MULTIPLE-INSTANCE FEATURE RANKING

by

ANDREW CLARK LATHAM

Submitted in partial fulfillment of the requirements

for the degree of Master of Science

Department of Electrical Engineering and Computer Science

CASE WESTERN RESERVE UNIVERSITY

January, 2016

CASE WESTERN RESERVE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

We hereby approve the thesis of

Andrew Clark Latham

candidate for the degree of **Master of Science***.

Committee Chair

Dr. Soumya Ray

Committee Member

Dr. Harold Connamacher

Committee Member

Dr. Michael Lewicki

Date of Defense

August 18, 2015

*We also certify that written approval has been obtained

for any proprietary material contained therein.

Contents

List of Tables

List of Figures

Acknowledgments

I'd like to begin by thanking my advisor, Dr. Soumya Ray, who has provided excellent guidance over the last two years and played a critical role in developing not just my understanding of machine learning but also my ability to think critically, overcome obstacles, and reason about research topics in a way that I never would have been able to do on my own. The work I have done with Professor Ray has been incredibly rewarding and inspired a passion for learning and exploring that will stay with me long after I leave this institution.

I would also like to thank Gary Doran, with whom I worked during my first year and a half of research. When I first started studying these topics, Gary was always happy to explain complicated ideas in an intuitive fashion, and had seemingly infinite patience when introducing me to the theoretical pillars that would later form the foundation for my own work. Gary also built the experimental framework that I used to execute the experiments in this thesis.

In addition to Professor Ray, my committee is composed of Dr. Harold Connamacher and Dr. Michael Lewicki. In my time at Case Western Reserve University, the classes I took from these professors have proven to be among the most useful both in general and, in particular, in the writing of this thesis; from Professor Connamacher I learned the basics of mathematical reasoning in the context of computer science, and from Professor Lewicki I learned how to approach machine learning from a probabilistic perspective. Professor Connamacher also took the time to send me a

detailed critique of this thesis prior to its submission, which has helped me to clarify and improve the explanations of many key concepts.

I would like to thank all my friends at Case Western Reserve University, but especially Larry Muhlstein, Ben Cowen, and Greg Penzias, who have proven to be great sources of advice and camaraderie. In the final week before completion of this thesis, Greg allowed me to use his allocated server time to run my experiments, which allowed me to improve the experimental results section.

Finally and most importantly, I would like to thank my family: my mom, dad, my sister Olivia, and my grandparents, who have been incredibly helpful and supportive over the years that it has taken me to get to this point and always give great advice. I couldn't have done it without you.

Multiple-Instance Feature Ranking

Abstract

by

ANDREW CLARK LATHAM

Multiple-instance learning is a subfield of machine learning in which training data is provided as labeled sets of instances called "bags," with the instance labels themselves unknown. Multiple-instance learning has many important practical applications, including the drug discovery, image retrieval, and text classification problems. In this thesis I will investigate the problem of feature ranking, where the most important features are determined based on some evaluation metric, in the context of multiple-instance learning. In order to rank features well, the instance labels are required; however, in multiple-instance learning, only the labels of the bags are available. I will investigate the implications of giving every instance the label of its bag, and using the resulting supervised dataset to evaluate and rank features. Even though this introduces additional one-sided noise to the data set, I provide a theoretical analysis that shows that in many situations a relevant set of features can be recovered, where "relevant" is defined using the feature set that would be found if the true instance labels were available, removing the limitations of the multiple-instance learning problem entirely. I describe the factors that control when a good feature ranking can be learned when both accuracy and AUC are used as scoring functions. Finally, I evaluate the performance of several dimensionality reduction algorithms on a number of multiple-instance learning datasets and find that, in addition to being fast and relatively simple, the algorithms proposed in this thesis are competitive with existing techniques and can often improve the performance of a classifier.

