaller or equal to the $L1$ norm. Therefore as $\delta$ increases,
our test is exceedingly better than the trivial test in terms of rejection speed. See figure 4.3 for a comparison between these stopping boundaries.

## 4.5 The Linear Sequential Thresholded Sum Test

The simplicity of the curtailed method was that neither the joint probability nor the stopping time probability need to be calculated explicitly to upper bound the conditional error probability. Moreover, actually calculating these probabilities under a curved boundary without distributional assumptions might not be analytically solvable. The curved boundary gave us a constant conditional error probability throughout the curve (for every coordinate \(i\).) This is a rather conservative boundary. A more aggressive boundary allows higher decision error rates at the beginning of the random walk and lower decision error rates at the end. Such a boundary would essentially stop more random walks early on, and less later on. In machine learning, this approach has the natural interpretation that we want to shorten the feature evaluation for obvious un-important samples, but we want to prolong the evaluations for samples we are not sure about. A linear boundary achieves this exact “error spending” characteristic. Also, by using a flat stopping boundary and the Central Limit Theorem, we can estimate all of the probabilities in equation 4.1 with high accuracy.

The linear boundary STST is developed by requiring that the joint distribution of stopping time and the sum ending below the required importance threshold is less than a required error rate \(\delta\)

\[
P(S_n < \theta, \text{stop before } n) = P(S_n < \theta, \max_i S_i > \tau) \leq \delta.
\]

This test is related to the curved boundary by the following relationship

\[
P(S_n < \theta, \max_i S_i > \tau) = P(S_n < \theta | \max_i S_i > \tau)P(\max_i S_i > \tau).
\] (4.33)

Unlike the curved boundary, we now fix the stopping threshold to a constant value across the entire walk \(\tau\). We derive the constant threshold by analyzing the following joint distribution that describes a decision error event slightly differently than its previous conditional
Figure 4.7: This figure demonstrates the reflection principle. The principle creates a reflection of all paths from the stopping boundary to the importance threshold. Thereby equating the probability of making a decision error to the probability of passing the new lower decision threshold.

Equation 4.34 is derived using the reflection principle (see Figure 4.7 and Appendix A), and equation 4.35 is its standardization $S_n \sqrt{\text{var}(S_n)} \sim N(0,1)$.

**Lemma 4.** Let an (upper) decision error be defined by the joint event $\{S_n \leq \theta, S_i > \tau\}$. If $X_1, ..., X_n$ are independent, then upper bounding the probability of making a decision error

$$P(S_n < \theta, \text{stop before } n) \leq \delta,$$

is equivalent to the following upper bound

$$\frac{2\tau - \theta + ES_n}{\sqrt{\text{var}(S_n)}} = \Phi^{-1}(1 - \delta),$$

(4.36)
where $\Phi^{-1}$ is the inverse normal cumulative distribution function.

**Proof.** Since $X_1, ..., X_n$ are independent, the sum $S_n = X_1 + ... + X_n$ is approximately normally distributed by the Central Limit Theorem. By standardizing $S_n$ we can upper bound the probability of making a decision error with the inverse standard normal distribution function $\Phi^{-1}$. From equation 4.35 we have

$$P(S_n < \theta, \max_i S_i > \tau) \leq P \left( \frac{S_n - ES_n}{\sqrt{\text{var}(S_n)}} < \frac{2\tau - \theta + ES_n}{\sqrt{\text{var}(S_n)}} \right).$$

By the independence assumption $(S_n - ES_n)/\sqrt{\text{var}(S_n)} \sim N(0, 1)$, which implies $\frac{2\tau - \theta + ES_n}{\sqrt{\text{var}(S_n)}} \leq \Phi^{-1}(1 - \delta)$.

Setting the probability of this decision error event to $\delta$ we get the following stopping rule

$$\frac{2\tau - \theta + ES_n}{\sqrt{\text{var}(S_n)}} = \Phi^{-1}(1 - \delta),$$

which yields the flat boundary STST stopping threshold

$$\tau = \frac{1}{2} \left( \theta - ES_n + \sqrt{\text{var}(S_n)} \Phi^{-1}(1 - \delta) \right).$$

(4.38)

For the curved STST the conditional decision error rate was fixed at $\delta$ throughout the entire curve. Unlike the curved boundary, this linear STST boundary has a varying conditional error rate throughout the random walk.

$$P(S_n < \theta | \text{stop at } i) \approx P(S_{in} < \theta - \tau | S_i = \tau)$$

$$= P \left( \frac{S_{in} - ES_{in}}{\text{std}(S_{in})} < \frac{\theta - \tau - ES_{in}}{\text{std}(S_{in})} \right)$$

$$= \Phi \left( \frac{\theta - \tau - ES_{in}}{\text{std}(S_{in})} \right).$$

(4.41)

If we assume that the random variables have zero expectation $E(X_i) = 0$, then since $\sqrt{\text{var}(S_{in})}$ decreases as $i$ grows, the conditional probability of making an error also decreases as $i$ grows. This phenomenon shows the main difference between the curved boundary and the linear boundary. The linear boundary starts with higher decision error rates and produces a faster stopping boundary whereas the curved boundary maintains the same error rate throughout the summation, and therefore produces a more conservative stopping boundary. See Figure 4.3 for a comparison of these boundaries.
(a) The tradeoff between accuracy and speed. The Flat boundary STST maintains a very close error rate (solid blue) to the required one (solid black), while speeding up computation significantly (solid red). For a sorted weighted random walk, as one would obtain when testing a classifier, order of magnitude speedups are shown with very little error tradeoff (solid square red).

(b) The tradeoff between accuracy and speed. The linear boundary STST (adaptive flat) maintains a very close error rate (solid blue) to the required one (solid black), while speeding up computation significantly (solid red). For a sorted weighted random walk, as one would obtain when testing a classifier, 2x speedups are shown with very little error tradeoff (solid square red).

Figure 4.8: Comparison of the flat boundary for the joint decision stopping thresholds, vs. the adaptive flat boundary and the conditional decision stopping thresholds.

