MACHINE LEARNING ON A BUDGET

KIRILL TRAPEZNIKOV

Dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
BOSTON UNIVERSITY
COLLEGE OF ENGINEERING

Dissertation

MACHINE LEARNING ON A BUDGET

by

KIRILL TRAPEZNIKOV
B.S., Boston University, 2007
M.S., Boston University, 2011

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
2013
Acknowledgments

First, I would like to thank my family for all their support. Most importantly, I would like to thank my mom for always being there and my grandfather for his encouragement. Second, I would like to thank my advisors, Venkatesh Saligrama and David Castañón, for their guidance. Lastly, this work would not have been possible without the funding provided by the National Science Foundation, National Geospatial Intelligence Agency and U.S. Department of Homeland Security.
ABSTRACT

In a typical discriminative learning setting, a set of labeled training examples is given, and the goal is to learn a decision rule that accurately classifies (or labels) unseen test examples. Much of machine learning research has focused on improving accuracy, but more recently costs of learning and decision making are becoming more important. Such costs arise both during training and testing. Labeling data for training is often an expensive process. During testing, acquiring or processing measurements for every decision is also costly. This work deals with two problems: how to reduce the amount of labeled data during training, and how to minimize measurements cost in making decisions during testing, while maintaining system accuracy.

The first part falls into an area known as active learning. It deals with the problem of selecting a small subset of examples to label, from a pool of unlabeled data, for training a good classifier. This problem is relevant in many applications where a large collection of unlabeled data is readily available but to label an instance requires using an expensive expert (a radiologist annotating a medical image). We study active learning in the boosting framework. We develop a practical algorithm that labels examples to maximally reduce the space of feasible classifiers. We show that, under
certain assumptions, our strategy achieves the generalization error performance of a system trained on the entire data set while only selecting logarithmically many samples to label.

In the second part, we study sequential classifiers under budget constraints. In many systems, such as medical diagnosis and homeland security, sensors have varying acquisition costs, and these costs account for delay, throughput or monetary value. While some decisions require all measurements, it is often unnecessary to use every modality to classify every example. So the problem is to learn a system that, for every decision, sequentially selects sensors to meet a measurement budget while minimizing classification error. Initially, we study the case where the sensor order in which measurement are acquired is given. For every instance, our system has to decide whether to seek more measurements from the next sensor or to terminate by classifying based on the available information. We use Bayesian analysis of this problem to construct a novel multi-stage empirical risk objective and directly learn sequential decision functions from training data. We provide practical algorithms for binary and multi-class settings and derive generalization error guarantees. We compare our approach to alternative strategies on real world data. In the last section, we explore a decision system when the order of sensors is no longer fixed. We investigate how to combine ideas from reinforcement and imitation learning with empirical risk minimization to learn a dynamic sensor selection policy.
## Contents

1 Introduction .................................................. 1
   1.1 Learning a Classifier .................................. 2
   1.2 Cost of Training a Classifier ........................ 3
       1.2.1 Active Learning .................................. 4
       1.2.2 Contribution ..................................... 6
   1.3 Learning with Test-Time Costs ....................... 7
       1.3.1 Contribution ..................................... 8
   1.4 Organization ............................................ 9

2 Active Learning ............................................. 11
   2.1 Motivation ............................................. 12
   2.2 Active Learning Problem ............................... 12
   2.3 Background ............................................ 14
       2.3.1 Version Space Methods ............................ 14
       2.3.2 Margin Based Methods .............................. 16
       2.3.3 Other Approaches ................................ 18
       2.3.4 Boosting ......................................... 19
   2.4 Active Boosted Learning ................................. 24
       2.4.1 ActBoost Algorithm ................................. 25
       2.4.2 Randomly Sampling a Polyhedron ................. 28
       2.4.3 ActBoost vs. Query-by-Boosting (QBB) .......... 29
   2.5 Theory .................................................. 33
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5.1 Volume Convergence Rate</td>
<td>34</td>
</tr>
<tr>
<td>2.5.2 Error Convergence</td>
<td>38</td>
</tr>
<tr>
<td>2.5.3 LP for Bounding Coherence</td>
<td>41</td>
</tr>
<tr>
<td>2.5.4 Undersampling vs. Oversampling</td>
<td>42</td>
</tr>
<tr>
<td>2.6 Experiments</td>
<td>44</td>
</tr>
<tr>
<td>2.6.1 Unbiased Initialization</td>
<td>45</td>
</tr>
<tr>
<td>2.6.2 Biased Initialization</td>
<td>45</td>
</tr>
<tr>
<td>3 Sequential Sensor Selection</td>
<td>50</td>
</tr>
<tr>
<td>3.1 Ordered Sequence of Sensors</td>
<td>51</td>
</tr>
<tr>
<td>3.2 Contribution: Sequential Reject Classifiers</td>
<td>55</td>
</tr>
<tr>
<td>3.3 Related Work</td>
<td>56</td>
</tr>
<tr>
<td>3.3.1 Single Stage Reject Classifiers</td>
<td>56</td>
</tr>
<tr>
<td>3.3.2 Generative &amp; Parametric Modeling</td>
<td>57</td>
</tr>
<tr>
<td>3.3.3 Discriminative Learning Approaches</td>
<td>57</td>
</tr>
<tr>
<td>3.3.4 Other Cost Sensitive Methods</td>
<td>59</td>
</tr>
<tr>
<td>3.4 Sequential Reject Classifiers in a Binary Setting</td>
<td>60</td>
</tr>
<tr>
<td>3.4.1 Problem Formulation</td>
<td>60</td>
</tr>
<tr>
<td>3.4.2 Bayesian Setting</td>
<td>62</td>
</tr>
<tr>
<td>3.4.3 Reject Classifier As Two Binary Decisions</td>
<td>65</td>
</tr>
<tr>
<td>3.4.4 Stage-wise Empirical Minimization</td>
<td>67</td>
</tr>
<tr>
<td>3.4.5 Algorithm</td>
<td>69</td>
</tr>
<tr>
<td>3.4.6 Generalization Error</td>
<td>73</td>
</tr>
<tr>
<td>3.4.7 Experiments</td>
<td>74</td>
</tr>
<tr>
<td>3.5 Sequential Reject Classifiers in a Multi-Class Setting</td>
<td>85</td>
</tr>
<tr>
<td>3.5.1 Multi-Class Problem Setting</td>
<td>85</td>
</tr>
<tr>
<td>3.5.2 Multi-Class Bayesian Analysis</td>
<td>86</td>
</tr>
</tbody>
</table>
3.5.3 Stage-Wise Empirical Risk Minimization .................................. 88
3.5.4 Multi-class decision with a reject option .................................... 88
3.5.5 Algorithm .................................................................................. 91
3.5.6 Complexity of a Multi-Stage System .......................................... 94
3.5.7 Experiments .............................................................................. 95

4 Future Work .................................................................................. 102

4.1 Towards Dynamic Sensor Selection .............................................. 102
4.2 Notation ....................................................................................... 102
4.3 Risk of Using a Policy ................................................................. 107
4.4 Cost-to-go and Reinforcement Learning Approach ......................... 107
4.5 Imitation Learning ....................................................................... 109
  4.5.1 Learning a policy for every stage .............................................. 111
4.6 Novel Risk Based Formulation ..................................................... 112
  4.6.1 Reduction to Weighted Classification ....................................... 113
  4.6.2 Risk based forward imitation learning algorithm ...................... 115
4.7 Common Issues ........................................................................... 115
4.8 Simulation .................................................................................... 117

5 Conclusion .................................................................................... 121

5.1 Future Directions ....................................................................... 122

A ActBoost ......................................................................................... 123

A.1 Connection between Query By Boosting and SVM Active Learning .. 123
A.2 More Details On Boosting ............................................................ 126
A.3 Proof of Theorem 2.1 ................................................................. 127
A.4 Proof of Lemma 2.2 .................................................................... 129
  A.4.1 Chernoff Bound on a Gamma distribution ............................. 133
  A.4.2 Integral of the Chernoff Bound on a Gamma distribution ....... 134
List of Tables

3.1 Dataset Descriptions ......................................................... 77
3.2 Performance illustration for different datasets (quantitate view of the curves). Datasets have 2 sensing modalities. Centralized denotes the test error obtained with all modalities. Last three columns denotes performance for different approaches. Performance is measured by the average number of examples requiring 2nd stage to achieve error close to centralized. Utility approach does not work for last three datasets due to high-dimensionality issues. We note the significant gains of our approach over competing ones of many interesting datasets. ........................................ 79
3.3 Dataset Descriptions .......................................................... 96
3.4 In this table we report an average percent of the maximum budget required to achieve the target error rate. The target rate is chosen to be close to the error of the centralized strategy. Thus if there is a maximum of 2 stages and we obtain a value of 28% for our strategy it means that for only 28% of examples a 2nd stage is utilized without any degradation in error. ................................................................. 101
B.1 Stage Complexity: we use polynomial kernel classifiers. This table displays the degree of the polynomial kernel used at each stage for the rejector and the stage classifier .................................................... 148
List of Figures

2.1 Active Learner, in conjunction with an oracle, labels a small subset of examples to learn a good classifier .......................... 13

2.2 Illustration of a generalized binary search on a one dimensional example. Each row of points represents a new iteration. The version space consists of threshold classifiers on this interval (represented by black boxes). Examples belong to two classes (blue circles and red squares). Solid color indicates already labeled examples. The green rhombus indicates an unlabeled example that approximately bisects the current version space and is to be labeled next. Generalized binary search quickly eliminates a large portion of the version space after every new labeled example and converges to the target classifier .......................... 15

2.3 Margin based active learning methods iterate between two steps: (1) selecting an unlabeled example that is most ambiguous with respect to the current model estimate and (2) labeling the selected example and retraining (updating) the model .......................... 17

2.4 Iterations of margin based active learning. Labeled examples are shown in solid color. The next unlabeled candidate to label is outlined by a diamond. This candidate is an example lying closest to the current estimate of the classifier (a linear boundary in this scenario). Once a new point is labeled, classifier is retrained .......................... 18

2.5 In this illustration, one dimensional thresholds (stumps) are linearly combined to produce a more complex boundary .......................... 20
2.6 Version space is a polyhedron in a high dimensional space. New labeled examples appear as new constraints in the version space. As more examples are labeled, the version space shrinks.

2.7 Selecting an example that approximately bisects the current version space. Once the label is revealed either (approximate) half is eliminated.

2.8 Instead of computing a high dimensional integral, volume and finding an example that achieves the most disagreement is accomplished by drawing random samples.

2.9 Initialization bias is tested on a 1D linear set (in 2.9(a)) consisting of three clusters. ActBoost and QBB are initialized with examples drawn only from the first two clusters. Both ActBoost and QBB are trained using Adaboost on their respective labeled examples. ActBoost is robust to initialization bias while QBB does not find the third cluster until all the first two clusters are labeled.

2.10 Stump Classifiers.

2.11 Accuracy of the worst case classifier vs. # labeled examples. For each curve, only the specified fraction of hypotheses is allowed in the version space. The true sparsity of the target classifier is 3%.

2.12 The parameter $K$ required for maintaining graph connectedness in the case of undersampled and oversampled weak learners. For each case shown: 1D dataset with stumps, the problem matrix $H$ and the neighborly graph reduction. Oversampling weak learners results in increase in $K$.

2.13 Two dimensional datasets: Gaussian Clusters 2.13(a), Box Dataset 2.13(b), Banana Dataset 2.13(c)
Parameter Selection: Accuracy vs # labeled examples as a function of ActBoost classifier samples $D$, more samples provide a better approximation of the GBS metric integral, resulting in better performance. \[ (2.14(a)) \], and as a function of Hit and Run iterations (HT): changing HT does not change performance. \[ 2.14(b) \]... 47

Accuracy vs. # labeled examples. 2D datasets: BANANA \[ (2.15(a)) \] and BOX \[ (2.15(b)) \]. ActBoost(sp) learns in the sparse version space and performs better than ActBoost which operates on the full version space. Multivariate Dataset: MUSHROOM \[ (2.15(c)) \]. ActBoost does not show significant performance improvement over QBB. 48

Robustness to Initialization Bias: binary datasets consisting of three clusters are formed from multi-class datasets (except for gauss clusters). Both algorithms are initialized only from the first two clusters; ActBoost is not affected. 49

Multi-Stage System consists of $K$ stages. Each stage is a classifier with a reject option. The system incurs a penalty of $c_{k+1}$ at $k$th stage if it rejects to seek more measurements. The $k$th classifier only sees the first $k$ sensing modalities in making a decision. 52
3.2 Advantage of a 2 stage classifier: 10 samples, binary (squares, circles).

The red line is the optimal decision when using only the 1st stage modality.

The blue line is optimal if using both. The curve is classification error vs. samples rejected (cost). The red point corresponds to classifying everything at stage 1. The blue corresponds to rejecting everything and classifying using both modalities. The green is a partial reject strategy. The samples outside the green region are classified using only the first modality, and samples inside the region are rejected to stage 2 and are classified using both modalities. Note that blue and green have the same error, while the reject strategy (green) has to use 2nd stage sensor only for $\frac{1}{2}$ of examples, reducing the cost by a factor of 2.

3.3 Optimal Reject Region can be expressed as the disagreement region of two binary classifiers ($f_n$ and $f_p$).

3.4 Comparison of Global to Utility on (a) quantized two gaussian clusters and (b) mammogram dataset. Reject Rate vs System Error. Reject Rate is the fraction of examples with measurements from both stages. Our approach outperforms Utility possibly because we do not need to estimate probability likelihoods.

3.5 Three datasets are evaluated: pima, polyps and threat. Reject Rate vs Error Rate for a varying $\alpha$. Reject Rate is the fraction of examples with measurements from both stages. Global and Myopic are compared. Global (our approach) has a better performance over all while Myopic does better in some situations.
Two Stage ROC using the global surrogate method. Each ROC curve corresponds to a different value of $\alpha$. The legend displays average reject rate for $\alpha$’s. Note, the circle marker ROC corresponds to the centralized system (100% reject rate). Very good performance can be achieved by requesting only 50% of instances to be measured at the second stage.

Three Stage System. The color maps error. A point on the map corresponds to a performance of a particular multistage classification strategy. The vertical axis is the fraction of examples for which only IR and PMMW measurements are used in making a decision. The horizontal axis is the fraction of examples for which all three modalities are used. An example red point in the figure, $\{.4, .15, .195\}$, correspond to a system where 40% of examples use IR and PMMW, 15% use only $IR$ and the rest of data (45%) use all the modalities. And this strategy achieves a system error rate of 19.5%.

To illustrate our parametrization, consider a binary classification setting. $d^k(\cdot)$ is a hyperplane and the confidence $\sigma_{d^k}(\cdot)$ is the distance to this hyperplane. A possible reject region is constructed by thresholding the confidence by a rejector $g^k(\cdot)$. Note how the width of the reject region varies with $x^k$ because $g^k$ is a function of $x^k$.

The figure illustrates the simplified optimization problem for $g^k(x^k)$ in Lemma 2. The objective is to find a rejector function to fit the decision regions in the figure. The data in the green region has cost-go smaller than the risk of classifying at the current stage and therefore is to be rejected. The data outside the green has higher cost-to-go than misclassification risk and is to be not rejected.
We display the decision boundaries of our method and the myopic approach for a fixed budget of 1.3. 1st stage classifier, $d^1$, is in blue. 2nd stage classifier, $d^2$, is black. The space that is rejected to 2nd stage is in green. Observe how our method only rejects the area around the first blue boundary. In contrast, myopic uniformly rejects samples around both boundaries even if the samples will be misclassified at the second stage. This is because our strategy anticipates that the 2nd stage classifier cannot really classify examples around the second blue boundary and does not suffer the acquisition cost for those examples. This results in higher error for the same budget for myopic.

Here, we compare our method to myopic on the MNIST data. We construct four stages of increasing resolution by averaging the original digit images. The experiment demonstrates the advantage of our approach. Also note that the performance of a full resolution sensor can be achieved using a much lower resolution measurement.

(a - c) illustrate error vs budget trade-off for our method and myopic various dataset. Clearly, our method is superior to myopic and can achieve performance of a centralized classifier (black diamond) with a significantly lower acquisition budget.

(a - c) illustrate error vs budget trade-off for our method and myopic various dataset. Clearly, our method is superior to myopic and can achieve performance of a centralized classifier (black diamond) with a significantly lower acquisition budget.
4.1 Schematic of an adaptive sensor selection system. Initially, no measurements are collected and the policy \( \pi(s) \) selects the first sensor to collect. Based on the first sensor information, the policy either decides to stop and classify using \( F(s) \) or select the next sensor. The procedure is repeated until either all the sensors are exhausted or an example is classified. Note that every example may follow a different sequence of sensors.

4.2 Simulated Example

4.3 Three approaches are compared: our novel risk-based imitation learning, fixed order strategy and an oracle.

A.1 Version Space for the two algorithms: solid lines represent hyperplanes imposed by labeled examples, dotted line is the example to be labeled, the circle is the largest radius ball inscribed in the linear constraints: (a) SVM Active Learning: the plane of the page is the surface of the unit-norm sphere \( ||q||_2 = 1 \); and (b) QBB Active Learning: dashed triangle is the probability simplex, note how the solution \( q_a \) is shifted towards a vertex away from the center.
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1D</td>
<td>one dimensional</td>
</tr>
<tr>
<td>2D</td>
<td>two dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>three dimensional</td>
</tr>
<tr>
<td>ActBoost</td>
<td>active boosted learning</td>
</tr>
<tr>
<td>AMMW</td>
<td>active millimeter wave</td>
</tr>
<tr>
<td>CAD</td>
<td>computer aided design</td>
</tr>
<tr>
<td>CT</td>
<td>computed tomography</td>
</tr>
<tr>
<td>DP</td>
<td>dynamic programming</td>
</tr>
<tr>
<td>ERM</td>
<td>empirical risk minimization</td>
</tr>
<tr>
<td>IDS</td>
<td>intrusion detection security</td>
</tr>
<tr>
<td>IR</td>
<td>infrared</td>
</tr>
<tr>
<td>GBS</td>
<td>generalized binary search</td>
</tr>
<tr>
<td>LP</td>
<td>linear programming</td>
</tr>
<tr>
<td>PAC</td>
<td>probably approximately correct</td>
</tr>
<tr>
<td>POMDP</td>
<td>partially observable Markov decision process</td>
</tr>
<tr>
<td>PMMW</td>
<td>passive millimeter wave</td>
</tr>
<tr>
<td>ROC</td>
<td>receiver operating characteristics</td>
</tr>
<tr>
<td>QBB</td>
<td>query by boosting</td>
</tr>
<tr>
<td>SVM</td>
<td>support vector machine</td>
</tr>
<tr>
<td>UCI</td>
<td>University of California Irvine</td>
</tr>
<tr>
<td>VC</td>
<td>Vapnik Chervonenkis</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Machine learning occupies an important role in many scientific and engineering problems. Its applications include regression, clustering, classification and many others. In this work, we limit ourselves to classification. In classification, given a set of training examples consisting of measurements and a corresponding class membership (a label), the goal is to learn a classifier that correctly labels unseen examples. Classification is a well studied area in machine learning with majority of effort concentrated on improving generalization error (error on examples outside the training set). However, recently cost of training classifiers and cost of making decisions during testing have gained importance. For learning, a labeled training set is required. Labeling is typically time consuming or expensive because it requires an expert to inspect an unlabeled example and assign it a label. The problem of reducing the number of labeled examples in order to train a good classifier is known as active learning, and it will be the focus of Chapter 2. In making a decision during a testing phase, a system incurs the cost of acquiring measurements. Different sensors have varying acquisition costs, and using every sensor to make every decision may be prohibitive due to budget constraints. In Chapter 3, we study the problem of reducing average acquisition cost in classification. In particular, we investigate how to learn a system that, for every decision, sequentially selects sensors in order to trade off system error and acquisition cost.
Organization: This chapter is organized in the following manner. The section 1.1 gives a short overview of supervised classification and ingredients required for learning a successful classifier. In section 1.2, we discuss the cost of labeling a classifier, the problem of active learning and our contribution in this area. In section 1.3, we introduce the problem of learning with test-time costs and summarize our contribution. Section 1.4 explains organization of the rest of the thesis.

1.1 Learning a Classifier

Let us assume that nature generates an example according to some unknown distribution. An example consists of measurements $x$ which can be collected using various sensors. An example also belongs to a particular group or a class $y \in \mathcal{Y}$ unknown to the sensor. We would like to find a classifier which is a function that takes in an unlabeled example and assigns to a class: $F(x) \in \mathcal{Y}$. We would like this classifier $F(x)$ to correctly classify any new example generated by $P(x, y)$:

$$\min_{F} \mathbb{E}_{x,y} \left[ 1[F(x) \neq y] \right]$$

From hypothesis testing, if we know the probability model, the optimal decision rule is to pick a class that maximizes the posterior belief:

$$F(x) = \arg \max_{y} P(y = y' \mid x)$$

However, in many cases, the model is not known or hard to estimate reliably. In such cases, we turn to supervised learning. Instead of modeling the posterior, we directly learn the decision rule by approximating the expected error by an empirical risk. To construct an empirical risk, we are required to collect a sample of training examples independently generated by the underlying distribution. And for each
training example, we also need to determine its class membership (ground truth):

\[
\min_{F \in \mathcal{F}} \sum_{i=1}^{N} 1_{[F(x) \neq y]}
\]  

(1.3)

In addition, we restrict the decision function to be part of a particular regularized family of classifiers \( \mathcal{F} \) so as to not end up with a look up table on the training set and improve the generalization of \( F(x) \). For example: \( \mathcal{F} \) can be a family of linear functions of \( x \). Also, since \( 1_{[\cdot]} \) is not continuous and not convex, a typical approach is to minimize a surrogate \( C[\cdot] \geq 1_{[\cdot]} \) instead. Most classification algorithms fit this description.

### 1.2 Cost of Training a Classifier

But regardless of classification algorithm, the training procedure requires two ingredients:

- training examples, \( x_1, x_2, \ldots \)

- their corresponding labels, \( y_1, y_2, \ldots \)

While collecting examples may be an easy task, obtaining the correct label can turn out to be challenging. We list two examples to demonstrate this point.

Consider the problem of learning a classifier to diagnose whether a patient has a particular disease based on a performed medical imaging procedure (X-ray scan). In this setting, \( x \), is the image collected, and \( y \), in the simplest case, is a binary opinion of a radiologist whether the image shows that this disease is present. In a more complicated case, a radiologist needs to annotate the image, highlighting the areas that indicate abnormalities. In automatic text article categorization, the task is to automatically determine the subject of an article (i.e. politics or sports). Here, training example is an article. The number of articles available on the internet is
very large. To label an article, a person has to read it (or at least skim it), a time consuming process.

To learn a classifier that has good performance requires many such training examples. A natural question arises. How can we limit the number of times we have to ask a radiologist (an expert) to label a particular piece of data?

If we have a large corpus of unlabeled data readily available then the problem is how to select a subset of that data to label in order to learn a good classifier. For example, clearly if two images are exactly the same then there is no need for labeling both of them. This can be extended even further. If two images are not exactly the same but very alike (with respect to some similarity measure) then possibly both labels are also not necessary. If some examples are easily differentiable while others are more ambiguous then the latter may be more valuable in learning a good classifier. The study of these issues is dealt in the context of active learning.

1.2.1 Active Learning

Active learning is a type of semi-supervised learning. Semi-supervised learning is an area of machine learning that studies learning in the presence of partially labeled data. While there are many active learning approaches, we will concentrate on one type of active learning known as version space approach (Freund et al., 1997). Recall the objective is to learn a good classifier. Assume, for now, that there exists a perfect separator; there is a target classifier (from a particular family of classifiers) that can perfectly distinguish between two classes and achieve zero error. We are limiting ourselves to a binary setting.

This implies that for any collection of data examples, there exists a classifier that perfectly separates, and this classifier may not necessary be unique. So we can further assumes that there is a space of classifiers that perfectly separate a given collection of labeled examples. This is known as version space.
As the number of labeled examples increases, size of the version space will decrease. So a natural strategy is to iteratively label examples. Once the label for an example is revealed a fraction of the version space is eliminated. The problem is how to choose an example that will eliminate a large number of potential classifiers. Since the label is not known, the best one can do is pick an example that has the greatest disagreement among the classifiers in the current version space. Once the label is revealed then ideally half of the version space will be eliminated. This procedure is known as generalized binary search (Nowak, 2009).

However, there are several issues. Existence of version space is not always guaranteed. If there is no classifier (from a particular family) that can achieve zero error then the version space is empty. It is unclear how to performing generalized binary search when the space of classifiers is uncountably infinite. How to guarantee, that labeling examples to decrease version space, also decreases generalization error of a classifier trained on the examples (or sampled from the final version space)?

Due to some of these issues, researchers (Campbell et al., 2000; Tong and Koller, 2001) have considered margin based active learning. In these methods, a small number of samples is labeled initially, and a classifier is trained on this initial set. Next, a classifier is evaluated on the unlabeled examples, and an example with lowest classification confidence (margin) is labeled. The classifier is retrained with this new training example and the process is repeated. However, this method is sensitive to bias in the initial labeled set. The process will only label examples in the vicinity of the initial set and never explore the full example space.

The issues with version space methods and margin based active learning motivates us to study active learning in the boosting framework.
1.2.2 Contribution

Our work on Active Boosted Learning is described in Chapter 2. Boosting is a method of combining simple classifiers (weak learners) to construct a complex decision boundary. We develop an active learning algorithm in version space of boosted classifiers. The reason for working in the space of boosted classifiers is due to the weak learning assumption. The assumption states that for any distribution over a training set there exists a weak learner with performance better than random guessing. If weak learning assumption holds then there exists a boosted classifier with perfect separation (Freund et al., 1996) therefore guaranteeing a non-empty version space.

Next, we highlight our major contributions:

- A novel active learning strategy that reduces the space of feasible boosted classifiers
- Characterize reduction rate of the version space in terms of the sign pattern of weak learners on the training set
- Show that under sparsity assumptions our active learning strategy labels logarithmically many examples to achieve performance of a classifier trained on the entire training pool
- Present a practical convex relaxation algorithm that scales polynomially with the number of weak learners
- Demonstrate advantage of our strategy on several datasets, especially its robustness to initialization bias
1.3 Learning with Test-Time Costs

So far we discussed costs involved during a training phase of a classifier. In this section, we introduce costs that arise during testing.

Assume we collected our data, we employed an expert to label it and trained a classifier, \( F(x) \). Now, in the field, we would like to use our system to classify unseen examples. Recall that in order to obtain measurements, we had to utilize a collection of sensors. Let \( x \) have \( K \) components, and in order to acquire \( k \)th component, \( x_k \), we need a separate sensor. If we use this sensor, we incur a cost of \( c_k \). For a particular example \( x \), we can define an acquisition cost as a sum of all the sensor costs used in making a decision.

Now, if our classifier \( F(x) \) is non-adaptive then, for every decision, we collect all the measurements (use every sensor), then the average cost of using \( F(x) \) will always be the same.

However, in many applications, some decisions are easy and require only few measurements while other decisions are more difficult and require the use of all the sensors. We explain the following example to provide some intuition.

**Motivating Example:** Consider the problem of classifying whether a luggage bag contains a threat (a dangerous object) or not. We have access to two sensors: an x-ray imager and a human operator that inspects a bag. An imager is a fast modality but may not provide information to correctly classify every bag. A human is slow and expensive but has the highest accuracy. Ideally, we would like to use both sensors on every example but this is not possible since the delay (or the cost) will be too high.

An adaptive system will use the imager first and, based on this cheap modality, decide whether to classify or to reject (request) the more expensive modality. So only the more ambiguous examples will require both the cheap and the expensive sensors,
and (hopefully) the majority will be classified using only the cheap modality.

This is a problem of sequential sensor selection and is studied in Chapter 3. During training, all the measurements are assumed to be available, and the objective is to minimize the trade-off between average acquisition cost and classification performance during testing phase.

\[
\min_F \mathbb{E}_{x,y} \left[ \text{error}(F) + \alpha \text{cost}(F) \right]
\]

The trade-off is captured by the \( \alpha \) parameter. If \( \alpha \) is small then the emphasis is on minimizing error and ignoring cost. For large \( \alpha \), sacrifices in error have to be made in order to achieve smaller cost. The goal is to learn a system that, for every decision, sequentially selects sensors to optimize the above objective.

### 1.3.1 Contribution

In the first part of Chapter 3, we assume that the order in which sensors can be used is fixed. This is motivated by systems where the order of the stages is fixed by the physics of the problem. There are \( K \) stages and each stage is associated with a sensor. The stages are arranged from the least expensive (least informative) to the most expensive (and most informative). The fixed order simplifies the problem to learning a decision function at every stage that either decides to classify an example using the available measurement or rejects the example to the next stage.

Our contribution in this setting is the following:

- Decompose an expected risk for the system in a convenient stage-wise form
- Approximate the expected risk with a multi-stage empirical risk
- Parametrize a reject decision: as a disagreement between two biased classifiers in the binary setting and as a separate decision function in the multi-class setting
• Derive algorithms that minimize the empirical risk by reducing to a series of supervised learning problems

• Deriving generalization error guarantees

In the last chapter, we investigate sequential decision system in a general setting and explore potential directions for future research in this setting. The order of sensors is no longer fixed, and the goal is to learn a system that sequentially selects sensors for every decision. We investigate how to combine ideas from imitation learning with empirical risk minimization to learn a dynamic sensor selection policy. In imitation learning, researchers assume existence of an oracle, a near optimal policy. An approximate policy is trained to imitate the oracle on the training set. However, the oracle may be too powerful for an approximate policy to learn to imitate successfully. Therefore, we introduce a empirical risk based formulation. Instead of learning to match the oracle directly, we utilize the oracle to compute risks of taking actions (selecting sensors). Using these risks on the training data, we learn a policy to minimize a cost-sensitive classification problem.

1.4 Organization

The rest of this thesis is organized in the following manner:

Chapter 2 will describe our work on active boosted learning. We will introduce the problem in more detail, summarize related work, explain our active boosted learning algorithm, provide theoretical results and illustrate performance with numerical experiments.

Chapter 3 will deal with sequential sensor selection. Again, we will explain the problem in more detail and describe related work in this area. Then, at first, we consider the setting of fixed sensor order. In this setting, we present results on
learning sequential reject classifiers in the case of binary classification. Next, we extend to a multi-class setting.

In Chapter 4, we present directions for future research. We investigate how to learn a sequential sensor selection system when the order of sensors is not longer fixed.

In Chapter 5, we provide some concluding remarks.

Proofs that are not found directly after theorems/lemmas are located in the appendix.
Chapter 2

Active Learning

The first part of the thesis deals with reducing labeling cost when training decisions systems. This area of machine learning is known as active learning. On a high level, the problem is to select a small subset of examples to label, from a pool of unlabeled data, for training a good classifier. We study the problem in the setting of boosted classifiers. In contrast to much of the recent efforts, which has focused on selecting the most ambiguous unlabeled example to label based on the current learned classifier, our approach is to select examples to maximally reduce the volume of the version space of feasible boosted classifiers. We show that under suitable sparsity assumptions, this strategy achieves the generalization error performance of a boosted classifier trained on the entire data set while only selecting logarithmically many unlabeled samples to label. We also establish a partial negative result, in that without imposing structural assumptions it is difficult to guarantee generalization error performance. We explicitly characterize our convergence rate in terms of the sign pattern differences produced by the weak learners on the unlabeled data. We also present a convex relaxation to account for the non-convex sparse structure and show that the computational complexity of the resulting algorithm scales polynomially in the number of weak learners. We test ActBoost on several datasets to illustrate its performance and demonstrate its robustness to initialization. The work presented here is partially published in Trapeznikov et al., 2011.
Organization: In section 2.1, we motivate the problem. In section 2.2, we describe active learning problem in more detail. In section 2.3, we provide background material. In section 2.4, we describe our Active Boosted Learning method. In section 2.5, we present theoretical results. In section 2.6, we provide numerical experiments.

2.1 Motivation

Any supervised machine learning algorithm requires a labeled set of examples to train a classifier. A labeled example consists of a feature vector and a label indicating its class membership. We will only consider binary classification in this work. To obtain a label requires an expert to analyze an unlabeled example. Active learning deals with the problem of selecting a small subset of unlabeled data for an expert to label in order to train a good classifier. In practice, only a small fraction of examples aids the learning process. Consider any binary classification example where two classes are separated by a boundary. Only the examples in the vicinity of this boundary provide useful information for learning the target classifier; the labels for the rest of the examples are not needed.