Chapter 1

Introduction

Supervised learning is one of the central challenges of machine learning. In this problem, a machine is given a dataset of objects, each of which is associated with a label, and must learn to predict labels for new objects based on what it has learned from that dataset. Supervised learning has seen a wide variety of applications, from object detection in self-driving cars to handwriting recognition to disease prevention. In this thesis, I will focus on the classification problem in supervised learning, where the labels are restricted to a small set of possible values, typically just "true" or "false" to indicate whether an object has some property.

One of the key problems in supervised learning is how to represent abstract problems or concepts in a computer-understandable way. Standard supervised learning represents each object by a vector of values, called features, each of which describes some information about the object. For example, when teaching a machine to recognize images of books, it may be presented with a collection of vectors and associated labels, where each vector contains a set of features describing properties of the image it represents, such as its color, and each label indicates whether or not the image is of a book. Given a new vector, the goal is to say whether or not it represents an image of a book.

An important issue in data representation concerns the features. Although it would seem ideal to have as much information as possible about an object, having too many features can result in a number of problems in practice, including the "curse of dimensionality," the phenomenon of algorithms having poor performance when presented with too many features relative to the number of instances. Dimensionality reduction is the problem of reducing the number of features used to describe an object to a much smaller number, either through feature selection, which attempts to discern which features are the most interesting or relevant and only use those, or through feature construction, which generates entirely new features based on those given [Guyon and Elisseeff, 2003]. Performing dimensionality reduction makes the learning process faster and often leads to better performance on the classification task.

In many cases, the simple method of data representation used in supervised learning may be highly inconvenient or even impossible, for reasons that will be discussed in section 2.3. The multiple-instance representation expands the range of possible data representations by allowing an object to be described as an arbitrarily-large multiset of vectors rather than a single vector [Dietterich et al., 1997]. This is a much more realistic representation when the property being studied describes a collection of things rather than a single thing. Furthermore, the use of the multiple-instance representation allows for objects to be broken down into consistently-described components. For example, when humans are determining whether an image has the property "book", we don't look at the entire picture; rather, we look at individual entities in the picture to see if any of those entities is a book. The multiple-instance representation allows for an image to be represented by a set of vectors describing the different entities in the image, matching this intuition.

In this thesis, I will study the problem of feature selection for multiple-instance learning. Feature selection is typically quick, relatively simple and, by selecting a

subset of the original features, allows for a human understanding of which features are the most useful or powerful. Specifically, I will study the problem of feature ranking, which considers the features independently according to some scoring function, thus revealing properties of the individual features by which they may be compared and ranked.

One of the central challenges of multiple-instance learning is that while multisets of vectors are labeled, the vectors themselves typically are not. A standard assumption made about the connection between the vector labels and the multiset labels is that the label of a multiset is true if the label of one of its vectors is true, and false only if all the labels of its vectors are false. This assumption matches the intuition of a wide variety of multiple-instance learning applications; for example, in the book-labeling task, an image is labeled to have the property "book" if at least one of the entities in the image is a book. In order to handle the issue of the instance labels not being available, many algorithms for multiple-instance learning have used the strategy of giving each instance the label of its bag, which can be shown to convert the data set into an ordinary supervised learning data set with one-sided noise [Blum and Kalai, 1998]. Recent results have shown the seemingly counter-intuitive result that this "bag-labeled instances" strategy can work well for classification [Ray and Craven, 2005, Doran and Ray, 2014]. In this thesis, I will use the bag-labeled instances strategy to perform feature selection for multiple-instance learning, a study of which will reveal certain properties about when this strategy can lead to good performance.

To perform feature ranking, it is necessary to compute a score for an individual feature. In order to analyze scoring functions in this context, I will show that the process of assigning a score to a feature can be conceptualized as assigning a score to a one-feature dataset, where the labels are assigned based on a probabilistic concept. A probabilistic concept is a function which assigns to an object a probability that it will be positive-labeled, and then gives each appearance of the object a label based on

that probability [Kearns and Schapire, 1993]. This is in contrast with deterministic concepts, which always assign the same label to an object. To give an example of a probabilistic concept, suppose an object describes the weather conditions of a particular day, and a label represents whether or not it will rain. If the object representing those weather conditions appears multiple times in a data set, it is reasonable to expect that it may have a positive label in some appearances and a negative label in others. In this scenario, the object can be described as having been labeled positive with some probability. In this thesis I will illustrate why probabilistic concepts allow for a valuable perspective from which to study feature selection, and analyze several scoring functions for multiple-instance feature selection using this perspective.