We assumed that there is no trend $EX_i = 0$ to apply the reflection principle. However, typically there will be a trend, and the random walk needs to be de-trended in order to apply the reflection principle. The de-trended linear STST

$$P(S_n < \theta, \text{stop before } n) \lesssim P(S_n < \theta, \max_i (S_i - \tau_i) > 0)$$

$$= P(S_n - ES_n < \theta - ES_n, \max_i (S_i - ES_i - (\tau_i - ES_i)) > 0).$$

If we take $\tau_i - ES_i$ to be independent of $i$, i.e., a constant, then we can apply the reflection...
principle as before to get

\[ P(S_n - ES_n < \theta - ES_n, \max_i (S_i - ES_i - (\tau_i - ES_i)) > 0) \]

\[ = P(S_n - ES_n > 2(\tau_i - ES_i) - (\theta - ES_n)) \]

\[ = P \left( \frac{S_n - ES_n}{\sqrt{\text{var}(S_n)}} > \frac{2(\tau_i - ES_i) - \theta + ES_n}{\sqrt{\text{var}(S_n)}} \right). \]  

The corresponding de-trended STST stopping thresholds

\[ \tau_i = ES_i + \frac{1}{2} \left( \theta - ES_n + \sqrt{\text{var}(S_n)} \Phi^{-1}(1 - \delta) \right). \]  

The flat boundary and the de-trended flat boundary tests performance are depicted in figure 4.8.

To increase the readability of the thesis we only state the upper flat de-trended stopping boundary. The derivation of the lower boundary follows the derivation of the upper boundary. The only difference between then is that instead of bounding the probability of the event \( \{ S_n < \theta, \max_i S_i > \tau \} \) we bound the probability of \( \{ S_n > \theta, \min_i S_i < \tau \} \). The flip in the inequality causes the subtraction of \( \Phi^{-1}(1 - \delta) \) instead of its addition in equation 4.38. Finally this gives the de-trended stopping boundary

\[ \tau_i = ES_i + \frac{1}{2} \left( \theta - ES_n - \sqrt{\text{var}(S_n)} \Phi^{-1}(1 - \delta) \right). \]

4.6 The Constant Sequential Thresholded Sum Test

In some scenarios \( \{ S_n < \theta \} \) is a rare event. For example, in face detection the overwhelming majority of image patches are non-faces. In this scenario the joint event \( \{ S_n < \theta, \text{stop} \} \) which we used to derive the flat stopping boundary would be even more rare. Controlling the probability of this event is practically useless when our task is to control the decision error rate. This is due to the fact that the probability of \( P(S_n < \theta) \) is already small, and therefore the joint probability of the decision error event \( P(S_n < \theta, \text{stop}) \) is even lower by definition. For example, when we are interested in finding the decision boundary that corresponds to a decision error rate \( P(S_n < \theta, \text{stop}) = 1\% \), if \( P(S_n < \theta) = 1\% \) it is possible that \( P(\text{stop}) = 1 \) and therefore all random walks would be stopped and predicted \( S_n < \theta \).
To correct this error, we once again need to look at a different definition of the decision error. We now condition the joint probability in the following way

\[ P(\text{stop before } n | S_n < \theta) = \frac{P(\text{stop before } n, S_n < \theta)}{P(S_n < \theta)}. \] (4.48)

We stated in equation 4.48 a conditional probability function which is conditioned on the examples of interest. Therefore in this case we are interested in limiting the decision error rate for examples that are important. To upper bound this conditional we will make an approximation that will allow us to apply boundary-crossing inequalities for a Brownian bridge. To apply the Brownian bridge to our conditional probability we approximate it by

\[ P(\text{stopped before } n | S_n < \theta) = P(\max_i S_i > \tau | S_n < \theta) \approx P(\max_i S_i > \tau | S_n = \theta). \]

Since we assume that the event \( \{S_n < \theta\} \) is rare (equivalently that \( EX_i > 0 \)), then we can approximate the inequality with an equality, which gives a Brownian bridge. Now we need to calculate boundary crossing probabilities of the Brownian bridge and a constant threshold. A review of the Brownian bridge can be found in Appendix A.

**Lemma 5.** The Brownian bridge Stopping Boundary. Let \( T_\tau = \inf\{i : S_i = \tau\} \) be the first hitting time of the random walk and constant \( \tau \). Then the probability of the following decision error is \( P(T_\tau < n | S_n = \theta) = e^{-\frac{2\tau(\tau - \theta)}{\text{var}(S_n)}}. \)

**Proof.** See Appendix ??.

**Theorem 13.** The Simplified Brownian bridge boundary \( \theta = 0 \), \( \tau = \sqrt{\text{var}(S_n)} \sqrt{\log \frac{1}{\sqrt{\delta}}} \) makes approximately \( \delta \) decision mistakes \( P(T_\tau < n | S_n < 0) \).

**Proof.** By approximating \( P(S_i > \tau | S_n < \theta) \approx P(S_i > \tau | S_n = \theta) \) and setting this probability to \( \delta \) we get the Brownian bridge boundary

\[ P(S_i > \tau | S_n < \theta) \approx \exp \left\{ -\frac{2\tau(\tau - \theta)}{\text{var}(S_n)} \right\} = \delta. \] (4.49)

Solving,

\[ \tau^2 - \tau \theta = \text{var}(S_n) \log \frac{1}{\sqrt{\delta}} \] (4.50)

\[ (\tau - \theta)^2 - \frac{1}{4} \theta^2 = \text{var}(S_n) \log \frac{1}{\sqrt{\delta}} \] (4.51)

\[ \tau = \theta + \sqrt{\frac{1}{4} \theta^2 + \text{var}(S_n) \log \frac{1}{\sqrt{\delta}}}. \] (4.52)
If we simplify this boundary by setting $\theta$ to zero, we get the theorem’s boundary

$$
\tau = \sqrt{\text{var}(S_n)} \sqrt{\log \frac{1}{\sqrt{\delta}}}.
$$

There are two appealing things about this boundary, the first that it’s not dependent on $ES_i$, and that it is always positive. Secondly, when using this boundary for prediction, we can directly see the implication on the error rate of the classifier, since the decision error essentially becomes a classification error, a fact that is also clearly evident throughout the experiments.

### 4.7 Average Stopping Time for the Curved and Constant STST

We are able to show that the expected number of features evaluated for the Curved and the Constant STST boundaries is in the order of $O(\sqrt{n})$. This can be obtained by limiting the range of values $X_i$ can take.

**Theorem 14.** Let $|X_i| \leq k$, and let $EX_i > 0$. Let the stopping time of the Brownian bridge be defined by $t = \inf\{i : S_i \geq \sqrt{\text{var}(S_n)} \log \delta^{-0.5}\}$. Then the expected stopping time is in the order of $O(\sqrt{n})$.