To reduce the number of labeled examples is important in situations where a large amount of unlabeled data is readily available but labeling it is a costly process. For example in medical imaging, many patients undergo various imaging procedures but to label a medical image requires a specialist’s time. In luggage scanning applications, while many scans of bags may be available, to properly label an object requires a human either analyzing the image or physically opening the bag. Any situation where labeling is either costly or time consuming can benefit from active learning.

2.2 Active Learning Problem

Let us outline a pool-based active learning problem. We are give the following:
Figure 2·1: Active Learner, in conjunction with an oracle, labels a small subset of examples to learn a good classifier

- Unlabeled training set: \( \{ x_1, x_2, \ldots, x_M \} \)
- An oracle or an expert that can reveal associated binary labels for a fixed cost:
  \[ \{ y_1, y_2, \ldots, y_M \}, \ y_i \in \{ +1, -1 \} \]
- A supervised learning algorithm that learns a classifier \( f(\mathbf{x}) \in \{ +1, -1 \} \) from a labeled training set
- A specified budget \( m \): maximum number of labeled examples or oracle calls allowed

We would like to invoke an oracle to label a small subset of examples \( L \subset X \) such that \( |L| \leq m \). Let \( f_L \) be a classifier trained on \( L \). The objective for an active learning algorithm is to choose \( m \) examples to label in order to achieve the best performance on the entire dataset:

\[
\min_{L \ s.t. \ |L| \leq m} \sum_{i=1}^{M} 1[ f_L(x_i) \neq y_i ] \tag{2.1}
\]

The problem is how to label the best \( m \) examples in order to achieve low error on the entire training pool. This is illustrated in Figure 2·1.
Streaming Scenario  Another version of active learning is in a streaming setting. Here, active learner does not have an ability to query (or choose) examples from a pool because the pool is so large that querying becomes computationally intractable. Instead, unlabeled examples arrive randomly one at a time. For each arrival a decision has to be made: label or not. Once an example has been decided it cannot be revisited again. In this work, we will only consider a pool based active learning problem.

2.3 Background

Active learning has been well studied by several researchers (see [Settles, 2010]). Most approaches are of greedy form. The labeled set is build up iteratively one (or few) examples at a time. The decision on which example to label next is based on the previously labeled set. Let $L^t$ be a set of examples labeled at a time. Initially, $L^0$ is an empty set. As an active learner labels one example at a time, we construct a sequence of nested sets:

$$L^0 \supset L^1 \supset \ldots \supset L^t$$  \hspace{1cm} (2.2)

2.3.1 Version Space Methods

One stream of work is to keep track of so called version space. The space of all classifiers (from a particular family) that agree with current labeled set. Consider a family of classifiers $\mathcal{F}$. The version space at a time $t$ is a set of all classifiers in this family that agrees with the current labeled set.

$$V^t = \{f \in \mathcal{F} \mid f(x_i) = y_i, \forall (x_i, y_i) \in L^t\}$$  \hspace{1cm} (2.3)

We get a similar nested sequence of version spaces for every time:

$$V^0 \supset V^1 \supset \ldots \supset V^t$$  \hspace{1cm} (2.4)
As more examples are labeled than the version space volume (with respect to some measure) decreases.

This construction assumes a separable scenario: there exists a classifier that correctly separates all the data in the unlabeled pool: \( f^*(x_i) = y_i, i = 1 \ldots M \). This assumption guarantees a non-empty version space at every iteration. Also note that version space at every time also contains the target classifier: \( f^* \in V^t, t = 1 \ldots M \). So a natural approach is to attempt to maximally reduce version space \( V^t \) with every new labeled example.

Figure 2-2: Illustration of a generalized binary search on a one dimensional example. Each row of points represents a new iteration. The version space consists of threshold classifiers on this interval (represented by black boxes). Examples belong to two classes (blue circles and red squares). Solid color indicates already labeled examples. The green rhombus indicates an unlabeled example that approximately bisects the current version space and is to be labeled next. Generalized binary search quickly eliminates a large portion of the version space after every new labeled example and converges to the target classifier.

This idea of reducing the version space in active learning has been considered
by several researchers ([Seung et al., 1992] [Gilad-Bachrach et al., 2005] [Abe and Mamitsuka, 1998] [Freund et al., 1997] [Tong and Koller, 2001] [Nowak, 2009]) for the so called separable case. The Query By Committee (QBC) of [Freund et al., 1997] is a version space approach described usually in a streaming scenario. The QBC selects an unlabeled instance to label if two random classifiers chosen from the version space disagree. [Freund et al., 1997] show that if the set of classifiers have a finite VC dimension, the classifiers and the unlabeled data points are chosen from a known prior distribution, and the so called information gain condition is satisfied, the number of examples labeled is small.

Authors in [Nowak, 2009] present a more general frame for version space active learning, known as generalized binary search. Since the goal is to reduce version space. As a new example is labeled, it eliminates all the classifiers that do not agree with its label. However, since the label is not known beforehand, a generalized binary search strategy is to pick an example that approximately halves the version space. This implies, the strategy will pick an example such that half of classifiers in the version label it as +1 and the other half will label it −1. Once the true label is revealed, half of the space is eliminated. Please refer to Figure 2.2 for a simple example. However, the strategy only works in finite classifier families. Our work on Active Boosted Learning will extend the idea of generalized binary search to continuously parametrized classifier families.

2.3.2 Margin Based Methods

One problem with version space methods is a difficulty in sampling the version space for most classifier families. The approaches also do not guarantee a non-empty version space implying that a chosen classifier family may not perfectly separate the dataset.

These issues lead researchers to consider strategies for finding the most ambiguous/uncertain examples based on the current learned classifier (see [Campbell...
Figure 2.3: Margin based active learning methods iterate between two steps: (1) selecting an unlabeled example that is most ambiguous with respect to the current model estimate and (2) labeling the selected example and retraining (updating) the model.

A classifier imposes a boundary in the example space. Proximity to this boundary is one metric of classification confidence. The classifier is less confident on the examples located near the boundary and more confident on the examples in the periphery, so a natural query strategy is to request a label for an example closest to the boundary. Iterations of a margin based strategy is illustrated in Figure 2.4

The fault in margin based strategies is poor performance in the presence of initialization bias. Initialization bias occurs when the initial labeled set is not representative of the entire example space. When a classifier is trained on it, its boundary will be a poor estimate of the true one. If only examples closest to this poor boundary are labeled then the boundary estimate will not improve or improve very slowly.

Exploring this method in the Support Vector Machine framework, Campbell et al., 2000, Tong and Koller, 2001, Schohn and Cohn, 2000, propose a simple algorithm that refines a current SVM boundary by querying examples with the smallest margin. In Tong and Koller, 2001, the SVM solution is shown to approximate the center of the version space, thus examples lying closest to it were likely to bisect it. The effectiveness of the method rests on how well the center of an inscribed circle estimates the center of a polyhedron. To overcome this problem, Gilad-Bachrach et al., 2005 presents a Query by Committee in SVM version space using Hit and Run sampling. All these methods assume separability in higher dimensional kernel space. However,
if this assumption happens to be false, the algorithm loses its theoretical foundation.

Active learning has also been explored in the context of boosted weak classifiers which we will describe in the next section. In this framework, \cite{Abe and Mamitsuka, 1998} follows the margin learning philosophy in their Query-by-Boosting method (QBB). In their method, the query point is the minimizer of the weighted sum of weak classifiers. The reasoning is again to select the least confident example. Since each weak classifier is a hard decision rule, the strong classifier is a linear combination of piecewise constant functions, so the Query-by-Boosting metric lacks the sense of distance to the boundary as in the SVM methods, and the method is still sensitive to poor initialization. Unfortunately, no performance guarantees for QBB can be provided since it suffers from initialization bias as well. For a detailed discussion of similarity between SVM and boosting margin-based active learning please see Appendix Section A.1

This suggests to investigate active learning algorithms in the boosting framework that do not rely on previous classifier estimates.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{margin_active_learning_iterations.png}
\caption{Iterations of margin-based active learning. Labeled examples are shown in solid color. The next unlabeled candidate to label is outlined by a diamond. This candidate is an example lying closest to the current estimate of the classifier (a linear boundary in this scenario). Once a new point is labeled, classifier is retrained.}
\end{figure}

\subsection{Other Approaches}

For the agnostic (non-separable) case, \cite{Dasgupta et al., 2007, Balcan et al., 2009} present algorithms independent of any specific classifier family. The algorithms operate in a streaming scenario where examples arrive sequentially and are either labeled
or ignored. The methods eliminate possible hypotheses by employing a region of uncertainty: a set of examples on which the current set of hypotheses disagrees. The algorithms suffer from computational complexity since they require enumeration of an entire hypothesis class and a supervised learner that is able to optimize a 0–1 loss. In [Beygelzimer et al., 2009] use a different approach that extends to more general loss functions (i.e. hinge loss, logistic loss and etc). The algorithm maintains a set of candidate hypothesis whose loss is not too great on the current labeled set. This set is used to compute a probability of labeling the next arriving example. [Dasgupta et al., 2007] redefine version space as:

\[
    f_t = \arg\min_{h \in V_t} \text{error}\{f, L^t\}
\]

\[
    V_{t+1}^t = \{ f \in V_t | \text{error}\{f, L^t\} \leq \text{error}\{f_t, L^t\} + \Delta_t \}
\]

This modified version space consists of all classifiers that are within \(\Delta_t\) error of the best classifier in the versions space. However, in practice this version space is hard to implement.

2.3.4 Boosting

Before we begin explanation of Active Boosted Learning, let us quickly overview a type of supervised classification approach known as boosting. For more details and discussion, an interested reader can refer to Appendix Section A.2.

Boosting is a well studied problem on how to combine weak hypotheses to yield a strong classification rule. Weak hypothesis is any hard-decision classifier. A weighted sum of weak classifiers is one method of boosting for which [Freund et al., 1996] provides a simple but powerful algorithm known as Adaboost.

The objective is to find a linear combination of weak hypotheses. We designate weak hypotheses as functions \(h_j(x) \in \{+1, -1\}\). Consider a finite set of weak hy-
Figure 2.5: In this illustration, one dimensional thresholds (stumps) are linearly combined to produce a more complex boundary.

The optimzation problem is to find a weight vector $q$ that minimizes indicator loss:

$$
\min_{q \geq 0} \sum_{i=1}^{M} 1_{\{y_i h_i^T q \leq 0\}}
$$

The weights can be constrained to be nonnegative since we can assume there exists a complement for each weak hypothesis. Since indicator functions are hard to optimize, surrogates are used instead. Let $C(z)$ be a convex surrogate that upper-bounds the indicator function:

$$
C(z) \geq 1_{|z| \leq 0}
$$

The optimization problem changes:

$$
\min_{q \geq 0} \sum_{i=1}^{M} C(y_i h_i^T q)
$$

Coordinate Descent Boosting  Ada-boost is a special case of coordinate descent boosting algorithms. Given a cost surrogate $C(z)$, coordinate descent boosting it-
eratively selects a weak hypothesis \( h_{j_t} \) and a weight \( q_{j_t} \) for that hypothesis that minimizes the cost. The final strong classifier consists of the selection from \( T \) iterations: \( \text{sgn}(\sum_{t=1}^{T} q_{j_t} h_{j_t}(x)) \). To perform one coordinate descent iteration requires three steps.

(1) Find Direction: First, we need to find the coordinate (direction) of maximum descent. Let be \( \mathbf{q}^{t-1} \) be weight vector at iteration \( t - 1 \). We find the negative of the partial derivative of the objective with respect to \( q_j \):

\[
d_j = \sum_{i=1}^{M} -C'(y_i h_i^T \mathbf{q}^{t-1}) y_i h_{ij}
\]

(2.10)

\[
C'(z) = \frac{\partial C'(z)}{\partial z}
\]

(2.11)

Then we pick the coordinate with maximum descent:

\[
j^t = \arg \max_{j=1\ldots N} d_j
\]

(2.12)

If we find that we cannot descend anymore \( (d_{j^t} \leq 0) \) the we reached the minimum and terminate.

(2) Find Step Size: Second, we need to find the step size \( \alpha^t \) to take in the direction \( j^t \). This translates to a line search problem:

\[
\alpha^{t+1} = \arg \min_{\alpha > 0} \sum_{i=1}^{M} C(y_i h_i^T \mathbf{q}^{t-1} + \alpha y_i h_{j^t}(x_i))
\]

(2.13)

As before, our cost function \( C(z) \) is differentiable. So we take the derivative of the above expression and set it to zero:

\[
\sum_{i=1}^{M} -C'(y_i h_i^T \mathbf{q}^{t-1} + \alpha^t y_i h_{j^t}(x_i)) y_i h_{j^t}(x_i) = 0
\]

(2.14)
Optimal step size $\alpha^{t+1}$ is the solution to the above equation. In some cases, the solution can be found analytically. Line search methods such as backtracking or newton method can be employed.

(3) **Update Step:** Finally, we update the current solution (just the $j^{t+1}$th component)

$$q_{j^t}^{t+1} = q_{j^t}^t + \alpha^{t+1}$$ (2.15)

**Weak Learner Assumption** [Freund et al., 1996] Note in order to descent, $d_{j^t}$ must be positive. This implies the famous weak learner assumption:

$$\max_{j=1,...,N} \sum_{i=1}^{M} -C'(y_i h_i^T q^t) y_i h_j(x_i) > 0$$ (2.16)

There must exist a weak hypothesis $j^*$ such that the weighted gain is greater than zero. We can rewrite:

$$y_i h_j(x_i) = 1 - 2 \mathbf{1}_{(h_j(x_i)\neq y_i)}$$ (2.17)

We substitute and get the weak learner assumption:

$$\min_{j=1,...,N} \sum_{i=1}^{M} s_i \mathbf{1}_{(h_j(x_i)\neq y_i)} < \frac{1}{2}$$ (2.18)

The assumption requires an existence of a weak hypothesis with training error of less than half for any positive weights $\{s_i\}_{i=1}^{M}$ (better than random guessing). Here, $s_i = -C'(y_i h_i^T q^t)$ and is always positive due to convexity of $C(z)$

Note that as long as the weak learner assumption holds, there is always a decent coordinate. This implies that the objective function is reduced at every iteration. The objective is the sum of cost surrogates $C(z)$. If $C(z)$ is convex then the objective function is convex. And if $C(z)$ approaches zero as $z$ goes to positive infinity then
training error will be driven down to zero with the number of iterations.

**Examples of Boosting Algorithms:** There are many variations for the cost surrogate $C(z)$. The most popular are found in Ada-boost $C(z) = \exp(-z)$. Another interesting one is Logiboost $C(z) = \log(1 + \exp(-z))$. While some experiments suggest Logiboost is more robust to outliers and label noise than Adaboost, since both cost functions are convex, they will both overtrain given enough iterations.
2.4 Active Boosted Learning

Having gained some familiarity with the version space in active learning and boosted classifiers, we are ready to explain our active learning algorithm.

We develop a novel active learner based on boosting. Boosting is a method for combining decisions of weak classifiers to form a strong classifier. A strong classifier is parameterized by a probability weight vector on the weak classifiers. The significance of boosted classifiers is that if the weak learning assumption holds, a probability weight vector and the associated strong classifier can be constructed with training error essentially equal to zero [Freund et al., 1996].

This feature of boosted classifiers motivates considering an active boosted learner based on the version space approach. In our context, the version space is the set of all probability vectors that correctly classify the current labeled training set.

Our approach is similar in spirit to QBC by [Freund et al., 1997] but with important differences. We do not employ a PAC (probably approximately correct) framework or the streaming scenario. We do not assume the information gain conditions or bounds on VC dimension. Instead we draw upon margin based generalization bounds for boosted classifier. Our analysis draws upon the concept of Generalized Binary Search of [Nowak, 2009]. We extend [Nowak, 2009] to the case of continuous classifier spaces, more precisely the space of probability vectors on weak learners. We explicitly relate our convergence rate to a function of the hamming distances of the weak classifiers.

Our active boosted learning (ActBoost) algorithm, at each time, selects examples to approximately bisect the version space. We establish that the number of labeled examples $n$ necessary to reduce the volume of the version space to a fraction $\epsilon$ of the initial volume scales as $n = O(\log \frac{1}{\epsilon})$. ActBoost randomly samples from the version space and chooses an example with maximum disagreement among the sampled
boosted classifiers. We utilize the Hit and Run algorithm from [Lovász and Vempala, 2004] to uniformly sample from a convex body. ActBoost has polynomial computational complexity in the number of weak learners. Hit and Run has been used for SVM active learning by [Gilad-Bachrach et al., 2005].

Nevertheless, we show that in the context of boosting, if all the weak hypotheses are allowed to contribute to the final ensemble, reduction of version space does not guarantee improvement in generalization performance. This motivates imposition of sparsity. Sparsity in the number of boosted weak learners has been employed in the literature [Taylor et al., 2010] and has been shown to improve generalization. We show that under this assumption, our strategy achieves the generalization error performance of a boosted classifier trained on the entire data set while only selecting logarithmically many unlabeled samples to label.

2.4.1 ActBoost Algorithm

We denote by $\mathcal{X} = \{x_i\}_{i=1}^{B}$, the pool of unlabeled data. We let $\mathcal{H} = \{h_j(\cdot)\}_{j=1}^{N}$ denote a finite set of weak classifiers\footnote{ActBoost can be extended to a suitably parameterized continuous space of weak classifiers but we consider a finite set for technical simplicity}. Each weak classifier is a binary valued function $h_j : \mathcal{X} \rightarrow \{+1,-1\}$. Associated with each example $x_i \in \mathcal{X}$ is a binary label $y_i \in \{-1, 1\}$, which is revealed by querying an Oracle. We introduce the simplex of all positive weight vectors $Q$:

$$Q = \left\{(q_1, q_2, \ldots, q_N) \mid \sum_{j=1}^{N} q_j = 1, q_j \geq 0, j = 1, \ldots, N \right\}$$ (2.19)
The space of classifiers under consideration is the set of all weighted combinations of weak classifiers, namely, we define a classifier parametrized by a weight vector $\mathbf{q}$:

$$
\mathbf{q}(\mathbf{x}) := \text{sgn}(\sum_{j=1}^{N} h_j(\mathbf{x})q_j) = \text{sgn}(\mathbf{h}(\mathbf{x})^T \mathbf{q})
$$

(2.20)

where, $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}) \ h_2(\mathbf{x}) \ldots h_N(\mathbf{x})]^T$, $\mathbf{q} = [q_1, q_2, \ldots, q_N]$ and $\text{sgn}(\cdot)$ returns the sign of its argument. We refer to this set of classifiers as boosted classifiers.

Our goal is to select a small subset of examples $\mathbf{x}_i \in \mathcal{X}$ to label so that boosted training on this labeled set leads to a strong classifier, namely that $\sum_{i=1}^{N} 1_{[y_i \neq \mathbf{y}(\mathbf{x}_i)]} \mathbf{y}^T \mathbf{q}$ on the unlabeled data is small. Let $L^t$ be the set of labeled training examples at iteration $t$. Our version space is the set of all classifiers parameterized by the vector $\mathbf{q}$ that correctly classify the labeled training set at an iteration $t$:

$$
Q^t = \left\{ \mathbf{q} \in Q \mid y_i \sum_{j=1}^{N} h_j(\mathbf{x}_i)q_j \geq 0, \ \forall \ i \in L^t \right\}
$$

(2.21)

Note that under the weak learning assumption \cite{Freund96}, it is well known that there exists a set of weights, $q_j$, such that the above set is not empty.

Figure 2.6: Version space is a polyhedron in a high dimensional space. New labeled examples appear as new constraints in the version space. As more examples are labeled, the version space shrinks.

At every iteration, a new unlabeled example is labeled and added to $L^t$, and the version space $Q^t$ decreases such that: (see Figure 2.6)

$$
Q = Q^0 \supset Q^1 \ldots \supset Q^t \supset Q^{t+1}
$$

(2.22)
We want to maximally reduce the version space at every iteration. To do so we draw upon recent ideas from generalized binary search [Nowak, 2009]. Let $U^t$ be a set of unlabeled examples at iteration $t$. The goal is to pick an $x \in U^t$, so that one half of the volume of $Q^t$ labels this point $+1$ and the other $-1$. Once the label $y$ is revealed half of $Q^t$ will be eliminated. (see Figure 2.7)

Figure 2.7: Selecting an example that approximately bisects the current version space. Once the label is revealed either (approximate) half is eliminated.

If the space $Q$ is discrete then the query strategy is:

$$x^t = \arg \min_{x \in U^t} | \sum_{q \in Q^t} q(x)| \quad (2.23)$$

In our case, $Q$ being a continuous space of weight vectors our query strategy is to
pick:

\[
x^t = \arg \min_{x \in U^t} \left| \int_{q \in Q^t} q(x) dq \right| 
\]

(2.24)

\[
x^t = \arg \min_{x \in U^t} |Vol(Q^t_+) - Vol(Q^t_-)|
\]

(2.25)

where, \(dq\) represents the Lebesgue measure on the space of normalized weight vectors:

\[
Vol(Q^t_+) = \int_{q \in Q^t} 1_{[q(x)=1]} dq 
\]

(2.26)

\[
Vol(Q^t_-) = \int_{q \in Q^t} 1_{[q(x)=-1]} dq
\]

(2.27)

The integral is well defined since \(Q_t\) is a bounded polyhedron and the indicator functions described here are Lebesgue measurable.

### 2.4.2 Randomly Sampling a Polyhedron:

The expression in (2.24) is hard to evaluate so we approximate it by uniformly sampling from \(Q^t\). (See Figure 2.8) If we draw \(D\) random samples \(q_1, q_2, ..., q_D\) then approximation is the following:

\[
x^t = \arg \min_{x \in U^t} \left| \sum_{d=1}^{D} q^d(x) \right|
\]

(2.28)

This sampling from the version space is related to QBC algorithm of Freund et al. [Freund et al., 1997] [Seung et al., 1992] with one principal difference, namely, our goal is to find the best example to approximate the volume(s) of the version space\(^2\) while [Freund et al., 1997] employs it to determine whether or not an instance is to be labeled.

\(^2\)Note that the random variable \(q^d(x)\) is Bernoulli for each \(x\) and one can obtain a precise approximate characterization for sufficiently large \(D\) by using a combination of Chernoff and Union bounds.
Since our version space is a bounded polyhedron we can uniformly sample by employing the Hit and Run algorithm [Lovász and Vempala, 2004]. The “Hit” step generates a random direction and draws a line in that direction through the interior point. The “Run” step generates a new interior point by uniformly sampling along the interval defined by the line in the “Hit” step and the boundary of the polyhedron. As two steps are repeated, the generated interior points converge to a uniform sample from the polyhedron. For a given labeled set, we can employ a phase I optimization approach [Boyd and Vandenberghe, 2004] to find a feasible point and a cutting plane or barrier method [Boyd and Vandenberghe, 2004] to find a point near the center of the polyhedron. Recently, Kannan et. al. [Kannan and Narayanan, 2009] have developed techniques which produces a uniformly random sample with complexity scaling as $O(|L_t|N^2)$.

### 2.4.3 ActBoost vs. Query-by-Boosting (QBB)

As we described earlier recent effort in active learning has focused on finding examples that are most ambiguous for the currently trained classifier. QBB is such an algorithm based on boosting. In contrast the ActBoost algorithm is based on finding an example that approximately bisects the version space. We use 1D example with three clusters.
Algorithm 1 Hit and Run [Lovász and Vempala, 2004]

INPUT: \( L^t \) \{labeled set of examples\}, \( T_s \) \{number of iterations\}, \( q^0 \) \{initial feasible point\}

\[ Q^t \leftarrow \{q : q \in Q, y_i x_i^T q \geq v_0 \forall i | x_i \in L^t\}, Q \leftarrow \{q : q \geq 0, 1^T q = 1\}, \]

\( d_0 \leftarrow \frac{1}{\sqrt{N}} 1 - q^0 \) \{initial direction\} \( w = \frac{1}{\sqrt{N}} 1 \)

for \( s = 1 \) to \( T_s \) do

\( z \leftarrow N(0, I), z' \leftarrow [I - w w^T] z, d \leftarrow \frac{z}{||z||_2} \)

\{Generate a normal random variable, project it onto a hyperplane parallel to the simplex, and normalize to form a random direction\}

\[ r_{1i}^+ \leftarrow \min \{ \min_{r_{2i} \geq 0} r_i^1 , \min_{r_{2i} > 0} r_i^2 \}, \alpha^- \leftarrow \max \{ \max_{r_{1i} < 0} r_i^1 , \max_{r_{1i} > 0} r_i^2 \} \]

\( q^+_s \leftarrow q_s + \alpha^+ d, q^-_s \leftarrow q_s + \alpha^- d \) \{find two endpoints\}

\( q_{s+1} \leftarrow q_s^+ \alpha_s + q_s^- (1 - \alpha_s) \) \{generate a new interior point\}

end for

OUTPUT: \( q_{sample} \leftarrow q_{r_s} \) \{uniform random sample from \( Q^t \)\}

Algorithm 2 Active Boosted Learning (ActBoost)

INPUT: \( H \) \{problem matrix\}, \( T \) \{number of iterations\}

\( L^0 \leftarrow \emptyset, U^0 \leftarrow \mathcal{X}, Q^0 \leftarrow \{q|1^T q = 1, q \geq 0\}, t \leftarrow 0 \)

while \( t \leq T \) do

\( q^0 \leftarrow init(Q^t) \) \{compute initial classifier in version space: \( q^0 \in Q^t \)\}

for \( d = 1 \) to \( D \) do

\( q^d \leftarrow sample(q^0, Q^t) \) \{draw a uniform random samples from version space\}

end for

\( x^t \leftarrow \arg \min_{x \in U^t} | \sum_{d=1}^D q^d(x) | \) \{find the closest bisecting example\}

\( y^t \leftarrow label(x^t), Q^{t+1} \leftarrow Q^t \cap \{q|y^t h(x^t)^T q \geq 0\} \) \{label and update the version space\}

\( L^{t+1} \leftarrow L^t \cup \{x^t, y^t\}, U^{t+1} \leftarrow U^t \setminus x^t, t \leftarrow t + 1 \)

end while

OUTPUT: \( \{x_i, y_i\} \in L^T \) \{labeled set of examples\}
Figure 2.9: Initialization bias is tested on a 1D linear set (in 2.9(a)) consisting of three clusters. ActBoost and QBB are initialized with examples drawn only from the first two clusters. Both ActBoost and QBB are trained using Adaboost on their respective labeled examples. ActBoost is robust to initialization bias while QBB does not find the third cluster until all the first two clusters are labeled.

In Figure 2.9(b) to point out an important attribute of ActBoost, namely, that it does not suffer from initialization bias.

QBB in [Abe and Mamitsuka, 1998] labels an example with the smallest margin with respect to the current classifier:

$$x_{QBB} = \arg \min_{x_i \in U_t} \left| \sum_{j=1}^{N} h_j(x_i) q^t_j \right|$$  \hspace{1cm} (2.29)

(where $q^t$ are the weights of the adaboost classifier trained on the labeled data).

Initial training set $L^0$ is forced to sample from only the first two clusters. QBB
is a margin based algorithm and only queries examples in the vicinity of the current boundary estimate. Since the initialization is skewed to omit the third cluster, the second boundary is not detected until all the samples in the first two clusters are labeled (Figure 2.9(b)). ActBoost is not affected and starts to query from the third cluster much sooner. Margin based classifiers fail to explore the space far from the boundary. They heavily rely on the current classifier estimate and become sensitive to initialization.
2.5 Theory

Our main result is based on establishing that ActBoost achieves exponential reduction of version space volume. Nevertheless, Theorem 2.1 shows that if the set of weak learners is negation complete, there is no labeling strategy that can guarantee generalization performance. Consequently, structural assumptions on the target boosted classifier must be imposed. Under sparsity assumptions, which is practically well-motivated, we show that version space reduction results in generalization.

At stage \( t \) suppose \( L^t \) is the set of labeled examples and the version space becomes:

\[
Q^t = \{q \in Q \mid h(x)^T q \geq 0, x \in L^t\}
\]  

We suppose that our weak hypothesis set \( \mathcal{H} = \{h_j\}_{j=1}^N \) is negation complete, i.e., corresponding to each weak learner the hypothesis set admits its complement.

**Theorem 2.1.** Suppose the weak hypothesis set is negation complete and the weak learning hypothesis is satisfied. Then for any unlabeled example \( x_k \notin L^t \) there exists \( q_+, q_- \in Q^t \) such that both \( h(x_k)^T q_+ > 0 \) and \( h(x_k)^T q_- < 0 \) are satisfied. Consequently, regardless of the labeling strategy and at any stage \( t \) it follows that, \( \max_{q, q' \in Q^t} |q(x) - q'(x)| = 1 \).

Thus the version space always contains two classifiers that disagree on any unlabeled example. This implies that one cannot obtain useful information on the unlabeled samples based on the labeled samples. This situation calls for imposing additional constraints on the hypothesis set. Generalization ability of boosting has been shown to depend on two aspects: the margin [Schapire et al., 1997a] and the sparsity in the hypotheses weight vector \( q \) [Taylor et al., 2010]. A boosted subset of weak hypotheses achieves better generalization error than an ensemble of a complete set. More so, [Koltchinskii and Panchenko, 2005] showed that generalization can be improved if the weights for a subset of weak hypothesis decay to zero at an exponential rate. And experimentally, majority of boosting algorithms when trained
on common datasets also favor sparse ensembles. Consequently, sparsity and margin assumptions on the boosted classifier trained on the entire data set appears to arise naturally and we will impose such conditions to relate volume reduction rate to error rates in Section 2.5.2. In the following section we quantify volume reduction rates when the initial version space is the entire simplex. We then extend these results to the sparse setting and relate version space volume to error rates.

2.5.1 Volume Convergence Rate

In this section we prove that the ActBoost algorithm has an exponential rate of convergence, namely, that to reduce the version space to an $\epsilon$ fraction of its original volume requires labeling $O(\log \frac{1}{\epsilon})$ unlabeled points. To establish this fact we introduce the concept of coherence and neighborliness for continuous spaces of weight vectors based on concepts developed by Nowak, 2009 for the generalized binary search problem.

For convenience we introduce the matrix, $H$ to denote,

$$H = [h_{ij}], \quad h_{ij} = h_j(x_i); \quad i = 1, 2, \ldots, B, \quad j = 1, 2, \ldots, N \quad (2.31)$$

In other words the jth column of the matrix $H$ denotes the sign-pattern for the jth weak learner on all the data points, while the ith row denotes the sign pattern of all the weak learners on data point $x_i$. We say that $x_i$ and $x_j$ are $K$-neighborly if the difference in the sign patterns across all classifiers is smaller than $K$:

$$d_h(x_i, x_j) = \sum_{k=1}^{N} 1[h_k(x_i) \neq h_k(x_j)] \leq K \quad (2.32)$$

The significance of $K$-neighborliness of two data points is described in the following important lemma. The lemma provides a connection between the discrete world of Hamming distances to volumes on continuous spaces.
Lemma 2.2. If \( x \) and \( x' \) are K-neighbors then, for any \( Q' \subset Q \), where \( Q \) is as in Equation 2.19,

\[
\frac{1}{2} \int_{Q'} 1_{[q(x) \neq q(x')] d\mathbf{q}} \leq \frac{2K + 1}{2N} \text{Vol}(Q)
\]  

(2.33)

where \( d\mathbf{q} \) is the Lebesgue measure on \( Q \).

**K-Connected Graph:** We introduce a graph based on the neighborliness property introduced above for later use. We form a graph \( G \) with nodes, \( x \in \mathcal{X} \). Two nodes \( x_i, x_j \), have an edge if they are K-neighborly. A graph is said to be K-connected if the K-NNG graph so formed is connected. Note that K-connectedness is a property of the matrix \( H \).

**Example:** We note that 1D stumps are 1-connected. Figure 2.10 depicts stump classifiers which are our weak classifiers. A 1D stump is defined as follows: \( h_j(x_i) = \text{sgn}(x_i - t_j) \), where \( t_j \) is a threshold.