There are three main contributions of this thesis:

1. I provide a framework for studying feature selection by using a probabilistic concept to represent the labeling function on a single-feature axis. In this context I analyze feature ranking for multiple-instance learning using the strategy of bag-labeled instances with accuracy as a scoring function, showing that in many situations it is possible to recover the most useful features using this approach.

2. I provide a similar analysis for multiple-instance feature ranking using the area under the ROC curve as a scoring function, showing similar results.

3. I perform an empirical evaluation, which reflects the theoretical description for when my multiple-instance feature ranking is likely to perform well, and which demonstrates that this approach is competitive with existing dimensionality reduction algorithms for multiple-instance learning and can often improve the performance of a classifier.

This thesis is organized as follows: In chapter two, I will introduce the relevant

concepts necessary for an understanding of the theoretical results I present. In chapter three I introduce the feature selection technique to be used for multiple-instance learning, describe a theoretical framework for studying feature selection when the bag-labeled instances strategy is used, and analyze the quality of the feature set learned when the accuracy of a deterministic concept is used as the scoring function for a feature. In chapter four I will extend this analysis to the case when the scoring function is the area under the ROC curve of a learned ranking concept. In chapter five, I will empirically evaluate the two approaches on a collection of common multiple-instance learning datasets and compare them to several pre-existing approaches. Finally, in chapter six I will describe some areas of interest for future work.

Chapter 2

Background and Related Work

In this chapter I will describe the relevant concepts necessary to understand the theoretical and empirical work I present. First, supervised learning will be described in the context of machine learning and, more broadly, the field of artificial intelligence, with support vector machines illustrated as an example of a supervised learning technique. Next, several key concepts from learning theory will be summarized, including probabilistic concepts, which are used extensively in the theoretical analysis. After that, multiple-instance learning will be described in detail, with particular focus on generative models and the idea of transforming multiple-instance datasets into supervised datasets with one-sided noise by using bag-labeled instances. Next, dimensionality reduction will be explained, with examples of both feature selection and feature construction techniques given. Finally, the evaluation metric of area under the ROC curve will be illustrated. Several related algorithms for dimensionality reduction in the context of multiple-instance learning will also be described and compared at the end of this chapter.

2.1 Supervised Learning

In the domain of artificial intelligence, many fields have made progress by writing an approximation of human knowledge into a machine, so that it has an internal understanding of the rules and intuitions that govern human behavior in that field. For instance, chess computers are able to analyze and evaluate board positions using certain rules based on human knowledge that they have been programmed to understand, such as "a queen is more valuable than a pawn" [Botvinnik, 1983, Campbell et al., 2002]. Human chess players, however, are not born with this sort of knowledge; instead, they typically learn it by playing many games of chess and developing an understanding of what behaviors will lead to success. This ability of humans to learn from experience is what the field of *machine learning* attempts to replicate. "Experience" in this context is some sort of data, which approximates the data available to humans to varying extents; for instance, self-driving cars attempt to learn a model for good driving based on the data they receive from the road, which can come in the form of physical indications and a camera, among other signals, similar to how human drivers learn an intuition for driving based on their physical experiences in the car and what they can see out the windshield [Leonard et al., 2008].