**Proof.**

$$
ES_T = ES_{T-1} + EX_T
$$

$$
\leq ES_{T-1} + k
$$

$$
\leq \sqrt{\text{var}(S_n)} \log \delta^{-0.5} + k
$$

The second inequality holds since the random walk only crossed the boundary for the first time at time $T$ and therefore was under the boundary at time $T - 1$. By applying Wald’s
CHAPTER 4. SEQUENTIAL STATISTICAL TESTS FOR THRESHOLDED SUMS

51

(a) A simulation of the Brownian bridge boundary with $X_i \sim N(0.05, 1)$. The boundary is conservative. (b) The boundary behaves similarly to what’s expected from theory. It computes in the order of $O(\sqrt{n})$ features.

Figure 4.9: Performance of the Brownian bridge boundary.

Equation $EST = ETEX$ we get

$$ET = \frac{\sqrt{\text{var}(S_n)} \log \delta^{-0.5} + k}{EX}$$

$$\leq \frac{c\sqrt{n} \log \frac{1}{\sqrt{\delta}} + k}{EX}$$

$$= O(\sqrt{n}),$$

where $c, k$, and $EX$ are constants.

We can derive the same expected stopping time for random walks which satisfy $S_n > \theta$ with the Curved STST. If we assume that $EX > 0$, and that the random variables are independent $\text{var}(S_{in}) < \text{var}(S_n)$ then we can create a more conservative curved boundary

$$\theta - ES_{in} + \sqrt{\text{var}(S_{in})} \Phi^{-1}(1 - \delta) < \theta + \sqrt{\text{var}(S_n)} \Phi^{-1}(1 - \delta).$$

Since $\Phi^{-1}(1 - \delta)$ is a constant, we get an $O(\sqrt{n})$ expected stopping time by the arguments of Theorem 14.
Part III

Algorithms and Experiments
Chapter 5

Attentive Online Margin Based Algorithms

Margin-based online learning algorithms compare the margin (or the score) of each example to a threshold to form a decision. Generalizing this process, most of the online margin-based algorithms compare a weighted sum of random variables to a threshold to make a decision.

We can therefore apply our stopping boundaries anywhere such thresholding occurs. We impose an independence assumption between the weak hypotheses, and we will traditionally cause an increase in the error rate of the classifier. However, as we have seen in Theorem 5, this added error decays exponentially fast as more base hypotheses are evaluated. Experimentally we also witness that these modified attentive algorithm indeed achieves generalization that is comparable, and sometimes even better, to the generalization of the base unmodified learning algorithm!

We applied the decision boundaries to three algorithms: Online Boosting, Perceptron, and Pegasos.

5.1 Curtailed Online Boosting

Online Boosting proposed by Oza and Russell (2001) is an online learning algorithm which converges to AdaBoost as more examples are observed. It adopts the exponential re-weighting scheme that is used by AdaBoost to weight the examples. Online Boosting
also has the sequential nature of AdaBoost where for every example all the features are evaluated in order, and the example weight is re-calculated at each coordinate.

We apply the STST filtering framework to speed-up Online Boosting by quickly rejecting examples of little importance. We measure the importance of an example by the size of its margin, the distance from the decision boundary geometrically. Online Boosting is modified by quickly rejecting examples which quickly obtain a large positive partial margin, and therefore are unlikely to have a full margin smaller than the importance threshold $\theta$.

Curtailed Online Boosting is detailed in algorithm 1. It is based on a slight modification of Online Boosting to the more prevalent AdaBoost weighting rule (Schapire and Singer 1999; Pelossof et al. 2008). The algorithm does not detail the bayesian feature selection for clarity. However, it can be incorporated similarly to Online Boosting. Also, in the case where we have a trained classifier and we are interested in speeding up the evaluation of the score of new examples, the STST thresholds can be pre-computed and stored. The test then becomes a fast and simple look-up table test.

5.2 The Attentive Perceptron

We propose the focus-of-attention mechanism to speed up the Perceptron algorithm by Rosenblatt (1958). Focus-of-attention speeds up the Perceptron algorithm by lowering the number of features evaluated throughout training and prediction. While the Perceptron evaluates all the features of each example, the Attentive Perceptron evaluates fewer features for uninformative examples, thereby achieving significant speedups and small losses in prediction accuracy. Focus-of-attention allows the Attentive Perceptron to stop the evaluation of features at any interim point and filter the example. This creates an attentive filter which concentrates computation on examples that are informative, and quickly filters ones that are not.

The focusing mechanism is inserted where the perceptron usually would test if it makes a classification error, and consequently update its model. Algorithm 2 shows the modification of the Perceptron algorithm to the novel Attentive Perceptron.
5.3 Attentive Pegasos

Pegasos by Shalev-Shwartz et al. (2010) is a simple and effective iterative algorithm for solving the optimization problem cast by Support Vector Machines. To solve the SVM functional it alternates between stochastic gradient descent steps (weight update) and projection steps (weight scaling). Similarly to the Perceptron these steps are taken when the algorithm makes a classification error while training. For a linear kernel, the total run-time of Pegasos is $\tilde{O}(d/(\lambda \epsilon))$, where $d$ is the number of non-zero features in each example, $\lambda$ is a regularization parameter, and $\epsilon$ is the error rate incurred by the algorithm over the optimal error rate achievable. By assuming that the features are independent, and applying the Brownian bridge STST we speed up Pegasos to $\tilde{O}(\sqrt{d}/(\lambda \epsilon))$ without losing significant accuracy. The algorithm is demonstrated in Algorithm 3.

We witness that when we apply the STST throughout learning, but predict with the entire set of weak hypotheses, the STST generally over-performs its budgeted equivalent. However, when we compare prediction with the Brownian bridge STST with budgeted prediction we see the significant improvement in accuracy obtained by the STST. Moreover, our algorithm even outperforms the original full Pegasos in prediction.