![Diagram of stumps](image)

**Figure 2.10:** Stump Classifiers

Finally, we need the notion of coherence. Let \( Q' \subset Q \), and denote by

\[
\rho^*(\mathcal{X}, Q') = \min_{\sum_i p_i = 1, p_i \geq 0} \sup_{q \in Q'} \left| \sum_{x_i \in \mathcal{X}} q(x_i) p_i \right|
\]

(2.34)

\(^3\)K-NNG: Nearest Neighbor Graph where only K-neighborly vertices are connected
where, $p_i$ denotes the weight of the $i$th data point. Note that by construction $0 \leq \rho^*(\mathcal{X}, Q') \leq 1$ and if $Q_1 \subset Q_2$ then $\rho^*(\mathcal{X}, Q_1) \leq \rho^*(\mathcal{X}, Q_2)$.

The significance of the above definition becomes clear in the following lemma.

**Lemma 2.3.** Suppose the graph induced by the matrix $H$ is $K$-connected and $\rho^*(\mathcal{X}, Q') < 1$ for some $Q' \subset Q$. Then for any $\rho$ s.t. $\rho^* \leq \rho < 1$, one the following statements must hold:

\[(1) \quad \left| \int_{Q'} q(x) d\mathbf{q} \right| \leq \rho \text{Vol}(Q') \quad (2.35)\]
\[(2) \quad \text{Vol}(Q') < \frac{(2K + 1)}{N \rho} \text{Vol}(Q) \quad (2.36)\]

To build intuition into why the algorithm described in Section 2.4 results in reducing the size of the feasible set note that if at iteration $t$ we have

$$\left| \int_{Q^t} q(x) d\mathbf{q} \right| = |\text{Vol}(Q^t_+) - \text{Vol}(Q^t_-)| \leq \rho_t \text{Vol}(Q^t)$$

for some $\rho_t < 1$, where $Q^t_+$ is the region where $q(x) = 1$ and $Q^t_-$ for $q(x) = -1$. Without loss of generality suppose $+1$ is the true label for $x$. So $Q^t_-$ will be eliminated if $x$ is labeled. Then, $Q^{t+1}_+ = Q^t_+$. Substituting $\text{Vol}(Q^t_-) = \text{Vol}(Q^t) - \text{Vol}(Q^t_+)$ we obtain:

$$|\text{Vol}(Q^{t+1}) - (\text{Vol}(Q^t) - \text{Vol}(Q^t_+))| \leq \rho_t \text{Vol}(Q^t) \quad (2.37)$$

$$\Rightarrow \text{Vol}(Q^{t+1}) \leq \frac{(1 + \rho_t)}{2} \text{Vol}(Q^t) \quad (2.38)$$

However, it turns out that the Lemma 2.3 is still insufficient and we need a regularity (smoothness) condition.
Regularity Condition: Suppose \( \tilde{Q} \subset Q \) such that \( \text{Vol}(\tilde{Q}) \geq (1 - \eta) \text{Vol}(Q) \) for some fixed constant \( \eta > 0 \), then for any two data points, \( x \) and \( x' \) there is an \( \alpha > 0 \) such that,

\[
\int_{\tilde{Q}} 1_{[q(x) \neq q(x')] d\mathbf{q}} \geq \alpha \int_{Q} 1_{[q(x) \neq q(x')] d\mathbf{q}}
\]

(2.39)

Basically the regularity condition states that the disagreement volume,

\[
V_d(Q) = \int_{Q} 1_{[q(x) \neq q(x')] d\mathbf{q}}
\]

on the original simplex \( Q \) (see Eq. 2.19) cannot change arbitrarily if a small subset of \( Q \) is removed. In other words, if \( \Delta Q = Q \setminus \tilde{Q} \) has small volume then the disagreement volume, \( V_d(\Delta Q) \) has to be small as well. We are now ready to state our main theorem.

**Theorem 2.4.** Consider the algorithm of Section 2.4 where an unlabeled data point \( x \in U^t \) is picked at stage \( t \) as the minimizer to Equation 2.24. Further assume that the graph induced by the matrix \( \mathbf{H} \) is \( K \)-connected and the assumption given by Equation 2.39 is satisfied. Then to reduce the volume of the version space to a fraction \( \epsilon \) of its original volume requires

\[
n = \frac{\log \epsilon}{\log \lambda}
\]

iterations where

\[
\lambda = \max\left\{ \frac{1 + \rho^*}{2}, \frac{1}{2} (1 + (1 - \alpha) \frac{2K + 1}{N}) \right\}
\]

Dealing with Constant Offsets: Constant offsets lead to \( \rho^* = 1 \), which implies no reduction in version space. To see this consider the 1D case in Figure 2.10 with stump classifiers. Outer stumps \( h_1, h_4 \) (bold red) are offsets and are problematic because they appear as columns of either all 1’s or all -1’s in the problem matrix \( \mathbf{H} \). Note that for this situation \( \rho^* = 1 \) since all the weight can be assigned to \( q_1 \) or \( q_4 \). However, if the stumps \( h_1 \) and \( h_4 \) are removed it can be easily seen that \( \rho^* = 0 \). This example generalizes to stumps in multiple dimensions. This scenario makes
sense: if the target classifier is the outer stump then in our search process no query helps in significantly reducing the version space, and every example has to be labeled. However, completely removing the outer stumps may degrade the classification ability of the hypothesis set. Instead, we assume that the good boosted classifier does not concentrate all of its weight on the problematic stumps. Let $I_p$ be the problematic set, and the augmented version space

$$Q' = Q \cap \{ q \mid \sum_{j \in I_p} q_j \leq \eta \}$$

for some fraction $\eta$. Maximization of $q$ in (2.34) will be over the augmented version space $Q'$, and $q$ will no longer be able to put all the weight on the outer stumps resulting in $\rho^*$ strictly less than one. This constraint is equivalent to an assumption that the target classifier will not be trivial and will not exclusively consist of outer stumps.

### 2.5.2 Error Convergence

In this section we will discuss how reduction in version space $Q$ is related to generalization error. Based on our earlier arguments we impose sparsity and margin constraints on the target boosted classifier, namely,

$$\exists q \in S = \{ q \in Q \mid \|q\|_0 \leq p \} \text{ such that } y_i q^T h(x_i) \geq \theta \quad (2.40)$$

for all $x_i \in \mathcal{X}$ and for some $\theta > 0$ and where $\|\cdot\|_0$ is the so called $\ell_0$ norm and characterizes the number of non-zero elements in the vector $q$. Under the margin constraints [Schapire et al., 1997a] has shown that the generalization error scales as $O(\frac{\log |\mathcal{X}| \log p}{\theta^2 |\mathcal{X}|})^{1/2}$. Consequently, our problem reduces down to labeling sufficiently many points such that every element in our version space satisfies the margin constraint.
Our goal is to reduce the sparse version space such that

\[ S \supset S^1 \supset \ldots \supset S^t \]

Note that the set \( S \) is made of \( \binom{N}{p} \) \( p \)-sparse disjoint simplices. For notational convenience we denote

\[ \{s_1, s_2, \ldots, s_{\binom{N}{p}}\} = S \subset Q \]

Our modified query strategy is to find an example to bisect the sparse version space:

\[ x^* = \arg \min_{x \in U^t} \left| \sum_{r=1}^{\binom{N}{p}} \int q \in s_r \right| \left| \int q(x) \, dP(q) \right| \quad (2.41) \]

Note that our modified algorithm accounts for sparsity of the target boosted classifier but does not assume knowledge of \( \theta \). Analogous to the setup for volume reduction on the entire simplex in Section 2.5.1 we define \( \rho^* \) as

\[ \rho^*(\mathcal{X}, S') = \min_{\sum_i p_i = 1, p_i \geq 0} \sup_{q \in S'} \left| \sum_{x_i \in \mathcal{X}} q(x_i) p_i \right|, \text{ for } S' \subset S. \quad (2.42) \]

and \( \lambda = \max\{\frac{1+\rho^*}{2}, \frac{1}{2} (1 + (1 - \alpha) \frac{2K+1}{N})\} \). Define,

\[ f(\theta, p) = \inf_{q^* \in S} Vol(\{q \in S \mid \|q - q^*\|_1 \leq \theta/2\}) \quad (2.43) \]

where volume is taken with respect to the Lebesgue measure on the \( p \) sparse subspace.

**Theorem 2.5.** *Consider the strategy where an unlabeled data point \( x \in U^t \) is picked at stage \( t \) as the minimizer to Equation (2.41) and suppose the regularity conditions of Theorem 2.4 are satisfied for the sparse set \( S \). Let the number of stages \( n \) and hence the number of labeled samples satisfy

\[ n \geq \frac{\log \binom{N}{p} + \log \frac{1}{f(\theta, p)}}{\log \lambda}. \quad (2.44) \]

\[ \text{Here, we assume that the true sparsity } p \text{ is known.} \]
Then for all $q \in S^n$, it follows that

$$\text{Prob}(q(x) \neq y) \leq O\left(\frac{\log |\mathcal{X}| \log p}{\theta^2 |\mathcal{X}|} + \frac{\log(1/\delta)}{|\mathcal{X}|}\right)^{1/2}$$

(2.45)

with probability $1 - \delta$ for a $\delta > 0$ and where $|\mathcal{X}|$ is the size of the unlabeled data pool.

Figure 2.11 illustrates implication of our modified algorithm on Figure 2.9(a) for increasing knowledge of sparsity.

**Convex Surrogate:** To reduce the sparse subset instead of the full version space is combinatorially hard because the integral in Equation (2.41) has to be enumerated for every $p$-sparse segment and $p$ is also unknown. If we convexify this problem, then ActBoost amounts to reducing the convex hull of the sparse subspace at every iteration. To see this let $S^t = \{q \in Q^t \mid \|q_0\| = p\}$. Instead of selecting an example to bisect a non-convex subspace $S^t$, suppose we reduce the convex hull of $S^t$: $\mathcal{C}(S^t)$. Note that $Q = \mathcal{C}(S)$ and for a set of labeled examples: $Q^t \supset \mathcal{C}(S^t)$. At $t + 1$, if we
select an example to reduce the version space $Q^t$, it will also reduce $C(S^{t+1})$:

$$\min_{x \in U^t} Vol(C(S^{t+1})) \leq \min_{x \in U^t} Vol(Q^{t+1})$$ (2.46)

While we can reduce the convex hull of $S^t$, we cannot guarantee that the subspace $S^t$ is also reduced by the same amount. However, our simulations demonstrate that reducing the full version space results in generalization.

### 2.5.3 LP for Bounding Coherence

In Theorem 1 we saw that coherence (see Eq. 2.34) controls the convergence. In this section we present a linear programming solution for computing the coherence. First, we have the following result.

**Lemma 2.6.** Let $\rho^*(Q', X)$ be as in (2.34) and suppose the set of weak classifiers is balanced, namely, for each $h_j(\cdot) \in \mathcal{H}$ there is a corresponding element $h_k(\cdot) = -h_j(\cdot) \in \mathcal{H}$. Then,

1. $\rho^*(Q', X) < 1 \implies \exists z \geq 0, z \neq 0, z^T H = 0$ (2.47)
2. $\exists q \in Q', q \geq 0, Hq \geq 0 \implies \rho^*(Q', X) = 1$ (2.48)

Note that an example of balanced set of classifiers are stumps described in the previous section. While the above lemma characterizes when $\rho^* < 1$, it does not directly help in finding a bound. To this end we consider the following LP and its corresponding dual.

$$\max_{v,q} v, \text{ s.t. } Hq \geq v, 1^T q = 1, q \geq 0 \iff$$

$$\min_{\pi,z} \pi, \text{ s.t. } z^T H \leq \pi, 1^T z = 1, z \geq 0$$

If the value of the primal or dual is less than zero then $\rho^*(X, Q')$ is less than one. The dual variable $z$ is the probability distribution on the examples. Notice that the
constraint $1^T z = 1$ imposes sparsity. Only the examples with a corresponding non-zero $z_i$ are active constraints in the primal. The examples with zero $z_i$ can be removed and the solution of the problem will not change. In the expression \[2.34\], we can put equal weights $p_i$ only on the examples with non-zero $z_i$. Suppose the cardinality of the support of $z$ is $l$, then a bound on $\rho^*$ follows:

$$\rho^* \leq \max_{q \in Q} \left| \sum_{i: \lambda_i > 0} q(x_i) \frac{1}{l} \right| \Rightarrow \rho^* \leq \left| \frac{l-1}{l} - \frac{1}{l} \right| = 1 - \frac{2}{l} \quad (2.49)$$

where we have used the fact that $\rho^* < 1$. This implies that $q(x)$ cannot have the same sign for all $x \in \mathcal{X}$. Note that as the sparsity of $z$ increases, $\rho^*$ decreases.

### 2.5.4 Undersampling vs. Oversampling

The convergence rate in Theorem 1 depends on how large a $K$ is necessary to ensure $K$-connectedness. To build intuition, consider a 1D stump example with three stumps $h_1, h_2, h_3$ and four training samples $x_1, x_2, x_3, x_4$. Each row $h(x_i)$ in the matrix in Figure 2.12(a) captures how each example is classified. Note that $K = 1$ is sufficient to ensure connectedness.

Suppose we oversample our weak hypotheses by adding three more stumps in bold red (Figure 2.12(b)). Now performing the same graph construction and reduction, then we need $K = 3$ to ensure connectedness. Since the value $K$ increases this directly leads to a reduction of aggregate convergence rate $\lambda$. This implies that if we demand too much resolution in our target classifier and do not have an appropriately dense training set then the number of iterations increases exponentially with the disagreement factor $K$. Notice that the opposite case of oversampled training set does not hurt as it increases the set of possible queries. Note that increase in $K$ here is a direct outcome of redundant weak learners. Consequently, these weak learners can be removed in simple cases but for more complicated cases this reduction may be
Figure 2.12: The parameter $K$ required for maintaining graph connectedness in the case of undersampled and oversampled weak learners. For each case shown: 1D dataset with stumps, the problem matrix $H$ and the neighborly graph reduction. Oversampling weak learners results in increase in $K$. 
non-trivial.

2.6 Experiments

We compare three algorithms random query strategy (RANDOM), QBB from [Abe and Mamitsuka, 1998] and ActBoost. RANDOM uniformly samples a random $x^*_{random}$ from the unlabeled pool $U^t$ at time $t$. QBB relies on the Adaboost solution $q^t_0 = \text{adaboost}(L^t)$, computed on the current labeled set $L^t$: $x^*_{QBB} = \arg\min_{x \in U^t} |h(x)^T q^t_0|$. All three algorithms are then trained using Adaboost on their respective labeled sets.

Note ActBoost operates on the full version space in the simulation without knowledge of the sparsity.

The ActBoost algorithm has several parameters: number of samples $D$ drawn by the Hit and Run algorithm, the number of iterations to generate one sample for the Hit and Run algorithm, initial size of the labeled set $L^0$ (necessary for comparison with QBB). Figure 2.14(a) demonstrates performance vs $D$. As expected, more samples result in better performance as the query comes closer to bisecting the version space. Changing the number of iterations for the Hit and Run algorithm does not have much affect on performance confirming that the sampling has a quick mixing time.

For each simulation an unlabeled training pool $\mathcal{X}$ of 200 is sampled uniformly from a dataset. Each simulation is averaged over 100 trials. For all simulations, the number of sampled classifiers is fixed at $D = 8$\textsuperscript{5}. We chose to use Adaboost to train a classifier on the labeled set at every iteration. We then compared the performance of QBB, ActBoost and Random. Observe that the goal of active learning is to achieve performance of an entire training set while labeling and learning on only a fraction of examples. Following this philosophy, at each iteration, we compute the error against the entire training pool $\mathcal{X}$. Similar performance evaluation on 'query

\textsuperscript{5}$D = 16$ shows small improvement, but $D = 8$ is used to speed up computation
data’ for QBB is employed in [Abe and Mamitsuka, 1998]. Since the labeled set of points is relatively small compared with the amount of unlabeled data, this also serves as a characterization of the generalization performance.

We use stumps for weak hypotheses as defined previously. For a given dataset of size $B$ in $\mathbb{R}^{D_x}$, there are $2(B + 1)D_x$ possible weak hypotheses. However, if the data examples are categorical or integers then the number of stumps can be significantly reduced by eliminating redundancies. ActBoost is tested on several datasets from the UC Irvine Machine Learning Repository (except for synthetic).

### 2.6.1 Unbiased Initialization:

The initial set $L^0$ is resampled at every trial to avoid any initialization bias (which is addressed next). BOX and BANANA are two dimensional datasets (see suppl.), MUSHROOM is a 22 dimensional dataset. On the 2D datasets, we illustrate the advantage of learning in sparse version space (Fig 2.15(b) 2.15(a)). Instead of the entire space, ActBoost(sp) operates only on the 10% of the weak hypotheses. This subset also contains the true sparse support which is determined by training with an entire pool labeled. As our theoretical results suggest, ActBoost(sp) achieves better performance than ActBoost. The experiments in Fig 2.15 only show marginal advantage of ActBoost since the initialization bias is removed by averaging. The simulations also support the negative result of Theorem 1: a reduction of version space of the full ensemble does not guarantee a decrease in generalization error.

### 2.6.2 Biased Initialization:

We further illustrate the robustness of ActBoost to initialization bias (see Section 2.4.3). We transform a multi-class dataset into a binary dataset consisting of three clusters. IRIS is a four dimensional dataset of three classes. Classes 1 and 3 are combined into one class to form a binary dataset consisting of three clusters. DER-
Figure 2.13: Two dimensional datasets: Gaussian Clusters 2.13(a), Box Dataset 2.13(b), Banana Dataset 2.13(c)
Figure 2-14: Parameter Selection: Accuracy vs # labeled examples as a function of ActBoost classifier samples $D$, more samples provide a better approximation of the GBS metric integral, resulting in better performance. $^{2.14(a)}$, and as a function of Hit and Run iterations (HT): changing HT does not change performance. $^{2.14(b)}$
Figure 2.15: Accuracy vs. # labeled examples. 2D datasets: BANANA (2.15(a)) and BOX (2.15(b)). ActBoost(sp) learns in the sparse version space and performs better than ActBoost which operates on the full version space. Multivariate Dataset: MUSHROOM (2.15(c)). ActBoost does not show significant performance improvement over QBB.
MATOLOGY has 34 dimensions and 5 classes. Similarly, we designate the first three classes as clusters and combine the first and third to form two classes. SOY has 35 dimensions and 19 classes. We use classes 4, 8, 14 as clusters to form a binary dataset.

Gaussian Clusters is a synthetic two dimensional dataset consisting of three clusters. (see suppl.) For each experiment (Fig 2.16), we force the initial set $L^0$ to be sampled from only the first two clusters. This approach simulates the worst case initialization bias. For all datasets, ActBoost remains robust while QBB does not locate the third cluster until the first two are exhausted.

![Graphs showing performance of ActBoost and QBB on different datasets](image)

**Figure 2.16:** Robustness to Initialization Bias: binary datasets consisting of three clusters are formed from multi-class datasets (except for gauss clusters). Both algorithms are initialized only from the first two clusters; ActBoost is not affected.
Chapter 3

Sequential Sensor Selection

In this chapter we develop a framework for a sequential decision making under budget constraints for classification. Different sensors have varying costs for acquisition, and these costs account for delay, throughput or monetary value. Consequently, we seek methods for maximizing performance of the system subject to budget constraints.

We consider a scenario where the order in which sensors are utilized is given. So for each example, a sensor is first chosen for acquiring measurements and then based on the available information one decides (to reject) to seek more measurements from the next sensor or to terminate by classifying the example based on the available information. We formulate a multi-stage empirical risk objective and learn sequential decision functions from training data.

We study this problem in the setting of binary classification. We show that the optimal reject classifier at each stage is a combination of two binary classifiers, one biased towards positive examples and the other biased towards negative examples. We use this parameterization to construct stage-by-stage global surrogate risk, develop an iterative algorithm in the boosting framework and present convergence and generalization results.

Next, we extend our work to a multi-class setting. We show that reject decision at each stage can be posed as a supervised binary classification. We develop an iterative framework and for this parameterization derive bounds for the VC dimension of the multi-stage system to quantify the generalization error. We test our work on synthetic,
medical and explosives detection datasets. Our results demonstrate that substantial cost reduction without a significant sacrifice in accuracy is achievable.

The work presented here is partially published in [Trapeznikov and Saligrama, 2013][1][Trapeznikov et al., 2013][1][Trapeznikov et al., 2012][1].

**Organization:** In sections 3.1, we describe the setting of ordered sequence of stages. Section 3.2 summarized our contribution on sequential reject classifiers. Section 3.3 describes related work. In section 3.4, we present our work on sequential reject classifiers in a binary classification setting. Section 3.5 extends this to a multi-class setting.

### 3.1 Ordered Sequence of Sensors

In many applications, decision systems are composed of an ordered sequence of stages. Each stage is associated with a sensor or a physical sensing modality. Typically, a less informative sensor is cheap (or fast) while a more informative sensor is either expensive or requires more time to acquire a measurement. In practice, a measurement budget (or throughput constraint) does not allow all the modalities to be used simultaneously in making decisions. The goal in these scenarios is to attempt to classify examples with low cost sensors and limit the number of examples for which more expensive or time consuming informative sensor is required.

Modern passenger screening systems for explosives detection employ a suite of sensors such as X-ray backscatter scanners (cheap & fast), millimeter wave imagers (expensive & low-throughput), magnetometers, video, IR imagers in different bands, and/or physical (human) search. Such systems must maintain a throughput constraint in order to keep pace with arriving traffic. In clinical diagnosis, doctors use a suite of sensors for detecting and assessing the severity of (breast cancer) mammographic mass lesions (malicious or benign) including genetic markers, CT images
Figure 3-1: Multi-Stage System consists of $K$ stages. Each stage is a classifier with a reject option. The system incurs a penalty of $c_{k+1}$ at $k$th stage if it rejects to seek more measurements. The $k$th classifier only sees the first $k$ sensing modalities in making a decision.

from different views, 3-D CT tomographic reconstructions, optical tomography imaging, ultrasound imaging, elastography imaging, manual palpation, and biopsy, among others. Many of these sensors provide imagery input for individual human radiologist scoring. The different sensing modalities have diverse costs, in terms of health risks (radiation exposure) and monetary expense. Consequently, we associate each stage with a new sensing modality with higher costs associated with later stages.

An important aspect of homeland security and medical diagnosis applications is that each sensors/modality produce high dimensional measurements (such as images (X-Rays etc)). So, not only are the underlying distributions for the sensor measurements under different classes not known, but impossible to estimate from training data due to the inherent “curse of dimensionality.”

Many such examples share a common structure (see Figure 3-1), and we list some of its salient aspects below:

(A) Sensors & Ordered Stages: Each stage is associated with a new sensor measurement or a sensing modality. Multiple stages are an ordered sequence of sensors
or sensor modalities with later stages corresponding to expensive or time-consuming measurements. In many situations, there is often some flexibility in choosing a sensing modality from a collection of possible modalities. In these cases, the optimal choice of sensing actions also becomes an issue. While our methodology can be modified to account for this more general setting, we primarily consider a fixed order of stages and sensing modalities in this chapter. Many of the situations we have come across consist of a handful of sensors or sensing modalities. Consequently, for these situations, the problem of choosing a fixed sensor ordering is not justified since one could by brute force enumerate and optimize over the different possibilities. We consider extensions to adaptively select the order of modalities for every data example in the next chapter.

(B) Reject Classifiers: Our sequential decision rules either attempt to fully classify an instance at each stage or “reject” the instance on to the next stage for more measurements in case of ambiguity. For example, in explosives detection, a decision rule in the first stage, based on IR scan, would attempt to detect whether or not a person is a threat and identify the explosive type/location in case of a threat. If the person is identified as a threat at the first stage it is unnecessary (and indeed dangerous – the explosive could be detonated) to seek more information. Similarly in medical diagnosis if a disease is diagnosed at an early stage, it makes sense to begin early treatment rather than waiting for more conclusive tests.

(C) Information vs. Computation: Note that our setup can only use the partial measurements acquired up to a stage in making a decision. In other methods, such as detection cascades (Viola and Jones, 2001), the full measurement and therefore all the information is available to every stage. Therefore, any region in the feature space can be carved out with more complex regions in the measurement space, or equivalently complex features can be extracted but with higher costs. In contrast, we
have only partial measurements (or information) and so any feature or classifier that we employ has to be agnostic to unavailable measurements at that stage.

**Figure 3.2:** Advantage of a 2 stage classifier: 10 samples, binary (squares, circles). The red line is the optimal decision when using only 1st stage modality. The blue line is optimal if using both. The curve is classification error vs. samples rejected (cost) The red point corresponds to classifying everything at stage 1. The blue corresponds to rejecting everything and classifying using both modalities. The green is a partial reject strategy. The samples outside the green region are classified using only the first modality, and samples inside the region are rejected to stage 2 and are classified using both modalities. Note that blue and green have the same error, while the reject strategy (green) has to use 2nd stage sensor only for $\frac{1}{2}$ of examples, reducing the cost by a factor of 2.

The two stage example in Figure 3.2 illustrates some of the advantages of our scheme over the alternative scheme that first acquires measurements from all the sensing modalities, which we refer to as the centralized classifier. A reject classifier utilizes the 2nd stage sensor only for a fraction of the data but achieves the same performance as the centralized classifier.
3.2 Contribution: Sequential Reject Classifiers

Our work is based on the so called *Prediction Time Cost Reduction approach* ([Kanani and Melville, 2008]). Specifically, we assume a set of training examples in which measurements from all the sensors or sensing modalities as well as the ground truth labels are available. Our goal is to derive *sequential reject classifiers* that reduce cost of measurement acquisition and error in the *prediction (or testing) phase*.

We show that this sequential reject classifier problem can be formulated as an instance of a *partially observable Markov Decision Process (POMDP)* ([Bertsekas, 2005]) when the class-specific probability models for the different sensor measurements are known. In this case the optimal sequential classifier can be cast as a solution to a Stochastic Dynamic Program (DP). The SDP solution is a sequence of *stage-wise optimization* problems, where each stage problem is a combination of the cost from the current stage and the cost-to-go function that is carried on from later stages.

Nevertheless, class probability models are typically unknown; our scenarios produce high-dimensional sensor data (such as images). Consequently, unlike some of the conventional learning approaches ([Ji and Carin, 2007]), where probability models are first estimated to solve POMDPs, we have to adopt a non-parametric *discriminative learning* approach. We utilize the structure of the POMDP solution to empirically approximate the value of the cost-to-go function only at a discrete subset of the data-space. Next, instead of interpolating or parameterizing the cost-to-go function and learning it from data, we formulate an empirical discriminative objective that utilizes point-wise cost-to-go estimates evaluated on the training set and directly learn sequential decision rules that minimize this objective. Using this decomposition, we formulate a novel *multi-stage expected risk minimization (ERM) problem*. 
3.3 Related Work

The subject of this paper is not new and has been studied in the Machine Learning community as early as MacKay, 1992. Our work is closely related to the so called prediction time active feature acquisition approach in the area of cost-sensitive learning. The goal there is to make sequential decisions of whether or not to acquire a new feature to improve prediction accuracy. We summarize the important categories:

3.3.1 Single Stage Reject Classifiers

Our paper is also closely related to the topic of reject classifiers, which has also been investigated. However, in the literature reject classifiers have been primarily considered in a single stage scenario. In the Bayesian framework, Chow and Chow, 1970 introduced Chow’s rule for classification. It states that given an observation \( x \) and a reject cost \( \delta \) and \( J \) classes, reject \( x \) if the maximum of the posteriors for each class is less than the reject cost: \( \max_{k=1,\ldots,J} P(y=j|x) < \delta \). In the context of machine learning, the posterior distributions are not known, and a decision rule is estimated directly. One popular approach is to reject examples with a small margin. Specifically, in the context of support vector machine classifiers, Yuan and Casasent, 2003 Bartlett et al., 2008 Rodríguez-Díaz and Castañón, 2009 Grandvalet et al., 2008, define a reject region to lie within a small distance (margin) to the separating hyperplane and embed this in the hinge loss of the SVM formulation. El-Yaniv and Wiener, 2011 proposes a reject criteria motivated by active learning but its implementation turns out to be computationally impractical. In contrast, we consider multiple stages of reject classifiers. We assume an error prone second stage which occurs in such fields as threat detection and medical imaging.
3.3.2 Generative & Parametric Modeling

In a Bayesian setting, probability models are either known or the data is sufficiently low-dimensional that these models can be reliably estimated. Under these assumptions, [Ji and Carin, 2007] and [Kapoor and Horvitz, 2009] model the decision process and infer feature dependencies while taking acquisition costs into account. [Sheng and Ling, 2006], [Bilgic and Getoor, 2007], and [Zubek and Dietterich, 2002] study strategies for optimizing decision trees while minimizing acquisition costs. The construction is usually based on some purity metric such as entropy. [Kanani and Melville, 2008] propose a method that acquires an attribute if it increases an expected utility. However, all these methods require estimating a probability likelihood that a certain feature value occurs given the features collected so far. While surrogates based on classifiers or regressors can be employed to estimate likelihoods, this approach requires discrete, binary or quantized attributes. In contrast, our problem domain deals with high dimensional measurements (such as images consisting of thousands of pixels), so estimating probability densities reliably is not possible. Instead, we develop a discriminative learning approach and formulate a multi-stage empirical risk optimization problem to reduce measurement costs and misclassification errors.

3.3.3 Discriminative Learning Approaches

Our approach is the first framework to analyze the design of multi-class sequential decision systems in a non-Bayesian setting. Multiple stages of margin based reject classifiers have been considered in a time efficient feature extraction (TEFE) algorithm by [Liu et al., 2008] in the context image classification. This method employs a sequence of SVMs, each operating on features of increasing computational complexity. The main contribution of the work in [Liu et al., 2008] is in efficient training of each stage; the solution of previous stage is used to initialize SVM optimization problem
of the following stage. However, the method uses a myopic strategy that does not
take into account the performance of the entire system in learning the decisions. We
compare this myopic strategy in the Experiments section and demonstrate signifi-
cantly better performance. Besides the method mentioned above, we are not aware
of any other approaches that seek to reduce measurement budget in a multi-stage and
multi-class setting and are able to handle large dimensional training data.

Detection Cascades  Our multi-stage sequential reject classifiers bears close resem-
bblance to detection cascades. There is much literature on cascade design (see [Zhang
and Zhang, 2010] and references therein) but most cascades roughly
follow the set-up introduced by [Viola and Jones, 2001] to reduce computation cost
during classification. At each stage in a cascade, there is a binary classifier with a very
high probability of correct detection and a mediocre probability of false alarm. Each
stage makes a partial decision; it either detects an instance as negative or passes it
on to the next stage. Only the last stage in the cascade makes a full decision, namely,
whether the example belongs to a positive or negative class.

There are several fundamental differences between detection cascades and our
multi-stage reject classifiers. A key difference is the system architecture. Detection
cascades make partial binary decisions, delaying a positive decision until the final
stage. In contrast, multi-stage reject classifiers can make full classification decisions
at any stage. Conceptually, this distinction requires a fundamentally new approach;
detection cascades work because their focus is on unbalanced problems with few
positives and a large number of negatives; and so the goal at each stage is to admit
large false positives with negligible missed detections. Consequently, each stage can
be associated with a binary classification problem that is acutely sensitive to missed
detections. In contrast, our scheme at each stage is a composite scheme composed
of a classifier as well as a rejection decision. The rejection decision is itself a binary
classification problem. In practice, multi-stage reject classifiers arise in important areas such as medical diagnosis and explosives detection (item (B)). As a performance metric detection cascades tradeoff missed detections at the final stage with average computation. Multi-stage reject classifiers tradeoff average misclassification errors against number of examples that reached later stages (i.e. required more sensors or sensing modalities). For these reasons it is difficult to directly compare algorithms developed for multi-stage reject classifiers to those developed for detection cascades. Nevertheless, our goals and resulting algorithms are similar to some of the issues that arise in cascade design (see Chen et al., 2012 and references therein), namely, perform a joint optimization for all the stages in a cascade given a cost structure for different features.

### 3.3.4 Other Cost Sensitive Methods

Network intrusion detection systems (IDS) is an area where sequential decision systems have been explored. (see Fan et al., 2000, Lee et al., 2002, Cordella and Sansone, 2007). In IDS, features have different computation costs. For each cost level, a rule-set is learned. The goal is to use as many low cost rules as possible. In a related set-up, Fan et al., 2002, Wang et al., 2003 consider a more general ensemble of base classifiers and explore how to minimize the ensemble size without sacrificing performance. In the test phase, for a sample, another classifier is added to the ensemble if the confidence of the current classification low. Here, similar to detection cascades, the goal is to reduce computation time.
3.4 Sequential Reject Classifiers in a Binary Setting

Let us first examine a sequential decision in the binary setting. It allows for an intuitive parametrization of a reject option which we describe next. We will revisit the multi-class setting later.