Within the study of machine learning, there are several different approaches that attempt to approximate different scenarios of human learning. Perhaps the most common and well-studied is *supervised learning*, where the learner is given a set of inputs and outputs and the goal is to learn to predict, for a new input, what the correct output will be. This aligns with human learning of correct behaviors based on memory; humans remember previous experiences and the results of those experiences, and use that knowledge to tackle unfamiliar scenarios in a way that seems reasonable. Numerous learning tasks have been represented as cases of supervised learning, including speech recognition, detecting faces in images, and medical diagnosis [Murphy,

2012, Mohri et al., 2012]. To draw a parallel with human learning, doctors are typically able to diagnose diseases based on the symptoms that match with diseases that they have experience with either from their education or their previous patients. The goal of supervised learning is for a computer, provided with the same set of patients and diagnoses, to be able to also give accurate diagnoses for new patients.

To formalize this, a supervised learning algorithm is presented with a set of n *instances* (also called *examples*) X and the corresponding labels for each instance Y. An instance is represented as a vector containing d different measurements or *features*. While features may be any type of measurement, in this thesis, only the common case of exclusively real-valued measurements will be considered. When this is the case, because the feature values for an instance can be envisioned as coordinates on a Euclidean plane, features are often called *dimensions* and the instance is said to exist in a particular *feature space* $\mathbb{R}^d$. Each instance x_i is said to have been drawn from the space of all possible instances, or *instance space*, $\mathcal{X} \subseteq \mathbb{R}^d$, according to some probability distribution over the possible instances $D_{\mathcal{X}}$.

The labels can take on a variety of values, and the case where labels are real-valued is the well-studied problem of regression; however, in this thesis I will study the problem of classification, where each label can take on one of two values, such as the boolean values 0 and 1. The space of possible labels $\mathcal{Y}$, then, is limited to those two values, i.e. $\mathcal{Y} = \{0, 1\}$. Supervised learning assumes that the instances are labeled with a deterministic function $f : \mathcal{X} \mapsto \mathcal{Y}$, and generally attempts to find a good approximation of that function given the limited data provided. Functions that map from instances to their labels are called *concepts*, and the function f is known as the *target concept* while the approximation of f that the learner finds is called the *hypothesis* h.

To continue with the analogy of diagnosis, a doctor may be trying to diagnose whether or not someone has cancer, and thus the label space is limited to "has

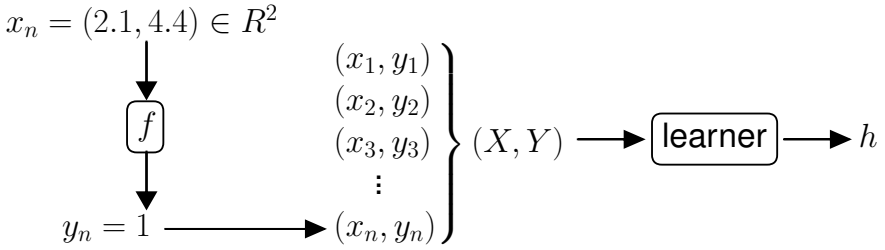


Figure 2.1: An illustration of the process of supervised learning. Instances are labeled with the unknown target concept f to form a dataset of n examples (X, Y), which is used by a supervised learning algorithm to find a hypothesis h that approximates f.

cancer" (1) or "does not have cancer" (0). Each patient is an instance, and the different measurements, tests and attributes the doctor takes into account provide the features — for instance, the age of a person might be one feature. The instance space $\mathcal{X}$ is the space of all possible people; however, certain types of people are more common than others — it is more common to see 40-year-old patients than 100-year-olds. Thus the distribution $D_{\mathcal{X}}$ will provide more instances of common types of patients. The laws of nature that determine, based on the doctor's analysis, whether or not someone has cancer have an analogy in f, and the doctor (or learning machine) would ideally like to learn a set of rules, or hypothesis h, that approximates those laws as closely as possible so that diagnoses can be right as often as possible. This process of learning is illustrated in Figure 2.1

Several questions immediately present themselves. What does it mean to find a "good" approximation of the labeling function f? How will such an approximation be learned? And how can abstract concepts and methods of human understanding be consistently represented using sets of features?