5.4 Calculation of $ES_n$ and $Var(S_n)$ of weighted sums

We are interested in calculating the basic statistics of the weighted sum. Since our specific application of this calculation involves weights that slowly change until convergence, we are actually interested in the expectation and variance of the random walks with the converged weights. We want to find a way to continuously update these statistics, and improve their estimation as the weights converge. Let $X_i$ be a random variable, and $w_i$ it’s corresponding weight. We use bold fonts as vector notation, and Bra-ket notation for dot products $\langle \cdot, \cdot \rangle$. We denote the weighted sum of independent random variables as $S_n(w) = \sum_{i=1}^{n} w_i X_i = \langle w, X \rangle$. Finally, we use the superscript to denote time, or the number of the example.
5.4.1 Parameter Estimation

All the decision boundaries that were developed throughout this section are dependent on the known required decision error rate $\delta$, and unknown statistics $ES_i$, and $Var(S_i)$. The unknown statistics are the mean and variance of the partial and full sums. To derive efficient stopping boundaries these quantities need to be estimated accurately. We estimate the expectation of a sum empirically by $ES_n = \sum_{i=1}^{n} E(X_i)$. In the case where $S_n$ is a weighted sum, the expectation is approximated by $ES_n = \sum_{i=1}^{n} w_i E(X_i)$. The variance of the random process is estimated through the expectation $var(S_n) = E(S_n^2) - (ES_n)^2$. This quantity is again estimated throughout our experiments by the empirical expectations $var(S_n) \approx \sum_{i=1}^{n} X_i^2 - (\sum_{i=1}^{n} X_i)^2$. If the $S_n$ is a weighted sum the variance is estimated by $var(S_n) \approx \sum_{i=1}^{n} w_i^2 X_i^2 - (\sum_{i=1}^{n} w_i X_i)^2$. The standard deviation is computed as $std(S_n) = \sqrt{var(S_n)}$. Since we deal with classification, we estimate the statistics for each class separately $E_y(S_i)$ and $var_y(S_i)$. To construct the boundary the appropriate statistics are used according to the example’s label.

Let us compute the expectation of $S_N$ if we only look at a set of weights $w$

$$E[S_n(w)] = E \left[ \sum_{i=1}^{n} w_i X_i \right] = \sum_{i=1}^{n} w_i E[X_i] = \sum_{i=1}^{n} w_i \sum_{l=1}^{m} \frac{X_{il}}{m} = \langle w, E(X) \rangle.$$ 

To compute $ES_n(w)$ we only need to keep $\sum_{l=1}^{m} X_{il}, i = 1, .., n$. Similarly, we expand the computation of the variance of the weighted sum

$$Var[S_n(w)] = Var \left[ \sum_{i=1}^{n} w_i X_i \right] = \sum_{i=1}^{n} (w_i)^2 Var[X_i]$$

$$= \sum_{i=1}^{n} (w_i)^2 (E[X_i^2] - E[X_i]^2) = \sum_{i=1}^{n} (w_i)^2 \left( \sum_{l=1}^{m} \frac{(X_{il})^2}{m} - \left( \sum_{l=1}^{m} \frac{X_{il}}{m} \right)^2 \right)$$

$$= \langle w^2, E[X^2] - E[X]^2 \rangle.$$ 

By keeping only $E[X_i^2]$ and $E[X_i]^2$ for $i = 1, .., n$, we can estimate the variance for any set of weights $w$.

These types of calculations are advantageous for an online algorithm that changes its weight vector as the learning progresses until convergence such as the Perceptron algorithm.
(see algorithm 2), since they do not require the entire history of the weights $w_1, \ldots, w^{T+1}$, but only the last weights vector $x^{T+1}$.

Also, since we keep the statistics for each coordinate we can generate "on-the-fly" boundaries with different sampling techniques, and for sorted coordinates. For example, in a random setting we randomly select $T$ coordinates $i_1, \ldots, i_T$, and update the corresponding expectations $E_X_{i_1}, \ldots, E_X_{i_T}$ and squared expectations $E_X^2_{i_1}, \ldots, E_X^2_{i_T}$. Then we compute the expectation $E_S_n$ by the summation of all the expectations of all the random variables $E_X, \ldots, E_X_n$. This can be achieved more efficiently by looking at the changes to the expectations and summing those changes to reflect the new expected sum.
Algorithm 1 Curtailed Online Boosting

Input: $h_1, \ldots, h_n; (x_1, y_1), \ldots, (x_m, y_m), \epsilon, \theta, \delta$

Initialize: $\lambda_i^+ = \epsilon, \lambda_i^- = \epsilon, \alpha_i = 0, z_i = 0, i = 1, \ldots, n$

Define: $u_{ji} = y_j h_i(x_j)$

for $j = 1$ to $m$ do
    $S_i = 0$
    for $i = 1$ to $n$ do
        $S_i = S_{i-1} + \alpha_i u_{ji}$
        if $S_i > \theta + \sqrt{z_i} \sqrt{\frac{1}{\delta^2}}$ then
            Jump to next example
        end if
    end for
end for

% Run Online Boosting
$d_1 = 1$

for $i = 1$ to $n$ do
    $\lambda_i^- \leftarrow \lambda_i^- + d_i 1_{[u_{ji}=-1]}$
    $\lambda_i^+ \leftarrow \lambda_i^+ + d_i 1_{[u_{ji}=+1]}$
    $\alpha_i = \frac{1}{2} \log \frac{\lambda_i^+}{\lambda_i^-}$
    $d_{i+1} = d_i e^{-\alpha_i u_{ji}}$
end for

for $i = 1$ to $n$ do
    $z_i = \sum_{j=i+1}^{n} \alpha_j^2$
end for

end for

Output: $\alpha_1, \ldots, \alpha_n$
Algorithm 2 Attentive Perceptron

Input: Dataset \( \{X^l, y^l\}_{l=1}^{m} \), learning rate \( \eta \), decision error rate \( \delta \)

Initialize: Set \( w = 0 \)

for \( l = 1, 2, \ldots, m \) do

if \( \exists i = 1, \ldots, n \text{ s.t. } y^l \sum_{j=1}^{i} w_i x_i \geq \sqrt{\sum_{j=1}^{n} w_j \cdot \text{var}'(x_j) \sqrt{\log \delta^{-0.5}}} \) then

Update \( \text{var}'(x_j), j = 1, \ldots, i \)

Jump to next example

else

\( w \leftarrow w + \eta y^l x^l \)

end if

end for

return \( w \)

Algorithm 3 Attentive Pegasos

Input: Dataset \( \{X^l, y^l\}_{l=1}^{m}, \lambda, \delta \)

Initialize: Choose \( w_1 \) s.t. \( \|w_1\| \leq 1/\sqrt{\lambda}, j = 0 \)

for \( l = 1, 2, \ldots, m \) do

if \( \exists i = 1, \ldots, n \text{ s.t. } y^l \sum_{j=1}^{i} w_i x_i \geq 1 + \sqrt{\sum_{j=1}^{n} w_j \cdot \text{var}'(x_j) \sqrt{\log \delta^{-0.5}}} \) then