We will first analyze the problem in the setting when all the probability models are known. When probability models are known, optimal strategies are given by the SDP solution, but it is unclear how to mimic these strategies in the empirical setting. However, if we restrict ourselves to a binary classification setting then we can transform reject decisions into binary classification problems. Specifically, we show that the optimal reject classifier at each stage is a combination of two binary classifiers, one biased towards positive examples and the other biased towards negative examples. The disagreement region of the two then defines the reject region.

We then approximate this empirical risk with a global surrogate. We present an iterative solution and demonstrate local convergence properties. The solution is obtained in a boosting framework. We then extend well-known margin-based generalization bounds ([Schapire et al., 1997b]) to this multi-stage setting. We tested our methods on synthetic, medical and explosives datasets. Our results demonstrate an advantage of multistage classifiers: cost reduction without a significant sacrifice in accuracy.

3.4.1 Problem Formulation

Let \((x, y) \in \mathcal{X} \times \{-1, +1\}\) be distributed according to an unknown distribution \(\mathcal{D}\). A data point has \(K\) features, \(x = \{x_1, x_2, \ldots, x_K\}\), and belongs to one of 2 classes indicated by its label \(y \in \{+1, -1\}\). A \(k\)th feature is extracted from a measurement acquired at \(k\)th stage. \(x_k\) is allowed to be a vector. We define a truncated feature vector at \(k\)th stage: \(x^k = \{x_1, x_2, \ldots x_k\}\). Let \(\mathcal{X}^k\) be the space of the first \(k\) features.
such that $x^k \in \mathcal{X}^k$.

The system has $K$ stages, the order of the stages is fixed, and $k$th stage acquires a $k$th measurement. At each stage, $k$, there is a decision with a reject option, $f^k$. It can either classify an example, $f^k(x^k) : \mathcal{X}^k \rightarrow \{-1, +1\}$, or delay the decision until the next stage, $f^k(x^k) = r$ and incur a penalty of $\alpha c_k$. Here, $r$ indicates the “reject” decision. $f^k$ has to make a decision using only the first $k$ sensing modalities. The last stage $K$ is terminal, a standard classifier.

Also note that the true cost $c_k$ is multiplied by a trade-off parameter $\alpha$. This is necessary because error and cost are in different units and are not directly comparable. By controlling $\alpha$, we can move the emphasis between error and cost.

Define the system risk to be,

$$R(f^1, \ldots, f^K, x, y) = \sum_{k=1}^{K} S^k(x^k)R_k(f^k, x^k, y)$$

(3.1)

Here, $R_k$ is the cost of classifying at $k$th stage, and $S^k(x^k) \in \{0, 1\}$ is the binary state variable indicating whether $x$ has been rejected up to $k$th stage.

$$R_k(x^k, y, f^k) = \begin{cases} 
\alpha c_{k+1}, & f^k(x^k) = r \\
w_p, & f^k(x^k) = 1, \land y = -1, \land f^k(x^k) \neq r \\
w_n, & f^k(x^k) = -1, \land y = 1, \land f^k(x^k) \neq r \\
0, & \text{otherwise}
\end{cases}$$

(3.2)

So if $x$ is active and is misclassified, the penalty is either $w_p$ or $w_n$. If it is rejected then the system incurs a penalty of $\alpha c_{k+1}$, and the state variable for that example...
remains at 1.

\[
S^{k+1}(x^{k+1}) = \begin{cases} 
S^k(x^k), & f^k(x^k) = r, \quad S^1 = 1 \\
0, & \text{else}
\end{cases} 
\] (3.3)

### 3.4.2 Bayesian Setting

In this section, we will digress from the discriminative setting and analyze the problem under the assumption that the underlying distribution \( D \) is known. In doing so, we hope to discover some fundamental structure that will simplify our empirical risk formulation in the next section.

If \( D \) is known the problem reduces to a POMDP, and the optimal strategy is to minimize the expected risk,

\[
\min_{f^1, \ldots, f^K} E_D \left[ R(f^1, \ldots, f^K, x^K, y) \right] 
\] (3.4)

If we allow arbitrary decision functions then we can equivalently minimize conditional risk,

\[
\min_{f^1, \ldots, f^K} E \left[ R(f^1, \ldots, f^K, x^K, y) \mid x \right] 
\] (3.5)

This problem—by appealing to dynamic programming—remarkably reduces to a single stage optimization problem for a modified risk function. To see this, we denote the cost-to-go,

\[
\delta^k(x^k) = \alpha c_{k+1} + \min_{f^k+1, \ldots, f^K} E \left[ \sum_{t=k+1}^K S^t(x^t)R_t(f^t, x^t, y) \mid x^k, S^k(x^k) = 1 \right] 
\] (3.6)

Note that we do not use the standard POMDP information sufficient statistic \( P(y \mid x^k) \). Instead, we use the information history, \( x^k \). This is another sufficient statistic, which does not require the knowledge of the distributions.
We also define the modified risk functional,

\[
\tilde{R}_k(x^k, y, f^k, \delta^k) = \begin{cases} \\
\delta^k(x^k), & f^k(x^k) = r \\
w_p, & f^k(x^k) = 1 \land y = -1 \land f^k(x^k) \neq r \\
w_n, & f^k(x^k) = -1 \land y = 1 \land f^k(x^k) \neq r \\
0, & \text{otherwise} \\
\end{cases}
\]

(3.7)

and prove the following theorem. To simplify our discussion, we consider equal error penalties: \(w_p = w_n = 1\). However, our approach can be easily extended to unbalanced error penalties as we will demonstrate in the experiments section.

**Theorem 3.1.** The optimal solution \(f^1, f^2, \ldots, f^K\) to the multi-stage risk in Eq. 3.5 decomposes to single stage optimization,

\[
f^k = \arg \min_f \mathbb{E} \left[ \tilde{R}_k(x^k, y, f, \delta^k) \mid x^k \right]
\]

(3.8)

and the solution is:

\[
f^k(x^k) = \begin{cases} \\
+1, & \Pr(y = 1 \mid x^k) > 1 - \delta^k(x^k) \\
-1, & \Pr(y = 1 \mid x^k) < \delta^k(x^k) \\
\text{reject}, & \delta^k(x^k) \leq \Pr(y = 1 \mid x^k) \leq 1 - \delta^k(x^k) \\
\end{cases}
\]

(3.9)

**Proof.** To simplify our derivations, we assume uniform class prior probability:

\[
\Pr[y = +1] = \Pr[y = -1] = \frac{1}{2}
\]

However, our results can be easily modified to account for a non-uniform prior. The expected conditional risk can be solved optimally by a dynamic program, where a DP recursion is,

\[
J_K(x^K, S^K) = \min_{f^K} \mathbb{E}_y \left[ S^K(x^K)R_k(y, x^K, f^K) \mid x^K \right]
\]

(3.10)

\[
J_k(x^k, S^k) = \min_{f^k} \left\{ \mathbb{E}_y \left[ S^k(x^k)R_k(y, x^k, f^k) \mid x^k \right] + \mathbb{E}_{x^{k+1}, \ldots, x^K} \left[ J_{k+1}(x^{k+1}, S^{k+1}) \mid x^k \right] \right\}
\]

(3.11)
Consider \( k \)th stage minimization, \( f^k \) can take 3 possible values \( \{-1, +1, r\} \) and \( J_k(x^k, S^k) \) can be recast as a conditional expected risk minimization,

\[
J_k(x^k, S^k = 1) = \min_{f^k} \left\{ P_y [y = 1 \mid x^k], 1 - P_y [y = 1 \mid x^k], \right. \\
\left. \alpha c_k + \mathbb{E}_{x^{k+1} \ldots x^K} \left[ J_{k+1}(x^{k+1}, 1) \mid x^k \right] \right\}
\]

(3.12)

Define,

\[
\delta(x^k) = \alpha c_{k+1} + \mathbb{E}_{x^{k+1} \ldots x^K} \left[ J_{k+1}(x^{k+1}, S^{k+1} = 1) \mid x^k \right]
\]

(3.14)

and rewrite the conditional risk in 3.13,

\[
f^k = \arg \min_{f^k} \left\{ P_y [y = 1 \mid x^k], 1 - P_y [y = 1 \mid x^k], \delta(x^k) \right\}
\]

(3.15)

Reject is the optimal decision if,

\[
P_y [y = 1 \mid x^k] \geq \delta(x^k) \text{ and } 1 - P_y [y = 1 \mid x^k] \geq \delta(x^k) \implies \delta(x^k) \leq P(y = 1 \mid x^k) \leq 1 - \delta(x^k)
\]

(3.16)

(3.17)

If reject is not the optimal strategy then a class is chosen to maximize the posterior probability:

\[
f^k(x^k) = \begin{cases} 
+1, & P(y = 1 \mid x^k) > 1 - \delta^k(x^k) \\
-1, & P(y = 1 \mid x^k) < \delta^k(x^k)
\end{cases}
\]

(3.18)

which is exactly our claim.

\[\square\]

The main implication of this result is that if the cost-to-go function \( \delta^k(x^k) \) is known then the risk \( \tilde{R}_k(\cdot) \) is only a function of the current stage decision \( f^k \). Therefore, we can ignore all of the other stages and minimize a single stage risk. Effectively, we decomposed the multi-stage problem in Eq. 3.5 into a stage-wise optimization in Eq. 3.8.
Note that the modified risk functional, \( \tilde{R}_k \), is remarkably similar to \( R_k \) except that the modified reject cost \( \delta^k(x^k) \) replaces the constant stage cost \( \alpha c_k \). Also, consider the range for which \( \delta^k(x^k) \) is meaningful. If we have 2 classes then a random guessing strategy would incur an average risk of \( \frac{1}{2} \). Therefore the risk for rejecting, \( \delta^k(x^k) \leq \frac{1}{2} \) in order to be a meaningful option. The work in \cite{Chow:1970} contains a detailed analysis of single stage reject classifier in a Bayesian setting.

![Figure 3-3: Optimal Reject Region can be expressed as the disagreement region of two binary classifiers (\( f_n \) and \( f_p \)).](image)

### 3.4.3 Reject Classifier As Two Binary Decisions

Consider a stage \( k \) classifier with a reject option from Theorem 3.1:

\[
f^k(x^k) = \begin{cases} 
+1, & P(y = 1 \mid x^k) > 1 - \delta^k(x^k) \\
-1, & P(y = 1 \mid x^k) < \delta^k(x^k) \\
\text{reject,} & \delta^k(x^k) \leq P(y = 1 \mid x^k) \leq 1 - \delta^k(x^k)
\end{cases}
\] (3.19)

It is clear from the expression that we can express the decision regions in terms of two binary classifiers \( f_n \) and \( f_p \). Observe that for a given reject cost \( \delta^k(x^k) \), the reject region is an intersection of two binary decision regions. To this end we further modify the risk function in terms of agreement and disagreement regions of the two classifiers,
\[ L_k(x^k, y, f_n, f_p, \delta^k) = \begin{cases} \delta(x^k), & f_n(x^k) \neq f_p(x^k) \\ 1, & f_n(x^k) = f_p(x^k) \land f_p(x^k) \neq y \end{cases} \]  

(3.20)

Note that the above loss function is symmetric between \( f_n \) and \( f_p \) and so any optimal solution can be interchanged. Nevertheless, we claim:

**Theorem 3.2.** Suppose \( f_n \) and \( f_p \) are two binary classifiers that minimize

\[ E_y \left[ L_k(x^k, y, f_n, f_p, \delta^k) \mid x^k \right] \]

over all binary classifiers \( f_n \) and \( f_p \). Then following resulting reject classifier:

\[ f^k(x^k) = \begin{cases} f_p(x^k), & f_n(x^k) = f_p(x^k) \\ \text{reject}, & f_n(x^k) \neq f_p(x^k) \end{cases} \]  

(3.21)

is the minimizer for \( E_y \left[ \bar{R}_k(x^k, y, f, \delta^k) \mid x^k \right] \) in Theorem 3.1 and the \( k \)th stage minimizer in Eq. 3.4.

**Proof.** For a given \( x^k \) and \( \delta(x^k) \),

\[ \min_{f} E_y \left[ \bar{R}_k(x^k, y, f, \delta^k) \mid x^k \right] = \]  

(3.22)

\[ \min_{f} \left\{ P_y \left[ y = -1 \mid x^k \right], P_y \left[ y = +1 \mid x^k \right], \delta(x^k) \right\} \]  

(3.23)

\[ \min_{f, f_p, f_n} E_y \left[ L_k(x^k, y, f_p, f_n, \delta^k) \mid x^k \right] = \]  

(3.24)

\[ \min_{f_p, f_n} \left\{ P_y \left[ y = -1 \mid x^k \right], P_y \left[ y = +1 \mid x^k \right], \delta^k(x^k) \right\} \]  

(3.25)

By inspection, the decomposition in 3.21 is the optimal Bayesian classifier minimizing \( E_y \left[ \bar{R}_k(x^k, y, f, \delta^k) \mid x^k \right] \).
We refer to Fig 3.3 for an illustration. We can express the new loss compactly as follows:

\[ L_k(x^k, y, f_p, f_n, \delta^k) = 1_{[f_p(x^k) \neq y]} 1_{[f_n(x^k) \neq y]} + \delta^k(x^k) 1_{[f_p(x^k) \neq f_n(x^k)]} \]  

(3.26)

Note that in arriving at this expression we have used: \( 1_{[a \neq c]} 1_{[a = b]} = 1_{[a \neq c]} 1_{[b \neq c]} \) for binary variables \( a, b, c \).

In summary, in this section, we derive the optimal POMDP solution and decouple a multi-stage risk to single stage optimization. Then, for the binary classification setting, we derive an optimal representation for a reject region classifier in terms of two biased binary decisions:

\[
\min_{f_k} \mathbb{E}[R(x, y, \ldots, f_k, \ldots)] \rightarrow \min_{f_k} \mathbb{E}[	ilde{R}_k(x^k, y, f^k, \delta^k)] \rightarrow \min_{f_k^p, f_k^n} \mathbb{E}[L_k(x^k, y, f_k^p, f_k^n, \delta^k)]
\]  

(3.27)

### 3.4.4 Stage-wise Empirical Minimization

In this section, we assume that the probability model \( D \) is no longer known and cannot be estimated due to high-dimensionality of the data. Instead, our task is to find multi-stage decision rules based on a given training set: \((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\).

We will take advantage of the stage-wise decomposition of the POMDP solution in Theorem 1 and parametrization of reject region in Theorem 2 to formulate an empirical version of the stage risk \( L_k(\cdot) \) in Eq. 3.26. However, this requires the knowledge of the cost-to-go, \( \delta^k : X^k \rightarrow \mathbb{R} \). Instead of trying to learn this complex function, we will define a point-wise empirical estimate of the cost-to-go on the training data:

\[ \delta^k(x_i^k) \rightarrow \delta_i^k, i = 1, 2, \ldots N \]  

(3.28)

and use it to learn the decision boundaries directly.
Note that by definition, \( \delta_k(x^k_i) \) is a only function of \( f^{k+1}, \ldots, f^K \). So the cost-to-go estimate is conveniently defined by the recursion,

\[
\delta_{i}^{k-1} = L_k(x^k_i, y_i, f^k_p, f^k_n, \delta_i^k) + \alpha c_k, \forall i
\]  

(3.29)

Now, we can form the empirical version of the risk in Eq 3.51 and optimize for a solution at stage \( k \) over some family of functions, \( \mathcal{F}^k \).

\[
\{ f^k_p(x^k), f^k_n(x^k) \} = \arg \min_{f_p, f_n \in \mathcal{F}^k \times \mathcal{F}^k} \frac{1}{N} \sum_{i=1}^{N} S_i^k L_k(x^k_i, y_i, f_p, f_n, \delta_i^k)
\]  

(3.30)

Observe that, as in standard setting, we need to constrain the class of decision rules \( f^k_p, f^k_n \in \mathcal{F}_k \times \mathcal{F}_K \) here. This is because with no constraints the minimum risk is equal to zero and can be achieved in the first stage itself.

Note, our stage-wise decomposition significantly simplifies the ERM. The objective in Eq. 3.30 is only a function of \( f^k_p, f^k_n \) given \( \delta^k_i \) and the state \( S^k_i \). To minimize an empirical version of a multi-stage risk in Eq. 3.4 is much more difficult due to stage interdependencies.

Given \( \delta^k_i \) and all the stages but the \( k \)th, we can solve 3.30 by iterating between \( f^k_p \) and \( f^k_n \). To solve for \( f^k_p \), we fix \( f^k_n \) and minimize a weighted error

\[
f^k_p = \arg \min_{f \in \mathcal{F}^k} \sum_{i=1}^{N} w_i \mathbf{1}[f(x^k_i) \neq y_i], \quad w_i = S^k_i \left[ \mathbf{1}[f^k_n(x^k_i) \neq y_i] + \delta^k_i - 2 \mathbf{1}[f^k_p(x^k_i) \neq y_i] \delta^k_i \right]
\]  

(3.31)

We can solve for \( f^k_n \) in the same fashion by fixing \( f^k_p \),

\[
f^k_n = \arg \min_{f \in \mathcal{F}^k} \sum_{i=1}^{N} w_i \mathbf{1}[f(x^k_i) \neq y_i], \quad w_i = S^k_i \left[ \mathbf{1}[f^k_p(x^k_i) \neq y_i] + \delta^k_i - 2 \mathbf{1}[f^k_n(x^k_i) \neq y_i] \delta^k_i \right]
\]  

(3.32)

To derive these expressions from 3.30 we used another identity for any binary vari-
ables $a$, $b$, $c$

$$1_{[a \neq b]} = 1_{[a \neq c]} + 1_{[b \neq c]} - 21_{[a \neq c]}1_{[b \neq c]}$$  (3.33)

Note the advantage of our parametrization from Theorem 2. We converted the problem from learning a complicated three region decision to learning two binary classifiers ($f_p$, $f_n$), where learning each of the binary classifiers reduces to solving a weighted binary classification problem. This is desirable since binary classification is a very well studied problem, and existing machine learning techniques can be utilized here, as we will demonstrate in the next section.

3.4.5 Algorithm

Minimizing the indicator loss is a hard problem. Instead, we take the usual ERM (empirical risk minimization) ([Friedman et al., 2001]) approach and replace it with a surrogate. We introduce an algorithm in the boosting framework based on the analysis from the previous section. Boosting is just one of our many possible machine learning approaches that can be used to solve it. We use boosting because it is easy to implement and is known to have good performance.

Recall that boosting is a way to combine simple classifiers to form a strong classifier. We are given a set of such weak classifiers $\mathcal{H} = \{h_1(x), h_2(x) \ldots h_M(x)\}$, $h_j(x) \in \{-1, +1\}$. Note that the set of weak classifiers need not be finite. Also, denote $\mathcal{H}_k \subset \mathcal{H}$ as a subset of weak classifiers that operate only on the first $k$ measurements of $x$. $h_j(x) = h_j(x^k)$ if $h_j \in \mathcal{H}_k$.

The strong classifier is the linear combination:

$$F(x) = \text{sgn} \sum_{h_j \in \mathcal{H}} q_j h_j(x)$$  (3.34)

Boosting algorithm operates in rounds. Each round can be viewed as a coordinate
descent step in the space of weak learners. In each round, a weak learner \( h_j(x) \) (a descent direction) is selected to be added to the linear combination classifier, \( F(x) \). Then a weight \( q_j \) (an optimal step size in that direction) is computed. This is repeated until termination criteria is reached. Boosting algorithms are described in more detail in Section 2.5.

**Global Surrogate:** In our algorithm, we use the sigmoid loss function \( C(z) = \frac{1}{1+\exp(z)} \) to approximate the indicator. Similar sigmoid based losses have been used in boosting before (Masnadi-Shirazi and Vasconcelos, 2009). Each subproblem (3.31) reduces to boosting a weighted loss.

To solve for stage \( k \), we keep the rest of the stages constant. To find \( f^k_p = \sum q_j h_j(x) \), we fix \( f^k_n \) and solve:

\[
f^k_p = \arg \min_{q_1,q_2,...} \sum_{i=1}^{N} w_i C \left( y_i \sum_{h_j \in \mathcal{H}^k} q_j h_j(x_i) \right)
\]  

(3.35)

Note that the weights \( w_i \), state variables \( S^k_i \) and cost-to-go \( \delta^k_i \) are also expressed in terms of the \( C(z) \) instead of \( 1_{[z]} \):

\[
w_i = S^k_i \left[ C(y f^k_p(x_i)) + \delta^k_i - 2 C(y f^k_p(x_i)) \delta^k_i \right]
\]  

(3.36)

To solve for \( f^k_n \), we solve the same problem but keep \( f^k_p \) constant instead:

\[
f^k_n = \arg \min_{q_1,q_2,...} \sum_{i=1}^{N} w_i C \left( y_i \sum_{h_j \in \mathcal{H}^k} q_j h_j(x_i) \right)
\]  

(3.37)

\[
w_i = S^k_i \left[ C(y f^k_p(x_i)) + \delta^k_i - 2 C(y f^k_p(x_i)) \delta^k_i \right]
\]

Note that the terms \( \delta^k_i \) and \( S^k_i \) do not depend on stage \( k \) and remain constant when solving for \( f^k_p \) and \( f^k_n \). For the ease of notation, we define a new term \( C_r \) that indicates if \( x_i \) is rejected at a \( k \)th stage. The term is close to one if \( f^k_p \) and \( f^k_n \) disagree (reject)
and small if they agree.

\[ C_r(f^k_p, f^k_n, x^k_i, y_i) = C(y_i, f^k_p(x^k_i)) + C(y_i, f^k_n(x^k_i)) - 2C(y_i, f^k_p(x^k_i))C(y_i, f^k_n(x^k_i)) \]  

(3.38)

The expressions for state variables and cost-to-go are now simplified.

\[ S_{i}^{k+1} = S_{i}^{k}C_r(f^k_p, f^k_n, x^k_i, y_i), \quad S_{i}^{1} = 1 \]  

(3.39)

The state variable remains greater than zero as long as \( x_i \) is rejected at every stage. The expression for cost-to-go at \( k \)th stage is:

\[
\delta_i^k = \alpha c_{k+1} \text{ meas. cost} + \underbrace{C(y_i, f^{k+1}_p(x_i^{k+1}))C(y_i, f^{k+1}_n(x_i^{k+1}))}_{\text{err. penalty if not rejected at stage } k+1} + \underbrace{\delta_i^{k+1}C_r(f^{k+1}_p, f^{k+1}_n, x_i^{k+1}, y_i)}_{\text{cost-to-to; if rejected at stage } k+1} 
\]  

(3.40)

(3.41)

The last two terms are simply a surrogate for \( L_k(\cdot) \) from 3.26 in terms of \( C(\cdot) \).

For the last stage (a standard binary classifier), we fix the first \( K - 1 \) stages and solve:

\[
f^K = \arg \min_{q_1, q_2, \ldots} \sum_{i=1}^{N} S_i^K C \left( y_i \sum_{h_j \in H^K} \frac{q_j h_j(x_i)}{q_h^{\sum_{j=1}^{H^K} h_j(x_i)}} \right) \]  

(3.42)

Our algorithms performs cyclical optimization over the stages. To initialize \( f^k_n, f^k_p \) for \( k = 1 \ldots K - 1 \), we simply hard code \( f^k_p \) to classify any \( x \) as +1 and \( f^k_n \) as -1 so that all \( x \)'s are rejected to the last stage. Using these nominal classifiers, we compute \( S_i^k \) and \( \delta_i^k \) according to equations 3.39 and 3.41 respectively.

At a stage \( k \), for a fixed \( \delta_i^k \) and \( S_i^k \), we alternate among minimizing \( f^k_p \) and \( f^k_n \) according to equations 3.35 and 3.37. In practice, we found that one iteration is sufficient.

Given a new estimate of stage \( k \), we update \( \delta_i^s \) for \( s > k \) and \( S_j^s \) for \( s < k \) and
then move on to optimizing another stage \( k' \). Given an estimate for stage \( k' \), we again update the state variables and cost-to-go for the rest of the system.

The stages are optimized in the following order. We start with the last stage and make our way backwards to the first stage. Then do a forward pass from 1st stage to last. These forward and back passes are repeated until convergence. See Algorithm 3.

---

**Algorithm 3** Global Algorithm

**INPUT:** \( \{x_i, y_i\}_{i=1}^N \), \( \{\mathcal{H}_k\}_{k=1}^K \) \{Weak Learners for each stage\}, \( \{\delta_k\}_{k=1}^K \) \{costs\}, \( D \) \{Loop Iterations\}

**INITIALIZE:** \( f_k^p(x) \leftarrow +1 \), \( f_k^n(x) \leftarrow -1 \), for \( k = 1 \ldots K - 1 \) \{first \( K - 1 \) stages reject everything\}

**for** \( d = 1, \ldots, D \) **do**

**for** \( k = K, \ldots, 1, 2, \ldots K - 1 \) **do**

\{Start from the last stage then iterate to the first stage and then back to last stage\}

**if** \( k < K \) **then**

Find \( f_k^p \) by solving boosting subproblem in 3.35

Find \( f_k^n \) by solving boosting subproblem to 3.37

**else if** \( k = K \) **then**

\{Last Stage\}

Find \( f^K(x) \) by solving boosting subproblem in 3.42

**end if**

Update \( \delta_s^s \) for \( s > k \) and \( S_s^s \) for \( s < k \)

**end for**

**end for**

\( F_k(x^k) \leftarrow \begin{cases} 
\text{sgn}(f_k^p(x^k)), & \text{if } \text{sgn} f_k^p(x^k) = \text{sgn} f_k^n(x^k) \\
\text{reject}, & \text{if } \text{sgn} f_k^p(x^k) \neq \text{sgn} f_k^n(x) 
\end{cases} \)

**OUTPUT:** \( F^1, F^2, \ldots, F^K \)

---

Our formulation allows us to form a surrogate for the entire risk in Equation 3.1, not just for each subproblem. This enables us to prove the following theorem.

**Theorem 3.3.** Our global surrogate algorithm converges to a local minimum.

**Proof.** This is simply due to a fact that we are minimizing a global smooth cost function by coordinate descent over \( q_1^1, q_1^2, q_2^2, \ldots, q^K \). Here, \( q_k^p \) is the vector of
weak learner weights parametrizing $f^k_p$. For the derivation of three stage system global cost refer to Appendix B.1.

However, since the global loss and the loss for each subproblem are non-convex programs, there is no global optimality guarantee. Theorem 3.3 ensures that our algorithm terminates.

**Regularization to reduce overfitting:** To reduce overtraining, we introduce a simple but effective regularization. For any loss $C(z)$ and a parameter $\lambda$, we introduce a multiplicative term to the cost function:

$$\min_q \exp(\lambda |q|) \sum_{i=1}^{N} C(y_i \sum_{h_j \in H} q_j h_j(x_i))$$

(3.43)

Here, $C'(z) = \frac{dC(z)}{dz}$. The term $\exp(\lambda |q|)$ limits how large a step size for a weak hypothesis can become. It also introduces a simple stopping criteria: abort if

$$\frac{\sum_{i=1}^{n} C'(y_i f_t(x_i)) y_i h_{t+1}(x_i)}{\sum_{i=1}^{n} C(y_i f_t(x_i))} \leq \lambda$$

This corresponds to a situation when no descent directions (weak hypothesis $h_{t+1}$) can be found to minimize the cost function.

**3.4.6 Generalization Error**

Our system is composed of margin maximizing classifiers, therefore it is appropriate to derive generalization error bounds based on margins. It turns out that we can employ maximum margin generalization techniques from Schapire et al., 1997b to derive error bounds for a two stage version of the system. A two stage system consists of three boosted binary classifiers:

$$f^1_p(x^1) = \sum_{h_j \in H^1} q^p_j h_j(x^1), \quad f^1_n(x^1) = \sum_{h_j \in H^1} q^n_j h_j(x^1), \quad f^2(x^2) = \sum_{h_j \in H^2} q^2_j h_j(x^2)$$
Theorem 3.4. Let \( D \) be a distribution on \( \mathcal{X} \times \{+1, -1\} \), and let \( S \) be a sample of \( m \) examples chosen independently at random according to \( D \), and a rejected subsample of size \( m_r \), \( S_r = \{ x \in S | f^1_p(x) \neq f^1_n(x) \} \). Assume that the base-classifier spaces \( \mathcal{H}_1 \) and \( \mathcal{H}_2 \) are finite, and let \( \delta > 0 \). Then with probability at least \( 1 - \delta \) over the random choice of the training set \( S \), all boosted classifiers \( f^1_n, f^1_p, f^2 \) satisfy the following bound for all \( \theta_1 > 0 \) and \( \theta_2 > 0 \):

\[
P_D[y f^1_n(x) \leq 0, y f^1_p(x) \leq 0] + P_D[y f^2(x) \leq 0, f^1_n(x) \neq f^1_p(x)] \leq \]

\[
P_S[y f^1_n(x) \leq \theta_1, y f^1_p(x) \leq \theta_1] + P_{S_r}[y f^2(x) \leq \theta_2] + \]

\[
O \left( \frac{1}{\sqrt{m}} \left( \frac{\log m \log |\mathcal{H}_1|}{\theta_1} + \log \frac{1}{\delta} \right)^{\frac{3}{2}} \right) + O \left( \frac{1}{\sqrt{m_r}} \left( \frac{\log m_r \log |\mathcal{H}_2|}{\theta_2} + \log \frac{1}{\delta} \right)^{\frac{1}{2}} \right)
\]

\[ (3.44) \]

Proof. The proof extends the approach in [Schapire et al., 1997b] to a two stage system. For complete details please refer to the Appendix [B.2] to a two stage system. For complete details please refer to the Appendix [B.2].

The two stage system can be compactly expressed:

\[
F(x) = \begin{cases} 
\text{sgn} f^1_p(x^1), & \text{sgn} f^1_p(x^1) = \text{sgn} f^1_n(x^1) \\
\text{sgn} f^2(x^2), & \text{sgn} f^1_p(x^1) \neq \text{sgn} f^1_n(x^1)
\end{cases}
\]

\[ (3.45) \]

The system error is a sum of two terms: error at the 1st stage + error at the 2nd stage. Theorem 4 states the generalization error of \( F(x) \) is bounded by the empirical margin error over the training set \( S \) plus a term that is inversely proportional to the margins and the number of training samples at that stage. An interesting observation is that \( m_r \), number of samples that reach the 2nd stage, depends on the reject classifier at the 1st stage. So if very few examples make it to the second stage then we do not have strong generalization.

3.4.7 Experiments

The goal is to demonstrate that a large fraction of data can be classified at an early stage using a cheap modality. In our experiments, we use four real life datasets with
measurements arising from meaningful stages. We compare our algorithm to two methods:

**Myopic:** An absolute margin of a classifier is a measure of how confident a classifier is on an example. Examples with small margin have low confidence and should be rejected to the next stage to acquire more features. This approach is based on reject classification ([Bartlett et al., 2008](#)). We know from Claim 1 that the optimal classifier is a threshold of the posterior. For each stage, we obtain a binary boosted classifier, $f^k(\cdot)$, trained on all the data. We then threshold the margin of the classifier, $|f^k(x^k)|$. It is known that given an infinite amount of training data, boosting certain losses (sigmoid loss in our case) approaches the log likelihood ratio, $f(x) = \frac{1}{2} \log \frac{P(y=1|x)}{P(y=-1|x)}$ ([Masnadi-Shirazi and Vasconcelos, 2009](#)). So a reject region for a given threshold $t_k$ is defined: $\{x \mid |f^k(x)| \leq t_k\}$. This is a completely myopic approach as the rejection does not take into account performance of later stages. This method is very similar to TEFE ([Liu et al., 2008](#)) which also uses absolute margin as a measure for rejection. The difference is that our myopic strategy is a boosting classifier not an SVM as used in TEFE.