What constitutes a good approximation of f is a question that can have several answers depending on the scenario in question. Different measurements of goodness may value different things — for instance, in medical diagnosis, it is much more important to be correct about people who do have cancer than it is to be correct

about those who don't, as deaths from overly-optimistic diagnoses are much more consequential than inconveniences of overly-pessimistic diagnoses. A common measurement of the goodness of a hypothesis is presented in section 2.5. The question of how to learn a good approximation is a vibrant field of study, and new algorithms are frequently proposed; recently, support vector machines (subsection 2.1.1) have found good success on a wide variety of tasks. Finally, understanding how to represent concepts is key to learning, and in fact it is one of the areas where these artificial intelligence algorithms are still heavily dependent on human expertise, as humans must select and fine-tune the appropriate features for learning. Even in this field, however, there still exist ways for computers to make their own decisions about how data should be represented, one of which is studied in this thesis.

2.1.1 Support Vector Machines

The *support vector machine* (SVM) is an example of a supervised learning algorithm [Vapnik, 1982]. In its most basic form, the support vector machine learns a line that attempts to separate the positive and negative instances as best as possible. Consider Figure 2.2, which demonstrates a dataset and the SVM that classifies it. The *decision boundary*, indicated by a solid line, is the line that separates the positive points from the negative points, taking the linear form $w \cdot x + b = 0$ for some weight vector w and intercept constant b, and the *support vectors* are the positive and negative points closest to the decision boundary. The positive and negative support vectors each have a hyperplane parallel to the decision boundary running through them, indicated by the dashed lines. The support vector machine attempts to find a line that maximizes the *margin*, the distance between the positive and negative hyperplanes.

In its most basic form, the optimization problem used to find an optimal SVM is:

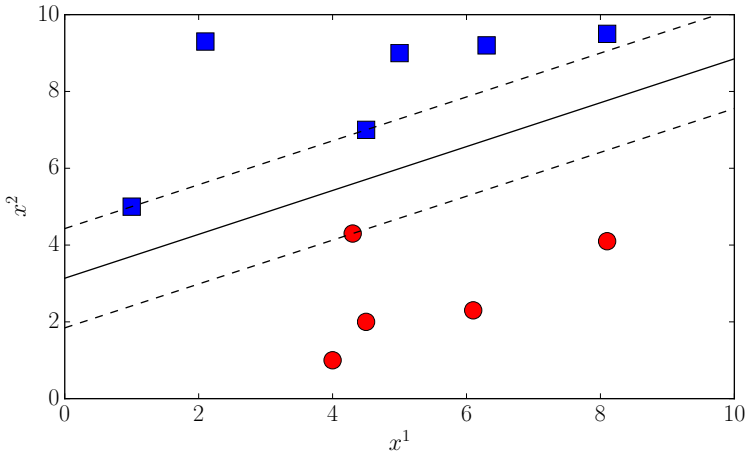


Figure 2.2: A support vector machine classifier, which separates the positive points (blue squares) from the negative points (red circles). The hyperplanes are indicated by dashed lines.

$$\begin{aligned} &\min_{w,b} \frac{1}{2}||w||^2 \\ &\text{s.t. } y_i(w \cdot x_i - b) \geq 1 \end{aligned} \tag{2.1}$$

where (x_i, y_i) is a particular instance/label pair, $i = 1...n$, and the labels are assumed to be from $\mathcal{Y} = \{-1, 1\}$. $||w||$ decreases as the margin increases, so intuitively this is maximizing the margin as much as possible while ensuring that all points are classified correctly.