Update \( \text{var}'(x_j), j = 1, \ldots, i \)

\( w^l = w^{l-1} \)

Jump to next example

else

Set \( \mu_l = \frac{1}{\lambda} \)

Set \( w_{l+\frac{1}{2}} = (1 - \mu_l \lambda)w_l + \mu_l y x \)

Set \( w_{l+1} = \min \left\{ 1, \frac{1}{\|w_{l+\frac{1}{2}}\|} \right\} \)

end if

end for

return \( w^{m+1} \)
Chapter 6

Experiments

We conducted several experiments to test the speed, generalization capabilities, and predictive accuracy of the STST. The first is a two dimensional synthetic experiment which enables us to see how the algorithm spends its computational power. We can interpret this test as testing the focusing abilities of the STST perceptron. The second is an experiment that shows the effect of sorting the weights on the efficiency of the stopping boundary. The third and fourth are real world experiments on the MNIST dataset, which shows the speed advantage of Curtailed Online Boosting and Attentive Pegasos.

6.1 Synthetic data

The synthetic experiment was set up to test the computational efficiency of Curtailed Online Boosting. We created a 2D set to map how the algorithm spends its computation (Figure 6.1). Two translated sin waves were sampled from to create a set of positive and negative examples. These examples were then split randomly to the training and test sets. Each of the sets contains 100,000 examples. First AdaBoost was trained on the sets to obtain a set of 100 features. The features we used are thresholded planes. These features are not online-learnable, and therefore were set beforehand.

Figure 6.1 shows the stochastic focus-of-attention through a computational efficiency map of the algorithm. The figure shows how most of the computation is allocated to examples that are close to decision boundary, and substantially less computation is allocated
Figure 6.1: **Stochastic focus-of-attention.** Online Boosting, which is modified to have stochastic focus-of-attention, places the most computational effort into processing examples near the decision boundary. The training set is pictured in the left subfigure. In the right subfigure, the size and color of each point are proportional to the number of features evaluated throughout the curtailed online learning process. Most examples with a large margin are rejected early without a full evaluation of their features, whereas hard examples by the decision boundary get allocated high computational effort.

to examples that are easily classifiable. In the figure a random subset of 5000 examples from the test set and the resulting decision boundary found by Curtailed Online Boosting are plotted. We set \( \theta = 0, \delta = 0.2 \). The algorithm fully calculated the margin of only 1233 examples out of the 100,000 which is about 1% of the training set. The average number of hypotheses evaluated per example is 18 out of 100 which is a 5x speed-up over Online Boosting. Throughout the learning process, only 282 examples actually had a margin below the required threshold of 0, out of which 6% were not fully evaluated due to early stopping. This is far below the required error rate of 20% that we set.

Figure 6.2(a) compares the test error of AdaBoost, Online Boosting, and Curtailed Online Boosting as training progresses on the sin dataset. All three algorithms had the weak hypotheses fixed throughout training for comparison.
(a) Synthetic (sin data): Test error for the different boosting algorithms as more examples are trained on. Online Boosting and Curtailed Online Boosting approach the accuracy of batch AdaBoost as more examples are presented. However, Curtailed Online Boosting performs a full model update only on 1% of the examples.

(b) MNIST: Combined classifier test error as the number of training examples is increased. Online Boosting and Curtailed Online Boosting converge to the same error rate, however Curtailed Online Boosting is about 7 times faster.

Figure 6.2: Generalization Error on synthetic and MNIST data sets.

6.1.1 Sorting the Coordinates

To further test our decision boundaries under diversified conditions, we simulated random walks with \( X_i \sim U(-1, 1), w_i \in [-1, 1] \). The random walk was over \( n = 1,000 \) steps, and, results were generated for \( m = 50,000 \) such random walks. These random walks are plotted in Figure 6.3. The figure compares the shape of the boundary and stopping times for the Curved CLT boundary. On the left we show examples of random walks where the weights are not sorted, and on the right where the weights are sorted. Each set of subfigures is ordered as follows: (upper left) the boundary in red, and random walks that ended below \( \theta \) (lower left) the boundary in red and random walks that were decision errors (upper right).
random walks that ended above the decision boundary (lower right) the distribution of stopping times for random walks that ended above the decision boundary.

In both cases (sorted and not sorted) we can see that the boundary is pretty tight to the maximum values that the random walks which end below $\theta$ obtain. The top right subfigure shows the random walks that crossed the boundary, and which would result in savings in computation. The bottom right subfigure shows how sorting the weights improve the stopping time distribution. Also of interest is that the shape of the decision boundary changes to a more aggressive boundary when the weights are sorted.

6.2 MNIST

The MNIST dataset consists of $28 \times 28$ images of the digits $[0, 9]$. The dataset is split into a training set which includes 60,000 images, and a test set which includes 10,000 images. All the digits are represented approximately in equal amount in each set.
Table 6.1: MNIST test error in % for each classifier, and curtailment efficiency. Curtailed Online Boosting (COB) is compared to Online Boosting (OB) which is trained with the same average number of features that COB used throughout learning. COB always performs better than OB. COB achieves a speedup of 8x with only 0.01% loss in classification accuracy!

<table>
<thead>
<tr>
<th>Classification Error in %</th>
<th>Digit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoost, 1000 features</td>
<td></td>
<td>0.33</td>
<td>0.20</td>
<td>0.78</td>
<td>0.94</td>
<td>0.9</td>
<td>1.01</td>
<td>0.53</td>
<td>0.81</td>
<td>1.63</td>
<td>1.35</td>
</tr>
<tr>
<td>OB, 1000 features</td>
<td></td>
<td>0.34</td>
<td>0.21</td>
<td>0.81</td>
<td>0.97</td>
<td>0.9</td>
<td>0.99</td>
<td>0.5</td>
<td>0.84</td>
<td>1.66</td>
<td>1.37</td>
</tr>
<tr>
<td>COB</td>
<td></td>
<td>0.35</td>
<td>0.23</td>
<td>0.82</td>
<td>1.04</td>
<td>0.97</td>
<td>1.01</td>
<td>0.59</td>
<td>0.91</td>
<td>1.77</td>
<td>1.35</td>
</tr>
<tr>
<td>OB with COB avg. #feat.</td>
<td></td>
<td>0.5</td>
<td>0.31</td>
<td>1.32</td>
<td>1.55</td>
<td>1.55</td>
<td>1.46</td>
<td>0.87</td>
<td>1.28</td>
<td>2.75</td>
<td>2.17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Curtailed Online Boosting Computational Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>COB avg. #features</td>
</tr>
<tr>
<td>COB speedup</td>
</tr>
</tbody>
</table>