**Expected Utility/Margin:** An expected margin difference measures how a new attribute, if acquired, would be useful for an example. If this expected utility for an example is large then a new attribute should be acquired. This approach is based on the work by [Kanani and Melville, 2008](#). We train boosted binary classifiers on all the data for each stage: $f^k(x^k)$. Given the measurement at the current stage $x^k$, we compute an expected utility (change in normalized margin) of acquiring the next measurement $x_{k+1}$:

$$U(x^k) = \sum_{x_{k+1} \in \mathcal{X}_{k+1}} |f^k(x^k) - f^{k+1}([x^k, x_{k+1}])| P(x_{k+1} | x_k)$$

(3.46)
An $x^k$ is rejected to the next stage if its utility $U(x^k) \geq t_k$ is greater than a threshold. Here, $\mathcal{X}_{k+1}$ denotes the possible values that $x_{k+1}$ can take. Note this approach requires estimating $P(x_{k+1}|x^k)$, therefore the $(k+1)$th measurement has to be discrete or distribution needs to be parametrized. Due to this limitation, we only compare this method on two datasets.

While there are many different ways to estimate a probability likelihood we used a Gaussian mixture due to its computational efficiency, The number of mixture components is equal to the number of discrete values that $x_2$ can take from an alphabet $\mathcal{X}_2$. The conditional $P(x_1 | x_2 = j)$ is a gaussian whose parameters are learned from the training set. Using Bayes rule, $P(x_2 | x_1) = \sum_{x' \in \mathcal{X}_2} \frac{P(x_1 | x_2 = x')}{P(x_1 | x_2 = x')}$

**Performance Metric:** A natural performance metric is the trade off between system error and measurement cost. Note, for utility and myopic methods, it is unclear how to set a thresholds $t_k$ for each stage given a measurement cost $\delta_k$. For this reason, we only compare them in a two stages system. More than two stages is not-practical because we would need to test every possible $t_k$ for every stage $k$.

In a two stage setting, since every example has to pass through the first stage, only the cost of the second stage, $c_2$, affects the performance. The average measurement cost of the system is proportional to $c_1 + (\text{the fraction of examples rejected to the second stage}) \times c_2$. So knowing the exact cost of the second stage sensor, $c_2$, is not necessary. In our algorithm, we vary the error-cost trade-off $\alpha$ to generate a system error vs reject rate plot. For margin and utility, we sweep a threshold $t_k$. System error is the sum of 1st stage and 2nd stage errors. Reject rate is the fraction of examples rejected to the 2nd stage and require additional measurements. Low reject rate (cost) corresponds to higher error rate as most of the data will be classified at the first stage using less informative measurements. High reject rate will have performance similar to a centralized classifier, as most examples will be classified at the 2nd stage.
Set Up: In all our experiments, we use stumps as weak learners. A stump classifier $h_{d,g,s} \in \{+1,-1\}$ is parametrized by a threshold $g$ on $d$th dimension and a sign variable $s \in \{+1,-1\}$: $h_{d,g,s}(x) = s \times \text{sgn}(x_d - g)$. We chose stumps for their simplicity, computation speed and relatively good performance. While more complicated weak learners, such as decision trees can be used, they would only change the absolute performance of our experiments. The entire curves would just move vertically up or down. Our goal is to demonstrate the advantage of a multi-stage classifier relative to the centralized system (a system that uses all the measurements for all examples).

For each dataset and experiment, we randomly split the data 50/50 for training and testing. The results are evaluated on a separate test set, and the simulations are averaged over 50 monte-carlo trials. The number of iterations for each boosting subproblem is set to $T = 50$. In our global surrogate algorithm, the number of outer loop iterations is set to $D = 10$

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>1st Stage</th>
<th>2nd Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian Mixture</td>
<td>1000</td>
<td>1st dim</td>
<td>2nd dim</td>
</tr>
<tr>
<td>Mammogram Mass</td>
<td>830</td>
<td>3 CAD meas.</td>
<td>Radiologist Rating</td>
</tr>
<tr>
<td>Pima Diabetes</td>
<td>810</td>
<td>6 simple tests: BMI, sex, ..</td>
<td>2 blood tests</td>
</tr>
<tr>
<td>Polyps</td>
<td>310</td>
<td>12 freq. bins</td>
<td>126 freq. bins</td>
</tr>
<tr>
<td>Threat</td>
<td>1300</td>
<td>Images in IR, PMMW</td>
<td>Images in AMMW</td>
</tr>
</tbody>
</table>

Table 3.1: Dataset Descriptions

Discrete Valued Data Experiments: To compare our method to the utility approach, we consider discrete data. The first dataset is a quantized (with 20 levels) Gaussian mixture synthetic data in two dimension. The 1st dimension is stage one; the 2nd dimension is stage two. The second dataset is Mammogram Mass from UCI Machine Learning Repository. It is used to predict the severity of a mammography mass lesion (malicious or benign). It contains 3 attributes extracted from the CAD image and also an evaluation by a radiologist on a confidence scale in addition to the true biopsy results. The first stage are features extracted from the CAD image, and
the second stage is the expert confidence rated on a discrete scale 1 – 5. Automatic analysis of the CAD image is cheaper than employing an opinion of a radiologist.

Simulations in Figure 3.4 demonstrate that utility performs worse when compared to our approach. Our approach requires far fewer 2nd stage measurements to achieve an error rate close to the centralized strategy than the utility approach. This is possibly due to poor probability estimates in limited data setting.

**Figure 3.4:** Comparison of Global to Utility on (a) quantized two gaussian clusters and (b) mammogram dataset. Reject Rate vs System Error. Reject Rate is the fraction of examples with measurements from both stages. Our approach outperforms Utility possibly because we do not need to estimate probability likelihoods.

**Continuous Valued Data Experiments** We compare our global method to the myopic method on three datasets. The Pima Indians Diabetes Dataset (UCI MLR) consists of 8 measurements. Since the stages are not specified in this dataset, we group measurements with similar costs into separate modalities. 6 of the measure-
ments are inexpensive to acquire and consist of simple tests such as body mass index, age, pedigree. These we designate as the first stage. The other two measurements constitute the second stage and require more expensive procedures.

The polyp dataset consists of hyper-spectral measurements of colon polyps collected during colonoscopies ([Rodríguez-Díaz and Castañón, 2009]). The attribute is a measured intensity at 126 equally spaced frequencies. Finer resolution requires higher photon count which is proportional to acquisition time. For a first stage, we use a coarse measurement down-sampled to only 12 frequency bins. The second stage is the full resolution frequency response. Using the course measurements is cheaper than acquiring the full resolution.

The threat dataset contains images taken of people wearing various explosives devices. The imaging is done in three modalities: infrared (IR), passive millimeter wave (PMMW), and active millimeter (AMMW). All the images are registered. We extract many patches from the images and use them as our training data. A patch carries a binary label, it either contains a threat or is clean. IR and PMMW are the fastest modalities but also less informative. AMMW requires raster scanning a person and is slow but also the most useful.

<table>
<thead>
<tr>
<th>Name</th>
<th>Centralized</th>
<th>Utility</th>
<th>Myopic</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D Gaussian Mix</td>
<td>0.09</td>
<td>50%</td>
<td>-</td>
<td>30%</td>
</tr>
<tr>
<td>Mammogram</td>
<td>0.165</td>
<td>60%</td>
<td>-</td>
<td>15%</td>
</tr>
<tr>
<td>Pima Diabetes</td>
<td>0.26</td>
<td>-</td>
<td>60%</td>
<td>45%</td>
</tr>
<tr>
<td>Polyps</td>
<td>0.24</td>
<td>-</td>
<td>75%</td>
<td>50%</td>
</tr>
<tr>
<td>Threat</td>
<td>0.185</td>
<td>-</td>
<td>50%</td>
<td>45%</td>
</tr>
</tbody>
</table>

Table 3.2: Performance illustration for different datasets (quantitate view of the curves). Datasets have 2 sensing modalities. Centralized denotes the test error obtained with all modalities. Last three columns denotes performance for different approaches. Performance is measured by the average number of examples requiring 2nd stage to achieve error close to centralized. Utility approach does not work for last three datasets due to high-dimensionality issues. We note the significant gains of our approach over competing ones of many interesting datasets.
Figure 3-5: Three datasets are evaluated: pima, polyps and threat. Reject Rate vs Error Rate for a varying $\alpha$. Reject Rate is the fraction of examples with measurements from both stages. Global and Myopic are compared. Global (our approach) has a better performance over all while Myopic does better in some situations.
In Figure 3.5, global performs better than margin in most cases. On threat data, margin appears to be doing just marginally worse than global, however, we get only a few points on the curve with reject rates less than 50%. Due to the heuristic nature of margin, we cannot construct a multistage classifier with an arbitrary reject rate.

The goal is to reach the performance of a centralized classifier (100% reject rate) while utilizing the 2nd stage sensor only for a small fraction of examples. Overall, the results demonstrate the benefit of multi-stage classification: rejection rate can be set to less than 50% with only small sacrifices in performance. For the mammogram data, this implies that for half of the patients a diagnoses can be made solely by an automatic analysis of a CAD image without an expensive opinion of a radiologist. For the Pima data, similar error can be achieved without an expensive medical procedures. For the polyps dataset, a fast low resolution measurement is enough to classify a large fraction of patience. In the threat dataset, IR and PMMW are sufficient to decide whether or not a threat is present for the majority of instances without requiring a person to go through a slower AMMW scanner.

**Unbalanced False Positive and False Negative Penalties:** In medical diagnosis and threat detection, the penalty of false positives and false negatives is not equal. We can easily adapt our algorithm to account for such setting. Empirical Risk in 3.54 can be modified to include a penalty of $w_p$ for a Type I error and $w_n$ for a Type II error. The experiment in Figure 3.6 demonstrates our global algorithms in such scenario. For each reject $\alpha$, we compute an ROC curve. This allows to select an operating point of the system with a desired false alarm or detection rate. We also compute a corresponding average reject rate for each value of $\alpha$. So the highest reject rate corresponds to the best performance but also to the highest acquisition cost incurred by the system. Note that very good performance can be achieved by requesting only 50% of instances to be measured at the second stage.
Figure 3-6: Two Stage ROC using the global surrogate method. Each ROC curve corresponds to a different value of $\alpha$. The legend displays average reject rate for $\alpha$’s. Note, the circle marker ROC corresponds to the centralized system (100% reject rate). Very good performance can be achieved by requesting only 50% of instances to be measured at the second stage.

Three Stages: Lastly, we demonstrate a three stage system, we apply our algorithm to three stages of threat dataset. Note for margin it is unclear how to generalize it to
a multistage scenario and there is no way to define reject costs for different stages. We set the first stage to be IR, second PMMW and AMMW as third. There is no cost for acquiring \( IR \). We vary the costs for the PMMW (2nd) stage, \( c_2 \), and AMMW (3rd), \( c_3 \), to generate an error map (color in Figure 3·7). A point on the map corresponds to a performance of a particular multistage classification strategy. The vertical axis is the fraction of examples for which only IR and PMMW measurements are used in making a decision. The horizontal axis is the fraction of examples for which all three modalities are used. For example, a red point in the figure, \( \{.4, .15, .195\} \), correspond to a system where 40% of examples use IR and PMMW, 15% use only \( IR \) and the rest of data (45%) use all the modalities. And this strategy achieves a system error rate of 19.5%. Note that the support lies below the diagonal. This is because the sum or reject rates has to be less than one. Results demonstrate some interesting observations. While best performance (about 19%) is achieved when all the modalities are used for every example, we can move along the vertical lines and allow a fraction to be classified by IR and PMMW, avoiding AMMW all together. This strategy achieves performance comparable to a centralized system, (IR+PMMW+AMMW).
Figure 3.7: Three Stage System. The color maps error. A point on the map corresponds to a performance of a particular multistage classification strategy. The vertical axis is the fraction of examples for which only IR and PMMW measurements are used in making a decision. The horizontal axis is the fraction of examples for which all three modalities are used. An example red point in the figure, \{0.4, 0.15, 0.195\}, correspond to a system where 40% of examples use IR and PMMW, 15% use only IR and the rest of data (45%) use all the modalities. And this strategy achieves a system error rate of 19.5%.
3.5 Sequential Reject Classifiers in a Multi-Class Setting

Next, we extend our work to multi-class setting. We will perform analysis similar to the binary setting. First, examining the solution in the Bayesian setting will enable us to decompose the system risk in a stage-wise manner. Next step is to obtain a convenient parameterization of the rejection option in a multi-class setting. To simplify the problem, we have access to pre-trained classifiers at each stage. A reject decision becomes a function that thresholds a confidence of a classification decision at a stage.

We also derive bounds for generalization error for this new parametrization. We consider the binary classification setting for simplicity. In this setting our system turns out to be a Boolean fusion of binary decision functions. Using this insight, we derive an upper bound on the VC dimension of the multi-stage reject classifier. We show that the VC dimension of a K-stage system grows as $K \log K$ times the maximum complexity of any stage. Our approach also enjoys other advantages. We can utilize “black box” classifiers that are pre-programmed into a sensing modality. In this context, our problem reduces to learning reject regions at each stage assuming that there is a confidence associated with each decision. In this setting, the complexity of our system only depends on the complexity of the highest reject region which is typically not very high.

3.5.1 Multi-Class Problem Setting

The problem set up is similar to the binary setting. Data $(x, y) \in \mathcal{X} \times \{1, 2, \ldots C\}$ be distributed according to an unknown distribution $D$. A data point has $K$ features, $x = \{x_1, x_2, \ldots, x_K\}$, but now belongs to one of $C$ classes.

Again, the system has $K$ stages, the order of the stages is fixed, and $k$th stage acquires a $k$th measurement. At each stage, $k$, there is a decision with a reject
option, \( f^k \). It can either classify an example, \( f^k(x^k) : \mathcal{X}^k \rightarrow \{1, 2, \ldots, C\} \), or delay the decision until the next stage, \( f^k(x^k) = r \) and incur a penalty of \( c_{k+1} \). We can define the system risk in the multi-class setting to be the same as in the binary setting,

\[
R(f^1, \ldots, f^K, x, y) = \sum_{k=1}^{K} S^k(x^k) R_k(f^k, x^k, y) \tag{3.47}
\]

Here, \( R_k \) is the risk of classifying at \( k \)th stage, and \( S^k(x^k) \in \{0, 1\} \) is the binary state variable indicating whether \( x \) has been rejected up to \( k \)th stage.

\[
R_k(x^k, y, f^k) = \begin{cases} 
\alpha c_{k+1}, & f^k(x^k) = r \\
1, & f^k(x^k) \neq y \land f^k(x^k) \neq r \\
0, & \text{otherwise}
\end{cases}
\]

If \( x \) is active and is misclassified, the penalty is 1. Here, unlike the binary setting, we do not consider different types of misclassification errors. If it is rejected then the system incurs a penalty of \( \alpha c_k \), and the state variable for that example remains at 1.

\[
S^{k+1}(x^{k+1}) = \begin{cases} 
S^k(x^k), & f^k(x^k) = r, S^0 = 1 \\
0, & \text{else}
\end{cases}
\tag{3.48}
\]

Here, \( \alpha \) is again the trade-off parameter between error and cost.

### 3.5.2 Multi-Class Bayesian Analysis

As in the binary case, if \( D \) is known, the problem reduces to minimize the expected conditional risk,

\[
\min_{f^1, \ldots, f^K} \mathbb{E} [R(f^1, \ldots, f^K, x^k, y) \mid x] \tag{3.49}
\]
And the solution naturally, reduces to single stage optimization. Cost-to-go definition remains the same,

$$\delta^k(x^k) = \alpha_{c+1} +$$

as does the modified risk functional,

$$\tilde{R}_k(x^k, y, f^k, \delta^k) = \begin{cases} 
\delta(x^k), & f^k(x^k) = r \\
1, & f^k(x^k) \neq y \land f^k(x^k) \neq r \\
0, & \text{otherwise}
\end{cases}$$

and we prove the following theorem which is an extension of the binary setting result. (see Appendix B.3 for proof),

**Theorem 3.5.** The optimal solution $f^1, f^2, \ldots f^K$ to the multi-stage risk in Eq. 3.49 decomposes to single stage optimization,

$$f^k = \arg\min \mathbb{E} \left[ \sum_{t=k+1}^{K} S^t(x^t) R_t(f^t, x^t, y) \mid x^k, S^k(x^k) = 1 \right]$$

and the solution is:

$$f^k(x^k) = \begin{cases} 
\hat{y}, & \bar{P}(x^k) > 1 - \delta^k(x^k) \\
\text{reject}, & \bar{P}(x^k) \leq 1 - \delta^k(x^k)
\end{cases}$$

$$\hat{y} = \arg\max_j P(y = j \mid x^k), \quad \bar{P}(x^k) = \max_j P(y = j \mid x^k)$$

In the multi-class setting we are still able to decompose the multi-stage problem in Eq. 3.49 into a stage-wise optimization in Eq. 3.51. If the cost-to-go function $\delta(x^k)$ is known then the risk $\tilde{R}_k(\cdot)$ is only a function of the current stage decision $f^k$. Therefore, we can ignore all of the other stages and minimize a single stage risk.
3.5.3 Stage-Wise Empirical Risk Minimization

Here, our goal is to learn multi-stage decision rules from a given training set:

\[ (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N) \]

We will take advantage of the stage-wise decomposition in Theorem 3.5 and formulate an empirical version of the stage risk \( \tilde{R}_k(\cdot) \) in Eq. 3.51. However, this requires an empirical estimate of the cost-to-go, \( \delta^k(x_i^k) \to \delta_i^k \), since we are not estimating probability models. Note that by definition, \( \delta^k(x_i^k) \) is a only function of \( f^{k+1}, \ldots, f^K \). So the cost-to-go estimate is conveniently defined by the recursion,

\[
\delta_i^{k-1} = S_i^k \tilde{R}_k(x_i^k, y_i, f^k, \delta_i^k) + \alpha c_k, \quad \forall i
\]  

(3.53)

Now, we can form the empirical version of the risk in Eq 3.51 and optimize for a solution at a stage \( k \) over some family of functions, \( F_k \).

\[
f^k(x^k) = \arg \min_{f \in F_k} \frac{1}{N} \sum_{i=1}^N S_i^k \tilde{R}_k(y_i, x_i^k, f, \delta_i^k)
\]  

(3.54)

Note, the stage-wise decomposition significantly simplifies the ERM. The objective in Eq. 3.54 is only a function of \( f^k \) given \( \delta_i^k \) and the state \( S_i^k \). To minimize an empirical version of a multi-stage risk in Eq. 3.49 is much more difficult due to stage interdependencies.

3.5.4 Multi-class decision with a reject option

Recall that at each stage, \( f^k(x^k) \), is a \( C + 1 \) decision function where the extra decision is due to the reject option. Because of this additional decision, minimizing the empirical risk at each stage is still difficult. In order to simplify the problem, we factorize the reject option from the multi-class decision.

Assume that at each stage, our system has a fixed stage classifier, \( d^k : \mathcal{X}^k \to \)
\{1, \ldots, C\} and its associated confidence function \(\sigma_{d^k} : \mathcal{X}^k \rightarrow \mathbb{R}^+\). \(\sigma(\cdot)\) reports how confident \(d^k(\cdot)\) is in classifying \(x^k\). Our choice for \(\sigma(\cdot)\) (described in Section [3.5.5]) is based on the absolute margin of a binary classifier, which evidently is a popular heuristic for confidence [Bartlett et al., 2008]. Using this reduction, we propose the following parameterization of a multi-class classifier with a reject option at each stage.

\[ f^k(x^k) = \begin{cases} 
  d^k(x^k), & \sigma_{d^k}(x^k) > g^k(x^k) \\
  \text{reject}, & \sigma_{d^k}(x^k) \leq g^k(x^k) 
\end{cases} \tag{3.55} \]

We designate \(g(\cdot)\) as a rejector at stage \(k\). The reject region is constructed by thresholding the confidence measure \(\sigma(\cdot)\) by \(g(x)\). In the space where \(g(x)\) is small, few examples are rejected. In the space where \(g(x)\) is large, rejection is high. Note that \(g(x)\) varies with \(x\). This dependence on \(x\) is important because it enables \(g(x)\) to selectively reject specific regions in the space. Our choice in parameterization mimics the optimal reject region: \(\max_j \mathbb{P}(y = j \mid x^k) \leq 1 - \tilde{\delta}^k(x^k)\). Recall that the optimal binary classifier is \(\arg \max_j \mathbb{P}(y = j \mid x^k)\). So the reject region is the space around the boundary whose size varies as a function of \(\delta^k(x^k)\).

Furthermore, we can rewrite the empirical risk in Eq. (3.54) using our parameterization,

\[ \tilde{R}_k(x^k_i, y_i, d^k, g^k) = \begin{cases} 
  \mathbb{1}_{[d^k(x^k_i) \neq y_i]} & \mathbb{1}_{[\sigma_{d^k}(x^k_i) > g(x^k_i)]} \\
  \text{error penalty} & \text{not rejected} \\
  \delta^k_i & \mathbb{1}_{[\sigma_{d^k}(x^k_i) \leq g(x^k_i)]} \\
  \text{cost to go} & \text{rejected} 
\end{cases} \tag{3.56} \]

Next, if we use this simplified form and hold the rest of the system constant then minimizing Eq. (3.54) with respect to \(g^k(x^k)\) over a family of functions \(G^k\) reduces to a supervised learning problem: (see Appendix B.4 for proof)

**Lemma 3.6.** If \(d^k(x)\), \(S^k_i\) and \(\delta^k_i\) are held constant then minimization over \(g(\cdot)\) in
To illustrate our parametrization, consider a binary classification setting. $d^k(\cdot)$ is a hyperplane and the confidence $\sigma_{d^k}(\cdot)$ is the distance to this hyperplane. A possible reject region is constructed by thresholding the confidence by a rejector $g^k(\cdot)$. Note how the width of the reject region varies with $x^k$ because $g^k$ is a function of $x^k$.

**Figure 3-8:**

$\sigma_{d^k}(x^k) = g^k(x^k)$

\[ d^k(x^k) \]

**+1 region**

-1 region

**reject region**

\[ g^k(x^k) = \arg \min_{g \in G^k} \sum_{i=1}^{N} S^k_i |w_i| \mathbf{1}_{[b_i g(x^k_i) - z_i \leq 0]} \]

\[ w_i = \mathbf{1}_{[d^k(x^k_i) \neq y_i]} - \delta^k_i, \quad z_i = \sigma_{d^k}(x^k_i), \quad b_i = \text{sgn} w_i \]

This simplified problem closely resembles minimizing weighted binary misclassification error. The pseudo labels $b_i$ play an important role. Note the weight $w_i$ is the difference between the risk of the current stage $d^k(\cdot)$ and the cost of rejecting, $\delta^k_i$. The label $b_i$ is +1, if it is more costly to classify $x_i$ at present stage and −1 if the penalty for rejecting is higher than classifying. This optimization finds a rejector $g(\cdot)$ such that the examples of pseudo class +1 are rejected and examples of class −1 are classified. Pseudo class +1 consists of examples with higher misclassification risk than rejection cost. Recall that $S^k_i$ are just binary variables indicating whether $x_i$ is still active at stage $k$. 

Eq 3.54 reduces to:

\[ g^k(x^k) = \arg \min_{g \in G^k} \sum_{i=1}^{N} S^k_i |w_i| \mathbf{1}_{[b_i g(x^k_i) - z_i \leq 0]} \]

\[ w_i = \mathbf{1}_{[d^k(x^k_i) \neq y_i]} - \delta^k_i, \quad z_i = \sigma_{d^k}(x^k_i), \quad b_i = \text{sgn} w_i \]
Figure 3.9: The figure illustrates the simplified optimization problem for $g^k(x^k)$ in Lemma 2. The objective is to find a rejector function to fit the decision regions in the figure. The data in the green region has cost-to-go smaller than the risk of classifying at the current stage and therefore is to be rejected. The data outside the green has higher cost-to-go than misclassification risk and is to be not rejected.

In summary, given $S_i^k, \delta_i^k, d^k(x^k)$, to solve for the rejector $g^k(x^k)$ requires finding a binary decision with pseudo labels $b_i$ and weights $|w_i|$ on the training set with respect to the indicator loss offset by $z_i$'s.

3.5.5 Algorithm

In this section, using the simplified rejector subproblem from Lemma 2, we provide one possible implementation of the multi-stage in the setting of multi-class to binary reduction and explain our stage-wise optimization. In our problem, we assume that we are either provided with stage classifiers $d^1, d^2, \ldots, d^K$ or train them a-priori. So our objective is to find the rejectors $g^1, g^2, \ldots g^{k-1}$ at each stage.

**Embedding the reject option** Before we proceed to finding the rejectors, $g^k$, we explain how we implement pre-training of $d^k$. We utilize a well known technique for multi-class classification: reduction from multi-class to binary \cite{Allwein et al., 2001}. For each class $j \in \{1, 2, \ldots C\}$, we choose a binary codeword $p_j \in \{+1, -1\}^M$ of length $M$. Let

$$h(x^k) = [h_1(x^k) \ h_2(x^k) \ \ldots \ h_M(x^k)] \quad (3.58)$$
be a vector valued classifier such that \( h_m : \mathcal{X}^k \to \mathbb{R} \). This approach reduces a multi-class problem to finding \( M \) binary classification functions, \( h_m(\mathbf{x}^k) \), with respect to code labels \( p_jm \). For each sub-problem \( m \), we take the usual ERM approach and upper-bound the indicator error by a convex loss: \( 1[z] \leq C[z] \) and fix a family of classifiers \( \mathcal{H}^k \).

\[
h_m^k(\mathbf{x}^k) = \arg \min_{h \in \mathcal{H}^k} \sum_{i=1}^{N} C[ p_{y_i} h(\mathbf{x}^k_i) ]
\]  

(3.59)

We use the logistic loss function, \( C[z] = \log(1 + \exp(-z)) \) and set \( \mathcal{H}^k \) to be a family of polynomial kernel classifiers. Polynomial kernel classifier of degree \( q \) is parametrized by a vector \( \mathbf{a} \): \( h(\mathbf{x}) = \sum_{i=1}^{N} a_i (\mathbf{x}^T \mathbf{x} + 1)^q \). Given an output \( \mathbf{h}^k(\mathbf{x}^k) \), we use maximum projection decoding to assign a class estimate to the best matching codeword. We define a stage classifier as

\[
d^k(\mathbf{x}^k) = \arg \max_{j=\{1...C\}} \mathbf{p}_j^T \mathbf{h}^k(\mathbf{x}^k)
\]  

(3.60)

For example, in our implementation, we use one vs all coding. Here the length of the codeword is \( M = C \) and, for class \( j \), each element of the codeword \( \mathbf{p}_j \) is \(-1\) except that the \( j \)th position is \(+1\), and \( d^k(\mathbf{x}^k) = \arg \max_{j=\{1...C\}} h_j^k(\mathbf{x}^k) \).

For the confidence function, \( \sigma(\cdot) \), we select an absolute maximum projection: \( |\max_{j=\{1...C\}} \mathbf{p}_j^T \mathbf{h}^k(\mathbf{x}^k)| \). Our choice in \( \sigma(\cdot) \) is inspired by an absolute margin of a binary classifier, \( |h(\mathbf{x})| \), which is a popular heuristic measure of classifier confidence [Bartlett et al., 2008]. However, in a multi-class setting, we use the absolute value of the best matching projection onto the codeword as a measure of confidence instead of a single margin. For example, \( \mathbf{p}^T \mathbf{h}^k(\mathbf{x}^k) \) is maximized when the classifier output matches the codeword exactly. A small value of the projection indicates that \( \mathbf{h}^k(\cdot) \) has lower confidence in its classification.

Substituting multi-class to binary reduction into our parameterization in Eq. 3.55
Algorithm 4 Our Method

Input: $\{x_i, y_i\}_{i=1}^{N}, \{d^k(\cdot)\}_{k=1}^{K}, \sigma_d(\cdot), \{c^k\}_{k=1}^{K}, P$

Initialize: $S^k_i = 1, \forall (i,k)$

for $p = 1,2,\ldots,P$ do

for $k = K - 1, K - 2, \ldots, 1$ do

Update $\delta^k_i$ according to Eq. 3.53

Train $g^k(x^k)$ according to Eq. 3.62

Update $S^k_i$ according to Eq. 3.48

end for

end for

Output: for $k = 1, \ldots K - 1,$

$$f^k(x^k) = \begin{cases} d^k(x^k), & \sigma_d(x^k) > g^k(x^k) \\ \text{reject}, & \sigma_d(x^k) \leq g^k(x^k) \end{cases}$$

yields a multi-class decision with a reject option:

$$\hat{y} = \arg \max_j p_j^T h^k(x^k), \quad \bar{h}(x^k) = \max_j p_j^T h^k(x^k)$$

$$f^k(x^k) = \begin{cases} \hat{y}, & |\bar{h}(x^k)| > g^k(x^k) \\ \text{reject}, & |\bar{h}(x^k)| \leq g^k(x^k) \end{cases}$$

Stage-wise optimization: To compute a rejector $g^k(x^k)$, every stage except the $k$th is held constant. We upper-bound the $1[\cdot] \leq C[\cdot]$ in Lemma 2. For a convex loss $C[\cdot]$ and a family of polynomial kernels $G^k$, the resulting optimization is a convex program,

$$g^k(x^k) = \arg \min_{g \in G^k} \sum_{i=1}^{N} S^k_i |w_i| C [b_i(g(x^k_i) - z_i)]$$

Once $d^1, d^2, \ldots, d^K$ are precomputed, to train $g^k$’s, we proceed by cyclic optimization of stages one at a time in reverse order: $g^{K-1}, g^{K-2}, \ldots, g^1$. Note that the weights $w_i$’s capture the difference in risk between the current stage and the cost-to-go. The order of cyclic optimization is reversed due the recursive nature of the cost-to-go; $\delta^k_i$ is a function of the next stage. Initially, state variables $S^k_i$ are set to one for all
examples and stages. After the first pass through the stages outputs $g^k$’s, the $S_i^k$’s are updated. Using the updated state variables, $g^k$’s are retrained in the second pass and so on. In our experiments, we found that one pass is sufficient. For details refer to Algorithm 1. Here, $P$ is the number of passes of cyclic optimization over stages.

### 3.5.6 Complexity of a Multi-Stage System

Using this particular parametrization, we can bound the VC-dimension of the entire system in the binary classification setting. VC-dimension captures the complexity of a family of classifiers. The theory states that for the same training error, a classifier of low complexity has better generalization error than that of a high complexity. For more details refer to [Vapnik and Chervonenkis, 1971](#).

**Theorem 3.7.** Let $F(x)$ be the decision of our $K$-stage system in the binary classification setting, and $F(\cdot) \in \mathcal{F}$. Let $\mathcal{H}^k$ be the family of stage classifiers, $\mathcal{G}^k$ is the family of rejectors at each stage,

$$
\text{VC}[\mathcal{F}] \leq \phi_K \max_{k=1\ldots K-1} \{ \text{VC}[\mathcal{H}^k] + \text{VC}[\mathcal{G}^k], \text{VC}[\mathcal{H}^K] \}
$$

where $\phi_K = 2(3K - 2) \log (e(3K - 2))$  \hspace{1cm} (3.63)

**Proof.** (see Appendix B.5 for proof)

Remarkably, the complexity increases as $K \log K$ in the number of stages $K$ and is proportional to the most complex stage in the system. Also, note that since the rejector class $\mathcal{G}^k$ is typically of lower complexity than the stage classifiers, the overall complexity will be dominated by the VC dimension of stage classifiers $\max_k \text{VC}[\mathcal{H}^k]$. However, if we are provided with “black box” classifiers $d^k$, then the complexity is bounded by $\max_k \text{VC}[\mathcal{G}^k]$. In this case, $\sigma_{d^k}(x^k)$ is simply an affine transformation of the class $\mathcal{G}$ which does not affect its VC dimension. (Sontag, 1998)
3.5.7 Experiments

**Discriminative Myopic Strategy:** For comparison, we again consider a myopic strategy but now in a multi-class setting. This method is closely related to TEFE algorithm to [Liu et al., 2008]. The single stage multi-class classifier with reject option remains the same except that the confidence \( \sigma_{dk}(x^k) \) is thresholded by a constant \( t_k \) to achieve a reject option:

\[
 f^k_{myop}(x^k) = \begin{cases} 
 d^k(x^k), & \sigma_{dk}(x^k) > t_k \\
 \text{reject}, & \sigma_{dk}(x^k) \leq t_k 
\end{cases}
\] (3.64)

The threshold \( t_k \) is chosen such that the \( k \)th stage will reject a constant fraction of the \( N \) examples in the training set. This strategy is completely myopic because \( t_k \) is chosen without considering the performance of stages before or after the current stage. Disadvantage of such strategy is illustrated in Figure 3-10.