In general, it may not be possible to perfectly classify every point. When this is the case, the best SVM is one that satisfies its goals while minimizing the distance between misclassified points and the decision boundary. The soft-margin formulation of the SVM introduces a slack variable ξ_i for each instance x_i to represent the error of the SVM on x_i, and then tries to minimize the sum of the slack variables while simultaneously maximizing the margin, with a tradeoff parameter C controlling the

relative importances of the two goals:

$$\begin{aligned} &\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_i \xi_i \\ &\text{s.t. } y_i(w \cdot x_i - b) \geq 1 - \xi_i \\ &\xi_i \geq 0 \end{aligned} \tag{2.2}$$

This formulation works for finding *linear classifiers*, which represent the decision boundary in terms of a dot product between a weight vector and an instance, causing it to take the form of a straight line in the instance space. A key strength of support vector machines comes from their ability to represent nonlinear decision boundaries in the instance space. Conceptually, this is done by mapping the data from the instance space into a new feature space, which may have a much higher dimensionality than the instance space, and finding a linear classifier in this new feature space. In general, the high dimensionality of the new feature space makes such a transformation impractical; however, it is possible to formulate the SVM optimization problem above so that only a measurement of the similarity between the representations of instances in the new feature space is required, not the representations themselves. For many feature spaces, this similarity measurement can be computed in closed form; such a measurement is called a *kernel*, written $k(x_1, x_2)$ [Tsuda and Scholkopf, 2004]. For example, the *radial basis function* (RBF) kernel measures the similarity between two points x_1, x_2 as the value of x_2 on a multivariate gaussian drawn in the instance space around x_1:

$$k(x_1, x_2) = e^{-\gamma||x_1 - x_2||^2} \tag{2.3}$$

By converting to the dual formulation of the SVM problem, all uses of the points can

be represented by kernel functions:

$$\begin{aligned} &\min_{w,b,\xi} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) + C \sum_i \xi_i \\ &\text{s.t. } y_i(\sum_j \alpha_j k(x_j, x_i) - b) \geq 1 - \xi_i \\ &\xi_i \geq 0 \end{aligned} \tag{2.4}$$

2.2 Learning Theory

In this section, I will introduce some principles fundamental to the theoretical study of machine learning. The Probably Approximately Correct (PAC) learning framework gives a general description of what it means to learn a concept class, while the Vapnik-Chervonenkis (VC) dimension provides a measurement of the difficulty of learning a concept class, and Empirical Risk Minimization (ERM) provides a general algorithm for performing that learning. I describe a complexity bound on the number of instances required for ERM to PAC-learn a concept class of finite VC dimension. Also included is a discussion of probabilistic concepts, as well as the pseudo-dimension, an analogy of the VC dimension for probabilistic concept classes.

2.2.1 Probably Approximately Correct Learning

Not all learning problems are created equal — some will be relatively easy for a machine to handle, while others will be much more challenging, perhaps prohibitively so. Analyzing what exactly it means to be able to "learn" something, as well as in which scenarios learning is possible, requires a certain degree of formality, which is provided by the *Probably Approximately Correct* (PAC) computational learning framework [Valiant, 1984, Mohri et al., 2012].

Recall from section 2.1 that in supervised learning, instances are labeled with a

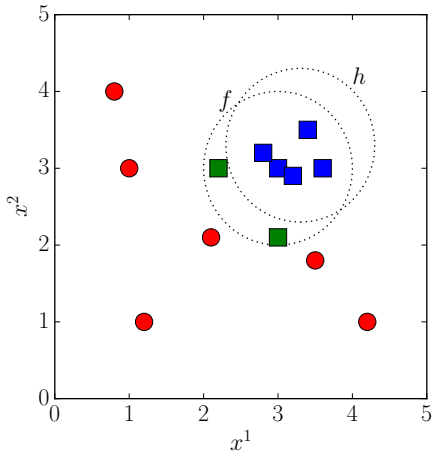


Figure 2.3: Illustration of concept approximation. f is the target concept, and h is the hypothesis when an algorithm is trained to find a unit circle concept on just the blue and red instances, with blue squares indicating positive instances and red circles indicating negative instances. The generalization error is represented by the regions covered by only one of f or h. If the positive points indicated by green squares were included in the dataset, h would come closer to perfectly matching f by moving down and left to include them. Given an infinite number of instances, f would be clearly visible.