6.2.1 Curtailed Online Boosting

We compared Batch Adaboost, Online Boosting, and Curtailed Online Learning on the 10 1-vs-all digit recognition problems. Similarly to the synthetic experiment, we trained a classifier in an offline manner with sampling to find a set of weak hypotheses. When training we normalized the images to have zero mean and unit variance. We used \( h_j(x) = \text{sign}(\|x_j - x\|_2 - \alpha_0) \) as our weak hypothesis. The weak learner found for every boosting round the vector \( x_j \) and threshold \( \alpha_0 \) that create a weak hypothesis which minimizes the training error. As candidates for \( x_j \) we used all the examples that were sampled from the training set at that boosting round. We partitioned the multi-class problem into 10 one-versus-all problems, and defined a meta-rule for deciding the digit number as the index of the classifier that produced the highest vote. Each of the digit classifiers was trained with 1,000 features. We set \( \delta^2 = 0.3 \), and \( \theta \) was set individually for each classifier in the range \( \theta \in [0, -0.5] \). The generalization error rate of the combination rule using each of the methods can be seen in figure 6.2(b). The generalization error rates for each classifier can
be seen in table 6.1. At the beginning of the training process the votes \( \alpha_j \) are a very bad estimate of the end votes. This causes the random walk to be highly unreliable, and the curtailment process inefficient. We therefore initialized our model with a small batch of the first 5,000 examples to avoid these estimation errors. At the beginning of the training process the weights \( \alpha_j \) are a very bad estimate of the end weights. This causes the random walk to be unreliable, and the curtailment process inefficient. As the training progresses the efficiency improves, and the decision error rate decreases. The efficiency results in table 6.1 exclude the full margin computation of the initial 5,000 out of 60,000 examples. If we were to add these examples then the average number of features evaluated would increase by about 66. The filtering error rate is not affected since we fully compute the margin of these examples. The results show that an order of magnitude speedup is possible while maintaining generalization accuracy.

6.2.2 Attentive Pegasos

We modified the Pegasos algorithm by stopping the computation of examples that were unlikely to have a negative margin early. We then computed the average number of features the algorithm computed for examples that were filtered, and compared it to a budgeted version of Pegasos, where only a fixed number of features are allowed to be evaluated for any example. We ran 1-vs-1 digit classification problems under different feature selection policies. With the first policy, we sorted the coordinates, such that coordinates with a large absolute weight before other features with lesser absolute weight. Then with the second, we ran experiments where the coordinates were selected by sampling from the weight distribution with replacement. Finally, with the third, we ran experiments where the coordinates were randomly permuted.

For each one of these scenarios, three algorithms were run 10 times on different permutations of the datasets and their results were averaged (Figures 6.4 and 6.5.) We first ran Attentive Pegasus under each of the coordinate selection methods. Then we set the budget for Budgeted Pegasos as the average number of features that we got through Attentive Pegasos. Finally, we ran the full Pegasos with a trivial boundary, which essentially computes everything. Both figures show that using the Brownian Bridge boundary can
Figure 6.4: Results for Attentive Pegasos, MNIST 2 vs 3, $\delta = 10\%$. Our Brownian bridge decision boundary (blue) processes only 49 feature on average (15 times faster than full computation), while achieving similar generalization as the fully trained classifier (red, middle subfigure). On the right subfigure, when the boundary is applied to prediction, Attentive Pegasus achieves a lower error rate than the full computation, and less than half the error of the Budgeted Boundary (green).

save in the order of 10x computation, and maintain similar generalization results to the full computation. Also, sorting under the Budgeted Pegasos is impossible since we need to learn the weights in order to sort them. Therefore we did not run Budgeted Pegasos with sorted weights. We can see in the middle subfigure, that the Attentive, Budgeted, and Full algorithms maintain almost identical generalization results. However, when we early stop prediction with the resulting model, the Brownian bridge gives the best predictive results, even better than what we get with the full computation but only computes a tenth of the feature values!
Figure 6.5: Results for Attentive Pegasos. MNIST 3 vs 10, $\delta = 10\%$. Our Brownian bridge decision boundary (blue) processes only 72 feature on average, while achieving similar generalization as the fully trained classifier. On the right, when the boundary is applied to classification, Attentive Pegasus gets a lower error rate than the full computation, and over a 2% advantage over the Budgeted Boundary.
Part IV

Discussion
Chapter 7

Summary

We developed several stopping boundaries to decide when to stop the computation of the margin or the score of a linear classifier. The stopping boundaries $\tau_i$ are set in such a way that performing the full computation would yield the same result as the partial computation with probability $1 - \delta$. We summarize the main boundaries that were developed throughout the thesis.

Let $X_1, ..., X_n$ be independent random variables. Let us denote a sum by $S_n = X_1 + ... + X_n$, which can be split to two parts: the partial and remainder sums $S_n = S_i + S_{in}$. The following boundaries give stopping thresholds, such that if a partial sum crosses them at coordinate $\min_i S_i > \tau_i$ the computation is stopped and the decision $S_n > \theta$ is made.

In linear prediction we set $X_i = w_i x_i$, which implies that if the score $S_n > \theta$ then the classifier predicts the positive class. For learning we set $X_i = y w_i x_i$, which implies that if the margin $S_n > \theta$ the classifier correctly classifies the example.

Throughout the thesis we developed three different classes of decision-making thresholds under different assumptions. The first is the Curved STST boundary which is the most conservative boundary, meaning that it tends to stop partial computations later than the other boundaries. The second is the Linear STST boundary which is a more aggressive boundary. Finally, we developed a very simple Constant STST boundary which is more aggressive than the curved boundary and less aggressive than the linear boundary.