**Performance Metric** A natural way to evaluate performance of a sequential decision system is to show the trade-off between system error and average acquisition budget. Recall that our algorithm requires parameters: \( c_1, \ldots, c_K \). \( c_k \) can be thought of as a sensor cost such that the cost of being classified at stage \( k \) is \( \sum_{l=1}^{k} c_l \). To achieve different operating points on the error vs budget curve, we can scale these parameters by a trade-off parameter: \( \alpha c_1, \ldots, \alpha c_K \). For small values of \( \alpha \), measurement costs are small so more examples are rejected down the stages resulting in higher average acquisition budget. For large \( \alpha \), acquisition costs are high resulting in smaller budget. If we sweep \( \alpha \), we generate the error vs budget operating points of our system. For the myopic method, we simply sweep the constant fraction rejected at each stage. In the experiments, we designate a centralized performance as a strategy that uses all sensors for every example. For more implementation details please refer to Appendix
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>State 3</th>
<th>Stage 4</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>synthetic</td>
<td>4,000</td>
<td>Sensor 1</td>
<td>Sensor 2</td>
<td>...</td>
<td>...</td>
<td>2</td>
</tr>
<tr>
<td>mam</td>
<td>830</td>
<td>CAD feat’s</td>
<td>expert rating</td>
<td>...</td>
<td>...</td>
<td>2</td>
</tr>
<tr>
<td>pima</td>
<td>768</td>
<td>weight, age, ..</td>
<td>glucose test</td>
<td>insulin test</td>
<td>...</td>
<td>2</td>
</tr>
<tr>
<td>threat</td>
<td>1230</td>
<td>PMMW img</td>
<td>IR img</td>
<td>AMMW img</td>
<td>...</td>
<td>2</td>
</tr>
<tr>
<td>covertype</td>
<td>581012</td>
<td>soils</td>
<td>wild. areas</td>
<td>elev, aspect, ..</td>
<td>...</td>
<td>7</td>
</tr>
<tr>
<td>letter</td>
<td>20000</td>
<td>pixel counts</td>
<td>moments</td>
<td>edge feat’s</td>
<td>...</td>
<td>26</td>
</tr>
<tr>
<td>mnist</td>
<td>70000</td>
<td>4 x 4 image</td>
<td>7 x 7 image</td>
<td>14 x 14 image</td>
<td>28 x 28 img</td>
<td>10</td>
</tr>
<tr>
<td>landsat</td>
<td>6435</td>
<td>Band 1</td>
<td>Band 2</td>
<td>Band 3</td>
<td>Band 4</td>
<td>7</td>
</tr>
</tbody>
</table>

**Table 3.3:** Dataset Descriptions
Figure 3.10: We display the decision boundaries of our method and the myopic approach for a fixed budget of 1.3. 1st stage classifier, $d^1$, is in blue. 2nd stage classifier, $d^2$, is black. The space that is rejected to 2nd stage is in green. Observe how our method only rejects the area around the first blue boundary. In contrast, myopic uniformly rejects samples around both boundaries even if the samples will be misclassified at the second stage. This is because our strategy anticipates that the 2nd stage classifier cannot really classify examples around the second blue boundary and does not suffer the acquisition cost for those examples. This results in higher error for the same budget for myopic.

Datasets We evaluate performance of our method on several datasets (see Table 1). Since for most datasets measurement cost is not specified, we consider uniformly increasing cost structure. A sample using the 1st stage sensor incurs a cost of 1. To reach the second stage sensor the cost is 2 and so on. So for a four stage system, if a sample passes all four stages, it incurs a cost of 4. To demonstrate the difference in decision regions between our and myopic strategies we use a binary two stage
Figure 3.11: Here, we compare our method to myopic on the MNIST data. We construct four stages of increasing resolution by averaging the original digit images. The experiment demonstrates the advantage of our approach. Also note that the performance of a full resolution sensor can be achieved using a much lower resolution measurement.

synthetic data with two dimensions corresponding to two sensors. (Figure 3.10) For another illustrative example, we convert a popular digit recognition data, MNIST, into a four stage decision system. (Figure 3.11) We designate the full resolution 28x28 pixel image as the last stage. To simulate the first three stages of increasing sensor quality, we average the original image down to three resolution levels, 4x4, 7x7 and 14x14 pixels. The next four datasets are from UCI. (Figure 3.12, 3.13). Landsat data consists of 3x3 pixel neighborhoods taken from a satellite image at four different hyper spectral bands. The objective is to correctly classify the soil type. We set four bands to be the four stages in our system. Covertype deals with classifying forest cover type. We set the first stage to be 10 measurements indicating soil type. The
Figure 3.12: (a - c) illustrate error vs budget trade-off for our method and myopic various dataset. Clearly, our method is superior to myopic and can achieve performance of a centralized classifier (black diamond) with a significantly lower acquisition budget.

Second stage is 4 measurements indicating wilderness area type. The last (3rd) stage consists of 40 measurements such as aspect, elevation, etc. Letter consists of features extracted from handwritten images. The 1st stage are 5 features describing letterbox
Figure 3.13: (a - c) illustrate error vs budget trade-off for our method and myopic various dataset. Clearly, our method is superior to myopic and can achieve performance of a centralized classifier (black diamond) with a significantly lower acquisition budget.

position and pixel counts. The 2nd stage consists of more complex features such as spatial moments. The last stage is most complex consisting of edge information. Pima is a dataset dealing with diabetes diagnoses with specified costs. The 1st stage
consists of 6 simple tests (1 dollar each) such as body mass index, age and etc. Next stage consists of a glucose blood test (17 dollars). The last stage is an insulin test (23 dollars). Threat dataset contains images taken of people wearing various explosives devices. The imaging is done in three modalities: infrared (IR), passive millimeter wave (PMMW), and active millimeter (AMMW). All the images are registered. We extract many patches from the images and use them as our training data. A patch carries a binary label, it either contains a threat or is clean. Since PMMW and IR are the fastest modalities but also least informative, we set them to stages 1 and 2. Stage 3 is an AMMW sensor that requires raster scanning a person and is slow but also the most useful. Overall, simulations demonstrate the advantage of our approach over a myopic strategy. In many datasets, performance close to the centralized (best) strategy can be achieved with much lower average budget. Table 3.4 summarizes our experiments.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Target Error</th>
<th>Myopic</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>synthetic</td>
<td>.147</td>
<td>52%</td>
<td>28%</td>
</tr>
<tr>
<td>pima</td>
<td>.245</td>
<td>41%</td>
<td>15%</td>
</tr>
<tr>
<td>threat</td>
<td>.16</td>
<td>89%</td>
<td>71%</td>
</tr>
<tr>
<td>covertype</td>
<td>.285</td>
<td>79%</td>
<td>40%</td>
</tr>
<tr>
<td>letter</td>
<td>.25</td>
<td>81%</td>
<td>51%</td>
</tr>
<tr>
<td>mnist</td>
<td>.085</td>
<td>90%</td>
<td>52%</td>
</tr>
<tr>
<td>landsat</td>
<td>.17</td>
<td>56%</td>
<td>31%</td>
</tr>
</tbody>
</table>

Table 3.4: In this table we report an average percent of the maximum budget required to achieve the target error rate. The target rate is chosen to be close to the error of the centralized strategy. Thus if there is a maximum of 2 stages and we obtain a value of 28% for our strategy it means that for only 28% of examples a 2nd stage is utilized without any degradation in error.
Chapter 4

Future Work

4.1 Towards Dynamic Sensor Selection

In the last chapter, the assumption was that the order of stages (and corresponding sensors) is fixed. In this chapter, we investigate how to learn a fully dynamic sensor selection policy. If the order of the sensors is no longer fixed then, at each stage, a system has to choose which sensor to acquire next based on previous measurements. We explore how to combine concepts from imitation learning with empirical risk minimization to learn a dynamic sensor selection policy. We introduce an empirical risk based problem that assumes access to an oracle on the training data. Instead of learning to match the oracle directly, we utilize the oracle to compute risks of selecting sensors. Using these risks on the training data, we learn a policy that minimizes a cost-sensitive classification problem.

4.2 Notation

The set-up is the same. Each example $\mathbf{x}$ has $K$ components. $k$th component represent a feature extracted from the $k$th sensor measurement. We will use the words features and measurements interchangeably to refer to a component of $\mathbf{x}$:

$$x_1, x_2, \ldots, x_K$$
Figure 4.1: Schematic of an adaptive sensor selection system. Initially, no measurements are collected and the policy $\pi(s)$ selects the first sensor to collect. Based on the first sensor information, the policy either decides to stop and classify using $F(s)$ or select the next sensor. The procedure is repeated until either all the sensors are exhausted or an example is classified. Note that every example may follow a different sequence of sensors.

For every example, there is an associated label $y \in \{1, \ldots, C\}$. The $K$ measurements and the label are distributed according to an unknown distribution $D$.

$$(x_1, x_2, \ldots, x_K, y) \sim D$$

Costs for acquiring each measurement are:

$$c_1, c_2, \ldots, c_K$$

In order to keep track of which sensor has been acquired for an example, let us introduce a state variable that encodes this information. It consists of three parts.

$$s = \{b, \bar{x}, z\}$$

The first part is a binary vector with $k$th element indicating if feature $k$ is acquired.

$$b_k = \begin{cases} 
1, & x_k \text{ acquired} \\
0, & \text{else} 
\end{cases}$$
Second part represents the actual measurements. The components corresponding to measurements that are not acquired are set to zero.

$$\tilde{x}_k = \begin{cases} x_k, & x_k \text{ acquired} \\ 0, & \text{else} \end{cases}$$

Third part is a special binary element, $z$, that indicates if is stopped and an example is classified.

So a state variable $s_i$ represents an example $x_i$ in a particular state. A similar formulation has been used in [Sergey Karayev, 2013] in dynamic feature selection for image classification.

**Classifier** We can access a classifier $F(s) \in \mathcal{Y}$ that can handle any subset of $K$ features of $x$ captured by $s$. We define the cost of stopping in a state $s$ and classifying using $F(\cdot)$ as:

$$c(s) = \mathbb{E}_{y, \{x_k \mid b_k = 0\}} [F(s) \neq y \mid s]$$

The expectation is over the components of $x$ that have not been acquired: $\{x_k \mid b_k = 0\}$. We assume we can either compute or estimate $c(s)$.

**Policy** A policy $\pi(s) \in \mathcal{A}(s)$ is a function that takes a state, $s$, as an input and returns an action, $a$, in a set $\mathcal{A}(s)$. Here, $s$ encodes features and values that have been acquired up to now.

$\mathcal{A}(s)$ is the set of actions that are allowed to be taken from state $s$. $\mathcal{A}(s)$ consists of all the features that have not been acquired and a stopping action.

$$\mathcal{A}(s) = \begin{cases} \{a \mid b_a = 0\} \cup \text{stop}, & z = 0 \\ \emptyset, & z = 1 \end{cases}$$
Recall that $z$ indicates whether acquisition has been stopped previously. Once acquisition has been stopped no more actions are allowed to be taken from that state.

**Remark:** Note that the state implicitly accounts for the time. This is because at most one sensor is allowed to be requested at every time. Therefore the binary sensor indicator can be summed to produce the time step or stage: $\sum_k b_k$. Consequently, the policy $\pi(s)$ appears to be stationary but indeed is time dependent. Here, time steps and stages will be used interchangeably.

**Trajectory** We introduce a transition function $T(s, a) = s'$ which determines the next transition state $s'$ from a state $s$ after taking an action $a$. In our setting transitions are deterministic and obey the following rules:

$$s' = T(s, a), \quad s' = \{b', \tilde{x}', z'\}, \quad s = \{b, \tilde{x}, z\}$$

(4.1)

$$b'_k = \begin{cases} 1, & a = k \land z = 0 \\ b_k, & \text{else} \end{cases}, \quad \tilde{x}'_k = \begin{cases} x_k, & a = k \land z = 0 \\ \tilde{x}_k, & \text{else} \end{cases}$$

(4.2)

$$k = 1, \ldots, K$$

(4.3)

$$z' = \begin{cases} 1, & a = \text{stop} \land z = 0 \\ z, & \text{else} \end{cases}$$

(4.4)

For an action $a = k$, the new acquired measurement, $x_k$, is generated from the probability distribution conditioned on the observed measurements, $\{x_j \mid b_j = 1\}$.

$$x_k \sim P(x_k, y \mid \{x_j\}_{j|b_j=1})$$
Note that once acquisition is stopped, the variable $z = 1$ and the state no longer changes. Initial state for all examples is the empty state, $s^0_i = s^0$:

$$s^0 = \{ b_k^0 = 0, \bar{x}_k^0 = 0, k = 1, \ldots K, z^0 = 0 \}$$  \hspace{1cm} (4.5)

If we apply a policy $\pi(s)$ to an example $x_i$ in a state, $s^t_i$, at time $t$ then an example transitions to the next state according to the transition rules:

$$s^{t+1}_i = T(s^t_i, \pi(s^t_i))$$

If we apply a policy $\pi(s)$ $K$ times starting from the initial state $s^0_i$ then we generate a $K + 1$ long trajectory for every example $x_i$.

$$\{s^0_i, s^1_i, \ldots, s^K_i\}$$

Please refer to Figure 4·1 for a visualization of an adaptive system.

**Immediate Cost**  For a classifier $F(s)$ and performance estimator $e(s)$, we define an immediate cost of taking an action $a$ in a state $s$

$$C(s, a) = \begin{cases} 
  e(s), & a = \text{stop} \land z = 0 \\
  0, & z = 1 \\
  \alpha c_a, & \text{else}
\end{cases}$$

Here, $\alpha$ is our usual trade-off parameter of cost vs error, and $c_a$ is the cost for the $a$th sensor.
4.3 Risk of Using a Policy

For a policy $\pi(s)$ and a training set the expected risk of using a policy for a sequence of states generated for an example $x$ by applying policy $\pi(s)$, $K$ times is:

$$E_{x,y} \left[ \sum_{t=0}^{K} C(s^{t}, \pi(s^{t})) \right]$$

(4.6)

However, we do not know the distribution of $x$. Instead, we have a training set: \( \{x_i, y_i\}_{i=1}^{N} \). Using this set we can approximate the expected risk with a sample average:

$$\sum_{i=1}^{N} \sum_{k=0}^{K} C(s^{t}_i, \pi(s^{t}_i))$$

(4.7)

We can formulate our objective as to find a policy that minimizes this risk:

$$\min_{\pi \in \Pi} \sum_{i=1}^{N} \sum_{k=0}^{K} C(s^{t}_i, \pi(s^{t}_i))$$

(4.8)

Here, $\Pi$ is a family of policies. Note that states are dependent on the policy, $s^{t+1} = T(s^{t}, \pi(s^{t}))$. So minimizing this sample average directly is difficult due to this interdependency. In order to investigate how to overcome this issue, we will introduce the cost-to-go and approximate policy iteration.

4.4 Cost-to-go and Reinforcement Learning Approach

For a policy $\pi(s)$, we define the cost-to-go of taking an action $a$ from state $s_i$ and following policy $\pi$ thereafter: $R_{\pi}(s, a)$.

$$R_{\pi}(s_i, a) = C(s_i, a) + R_{\pi}(s^{t}_i, \pi(s^{t}_i))$$
Here, \( s'_i = T(s_i, a) \) is induced by taking action \( a \) from state \( s_i \) for an example \( i \). Recall that state transition is deterministic. If an action is taken to acquire a sensor \( a \) then that sensor \( a \) will be acquired.

Using the cost-to-go, we define an alternative empirical risk minimization over a family of policies \( \Pi \):

\[
\min_{\pi \in \Pi} \sum_{i=1}^{N} R_{\pi}(s^0_i, \pi(s^0_i))
\]

One approach to solve this problem is to apply approximate policy iteration. [Bertsekas, 2012]. There is a rich literature on approximate policy iteration, however, here we will describe a simple approach. If we know the cost-to-go with respect to an optimal policy \( R^*(s, a) \), then at any state the optimal policy is simply:

\[
\pi^*(s) = \arg\min_{a \in A(s)} R^*(s, a)
\]

So knowing the optimal policy is equivalent to knowing the optimal cost-to-go. In approximate policy iteration, one parameterizes the cost-to-go as a tractable function of the state \( s \) and action \( a \). For example, let the cost-to-go be a linear function of the state for each action:

\[
\hat{R}_\Theta(s, a) = \theta^T_a s
\]

Note that now, the policy is completely described by the linear weight vectors for each action: \( \Theta = \theta_1, \theta_2, \ldots, \theta_K, \theta_{\text{stop}} \).

Given some initial policy \( \Theta_0 \), a training set \( \{x_i, y_i\} \) and access to a classifier, \( F(s) \), and its confidence estimator, \( e(s) \), the procedure will alternate between these steps:

1. Apply previous policy estimate \( \Theta_j \) to the training set to generate a trajectory

\[
s^0_i, s^1_i, \ldots, s^K_i, \ i = 1, \ldots, N
\]
2. Evaluate cost-to-go

\[ R_{\Theta_j}(s^t_i, a), \ a \in \mathcal{A}(s^t_i), \ i = 1, \ldots, N, \ t = 0, \ldots, K \]

3. Update the policy by solving a least squares problem for each action:

\[ \Theta_{j+1} = \arg \min_{\Theta} \sum_{i=1}^{N} \sum_{k=0}^{K} \sum_{a \in \mathcal{A}(s^t_i)} (R_{\Theta_j}(s^t_i, a) - \theta^T_a s^t_i)^2 \]

Variation of this procedure has been proposed in [Sergey Karayev, 2013] but with an additional retraining of \( F(s) \) after every iteration.

**Issues with this formulation** First, note that the least squares problem is coupled as the evaluation of cost-to-go depends on the parameter \( \Theta \). For this reason, the algorithm needs to iterate between evaluation and update step. In addition, the reinforcement learning approach attempts to learn the cost-to-go function while only the policy itself is necessary to make decisions.

To overcome both of these issues we introduce imitation learning.

### 4.5 Imitation Learning

Imitation learning is a research area popular in the robotics community. [Ross and Bagnell, 2010] [Ross et al., 2010]. In robotics, the dynamics may be so complicated that standard control approaches fail. A typical set-up is learning a controller to steer a robot. A robot has a forward oriented camera. The goal is to learn what action to take (i.e. which way to steer) given the image of what robot sees.

Imitation learning assumes we have access to an oracle policy \( \pi^* \). We discuss the issues that arise in defining an oracle for our sequential sensor selection setting in Section 4.7. However, in the robot example, an oracle is a human driving a robot. The
problem is to learn to imitate the oracle over all the states the robot may encounter. So our problem is simplified since we are simply trying to match the oracle,

$$\min_{\pi \in \Pi} \sum_{s \in S} 1[\pi(s) \neq \pi^*(s)]$$

(4.12)

Note that the cost-to-go of taking a particular action from each stage is encoded into the oracle.

The question is how to collect a representative set of states $S$ over which to train our system. Let us consider a naive approach. We have training data. Let us simply apply our oracle to generate trajectories for every training example and aggregate them into a sample $S$. Next, let us solve the problem above over this sample $S$ to produce a policy $\pi$. However, this strategy will fail because the oracle may be very good, and $\pi$ is not able to follow the oracle exactly and will make mistakes. This implies that the system may encounter states that were never visited during training. On these unknown states, $\pi$ will take incorrect actions and will further diverge from the right path. The system needs to learn how to correct its mistakes.

To overcome this issue, one can use DAGGER (data aggregation)-like algorithms (Ross et al., 2010). First, apply $\pi^*$ to an initial sample of states $S^0$ and generate $S^1$. Learn $\pi^1$ to imitate $\pi^*$ over the sample $S^1$.

$$\pi^1 = \arg\min_{\pi \in \Pi} \sum_{s \in S^1} 1[\pi(s) \neq \pi^*(s)]$$

Then generate $S^2$ by applying $\pi^1$ and learn $\pi^2$ over the union of the two $S^1 \cup S^2$ and so on.

$$\pi^k = \arg\min_{\pi \in \Pi} \sum_{s \in S^1 \cup S^2 \cup \ldots \cup S^k} 1[\pi(s) \neq \pi^*(s)]$$

This procedure is repeated for a large number of iterations $J$. At the end, the policy among $\pi^1, \pi^2, \ldots, \pi^J$ that best imitates the oracle over a separate cross-validation set
The advantage of such strategy is that the policy learns to imitate the oracle on the states that the policy itself generates.

However, the disadvantage of DAGGER is the need to iterate between learning $\pi^j$ over $\{S^0 \cup S^1 \cup \ldots \cup S^j\}$, and generating a new sample of states $S^{j+1}$. Also, learning over the union of visited states may become computationally difficult. Nevertheless, DAGGER has been applied in learning systems to reduce feature cost [He et al., 2012].

4.5.1 Learning a policy for every stage

One way to overcome the issue of learning of a large union of states is the so called forward training algorithm [Ross and Bagnell, 2010]. Since in our setting we only have $K$ measurements, the system will only run for at most $K$ time steps or stages. So let us explicitly learn $K$ policies $\pi_1, \pi_2, \ldots, \pi_K$ one for every time step instead of learning a single policy to be used at every step.

Given the $K$ policies, the evaluation process will be:

$$\pi_1(s_i^0) \rightarrow s_i^1, \quad \pi_2(s_i^1) \rightarrow s_i^2, \quad \ldots \quad \pi(s_i^K)$$

The algorithm proceeds as follows. We use the oracle $\pi^*$ only to evaluate the correct action for a particular sample of states $S$. Also note that a policy $\pi_i$ is applied only at the $t$th time step.

Initially at time 0, $S^0$ is a collection of empty states. First, we will learn $\pi_1$ over $S^0$. Next, we generate $S^1$ by applying $\pi_1$. Note that the sample of states produced by the oracle $\pi^*$ and an approximate policy $\pi^1$ need not (and most likely will not be) the same. Now, $\pi_2$ is trained over the sample $S^1$ generated by the policy $\pi_1$ at the previous time. $S^2$ is generated by $\pi_2$ and so on. The algorithm is described below:

---

1Some variations of this algorithm will randomly evaluate the oracle instead of the policy estimate part of the time.
Algorithm 5 Forward training imitation learning

Input: \( S^0 = \{ s^0_i \}_{i=1}^N \) \{Initial empty sample of states\}, \( \pi^*(s) \) \{oracle policy\}

for \( t = 0, 1, 2, \ldots K - 1 \) do

\[
\pi_{t+1} \leftarrow \min_{\pi \in \Pi} \sum_{i=1}^{N} 1[\pi(s^t_i) \neq \pi^*(s^t_i)]
\]

\( \{s^{t+1}_i\}_{i=1}^N \) \( \leftarrow \) evaluate \( \pi_{t+1} \) on \( \{s^t_i\}_{i=1}^N \)

end for

Output: \( \pi_1, \pi_2, \ldots, \pi_K \)

There are several advantages of such an algorithm. First, we only have to imitate an oracle \( K \) times, and at each time, training is over a training set of the same size \( N \). Another advantage is that \( K \) policies allow us to imitate a more complicated oracle more so than a single policy in DAGGER-like algorithms.

4.6 Novel Risk Based Formulation

However, one disadvantage of the forward training algorithm is the strict objective to exactly match the oracle. The oracle policy may be too complex to imitate exactly. So instead of using the expert to generate the correct actions (labels), we can use the expert \( \pi^* \) to compute the optimal costs-to-go, \( R_*(s, a) \), for taking a particular action at a state. In fact, knowing the oracle cost-to-go or the oracle policy is equivalent as one can be derived from the other.

Recall that we have access to training data with full measurements, \( \{x_i, y_i\}_{i=1}^N \), and a particular state \( s_i \) corresponds to measurements acquired for a training point \( x_i \) as indicated by non-zero entries of \( b_i \).

For simplicity, assume we have access to a collection of such states, \( S \). We define
a problem to minimize the cost-to-go over this set of states:

\[
\min_{\pi \in \Pi} \sum_{s \in S} R_*(s, \pi(s))
\]  

(4.13)

This formulation relaxes the demand on the estimated policy \( \pi \) to exactly imitate the expert. For example, assume for a state \( s \), two actions have similar costs: \( R_*(s, a_1) \approx R_*(s, a_2) \) but the minimum cost action is \( a_1 = \arg \min_{a'} R_*(s, a') \). In our formulation, the penalty for misclassifying \( \pi(s) \neq a_1 \) and \( \pi(s) \neq a_2 \) will also be approximately the same. This does not force \( \pi \) to follow the oracle exactly and allows more freedom.

Next, we describe how to reduce a cost-to-go problem to a more familiar supervised learning problem.

4.6.1 Reduction to Weighted Classification

We can rewrite the cost-to-go problem as a cost-sensitive supervised learning problem. And with the help of some recent results in this field (Zadrozny et al., 2003; Blatt and Hero, 2005; Trapeznikov et al., 2013; Wang and Saligrama, 2012), we can further
reduce to a weighted supervised learning problem.

\[
\arg \min_{\pi \in \Pi} \sum_{s \in S} R_s(s, \pi(s)) \tag{4.14}
\]

\[
= \arg \min_{\pi \in \Pi} \sum_{s \in S} \sum_{a \in A(s)} R_s(s, a) 1_{[\pi(s) = a]} \tag{4.15}
\]

\[
= \arg \min_{\pi \in \Pi} \sum_{s \in S} \sum_{a \in A(s)} R_s(s, a) \left[ 1 - 1_{[\pi(s) \neq a]} \right] \tag{4.16}
\]

\[
= \arg \min_{\pi \in \Pi} \sum_{s \in S} \sum_{a \in A(s)} \left[ \sum_{a \in A(s)} R_s(s, a) + \sum_{a \in A(s)} \left[ \max_{a' \in A(s)} R_s(s, a') - R_s(s, a) \right] 1_{[\pi(s) \neq a]} \right] \tag{4.17}
\]

\[
= \arg \min_{\pi \in \Pi} \sum_{s \in S} \sum_{a \in A(s)} \left[ \sum_{a \in A(s)} R_s(s, a) - \max_{a' \in A(s)} R_s(s, a') 1_{[\pi(s) \neq a]} \right] \tag{4.18}
\]

The second term may be dropped because it is constant and is no longer a function of \( \pi(s) \). This is due to:

\[
\sum_{a \in A(s)} 1_{[\pi(s) \neq a]} = |A(s)| - 1
\]

The result is a a weighted classification problem. If we define the weights as,

\[
w(s, a) = \max_{a' \in A(s)} R_s(s, a') - R_s(s, a)
\]

the weighted learning problem becomes:

\[
\arg \min_{\pi \in \Pi} \sum_{s \in S} \sum_{a \in A(s)} w(s, a) 1_{[\pi(s) \neq a]} \tag{4.19}
\]

Here, the two summations are simply iterating over an expanded data set. For each state instance in \( S \), we duplicate it \( |A(s)| \) times. This formulation is much easier to handle as many existing algorithms can now be utilized.
4.6.2 Risk based forward imitation learning algorithm

Using the reduction to supervised learning, we modify the forward training algorithm to solve a cost-sensitive problem instead of minimizing oracle mismatch.

The main difference is that we use the oracle policy to generate costs-to-go on the set of states generated by policy from the previous time. At time step 1, we compute the misclassification weights using the oracle and learn $\pi_1$ with respect to these weights. Next, we evaluate $\pi^1$ to generate the next set of states $S^1$ and again use the oracle to compute the weights. This is repeated to learn $K$ policies. The algorithm is described in Algorithm 6.

**Algorithm 6** Risk based imitation learning

Input: $\{x_i, y_i\}_{i=1}^N$ \{training data with full measurements\}, $R_* (s, a)$ \{oracle cost-to-go\}

$s_{0;i}=1 \leftarrow \text{empty state}$

for $t = 0, 1, 2 \ldots K - 1$ do

\[
w(s, a) = \max_{a' \in A(s)} R_* (s, a') - R_* (s, a)
\]

\[
\pi_{t+1} \leftarrow \min_{\pi \in \Pi} \sum_{i=1}^N \sum_{a \in A(s_i)} w(s^t_i, a) 1[\pi(s_i^t) \neq a]
\]

$\{s^{t+1}_{i,1}\}_{i=1}^N \leftarrow \text{evaluate } \pi_t \text{ on } \{s^t_i\}_{i=1}^N$

end for

Output: $\pi_1, \pi_2, \ldots, \pi_K$

4.7 Common Issues

So far, all the described methods including ours, have several major difficulties.

**Classifier** In all the discussions so far in this section, we assumed we have access to a classifier $F(s)$ and some measure of confidence $c(s)$. This classifier has to handle
missing measurements in \( \mathbf{x} \) because as the example traverses the system more and more measurements are being added. There is some work on dealing with missing or corrupted features. In [Laurens van der Maaten and Weinberger., 2013], authors propose to include empirical moments of the training data while learning the classifier. However, most common method is to duplicate the original training by randomly omitting features and training a classifier on this expanded data. Ideally, we would like to train a separate \( F(s) \) for every unique subset of sensors but this is prohibitive computationally.

**Oracle** The oracle does not have to be the optimal policy that can be applied to any example. The oracle only needs to operate on the training data and it has to be relatively fast to evaluate.

One example of an oracle is a linear classifier. \( F(s) = \text{sgn}(\sum_{i|b_i=1} w_i x_i) \). We only combine features that are available (indicator by a vector \( \mathbf{b} \)). The margin, \( y_i \sum_{i|b_i=1} w_i x_i \), captures confidence of classification. Again, we are considering a binary case, \( y \in \{+1, -1\} \). If the margin is a large positive number then the confidence is high. We can form an integer programming problem. For a particular state \( \{\mathbf{b}, \tilde{x}, z\} \)

\[
\max_{\gamma_i|b_i=0} \sum_{i|b_i=0} \gamma_i w_i x_i
\]

\[
s.t. \sum_{i|b_i=0} \gamma_i c_i + \sum_{i|b_i=1} c_i \leq B_0 \tag{4.22}
\]

This problem is known as a 0/1 knapsack and returns the subset of unused sensors that maximize the margin subject to the budget constrain \( B_0 \). However, this problem has to be solved for every training example and for every state that may be encountered during training. This can become prohibitively slow during training and only works for linear classifiers.\footnote{Since \( w_i \) may be negative, we can add an offset to all the terms and still retain the right objective}
Parameterization  The policy suffers from the same problem of missing features as $F(s)$. Here, we explicitly zero out the components of $x$ that have not been acquired yet. But whether this is a good approach is still unclear.

4.8 Simulation

In this section, we present some initial results on using our novel risk based imitation learning.

To simplify the problem, we generate a synthetic data set for our simulation. This enables us an access to an oracle and a classifier $F(s)$.

![Figure 4-2: Simulated Example](image)

Generated Data  Each example consists of 3 sensor measurements, and each measurement is a Gaussian random variable with a different variance. Examples can
belong to one of three groups, each group has a different permutation of the variances. For group 1, dimension 1 has the lowest variance, dimension 2 has the medium variance and dimension 3 had the highest variance. For group 2, the order for low, medium, high is 3rd, 2nd, 1st dimensions. For group 3, the order is 2,1,3. Classification performance depends on which measurements are acquired, the best being when all three are available.

We generate 1000 data points according to the following model. We use half for training and the other half for testing.

- Generate label \( y \sim \text{Uni}\{+1,-1\} \)
- Uniformly generate a group \( j \sim \text{Uni}\{1,2,3\} \)
- Generate three measurements:
  \[
  \begin{align*}
  x_1 &= y + n_{1,j} + t_j \\
  x_2 &= y + n_{2,j} + t_j \\
  x_3 &= y + n_{3,j} + t_j 
  \end{align*}
  \]
- \( n_{k,j} \sim \mathcal{N}(0,\sigma(k,j)) \)
- \( \sigma(k,j) \) is the noise variance for measurement \( k \) for group \( j \)
- \( t_j \) is a deterministic offset for group \( j \)

The optimal classifier for any example \( \mathbf{x} \) in any state \( s \) can be computed in closed form. Note that this requires the knowledge of the offset \( t_j \) and the group \( j \) from which \( \mathbf{x} \) was generated. We assume that this information is available at all times.
\[ F(s) = \begin{cases} +1, & \sum_{k|b_k=1} \frac{x_k}{\sigma(k,j)} \geq \frac{t_j}{2} \sum_{k|b_k=1} \frac{1}{\sigma(k,j)} \\ -1, & \sum_{k|b_k=1} \frac{x_k}{\sigma(k,j)} < \frac{t_j}{2} \sum_{k|b_k=1} \frac{1}{\sigma(k,j)} \end{cases} \] (4.23)

And the error confidence can also be computed \( e(s) \) using the erfc function:

\[ e(s) = \frac{1}{2} \text{erfc} \left( \frac{1}{2} \frac{\sum_{k|b_k=1} \frac{1}{\sigma(k,j)}}{\sqrt{\sum_{k|b_k=1} \frac{1}{\sigma(k,j)}}} \right) \] (4.24)

We assume the costs for each stage are uniform: \( c_1 = c_2 = c_3 = 1 \). And since there are only 3 stages, we can exactly compute the oracle cost-to-go, \( R^*(s, a) \) for any example \( x_i \) in any state by enumeration.