By controlling the decision error $P(S_n < \theta, \text{stop before } n) < \delta$ we developed the Curved
STST boundary

\[ \tau_i = \theta - ES_i + \sqrt{\text{var}(S_i)}\Phi^{-1}(1 - \delta), \]

where \( \Phi^{-1} \) is the inverse standard normal distribution function. Through a different method of controlling the decision error we can also derive the more aggressive \textit{Linear STST boundary}

\[ \tau_i = ES + \frac{1}{2} (\theta - ES + \sqrt{\text{var}(S)}\Phi^{-1}(1 - \delta)). \]

In some cases the event \( \{ S_n < \theta \} \) is rare. Therefore controlling the joint error event \( \{ \text{stop before } n, S_n < \theta \} \) is useless if we desire to limit the decision error rate. By controlling the error event \( \{ \text{stop before } n|S_n < \theta \} \) we can limit the decision error rate in such cases. The resulting boundary does not require the expectation of the partial sums, and is a constant boundary. \textit{The Constant STST boundary}

\[ \tau = \theta + \sqrt{\text{var}(S_n)} \sqrt{\log \frac{1}{\sqrt{\delta}}}. \]

We proved that if we assume that \( EX_i > 0 \) then the Curved and the Constant STST boundaries will compute on average \( O(\sqrt{n}) \) features on examples that satisfy \( S_n > \theta \).
Chapter 8

Future Work and Conclusions

8.1 Future work

We have developed three stopping boundary STST which apply to a very simple learning paradigm with linear decision-making. Further research is needed to develop stopping boundaries under other paradigms such as learning with kernels and learning with decision trees such as Freund and Mason (1999). Also, our boundaries only deal with single class learning, and extensions for multi-class learning should be developed. An especially interesting paradigm would be multi-class learning with feature sharing. Finally, future research should include extensions of the boundaries to the case where evaluating different features implies costs for the learner.

Along with different learning paradigms, theory needs to be developed to determine the generalization capabilities of algorithms that use our early stopping thresholds. Current theory does not hold since a different number of features are computed for every example.

The boundaries were developed by upper bounding the probability of passing the stopping threshold with the probability of stopping at the stopping threshold. There are a variety of statistical methods that estimate this overshoot, and can improve the performance of our boundaries. Similarly, when we developed the Constant STST boundary we upper bounded the event of passing the decision threshold by the event of hitting the decision thresholds. This allowed us to treat the random walk as a Brownian bridge - which is a crude approximation. A better approximation can be derived by looking at the expansion
CHAPTER 8. FUTURE WORK AND CONCLUSIONS

of the tail of the distribution below the stopping threshold. A more general improvement of the decision threshold is to develop stopping boundaries that are not dependent on the expectation or variance of the underlying data. These boundaries would instead be dependent on a different measure such as the VC dimension of the underlying classifier class.

This theory has concentrated mostly around the development of the theory behind the stopping boundaries. Further experimentation is needed to get a better sense of the performance of the Attentive algorithms when the underlying statistical assumptions do not hold. Also, experimentation in face detection would be of particular interest to the authors since this is the task that motivated this thesis. Experimentation with the performance of the STST thresholds when the learner is adaptive, and therefore changes its weights throughout deployment is also of particular interest.

8.2 Conclusions

We sped up online learning algorithms up to an order of magnitude without any significant loss in predictive accuracy. Surprisingly, in prediction Attentive Pegasos outperformed the original Pegasos, even though it computed an order of magnitude less features on digit classification tasks from the MNIST dataset. This was accomplished by applying early stopping thresholds to the computation of the margin and the score. In addition, we proved that the expected speedup under independence assumptions of the weak hypotheses is $O(\sqrt{n})$ where $n$ is the set of all features used by the learner for discrimination.

One of the side effects of this thresholding process is that it creates a natural attention mechanism for online learning algorithms. Examples that are easy to classify (such as background) are filtered quickly without evaluating many of their features. For examples that are hard to classify, the majority of their features are evaluated. This property of the stopping thresholds holds in any learning task without the algorithm having to model the underlying data generating distribution. By spending little computation on easy examples and a lot of computation on hard “interesting” examples the Attentive algorithms exhibit a stochastic focus-of-attention mechanism.

We arrived at this problem by generalizing the problem of setting early rejection thresh-
olds for non-faces in face detectors. These early rejection thresholds were critical in making
face detectors real-time. Traditionally in the Statistics community this type of decision-
making problem is addressed through likelihood models. An approach that does not apply
to discriminative learning where the underlying data generating process is unknown, and
thousands of features may be used. Therefore, we took the approach of modeling the linear
classifier’s summation of feature votes as a stochastic process to bridge the two commu-
nities - Discriminative Learning and Sequential Statistics. By doing so we were able to
solve different computation stopping boundaries for linear decision-making problems under
a minimal set of assumptions.

The decision-making problem that the classifier is faced with is to determine if the sum
of weighted weak-hypotheses votes is above or below a specified decision-making threshold.
The sum is referred to as the score in prediction, and as the margin in training. In both cases
all the weak hypotheses are evaluated to determine if the sum is larger than a threshold.
The evaluation of all the weak hypotheses is not required when a partial sum is significantly
larger than the decision-making threshold. By looking at the summation as a random walk
we were able to determine when enough weak hypotheses have been added, and the decision
of the learning algorithm will not change.
Part V

Appendices
Appendix A

Stochastic processes supplements

Lemma 6. The Central Limit Theorem. Let $X_1, ..., X_n$ be a sequence of mutually independent and identically distributed random variables each with mean and variance $EX, \sigma(X)^2$. Let $S_n = X_1 + ... + X_n$. Then for large $n$ the distribution of $S_n$ is approximately normal.

Standardizing $S_n$

$$Z_n = \frac{S_n - ES_n}{\sigma(S_n)} = \frac{S_n - nEX}{\sqrt{n}\sigma(X)}.$$  

The central limit theorem states that if $F_{Z_n}(z)$ is the CDF of $Z_n$, then

$$\lim_{n \to \infty} F_{Z_n}(z) = \lim_{n \to \infty} P(Z_n < z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-u^2/2} du = \Phi(z)$$

Therefore, $\lim_{n \to \infty} Z_n = N(0,1)$

Definition 2. Standard Brownian motion $\{W(t) : t \geq 0\}$ is a stochastic process (Wiener process) which has stochastic independent increments. Brownian motion is defined by three facts:

1. $W_0 = 0$.

2. $W_t$ is almost surely continuous.

3. Stationary and independent increments $W_t - W_s \sim N(0, t-s)$ for $0 \leq s \leq t$.

Brownian motion can be constructed as the limit of random walks. Let $S_n = X_1 + ... + X_n$ be a sum of independent random variables $X_i$. We construct a discrete approximation to Brownian motion by setting $W_{i\Delta t} = \frac{1}{\sqrt{n}} S_i$, where $\Delta t = 1/n$. As $n \to \infty$ by the central limit
theorem the distribution of $W_1^n = \frac{S_n}{\sqrt{n}}$ approaches the normal distribution with mean 0 and variance 1.