Our algorithm should learn which measurement to choose next in order to minimize this oracle cost-to-go.

**Comparison** We use closed form expressions for \( F(s) \), \( e(s) \) and concentrate on learning only the policy \( \pi(s) \).

We compare three methods:

- Our novel risk based imitation learning Algorithm 6 (SL Imitation) in Figure 4.3. \( \pi(s) \) is parameterized by a 2nd order polynomial kernel: \( \pi_k(s) = \sum_{i=1}^{N} q_i (s^T s_i^k + 1)^2 + b \), and a logistic loss is used to upper-bound the indicator loss.

- The oracle which exhaustively evaluates \( R^*(s, a) \) and picks the lowest cost action

- A fixed order strategy is a strategy such that every example will acquire measurements in the same order. First, we fix 1st sensor for which the error is
Figure 4.3: Three approaches are compared: our novel risk-based imitation learning, fixed order strategy and an oracle minimized. Next, fix the 2nd sensor such that the error is the smallest. And since there are only three sensors, the last sensor is already determined. This order is used in evaluating every example during testing.

Note that trying direct forward selection imitation learning (Algorithm 5) did not produce meaningful results.

For the simulation, the trade-off parameter $\alpha$ is swept to generate an error vs cost curves for each method. Our approach performs relatively close to the oracle and much better than a fixed order method. So extending reinforcement learning into a cost sensitive formulation appears to be a promising direction to take.
Chapter 5

Conclusion

In this work, we addressed two areas of cost in supervised learning: cost during training and cost during testing.

Our work on Active Learning extends the version space approach in the framework of boosted classifiers. The weak learning assumption guarantees the existence of version space and compact parameterization of boosted classifiers enables an efficient way to find examples that reduce this set of feasible classifiers. However, the large classification complexity of boosted classifiers also has drawbacks. In turns out that reducing version space at an exponential rate does not necessarily guarantee that the classifier learnt on the labeled examples has good generalization ability. In fact, we prove that boosting is so powerful that there always exist two classifiers sampled from any version space that disagree on every single unlabeled example. The reduction of version space only guarantees reduction in error when the target classifier is assumed to be sparse in the number of weak learners used. And this requires to carry out our procedure on all the sparse subsets of the version space, a combinatorially hard problem. Nevertheless, our bisecting procedure on the full version space can be viewed as a convex relaxation of the ideal strategy. And experiments demonstrate good performance in comparison to margin based active learning approaches, especially when initialization bias is an issue.

In the problem on reducing measurement cost during testing, we considered a scenario of fixed order of stages with a sensor associated with each stage. We introduced
a framework for multi-stage reject classifiers where at each stage a decision whether
to continue or stop and classify using current information is learn from training data.
We employ a decomposition in the setting of known probability models, to construct
a multi-stage empirical risk, and derive suitable parameterization to minimize this
risk in both binary and multi-class settings. Also, for each parameterization, gener-
alization guarantees are provided. Experiments on many datasets demonstrate the
advantage of our approach to a myopic strategy, a strategy. Our sequential deci-
sions system achieves performance close to the best (centralized) classifiers by only
requesting a fraction of the sensors.

Lastly, we explore dynamic sensor selection and illustrate a novel extension of
imitation learning to risk based formulation.

5.1 Future Directions

Dynamic sensor selection is the most promising direction to take for future research.
There are many issues that need to be resolved but there are also a lot of promising
practical algorithms and theoretical results from the imitation learning field. Some
interesting directions:

- Incorporate the uncertainty in the ground truth in order to construct a proper
  POMDP formulation.

- Training classifiers that can handle missing features but still achieve good per-
  formance.

- Designing oracle policies that are computationally efficient but also provide
  useful ground truth for the trained policy to imitate

- More efficient algorithms to solve cost-sensitive problems as reduction to regular
  supervised learning expands the data set making training much slower.
Appendix A

ActBoost

A.1 Connection between Query By Boosting and SVM Active Learning

In this section, we will provide intuition on the QBB algorithm from [Abe and Mamitsuka, 1998] by extending theory behind the version space SVM algorithm from [Tong and Koller, 2001]. As in our problem, each example is characterized by a vector $h(x)$. A hard margin SVM supervised learner finds a hyperplane that correctly separates a labeled set $L$: (we will assume zero offset)

$$q_{svm} = \arg \max_{q,v} v$$

$$s.t. \ y_{i} h^{T}(x_{i}) q \geq v \ \forall i \in L$$

$$||q||_{2} = 1$$

A margin $h^{T}(x)q$ is the distance from a vector $h(x)$ to the hyperplane with a unit-norm normal vector $q$, and the above program finds a hyperplane $q_{svm}$ that maximizes the minimum margin. An SVM active learning algorithm will label an example closest to this hyperplane: $x^{*} = \arg \min_{x \in U^{t}} |q_{svm}^{T} h(x)|$.

If we think of this problem in the space of possible hyperplane normals $q$, then the version space will lie on the surface of a unit sphere $||q|| = 1$. Each labeled example imposes a hyperplane with a normal vector $y h(x)$ cutting through the sphere. These constraints limit the version space to a section on the sphere surface. In Figure A.1
if we flatten the surface of the unit sphere on to the plane of the page then \( q_{svm} \) becomes the center of the largest diameter ball (circle in Figure A·1) inscribed in the polyhedron formed by linear constraints (thin lines in Figure A·1). If a ball is a good approximation to the polyhedron then the \( q_{svm} \) is a good estimate of the version space center. If we choose to label an example whose hyperplane (dashed line in Figure A·1) is closest to this center then it is likely to halve the version.

A similar logic can be applied to the QBB algorithm. The algorithm relies on the solution of the Adaboost supervised learner which approximates the following linear program: 

\[
q_a = \arg \max_{q \in Q} \min_{y_i} y_i h(x_i) \text{ where } Q = \{q | 1^T q = 1, q \geq 0\}.
\]

And QBB labels an example with the smallest margin: 

\[
x^* = \arg \min_{x \in U^T} |q_a^T h(x)|.
\]

The program can be transformed to mimic the SVM problem:

\[
q_a = \arg \max_{q,v} v||q||_2
\]

s.t. \( y_i h^T(x_i) \frac{q}{||q||_2} \geq v \forall i \in L \)

\[
1^T q = 1, q \geq 0
\]

Note that the differences from the SVM. The version space is constrained to lie on a probability simplex, and the program, in addition to maximizing the minimum margin, also tends to maximize the magnitude of \( q \). On a simplex, \( ||q||_2 \) is minimum at the center of the simplex, increases as we move towards the edges and is maximum at the vertices. So \( q_a \) is skewed center of a largest radius ball inscribed in polyhedron imposed by the labeled examples. The solution drifts away from the approximate center towards the vertices, favoring sparsity. So how well Ada-boost approximates the center of the version, depends on the geometry of the problem. If the version space happens to be near the center of the probability simplex then the approximation may be adequate. If the region is closer to the boundary of the simplex then the performance of QBB will deteriorate as the center estimate will not be valid. The
Figure A.1: Version Space for the two algorithms: solid lines represent hyperplanes imposed by labeled examples, dotted line is the example to be labeled, the circle is the largest radius ball inscribed in the linear constraints: (a) SVM Active Learning: the plane of the page is the surface of the unit-norm sphere $\|q\|_2 = 1$; and (b) QBB Active Learning: dashed triangle is the probability simplex, note how the solution $q_a$ is shifted towards a vertex away from the center.
example in Figure A.1 illustrates the bias towards the vertices. The hyperplane of the example to be labeled (dotted line) does not adequately halve the version space.

### A.2 More Details On Boosting

**Linear Programming Boosting** Boosting can be formulated as a linear program in [Demiriz et al., 2002]. The cost surrogate is a hinge loss function: \( C(z) = \max\{-z, 0\} \). Optimal weight vector on the weak hypothesis is the solution to the following LP. To simplify notation \( m_i = y_i h_i \).

\[
q_{LP} = \arg \max_{q,s,v} v - \sum_{i=1}^{M} s_i \tag{A.1}
\]

\[
m_i^T q \geq v - s_i, i = 1, \ldots, M \tag{A.2}
\]

\[
\sum_{j=1}^{N} q_j = 1 \tag{A.3}
\]

\[
q \geq 0, s \geq 0 \tag{A.4}
\]

First term in the cost \( v \) can be thought of as the minimum margin over the examples. The algorithm will maximize the minimum margin while minimizing the number of misclassified examples. This error is the second term in the cost: sum of the slack variables \( s_i \).

Note that due to the \( l_1 \) constraint, the solution tends to be sparse. The same is true for Ada-boost and other coordinate descent methods which we’ll summarize next.

**Weakness of the Weak Learner Assumption** As long as \( C(z) \) is a convex function which decreases to zero as \( z \) increases to positive infinity, coordinate descent boosting will drive training error to zero. \( \lim_{z \to \infty} C(z) = 0 \) This is the result of
Algorithm 7 Convex Cost Coordinate Descent Boosting

INPUT: \{\(x_i, y_i\)\}_{i=1}^{M} \{training set\}, \{\(h_j\)\}_{j=1}^{N} \{set of weak hypothesis\}, \(T\) \{number of iterations\}, \(C(z)\) \{cost surrogate for the indicator function, must be twice differentiable\}

\[ t \leftarrow 0, \quad q^0 \leftarrow 0, \quad m_{ij} \leftarrow 1_{\{h_j(x_i)=y_i\}}, \quad W(z) = -\frac{\partial C(z)}{\partial z}, \quad W'(z) = \frac{\partial W(z)}{\partial z} \]

\textbf{while} \( t \leq T \) \textbf{do}

\[ d_j \leftarrow \sum_{i=1}^{M} W(m_i^T q^t) m_{ij} \{\text{Compute descent directions}\} \]

\[ j^t \leftarrow \max_{j=1...N} d_j \{\text{Pick direction of maximum descent}\} \]

\textbf{if} \( d_{j^t} \leq 0 \) \textbf{then}

\[ \text{Terminate} \{\text{Check if can descent anymore}\} \]

\textbf{end if}

\[ g(\alpha) = \sum_{i=1}^{M} W(m_i^T q^t + \alpha m_{ij^t}) m_{ij^t} \]

\[ g'(\alpha) = \sum_{i=1}^{M} W'(m_i^T q^t + \alpha m_{ij^t}) m_{ij^t} \]

\[ \alpha^t \leftarrow \text{NEWTON}(g(\alpha), g'(\alpha)) \{\text{Compute step size using Newton’s Method}\} \]

\[ q_{j^t}^{t+1} \leftarrow q_{j^t}^t + \alpha^t \{\text{Update}\} \]

\[ t \leftarrow t + 1 \]

\textbf{end while}

OUTPUT: \( q_C \) \{weak hypothesis weights\}

the weak learner assumption guaranteeing a positive descent direction. However, if the weak learner assumption also holds for the training set with random label noise then coordinate descent boosting will overtrain on the noisy labeled examples. The generalization will suffer.

A.3 Proof of Theorem 2.1

\textit{Proof.} The weak learner assumption implies that for \( x_k \in U^t \)

\[ \exists q \geq 0 : y_k h(x_k)^T q > 0 \text{ and } y_i h(x_i)^T q > 0 \forall x_i \in L^t \quad (A.5) \]

Without loss of generality assume that \( y_k = -1 \). This implies that

\[ A = \{ q \geq 0, q \neq 0 | -h(x_k)^T q > 0 \text{ and } y_i h(x_i)^T q > 0 \forall x_i \in L^t \} \neq \emptyset \quad (A.6) \]
We are left to determine whether, there is a \( q \geq 0 \) such that, \( h(x_k)^T q > 0 \) and \( y_i h(x_i)^T q > 0 \ \forall x_i \in L^t \). Suppose there is no such \( q \), then we have that

\[
\n\n\]

\[ \forall q \geq 0 : h(x_k)^T q > 0 \text{ and } y_i h(x_i)^T q > 0 \ \forall x_i \in L^t \tag{A.7} \]

By assumption \( \mathcal{H} \) is negation complete that is \( \exists j, j^* : h_j(x) = -h_{j^*}(x) \). We can rewrite \( h(x) = \begin{bmatrix} h^{(1)}(x) \\ h^{(2)}(x) \end{bmatrix} \) s.t. \( h^{(1)}(x) = -h^{(2)}(x) \). If we define \( q = \begin{bmatrix} q^{(1)} \\ q^{(2)} \end{bmatrix} \) then we can rewrite the inner product as

\[
\]

\[ h(x)^T q = h^{(1)}(x)^T q^{(1)} + h^{(2)}(x)^T q^{(2)} = h^{(1)}(q^{(1)} - q^{(2)}) \]

If define vector \( \tilde{q} = \begin{bmatrix} q^{(1)} - q^{(2)} \\ q^{(2)} - q^{(1)} \end{bmatrix} \) where \( \tilde{q}_j = q_j - q_{j^*} \) then we can simplify the expression in [A.7] to:

\[
\n\]

\[ \forall \tilde{q} : h(x_k)^T \tilde{q} > 0 \text{ and } y_i h(x_i)^T \tilde{q} > 0 \ \forall x_i \in L^t \tag{A.8} \]

Note \( \tilde{q} \) is now allowed to be negative. This means that as \( \tilde{q}_i \) ranges over all the real numbers, the vector \( (h(x_k)^T \tilde{q}, y_1 h(x_1)^T \tilde{q}, \ldots, y_l h(x_l)^T \tilde{q}) \) does not intersect the first quadrant. In addition the complement of this set contains \( A \), which is convex and non-empty. Consequently, we can invoke the separating hyperplane theorem that separates the first quadrant from all the feasible vectors \( (h(x_k)^T \tilde{q}, y_1 h(x_1)^T \tilde{q}, \ldots, y_l h(x_l)^T \tilde{q}) \) as \( \tilde{q}_i, \forall i \) ranges over all real numbers. As a consequence we have hyperplane \( \lambda \geq 0 \) and \( \delta > 0 \) such that,

\[
\n\]

\[ \exists \lambda, \delta \geq 0 : \delta h(x_k)^T \tilde{q} + \sum_{i \in L^t} \lambda_i y_i h(x_i)^T \tilde{q} \leq 0 \ \forall \tilde{q} \tag{A.9} \]

\[
\n\]

\[ \exists \lambda, \delta \geq 0 : [\delta h(x_k)^T + \sum_{i \in L^t} \lambda_i y_i h(x_i)^T] \tilde{q} \leq 0 \ \forall \tilde{q} \tag{A.10} \]

\[ \implies \delta h(x_k) + \sum_{i \in L^t} \lambda_i y_i h(x_i) = 0 \tag{A.11} \]

Note that \( \lambda \) or \( \delta \) cannot be all zeros. For \( \delta \neq 0 \), equality in [A.11] implies that \( h(x_k) \) has to lie in the cone of \( y_i h(x_i) \)'s. \( h(x) \) is a vertex of \(+1, -1\) hypercube in \( N \) dimensions. A vertex \( h(x_k) \) of this hypercube lies in the cone of other vertices \( \{h(x_i)\}_{i \in L^t} \) if and only if \( k \in L^t \).

For \( \delta = 0 \), the equality in [A.11] cannot hold for \( \{y_i h(x_i)\}_{i \in L^t} \) that satisfy the weak
A.4 Proof of Lemma 2.2

Proof. We provide the main outline of the proof and skip some of the messy algebra. For simpler notation, let \( q(x) = \text{sgn}(\sum_{j=1}^{L} q_j h_j(x) - .5) \) where \( h_j(x) \in \{0, 1\} \). We emphasize that the weak learners map to zero or one. Any two samples \( x, x' \) are \( \delta \)-neighborly if:

\[
\frac{1}{2} \int_Q |q(x) - q(x')| \, dq \leq \delta \quad (A.12)
\]

The integral is the volume where \( q(x) \) and \( q(x') \) disagree:

\[
\int_Q 1_{[q(x) \neq q(x')]} \, dq \leq 2\delta \quad (A.13)
\]

Let \( S = \{ j \mid h_j(x) = h_j(x') \} \) and \( S^c = \{ j \mid h_j(x) \neq h_j(x') \} \):

\[
q(x) = \text{sgn}(\sum_{j \in S} q_j h_j(x) + \sum_{j \in S^c} q_j h_j(x) - .5) \quad (A.14)
\]

\[
q(x') = \text{sgn}(\sum_{j \in S} q_j h_j(x) + \sum_{j \in S^c} q_j h_j(x') - .5) \quad (A.15)
\]

Let \( S_1 = \{ j \mid h_j(x) = 1 \} \cap S^c \) and \( S_2 = \{ j \mid h_j(x') = 1 \} \cap S^c \) then

\[
q(x) = \text{sgn}(\sum_{j \in S} q_j h_j(x) + \sum_{j \in S_1} q_j - .5) \quad (A.16)
\]

\[
q(x') = \text{sgn}(\sum_{j \in S} q_j h_j(x) + \sum_{j \in S_2} q_j - .5) \quad (A.17)
\]
And \( q(x) \neq q(x') \) if and only if

\[
\sum_{j \in S} q_j h_j(x) < 0.5 \quad \text{and} \quad \sum_{j \in S_1} q_j > 0.5 - \sum_{j \in S} q_j h_j(x) \quad \text{and} \quad \sum_{j \in S_2} q_j < 0.5 - \sum_{j \in S} q_j h_j(x)
\]  

(A.18)

By the \( K \)-neighbor assumption: \( |S_1 \cup S_2| \leq K \). Let \( |S_1| = K - k_1 \) and \( |S_2| = k_1 \) and:

\[
\tilde{Q}(k_1) = \{ q \in Q \mid \sum_{j \in S} q_j h_j(x) < 0.5, \sum_{j \in S_1} q_j > 0.5 - \sum_{j \in S} q_j h_j(x), \sum_{j \in S_2} q_j < 0.5 - \sum_{j \in S} q_j h_j(x) \} 
\]  

(A.19)

(A.20)

It is easy to check that the case where \( |S_2| = 0 \) and \( |S_1| = K \) will have the greatest volume:

\[
Vol(\tilde{Q}(k_1)) \leq Vol(\tilde{Q}(0)) \quad \text{for} \quad 0 < k_1 \leq K
\]  

(A.21)

So let,

\[
\tilde{Q}(0) = \{ q \in Q, \sum_{j \in S_1} q_j > 0.5 - \sum_{j \in S} q_j h_j(x), \sum_{j \in S_2} q_j h_j(x) < 0.5 \}
\]  

(A.22)

\( Vol(\tilde{Q}(0)) \) is an upper bound for (A.13).

To compute the volume we recast the problem in terms of probabilities. Note that since the simplex \( Q \) is endowed with the Lebesgue measure we can think of \( q \) as a random variable uniformly distributed over \( Q \). However, the components of \( q \) are now dependent. To transform the problem into an independent set of random variables we consider exponentially distributed random variables.

Define the unnormalized IID random variable, \( q'_j \), where \( q'_j \) are IID exponentially distributed random variables with mean equal to \( \theta \). Then \( E[\sum_{j=1}^{N} q'_j] = \frac{N}{\theta} \). It is well known that such an exponentially distributed set of random variables when normalized exactly produces a uniform distribution over the simplex:

\[
q_j = \frac{q'_j}{\sum_{k=1}^{N} q'_k} \quad \Rightarrow \quad q'_j = q_j \sum_{k=1}^{N} q'_k
\]
By substitution of the unnormalized random variables we obtain,

\[
Pr\{\bar{Q}(0)\} = Pr\{q \in Q, \sum_{j \in S_1} q_j > .5 - \sum_{j \in S} q_j h_j(x), \sum_{j \in S} q_j h_j(x) < .5\} = Pr\left\{\sum_{j \in S_1} q_j' > .5\left(\sum_{j=1}^{N} q_j'\right) - \sum_{j \in S} q_j' h_j(x), \sum_{j \in S} q_j' h_j(x) < .5\left(\sum_{j=1}^{N} q_j'\right)\right\}
\]

To simplify this expression we consider the event,

\[
A = \left\{\left|\frac{1}{\theta} - \frac{1}{N} \sum_{j=1}^{N} q_j' \right| \leq \epsilon_2 \right\} \quad (A.23)
\]

Note that the event \(A\) can be cast in the familiar form of an empirical average being close to its empirical mean. Consequently, we expect that the probability of the complement, \(A^c\), of the event \(A\) is exponentially small in \(N\). We now proceed as follows:

\[
Pr\{\bar{Q}(0)\} \leq Pr\left\{\sum_{j \in S_1} q_j' > .5\left(\sum_{j=1}^{N} q_j'\right) - \sum_{j \in S} q_j' h_j(x), \sum_{j \in S} q_j' h_j(x) < .5\left(\sum_{j=1}^{N} q_j'\right), q_j' \in A\right\} \quad (A.24)
\]

\[
+ Pr(A^c) \quad (A.25)
\]

\[
\leq Pr\left\{\sum_{j \in S_1} q_j' > .5 \frac{N}{\theta}(1 - \epsilon_2) - \sum_{j \in S} q_j' h_j(x), \sum_{j \in S} q_j' h_j(x) < .5 \frac{N}{\theta}(1 + \epsilon_2), q_j' \in A\right\} \quad (A.26)
\]

\[
+ Pr(A^c) \quad (A.27)
\]

\[
\leq Pr\left\{\sum_{j \in S_1} q_j' > .5 \frac{N}{\theta}(1 - \epsilon_2) - \sum_{j \in S} q_j' h_j(x), \sum_{j \in S} q_j' h_j(x) < .5 \frac{N}{\theta}(1 + \epsilon_2)\right\} + Pr(A^c) \quad (A.28)
\]

where the first inequality follows from the union bound; the second inequality follows from the definition of event \(A\); the third inequality is a direct application of the union bound. We now ignore the second term since it is arbitrarily small for sufficiently large \(N\).

We are now in the familiar territory of a sum of IID random variables since \(S\) and \(S_1\) have no overlap. Note that \(\sum_{j \in S_1} q_j'\) is independent of \(\sum_{j \in S} q_j' h_j(x)\) and
each of these random variables are Γ distributed. By straighforward conditioning on \( \sum_{j \in S} q'_j h_j(x) \) we can simplify the expressions in Equation [A.28]. It follows that,

\[
\Pr\{\tilde{Q}(0)\} \leq \int_0^5 \Pr\{ \sum_{j \in S_1} q'_j > \frac{N}{\theta} \} \, dg
\]  

(A.29)

Let \( Z = \sum_{j \in S_1} q'_j \) which has a gamma distribution: \( \Gamma(K, \theta) \) and by the Chernoff bound(Section [A.4.1]),

\[
\Pr\{Z > \frac{N}{\theta} \} \leq \min_{t \geq 0} e^{-t \frac{N}{\theta}} \mathbb{E}[e^{tZ}]
\]

= \( \min_{t \geq 0} e^{-t \frac{N}{\theta}} (1 - \frac{t}{\theta})^{-K}, t < \theta \)

= \( (\frac{N}{K})^K e^K e^{-gN}, g > \frac{K}{N} \)

The integral in (A.29):

\[
= \int_0^{\frac{K}{N}} \Pr\{ \sum_{j \in S_1} q'_j > \frac{N}{\theta} \} \, dg + \int_{\frac{K}{N}}^5 (\frac{N}{K})^K e^K e^{-gN} \, dg
\]

(A.30)

The first term is upper-bounded by \( K/N \) since the integrand is positive and always less than 1. The second term is upper-bounded by:

\[
(\frac{N}{K})^K e^K \int_{\frac{K}{N}}^5 g^K e^{-gN} \, dg \leq (\frac{N}{K})^K e^K \int_{\frac{K}{N}}^{\infty} g^K e^{-gN} \, dg
\]

= \( \frac{1}{N} \sum_{p=0}^{K} \frac{K!}{(K-p)!K^p} \)

\[
\leq \frac{K+1}{N}
\]

Combining the bounds on the two terms, we have the upper bound:

\[
\Pr\{q(x) \neq q(x')\} \leq \frac{2K + 1}{N}
\]

(A.31)
And the disagreement volume:

\[
\int_Q 1_{[q(x) \neq q'(x')]} dq \leq \frac{2K + 1}{N} \text{Vol}(Q) \tag{A.32}
\]

And for any \( Q' \subset Q \):

\[
\int_{Q'} 1_{[q(x) \neq q'(x')]} dq \leq \int_Q 1_{[q(x) \neq q'(x')]} dq \leq \frac{2K + 1}{N} \text{Vol}(Q) \tag{A.33}
\]

\[\square\]

### A.4.1 Chernoff Bound on a Gamma distribution

\[
\Pr\{Z > g \frac{N}{\theta}\} \leq \min_{t \geq 0} e^{-tg \frac{N}{\theta}} \mathbb{E}[e^{tZ}] \tag{A.34}
\]

For a Gamma Random Variable \( Z \sim \Gamma(K, \theta) \) the moment generating function is

\[
\mathbb{E}[e^{tZ}] = (1 - \frac{t}{\theta})^{-K}, \text{ if } t < \theta \tag{A.35}
\]

Minimize the bound over \( 0 \leq t < \theta \):

\[
\mathcal{B}(t) = \frac{1}{e^{tg \frac{N}{\theta}} (1 - \frac{t}{\theta})^K} \tag{A.36}
\]

Let \( t = \gamma \theta \) and maximize \( \mathcal{B}^{-1}(\gamma) \) instead:

\[
\gamma^* = \text{argmax}_{0 \leq \gamma < 1} e^{\gamma N} (1 - \gamma)^K \tag{A.37}
\]

Take the derivative:

\[
\frac{d\mathcal{B}^{-1}}{d\gamma} = (1 - \gamma)^{K-1} e^{\gamma N}[-K + (1 - \gamma)gN] \tag{A.38}
\]
The derivative is zero only when the last product term is zero or:

$$\gamma^* = 1 - \frac{K}{gN}$$  \hspace{1cm} (A.39)

Note since $K << N$, $\gamma^* < 1$ and if $c \geq \frac{K}{N}$ then $\gamma^* \geq 0$. Plugging $\gamma^*$ back in:

$$B'(\gamma^*) = \left(\frac{N}{K}\right)^Ke^Kg^Ke^{-gN}, \text{ if } g > \frac{K}{N}$$  \hspace{1cm} (A.40)

## A.4.2 Integral of the Chernoff Bound on a Gamma distribution

$$\left(\frac{N}{K}\right)^Ke^K\int_{g_0}^{\infty} g^Ke^{-gN}dg = e^{-g_0N} \sum_{p=0}^{K} g_0^{K-p} \frac{K!}{(K-p)!K^p}$$  \hspace{1cm} (A.41)

Let $g_0 = \frac{K}{N}$, \hspace{2cm} 

$$= \frac{1}{N} \sum_{p=0}^{K} \frac{K!}{(K-p)!K^p}$$  \hspace{1cm} (A.42)

Define a term in this series as $A_p = \frac{K!}{(K-p)!K^p}$ and calculate the ratio of two succeeding terms:

$$r = \frac{A_p}{A_{p+1}} = \frac{K}{K - p} \geq 1$$  \hspace{1cm} (A.43)

The series is decreasing and the first term $A_0 = 1$ thus

$$\sum_{p=0}^{K} A_p \leq K + 1$$  \hspace{1cm} (A.44)

And the integral is bounded:

$$\left(\frac{N}{K}\right)^Ke^K\int_{g_0}^{\infty} g^Ke^{-gN}dg \leq \frac{K + 1}{N}, \text{ } g_0 = \frac{K}{N}$$  \hspace{1cm} (A.45)
A.5 Proof of Lemma 2.3

The proof closely follows [Nowak, 2009].

Proof. \( \exists p' \) such that

\[
\left| \sum_{i=1}^{B} q(x_i)p'_i \right| \leq \rho \quad \forall q \tag{A.46}
\]

Integrate both sides over \( q \in Q' \)

\[
\int_{Q'} \left| \sum_{i=1}^{B} q(x_i)p'_i \right| dq \leq \rho \text{ Vol}(Q') \tag{A.47}
\]

Integral of the absolute value is greater than the absolute value of the integral and interchange integration with addition:

\[
\left| \sum_{i=1}^{B} \int_{Q'} q(x_i)dq \right| p'_i \leq \rho \text{ Vol}(Q') \tag{A.48}
\]

If \( x \in X \) s.t. \( |\int_{Q'} q(x)dq| \leq \rho \text{ Vol}(Q') \) does not exist then \( |\int_{Q'} q(x)dq| > \rho \text{ Vol}(Q') \) for all \( x \in X \). Since (A.48) is a convex combination of \( \int_{Q'} q(x_i)dq \), if one term is negative there has to exist a positive term in order for the sum to be less than or equal to \( \rho \text{ Vol}(Q') \). Therefore \( \exists x, x' \) such that:

\[
\int_{Q'} q(x)dq > \rho \text{ Vol}(Q') \quad \text{and} \quad \int_{Q'} q(x')dq < -\rho \text{ Vol}(Q') \tag{A.49}
\]

If the pair \( Q, X \) is \( \delta \)-neighborly, there exists a sequence of \( x_i \)'s starting at \( x \) and ending in \( x' \). The sign will have to switch somewhere in the sequence. Let us redefine the pair \( x, x' \) to be where the sign switches. From before: \( \int_{Q'} q(x)dq - \int_{Q'} q(x')dq > 2\rho \text{ Vol}(Q') \). By \( \delta \)-neighborly assumption: \( |\int_{Q'} q(x)dq - \int_{Q'} q(x')dq| < \int_{Q'} |q(x) - q(x')|dq < 2\delta \text{ Vol}(Q) \). Combining the two inequalities: \( \text{ Vol}(Q') < \frac{\delta}{\rho} \text{ Vol}(Q) \). 

\( \square \)
A.6 Proof of Theorem 2.4

Proof. Let $\rho \geq \rho^* \{X, Q\}$ and at this stage we want to find an $x'$ to reduce version space $Q'$ by $\frac{1+\rho}{2}$ at stage $\tau$. Lemma 2.3 states that if that is not possible then

$$Vol(Q') \leq \frac{\delta}{\rho} Vol(Q) \quad (A.50)$$

For simplicity of notation call this the termination of stage 1 and let $\tau$ be the time stage 1 is terminated, namely, the condition above is realized.

To proceed we now restart the entire process by exchanging $Q$ with $Q_{\tau}$. We call this start of stage 2. To avoid confusion we denote the iterations in this stage by $t$.

Let $\rho_t \geq \rho^* \{X, Q_t\}$. Observe that since $Q_t \subset Q$, $\rho^*(X, Q_t) \leq \rho^*(X, Q)$ and we can set $\rho^* \{X, Q\} \leq \rho_t < 1$.

By following the proof of Lemma 2.3 at some time $t$ if an $x$ such that $|\int_{Q_t} q(x) dq | < \rho_t Vol(Q_t)$ does not exist than there must exist $x$ and $x'$ such that:

$$\int_{Q_t} q(x) dq - \int_{Q_t} q(x') dq > 2\rho_t Vol(Q') \quad (A.51)$$

Let $V_d(Q') = \int_{Q_t} \mathbf{1}_{q(x) \neq q(x')}) dq$. Let $Q_{tC} = Q \setminus Q_t$ and $Vol(Q_{tC}) \geq (1 - \frac{\delta}{\rho}) Vol(Q)$.

$$V_d(Q_t) + V_d(Q_{tC}) = V_d(Q) \quad (A.52)$$

By the regularity assumption (2.39), $V_d(Q_{tC}) \geq \alpha V_d(Q)$ and

$$V_d(Q_t) \leq (1 - \alpha)V_d(Q) \quad (A.53)$$

And by $\delta$-neighborly assumption, $V_d(Q) \leq \delta Vol(Q)$ and

$$V_d(Q_t) \leq (1 - \alpha)\delta V_d(Q) \quad (A.54)$$

Combining this expression with inequality $[A.51]$ we obtain:

$$Vol(Q_t) \leq \frac{1 - \alpha}{\rho_t} \delta V_d(Q) \quad (A.55)$$

The first statement of Lemma 2.3 states that for any two consecutive version space
\( Q^t \) and \( Q^{t+1} \) the following reduction is possible for \( \rho^* \leq \rho < 1 \) (\( \rho^* := \rho^* \{X, Q\} \))

\[
Vol(Q^{t+1}) \leq \frac{(1 + \rho)}{2} Vol(Q^t)
\]  

(A.56)

If this condition is not satisfied then the volume bound of Eq. A.55 must hold. Now note that the ratio of the volume bound at the termination of the previous stage \( \tau \) (see Eq. A.50) and at the termination of the current stage \( t \) (see Eq. A.55) is a constant equal to \( (1 - \alpha) \). Furthermore, we are guaranteed an exponential rate \( (1 + \rho_t)/2 \) of decay while going from termination of stage 1 to termination of stage 2. Consequently, we can reduce the volume at the previous stage \( \tau \) to the current stage \( t \) with at most a constant number of queries. For simplicity we assume that this is equal to one since the order-wise scaling of the number of queries does not change. Consequently, we can obtain:

\[
Vol(Q^{t+1}) = \frac{(1 - \alpha)\delta}{\rho} Vol(Q^t)
\]  

(A.57)

To obtain the worst case rate for each iteration we need:

\[
\lambda_0 = \min_{\rho^* \leq \rho \leq 1} \max \left\{ \frac{1 + \rho}{2}, \frac{(1 - \alpha)\delta}{\rho} \right\}
\]  

(A.58)

This expression simplifies to the situation when the two arguments are equal. This turns out to be \( \rho = \frac{1}{2} (\sqrt{1 + 8(1 - \alpha)\delta} - 1) \)

\[
\lambda_0 = \max \left\{ \frac{1 + \rho^*}{2}, \frac{1 + 0.5(\sqrt{1 + 8(1 - \alpha)\delta} - 1)}{2} \right\}
\]  

(A.59)

where \( \delta = \frac{2K + 1}{N} \). We now note that \( \sqrt{1 + z} \leq 1 + z/2 \). Consequently, we get,

\[
\lambda_0 \leq \lambda = \max \left\{ \frac{1 + \rho^*}{2}, \frac{1}{2} (1 + (1 - \alpha) \frac{2K + 1}{N}) \right\}
\]  

(A.60)

We can repeat this argument for Stage 3, Stage 4 and so on in an identical fashion. The volume of our final version space is required to be \( Vol(Q^n) = \epsilon Vol(Q) \).

\[
Vol(Q^n) = \lambda^n Vol(Q)
\]
\[ \epsilon = \lambda^n \implies n = \frac{\log \epsilon}{\log \lambda} \]

### A.7 Proof of Theorem 2.5

**Proof.** In the proof, all volume is taken with respect to the lebesgue measure on the \( p \) sparse subspace. If we can reduce the volume of sparse version space at each stage by \( \lambda \) then after \( n \) stages:

\[ Vol(S^n) = \lambda^n Vol(S) \quad (A.61) \]

There are \( \binom{N}{p} \) \( p \)-sparse disjoint segments: \( \{ s_1, s_2, \ldots, s_{\binom{N}{p}} \} = S \). Without loss of generality, we define the volume \( Vol(.) \) such that \( Vol(s_r) = 1 \) for \( r = 1, \ldots, \binom{N}{p} \) therefore

\[ Vol(S^n) = \lambda^n \left( \binom{N}{p} \right) \quad (A.62) \]

By assumption from Section 2.5.2 we define

\[ q_s = \arg\inf_{q^* \in S} Vol\{ q \in S \mid ||q - q^*||_1 \leq \frac{\theta}{2} \} \quad (A.63) \]

\[ f(\theta, p) = Vol\{ q \in S \mid ||q - q_s||_1 \leq \frac{\theta}{2} \} \quad (A.64) \]

If \( Vol(S^n) \leq f(\theta, p) \) then \( S^n \subset \{ q \in S \mid ||q - q_s||_1 \leq \frac{\theta}{2} \} \) and \( \forall q \in S^n \) (by the margin bound [Schapire et al., 1997a])

\[ Prob(q(x) \neq y) \leq O \left( \frac{\log |\mathcal{X}| \log p}{\theta^2 |\mathcal{X}| } + \frac{\log(1/\delta)}{|\mathcal{X}|} \right)^{\frac{1}{2}} \quad (A.65) \]
So we require:

\[
Vol(S^n) \leq f(\theta, p) 
\]

\[
n \log \lambda + \log \left( \frac{N}{p} \right) \leq \log f(\theta, p) 
\]

\[
n \geq \frac{\log (N)}{\log \frac{1}{\lambda}} + \log \frac{1}{f(\theta, p)} 
\]

(A.66)  \quad (A.67)  \quad (A.68)

\[\square\]

A.8 Proof of Lemma 2.6

Proof. If \( \rho^* < 1 \) then \( \exists q \in Q \) s.t. \( q^T h(x_i) > 0 \ \forall i \). Let us define a vector \( f(q) \in R^M \) with \( f(q)_i = q^T h(x_i) \) and a set \( F = \{ f(q) | q \in Q \} \). Since every component of \( f \) cannot be positive, the set \( F \) cannot lie in the first (positive) orthant. The set \( F \) is also convex, so there must exist a separating hyperplane with a normal vector \( z \geq 0 \).

This implies the following inequality:

\[
\sum_{i=1}^{M} z_i f(q)_i = \sum_{i=1}^{M} z_i \sum_{j=1}^{N} q_j h_j(x_i) \leq 0 
\]

(A.69)

At least one element of \( z \) must be non-zero to define a hyperplane. Let us interchange the summation:

\[
\sum_{j=1}^{N} q_j \sum_{i=1}^{M} z_i h_j(x_i) \leq 0 
\]

(A.70)

From earlier, we assume that for every weak hypothesis there exists a compliment: s.t. \( h_j(x) = -h_{j^*}(x) \) and \( h_j, h_{j^*} \in H \). For any weight vector \( q \), we can reassign the weight of \( h_j \) to its compliment \( h_{j^*} \) and make the left side in (A.70) greater than zero. But the inequality in (A.70) has to hold for all \( q \in Q \). This can only be true if every term in the summation is zero:

\[
\sum_{i=1}^{M} z_i h_j(x_i) = 0 \ \forall j 
\]

(A.71)

\[\square\]
Appendix B

Sequential Sensor Selection

B.1 Derivation for Theorem 3.3

Consider a three stage system. Define some terms

Error Indicator: $1_{[f(x) \neq y]} \rightarrow C(yf(x)) = \frac{1}{1 + \exp(yf(x))}$ \hspace{1cm} (B.1)

Reject Indicator: $1_{[f_p(x) \neq f_n(x)]} \rightarrow \hspace{1cm} (B.2)$

$C_{r}(f_p, f_n, x, y) = C(yf_p(x)) + C(yf_n(x)) - 2C(yf_p(x))C(y_nf(x))$ \hspace{1cm} (B.3)

Risk for three stages:

$R(f_p^1, f_n^1, f_p^2, f_n^2, f^3, x, y) = S^1 R^1 + S^2 R^2 + S^3 R^3$ \hspace{1cm} (B.4)

$S^1 = 1$ \hspace{1cm} (B.5)

$S^2(f_p^1, f_n^1, x, y) = C_r(f_p^1, f_n^1, x^1, y)$ \hspace{1cm} (B.6)

$S^3(f_p^1, f_n^1, f_p^2, f_n^2, x, y) = C_r(f_p^1, f_n^1, x^1, y)C_r(f_p^2, f_n^2, x^2, y)$ \hspace{1cm} (B.7)

$R^1(f_p^1, f_n^1, x, y) = C(yf_p^1(x^1))C(yf_n^1(x^1)) + \alpha c_2 C_r(f_p^1, f_n^1, x^1, y)$ \hspace{1cm} (B.8)

$R^2(f_p^2, f_n^2, x, y) = C(yf_p^2(x^2))C(yf_n^2(x^2)) + \alpha c_3 C_r(f_p^2, f_n^2, x^2, y)$ \hspace{1cm} (B.9)

$R^3(f_p^3, x, y) = C(yf^3(x^3))$ \hspace{1cm} (B.10)

(B.11)
Plug in all the terms:

\[ R(\cdot) = C(yf_p^1(x^1))C(yf_n^1(x^1)) + \alpha c_2 C_r(f_p^1, f_n^1, x^1, y) \]

\[ + C_r(f_p^1, f_n^1, x^1, y) \left\{ C(yf_p^2(x^2))C(yf_n^2(x^2)) + \alpha c_3 C_r(f_p^2, f_n^2, x^2, y) \right\} \]

\[ + C_r(f_p^1, f_n^1, x^1, y) C_r(f_p^2, f_n^2, x^2, y) C(yf^3(x^3)) \]

Minimize over \( f_p^1, f_n^1 \) and keep \( f_p^2, f_n^2, f^3 \) constant. We can rearrange the terms to get:

\[ \arg \min_{f_p^1, f_n^1} \sum_i R(f_p^1, f_n^1, f_p^2, f_n^2, f^3, x_i, y_i) = \]

\[ \arg \min_{f_p^1, f_n^1} \sum_i C(yf_p^1(x_i^1))C(yf_n^1(x_i^1)) + \delta_i^1 C_r(f_p^1, f_n^1, x_i^1, y_i) \]

\[ \text{such that:} \]

\[ \delta_i^1 = \alpha c_2 + \left\{ C(yf_p^2(x_i^2))C(yf_n^2(x_i^2)) + \alpha c_3 C_r(f_p^2, f_n^2, x_i^2, y_i) \right\} \]

\[ + C_r(f_p^2, f_n^2, x_i^2, y) C(yf^3(x_i^3)) \]

Minimize over \( f_p^2, f_n^2 \) and keep \( f_p^1, f_n^1, f^3 \) constant:

\[ \arg \min_{f_p^2, f_n^2} \sum_i R(f_p^1, f_n^1, f_p^2, f_n^2, f^3, x_i, y_i) = \]

\[ \arg \min_{f_p^2, f_n^2} \sum_i S_i^2 \left\{ C(yf_p^2(x_i^2))C(yf_n^2(x_i^2)) + \delta_i^2 C_r(f_p^2, f_n^2, x_i^2, y_i) \right\} \]

\[ \text{such that:} \]

\[ S_i^2 = C_r(f_p^1, f_n^1, x_i^1, y) \]

\[ \delta_i^2 = \alpha c_3 + C(yf^3(x_i^3)) \]
Minimize over $f_3$ and keep $f_p^1, f_n^1, f_p^2, f_n^2$ constant:

$$\arg \min_{f_3} \sum_i R(f_p^1, f_n^1, f_p^2, f_n^2, f_3, x_i, y_i) = \text{(B.25)}$$

$$\arg \min_{f_3} \sum_i S_i^3 C(y f_3(x_i^3)) = \text{(B.26)}$$

such that:

$$S_i^3 = C_r(f_p^1, f_n^1, x_i^1, y) C_r(f_p^2, f_n^2, x_i^2, y) \text{ (B.27)}$$

### B.2 Proof of Theorem 3.4

**Proof.** This will closely follow the proof of Theorem 1 in [Schapire et al., 1997b]. We have to bound two terms:

$$P_D[ yf_n(x) \leq \theta_1, yf_p(x) \leq \theta_1 ] \text{ and } P_D[ yf_2(x) \leq \theta_2, yf_n(x) \neq yf_p(x) ] \text{ (B.29)}$$

**First Term** Let us bound the first term. Define $C_N$ to be the set of unweighted averages over $N$ elements from $H_1$,

$$C_N = \{ f : x \rightarrow \frac{1}{N} \sum_{i=1}^{N} h_i(x) \mid h_i \in H_1 \} \text{ (B.30)}$$

Any weighed classifier $f = \sum_h q_h h(x)$ can be approximated by drawing an element from $C_N$ by choosing $h_1...h_N$ with prob. $q_h$.

We can express our first term as a sum of probabilities of disjoint events.

$$P_D[ yf_p(x) \leq 0, yf_n(x) \leq 0 ] = \text{(B.31)}$$

$$P_D \left[ yf_p(x) \leq 0, yf_n(x) \leq 0, yg_p(x) \leq \frac{\theta_1}{2}, yg_n(x) \leq \frac{\theta_1}{2} \right] \text{ (B.32)}$$

$$+ P_D \left[ yf_p(x) \leq 0, yf_n(x) \leq 0, yg_p(x) \leq \frac{\theta_1}{2}, yg_n(x) > \frac{\theta_1}{2} \right] \text{ (B.33)}$$

$$+ P_D \left[ yf_p(x) \leq 0, yf_n(x) \leq 0, yg_p(x) > \frac{\theta_1}{2}, yg_n(x) \leq \frac{\theta_1}{2} \right] \text{ (B.34)}$$

$$+ P_D \left[ yf_p(x) \leq 0, yf_n(x) \leq 0, yg_p(x) > \frac{\theta_1}{2}, yg_n(x) > \frac{\theta_1}{2} \right] \text{ (B.35)}$$
Further, we can write,

\[
P_D \left[ yf_p(x) \leq 0, yf_n(x) \leq 0 \right] \leq P_D \left[ yg_p(x) \leq \frac{\theta_1}{2}, yg_n(x) \leq \frac{\theta_1}{2} \right] \quad (B.36)
\]

\[
+ P_D \left[ yf_p(x) \leq 0, yf_n(x) \leq 0, yg_p(x) > \frac{\theta_1}{2}, yg_n(x) > \frac{\theta_1}{2} \right] \quad (B.37)
\]

The inequality holds for any \( g_p, g_n \). We take the expected value of the right hand side wrt to the distribution \( C \)

\[
P_D \left[ yf_p(x) \leq 0, yf_n(x) \leq 0 \right] \leq \mathbb{E}_C \left[ P_D \left[ yg_p(x) \leq \frac{\theta_1}{2}, yg_n(x) \leq \frac{\theta_1}{2} \right] \right] \quad (B.38)
\]

\[
+ \mathbb{E}_D \left[ P_{C_p, C_n} \left[ yg_p(x) > \frac{\theta_1}{2}, yg_n(x) > \frac{\theta_1}{2} \mid yf_p(x) \leq 0, yf_n(x) \leq 0 \right] \right] \quad (B.40)
\]

The last term inside the expectation is the probability that an average of \( N \) bernoulli random variables is larger than its expectation, we use a concentration result from Equation (4) in Theorem 1 of \cite{Schapire et al., 1997b}.

\[
P_{C_p, C_n} \left[ yg_p(x) > \frac{\theta_1}{2}, yg_n(x) > \frac{\theta_1}{2} \mid yf_p(x) \leq 0, yf_n(x) \leq 0 \right] \leq \exp \left( -\frac{N\theta_1^2}{8} \right) \quad (B.41)
\]

To bound the first we use the result from Equation (5) in Theorem 1 of \cite{Schapire et al., 1997b}. if we set \( \epsilon_N = \sqrt{(1/2m) \log((N + 1)|H_1|2N)/\delta_N} \), with probability at least \( 1 - \delta_N \),

\[
P_{D,C} \left[ yg_p(x) \leq \frac{\theta_1}{2}, yg_n(x) \leq \frac{\theta_1}{2} \right] \leq P_{S,C} \left[ yg_p(x) \leq \frac{\theta_1}{2}, yg_n(x) \leq \frac{\theta_1}{2} \right] + \epsilon_N \quad (B.42)
\]

for any choice of \( \theta \) and every distribution \( C \). Here, \( P_S \) is probability taken with respect to a randomly drawn sample of size \( m \) from \( D \).

By the same argument as in inequality \( B.37 \),

\[
P_{S,C_p} \left[ yg_p(x) \leq \frac{\theta_1}{2}, yg_n(x) \leq \frac{\theta_1}{2} \right] \leq \quad (B.43)
\]

\[
P_S \left[ yf_p(x) \leq \theta_1, yf_n(x) \leq \theta_1 \right] + \mathbb{E}_S \left[ P_{C_p} \left[ yg_p(x) \leq \frac{\theta_1}{2} \mid yf_p(x) > \theta \right] \right] \quad (B.44)
\]

The expressions inside the expectation can be bounded using the same Chernoff bound.
result from [B.41]
\[ P_C \left[ yg_p(x) \leq \frac{\theta_1}{2}, yg_n(x) \leq \frac{\theta_1}{2} \mid yf_p(x) > \theta_1, yf_p(x) > \theta_1 \right] \leq \exp \left( \frac{-N\theta_1^2}{8} \right) \] (B.45)

By setting \( \delta_N = \delta/(N(N + 1)) \), and combining the terms,
\[ P_D[yf_p(x) \leq 0, yf_n(x) \leq 0] \leq \]
(B.46)
\[ P_S[yf_p(x) \leq \theta_1, yf_n(x) \leq \theta_1] + 2\exp \left( \frac{-N\theta_1^2}{8} \right) + 2\sqrt{\frac{1}{2m} \log \left( \frac{N(N + 1)^2|H_1|^{2N}}{\delta} \right)} \] (B.47)

By setting, \( N = (4/\theta_1^2) \log(m/ \log |H_1|^2) \),
\[ P_D[yf_p(x) \leq 0, yf_n(x) \leq 0] \leq P_S[yf_p(x) \leq \theta_1, yf_n(x) \leq \theta_1] \]
(B.48)
\[ + \mathcal{O} \left( \frac{1}{\sqrt{m}} \left( \frac{\log m \log |H|^2}{\theta} + \log \frac{1}{\delta} \right)^{1/2} \right) \] (B.49)

**Second Term** Here we will bound the second term, \( P_D[yf_2(x) \leq \theta_2, yf_n(x) \neq yf_p(x)] \) Define a new distribution:
\[ D_r = \begin{cases} cD(x, y), & f_p(x) \neq f_n(x) \\ 0, & f_p(x) = f_n(x) \end{cases} \] (B.50)

Rewrite:
\[ P_D[yf_2(x) \leq \theta_2, yf_n(x) \neq yf_p(x)] \leq P_D[yf_2(x) \leq \theta_2 \mid yf_n(x) \neq yf_p(x)] \] (B.51)
\[ = P_{D_r}[yf_2(x) \leq \theta_2] \] (B.52)

Note that \( S_r \) is an iid sample from \( D_r \). Using Theorem 1 in [Schapire et al., 1997b],
\[ P_{D_r}[yf_2(x) \leq 0] \leq P_{S_r}[yf_2(x) \leq \theta_2] + \mathcal{O} \left( \frac{1}{\sqrt{m}} \left( \frac{\log m \log |H_2|}{\theta_2} + \log \frac{1}{\delta} \right)^{1/2} \right) \]

Collecting the two terms produces the desired result.
B.3 Proof of Theorem 3.5

To simplify our derivations, we assume uniform class prior probability: $P_y[y = \hat{y}] = \frac{1}{C}$, $\hat{y} = 1, \ldots, C$. However, our results can be easily modified to account for a non-uniform prior. The expected conditional risk can be solved optimally by a dynamic program, where a DP recursion is,

$$J_K(x^K, S^K) = \min_{f_K} E_y \left[ S^K(x^K) R_k(y, x^K, f^K) \mid x^K \right]$$

(B.53)

$$J_k(x^k, S^k) = \min_{f_k} \left\{ E_y \left[ S^k(x^k) R_k(y, x^k, f^k) \mid x^k \right] + E_{x^{k+1} \ldots x^K} \left[ J_{k+1}(x^{k+1}, S^{k+1}) \mid x^k \right] \right\}$$

(B.54)

Consider $k$th stage minimization, $f^k$ can take $C + 1$ possible values $\{1, 2, \ldots, C, r\}$ and $J_k(x^k, S^k)$ can be recast as a conditional expected risk minimization,

$$J_k(x^k, S^k = 1) = \min_{f_k} \left\{ P_y \left[ y \neq \hat{y} \mid x^k \right], \alpha c_{k+1} + E_{x^{k+1} \ldots x^K} \left[ J_{k+1}(x^{k+1}, 1) \mid x^k \right] \right\}$$

(B.55)

Define,

$$\delta^k(x^k) = \alpha c_{k+1} + E_{x^{k+1} \ldots x^K} \left[ J_{k+1}(x^{k+1}, S^{k+1} = 1) \mid x^k \right]$$

(B.56)

and rewrite the conditional risk in (B.55)

$$f^k = \arg \min_f \left\{ 1 - P_y \left[ y = \hat{y} \mid x^k \right], \delta^k(x^k) \right\}$$

(B.57)
Reject is the optimal decision if,

\[
\min_{\hat{y}} \{ 1 - P_y[y = \hat{y} \mid \mathbf{x}^k] \} \geq \delta^k(\mathbf{x}^k) \implies \max_{\hat{y}} \{ P_y[y = \hat{y} \mid \mathbf{x}^k] \} \leq 1 - \delta^k(\mathbf{x}^k)
\]  
(B.58)

If reject is not the optimal strategy then a class is chosen to maximize the posterior probability:

\[
f^k(\mathbf{x}^k) = \arg \max_{\hat{y} \in \{1, \ldots, c\}} \{ P_y[y = \hat{y} \mid \mathbf{x}^k] \}
\]  
(B.59)

which is exactly our claim.

\[\square\]

### B.4 Proof of Lemma 3.6

Define an auxiliary variable corresponding to the error penalty term and absolute value of the maximizing codeword projection respectively:

\[
e_i = 1_{[d^k(\mathbf{x}^k) \neq y_i]}, \quad z_i = \sigma_d^k(\mathbf{x}^k)
\]  
(B.60)

\[
\tilde{R}^k_i(\cdot) = e_i 1_{[g(\mathbf{x}^k) - z_i < 0]} + \delta_i^k 1_{[g(\mathbf{x}^k) - z_i \geq 0]}
\]  
(B.61)

\[
= e_i 1_{[g(\mathbf{x}^k) - z_i < 0]} + \delta_i^k \left\{ 1 - 1_{[g(\mathbf{x}^k) - z_i < 0]} \right\}
\]  
(B.62)

\[
= \{ e_i - \delta_i^k \} 1_{[g(\mathbf{x}^k) - z_i < 0]} + \delta_i^k
\]  
(B.63)

Define weights \(w_i = e_i - \delta_i^k\) and drop the \(\delta_i^k\) term since it does not depend on \(g(\cdot)\). Our goal is to minimize \(\sum S_i^k \tilde{R}^k_i\) over \(g\). We will split the summation into two sets:

\[
= \sum_{w_i \geq 0} S_i^k w_i 1_{[(g(\mathbf{x}^k) - z_i) \leq 0]} + \sum_{w_i < 0} S_i^k w_i 1_{[(g(\mathbf{x}^k) - z_i) \leq 0]}
\]  
(B.64)

\[
= \sum_{w_i \geq 0} S_i^k w_i 1_{[(g(\mathbf{x}^k) - z_i) \leq 0]} + \sum_{w_i < 0} S_i^k w_i \left\{ 1 - 1_{[(g(\mathbf{x}^k) - z_i) > 0]} \right\}
\]  
(B.65)
If discard the constant term $\sum_{w_i < 0} S^k_i w_i$ and introduce pseudo labels

$$b_i = \begin{cases} +1, & w_i \geq 0 \\ -1, & w_i < 0 \end{cases}$$

then,

$$\arg\min_g \sum_{i=1}^{N} S^k_i \tilde{R}^k_i = \arg\min_g \sum_{i=1}^{N} S^k_i \left| w_i \right| \mathbf{1}_{[b_i(g(x^k_i) - z_i) \leq 0]} \quad (B.66)$$

### B.5 Proof of Theorem 3.7

At each stage the reject decision can be expressed in terms of three boolean decisions:

$$\mathbf{1}_{[h^k(x^k)|-g^k(x^k)\leq 0]} = \mathbf{1}_{[h^k(x^k)>0]} \mathbf{1}_{[h^k(x^k)-g^k(x^k)\leq 0]} + \mathbf{1}_{[h^k(x^k)\leq 0]} \mathbf{1}_{[-h^k(x^k)-g^k(x^k)\leq 0]}$$

(B.67)

If the rejectors ($g^k \in G^k$) and stage classifiers ($h^k \in H^k$) belong to families with finite VC dimensions then the complexity of Decision 2 and Decision 3 is $\mathcal{VC}[G^k] + \mathcal{VC}[H^k]$.

The system classifier, $F$, is composed of $K$ stages. Each of the first $K - 1$ stages can be expressed as a boolean function of 3 boolean decisions. The last stage is a single boolean decision. So the output $F$ can be expressed as a boolean function of $3(K - 1) + 1 = 3K - 2$ functions. We know the VC dimension for each of the functions. Using this fact and Lemma 2 in [Sontag, 1998](#) we obtain our result.

### B.6 Implementation Details

For large datasets ($N > 1000$), we split them 50/10/40% into train, validation and test sets. The performance reported is on the test set. For smaller datasets ($N < 1000$), we perform 50 random 70/10/20% splits and report the average performance over the
trials. Each subproblem reduces to minimizing a weighted binary error problem with respect to a logistic loss. Polynomial kernel classifier of degree $q$ is parametrized by a vector $a$:

$$h(x) = \sum_{i=1}^{N} a_i (x_i^T x + 1)^q$$  \hspace{1cm} (B.68)

The optimization over the polynomial kernel classifier is performed using newton gradient descent method. Table 1 shows the degree of polynomial kernels used in our simulations.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\mathcal{H}^1$</th>
<th>$\mathcal{G}^1$</th>
<th>$\mathcal{H}^2$</th>
<th>$\mathcal{G}^2$</th>
<th>$\mathcal{H}^3$</th>
<th>$\mathcal{G}^3$</th>
<th>$\mathcal{H}^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>synthetic</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mam</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pima</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>threat</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>covertype</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>letter</td>
<td>7</td>
<td>2</td>
<td>7</td>
<td>2</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mnist</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>landsat</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table B.1:** Stage Complexity: we use polynomial kernel classifiers. This table displays the degree of the polynomial kernel used at each stage for the rejector and the stage classifier

### B.6.1 Minimizing Logistic Loss

Minimization of a logistic loss for binary classification problem can be easily performed using a Newton a method:

$$\min_{\mathbf{w}} [f(\mathbf{w})] = \frac{N}{\sum_{i=1}^{N} C [y_i \mathbf{w}^T \mathbf{x}_i]}$$  \hspace{1cm} (B.69)
For simplicity, let us assume the classifier is parameterized by a linear function $\textbf{w}$. Let us also the first derivative and the second derivative with respect to $\textbf{w}$:

$$
\frac{df(\textbf{w})}{d\textbf{w}_j} = \sum_{i=1}^{N} y_i x_{ii} L'[y_i \textbf{w}^T \textbf{x}_i] \quad (B.70)
$$

$$
\frac{d^2 f(\textbf{w})}{d\textbf{w}_j d\textbf{w}_k} = \sum_{i=1}^{N} x_{ij} x_{ik} L''[y_i \textbf{w}^T \textbf{x}_i] \quad (B.71)
$$

And for a logistic loss:

$$
L(z) = \log(1 + \exp(-z)) \quad (B.72)
$$

$$
L'(z) = \frac{-1}{1 + \exp(z)} \quad (B.73)
$$

$$
L''(z) = \frac{2}{\cosh(z/2)^2} \quad (B.74)
$$

A newton iteration update has the following form:

$$
\textbf{w}^{t+1} = \textbf{w}^t + a \nabla^2 f(\textbf{w}^t) \nabla f(\textbf{w}^t) \quad (B.75)
$$

The step size $a$ can easily be computed by a backtracking algorithm. Notice that this optimization can be performed for any smooth convex loss function $C[\cdot]$.

Often there is a need to add a regularization term to the objective. For example, this can be easily done for an $l_2$ regularization. For simplicity, let $d$ be the dimensionality of $\textbf{x}$ and let the last component be constant: $x_d = 1$

$$
\min_{\textbf{w}}[f(\textbf{w})] = \sum_{i=1}^{N} C [y_i \textbf{w}^T \textbf{x}_i] + \lambda \sum_{j=1}^{d-1} w_j^2 \quad (B.76)
$$
We can readily redefine the first and the second derivatives:

\[
\frac{df(w)}{dw_j} = \sum_{i=1}^{N} y_i x_{ii} L'[y_i w^T x_i] + 1_{[j<d]} \lambda w_j \quad \text{(B.77)}
\]

\[
\frac{d^2 f(w)}{dw_j dw_k} = \sum_{i=1}^{N} x_{ij} x_{ik} L''[y_i w^T x_i] + 1_{[j<d]} \lambda^2 \quad \text{(B.78)}
\]
References


Vita

Kirill Trapeznikov

http://blogs.bu.edu/ktrap/aboutme/

+1 440 840 0923

ktrapeznikov@gmail.com

240 Tappan St., Brookline, MA 02445

citizenship: U.S. citizen

Areas of Specialization

- *Machine Learning*: supervised, semi-supervised and unsupervised algorithms, generative and discriminative methods, robust classification, cost sensitive learning, feature extraction, dimensionality reduction and visualization

- *Statistical Signal Processing*: recursive estimation, image processing and reconstruction, inverse problems, detection theory, basic computer vision

- *Optimization Methods*: convex, non-convex, online, bayesian

Software Skills: MATLAB, Python, C/C++, \LaTeX, Cadence (IC,PCB), Verilog

Other Skills: Knowledge of analog and digital circuit design. Use of engineering lab equipment: spectrum analyzer, oscilloscope, function generator, various soldering tools, basic optical equipment
Education

**Boston University**, Boston, MA

*Doctor of Philosophy Candidate*, Electrical Engineering *September 2013*

Thesis Title: "Machine Learning on a Budget"

*Master of Science*, Electrical Engineering. GPA: 3.95/4.00 *December 2010*

*Bachelor of Science*, Electrical Engineering. GPA: 3.86/4.00 *September 2007*

Research and Professional Experience

Dept. of Electrical and Computer Engineering, Boston University, Boston, MA

*Graduate Research Assistant* *September 2008 - September 2013*

Information Sciences and Systems Lab. Research in machine learning and statistical signal processing, theory and methods: active learning, boosting methods, multi-stage sequential decision systems, cost-sensitive and budget constrained classification.

Projects:

- Multi-sensor fusion explosive detection: DHS funded project to increase system throughput by dynamic sensor selection for every decision.

- Machine learning under budgets: NSF funded project to investigate frameworks for training decisions systems under test-time budget constraints.

- Classification with privileged information: AFOSR project (collaboration with Longshortway, Inc.) to utilize privileged information in reducing acquisition cost when making decision. Duties include supervision of a first year graduate student.

*Research Advisors*: Venkatesh Saligrama, David Castañón.
Sandia National Laboratories, Solar Technologies, Albuquerque, NM


Projects in concentrated solar power dish technologies:

- Novel imaging system: automated mirror alignment and surface characterization tools using fringe reflection techniques. Design and implementation of algorithms and GUI in MATLAB and C.

- Heat engine simulator for control system development: circuit design and PCB layout.

Biomimetic Systems, Cambridge, MA

Technical Intern  Summer 2006

Validation and testing of hardware and algorithms for an acoustic direction finder system (gunshot localization).

Selected Publications


- K. Trapeznikov, V. Saligrama, "Supervised Sequential Classification Under Budget Constraints", Int. Conf. on Artificial Intell. and Stats., April 2013, (oral, 10% acceptance rate)


Invited Publications


Invited Talks

• Supervised Sequential Classification Under Budget Constraints, Graduation Day Talk, Information Theory and Applications Workshop, San Diego, 2013

• Multi-Stage Decision System, 8th Algorithm Development for Security Applications Workshop, Boston, 2012

Workshop Organization

• Co-organizer: Int. Conf. on Machine Learning 2013 Workshop on Machine Learning with Test-time budgets
Poster Presentations


- Multi-Stage Classifier Design, Research and Industrial Collaboration Conference (RICC), at Awareness and Localization of Explosive Related Threats (ALERT) DHS Center of Excellence, October, Boston, 2011

- Active Boosted Learning, Boston University Science Day, 2011

- Active Boosted Learning, Research and Industrial Collaboration Conference (RICC), at Awareness and Localization of Explosive Related Threats (ALERT) DHS Center of Excellence, October, Boston, 2010

Teaching and Mentorship Experience

- Teaching Assistant: hold office hours, teach discussion sections, provide help during lab hours, teach substitute lectures for: Electromagnetic, VLSI Design, Photonics, Stochastic Systems.

- Supervise junior graduate students on research projects.

Related Coursework

Course Projects

Evaluation of SVM Active Learning Methods, Algorithms for Target Detection in Clutter, Image De-blurring Methods, Compressed Sensing, Equalization Techniques

Languages: Fluent in English and Russian