**Definition 3.** Markov Property. Let $X_t$ be a standard Brownian motion. Let $\mathcal{F}_t$ represent the information contained in $X_s, s < t$ then

$$E(W_t|\mathcal{F}_s) = E(W_s|\mathcal{F}_s) + E(W_t - W_s|\mathcal{F}_s) = W_s = E(W_t|W_s).$$

Since $W_t - W_s$ is independent of $\mathcal{F}_s$, $E(W_t - W_s) = 0$.

**Lemma 7.** (introduction to stochastic processes - lawler) The reflection principle. Suppose $X_1, X_2, ..., X_n$ are independent random variables whose distribution is symmetric about the origin. Let $S_0 = 0, S_n = X_1 + ... + X_n$. Then for every $\tau > 0$,

$$P(\max\{S_0, ..., S_n\} > \tau) \leq 2P(S_n \geq \tau).$$

**Proof.** Let $T = \inf_i\{S_i > \tau\}$. Note

$$P(\max\{S_0, ..., S_n\} > \tau) = \sum_{j=0}^{n} P(T = j),$$

and

$$P(S_n \geq \tau) = \sum_{j=0}^{n} P(T = j, S_n \geq \tau) = \sum_{j=0}^{n} P(T = j)P(S_n \geq \tau|T = j).$$

However, independence and symmetry of the distribution of $X_1, ..., X_n$ show that

$$P(M_n \geq \tau|T = j) \geq P(S_n - S_j \geq 0|T = j) = P(S_n - S_j \geq 0) \geq \frac{1}{2}.$$

**Lemma 8.** Brownian bridge boundary crossing probability

$$P(T_\tau < n|S_n = \theta) = \exp\left\{\frac{-2\tau(\tau - \theta)}{\text{var}(S_n)}\right\}. \quad (A.1)$$

**Proof.** We can look at an infinitesimally small area $d\theta$ around $\theta$

$$P(T_\tau \leq n|S_n = \theta) = \frac{P(T_\tau < n, S_n = \theta)}{P(S_n = \theta)}. \quad (A.2)$$
The numerator can be developed to

\[ P(T_\tau < n, S_n = \theta) = P(T_\tau < n)P(S_n \in d\theta | T_\tau < n) \]  
\[ = P(T_\tau < n)P(S_n \in 2\tau - d\theta | T_\tau < n) \]  
\[ = P(S_n \in 2\tau - d\theta, T_\tau < n) \]  
\[ = P(S_n \in 2\tau - d\theta) \]  
\[ = \frac{1}{\sqrt{\text{var}(S_n)}} \phi \left( \frac{2\tau - \theta}{\sqrt{\text{var}(S_n)}} \right) d\theta \]  
\[ (A.3) \]

Similarly, the denominator

\[ \text{denominator} = \frac{1}{\sqrt{\text{var}(S_n)}} \phi \left( \frac{\theta}{\sqrt{\text{var}(S_n)}} \right) d\theta \]

Plugging back into \( A.2 \)

\[ P(T_\tau < n | S_n = \theta) = \frac{\phi \left( \frac{2\tau - \theta}{\sqrt{\text{var}(S_n)}} \right)}{\phi \left( \frac{\theta}{\sqrt{\text{var}(S_n)}} \right)} \]
\[ = \exp \left\{ -1 \frac{(2\tau - \theta)^2}{2 \text{var}(S_n)} + \frac{1}{2} \frac{\theta^2}{\text{var}(S_n)} \right\} \]  
\[ = \exp \left\{ -2\tau \frac{(\tau - \theta)}{\text{var}(S_n)} \right\}. \]  
\[ (A.8) \]

\[ (A.9) \]

\[ (A.10) \]

\[ \square \]

**Definition 4.** [Dudley 2010] A random variable \( T \) which is a function of \( X_1, X_2, \ldots \) is a stopping time if \( T \) has nonegative integer values and for all \( n = 1, 2, \ldots \) there is an event \( A_n \) such that \( T \leq n \) if and only if \( (X_1, \ldots, X_n) \in A_n \), while for \( n = 0, \{T = 0\} \) is either empty (the usual case) or the whole space. For a nonnegative integer-valued random variable \( X \), we have

\[ \sum_{j=1}^{\infty} P(X \geq j) = \sum_{j=1}^{\infty} \sum_{k \geq j} P(X = k) \]
\[ = \sum_{k=1}^{\infty} P(X = k) \sum_{j=1}^{k} 1 \]
\[ = \sum_{k=1}^{\infty} kP(X = k) = EX \leq +\infty \]  
\[ (A.11) \]
Lemma 9. ([Wald 1944], proof from Dudley (2010)) Wald’s Identity. Let $S_T$ be a sum of independent identically distributed random variables $X_1 + ... + X_T$, where $EX_i < \infty$. Let $S_n = X_1 + ... + X_n, i > T$ and $S_0 = 0$. Let $T = \{\inf_i S_i = a\}, T > 0$ where $a$ is a constant be a random variable with $ET < \infty$. Then $ES_T = ETEX$.

Proof. If $T = 0$ the identity is trivial. Otherwise

$$ES_T = \sum_{n=1}^{\infty} P(T = n)E(X_1 + ... + X_n | T = n)$$ (A.12)

$$= \sum_{n=1}^{\infty} P(T = n)\sum_{j=1}^{n} E(X_j|T = n)$$ (A.13)

$$= \sum_{j=1}^{\infty} \sum_{n=j}^{\infty} P(T = n)E(X_j|T = n)$$ (A.14)

$$= \sum_{j=1}^{\infty} E(X_j 1_{T=n})$$ (A.15)

$$= \sum_{j=1}^{\infty} E(X_j 1_{T \geq j})$$ (A.16)

$$= \sum_{j=1}^{\infty} E(X_j (1 - 1_{T \leq j-1}))$$ (A.17)

The event $\{T \leq j - 1\}$ is independent of $Y_j$, therefore

$$ES_T = \sum_{j=1}^{\infty} E(X_j)P(T \geq j) = EX \sum_{j=1}^{\infty} P(T \geq j) = EXET$$

by definition 4.

\[\square\]
Part VI

Bibliography
Bibliography