deterministic target concept f. Ideally it would be possible for a learning algorithm to precisely replicate this function; however, this may not always be possible. Not only can f be arbitrarily complex, but the learner must discover it using a limited sample of n instances. As shown in Figure 2.3, when there are many concepts that seem perfect given the few examples provided, it is hard to pick the exact concept f. In lieu of perfectly representing f, the next best result would be to find a hypothesis h that approximates f as closely as possible. A natural way of measuring how closely h approximates f is to consider how often h will predict a different output from f. This measurement is known as the *generalization error* and is measured with respect to the distribution over instances $D_{\mathcal{X}}$, because if some instances are more common than others then it is more important that h agree with f on those instances:

Definition 2.1 (Generalization error) *Given an instance distribution $D_{\mathcal{X}}$, a tar-*

get concept f and a hypothesis concept h, the generalization error, or risk, of h is

$$R(h) = \Pr_{x \in D_{\mathcal{X}}}[h(x) \neq f(x)] = \mathbb{E}_{x \in D_{\mathcal{X}}}\left[\mathbb{1}[h(x) \neq f(x)]\right]$$

In this definition, the *risk* is the expected value over the instance distribution of the *loss function* $\mathbb{1}[h(x) \neq f(x)]$. In some scenarios, the loss function may be changed; for instance, in some cases it is convenient to use the *quadratic loss*, $(h(x) - f(x))^2$ as an alternative loss function.

If f can not be found, then the next best scenario would be if it was always possible to find an h that has a very low generalization error. The instances used to construct the dataset, however, are assumed to be *independently and identically distributed* (IID). What this means is that they are all drawn from the same distribution $D_{\mathcal{X}}$, and when constructing the dataset the examples drawn so far have no impact on what the next example drawn will be. Because instances are IID, it is theoretically possible to get difficult datasets for the learner; for example, a dataset constructed of points drawn only from a certain region of the instance space would make it very difficult to form hypotheses regarding the regions about which the learner has no information. Even for an arbitrary number of instances n, very bad datasets are still theoretically possible, if highly improbable. Rather than demanding that a low-error h be found all the time, then, it is more sensible to request that such an h be found *most* of the time.

Finally, it is not very helpful if an algorithm can learn an h that has low error most of the time if that algorithm requires an extremely large number of instances to do so. Therefore, it is desirable for an algorithm to learn such a hypothesis h using a restricted number of examples. The PAC learning framework takes these three goals — finding a concept with low generalization error most of the time with not too many examples — as its definition of learnability.

A definition of learnability that extends only to a particular target concept f is not very useful, because learnability will have to be re-evaluated for every new target concept. Instead, the PAC learning framework considers groups of concepts that share similar characteristics. A *concept class* is a particular set of concepts. For example, the set of all circles of radius 1 may be a concept class $\mathcal{F}$, and if the target concept f is a circle of radius 1 then $f \in \mathcal{F}$. PAC results extend to entire classes of concepts, saying that all concepts with a particular set of properties are learnable.

Definition 2.2 (PAC Learning) *A concept class $\mathcal{F}$ is said to be PAC-learnable if there exists an algorithm $\mathcal{A}$ such that for any $\epsilon > 0$ and $\delta > 0$, for all distributions $D_{\mathcal{X}}$ on $\mathcal{X}$, for all target concepts $c \in \mathcal{C}$, given a sample X of size $n = O\left(\text{poly}(\frac{1}{\epsilon}, \frac{1}{\delta})\right)$, $\mathcal{A}$ will produce a concept h s.t.*

$$\Pr[R(h) \leq \epsilon] \geq 1 - \delta$$

2.2.2 VC dimension

PAC learning gives a way to describe whether or not concept classes are "learnable", but not whether certain concept classes are more learnable than others; however, clearly not all concept classes are equally difficult to learn. Consider a concept class that only contains two concepts, each representing a different region of $\mathbb{R}^2$ where points are positive, with the two regions partially overlapping. It would be surprising if a large number of points were required to determine which of the two is the target concept, because the first point drawn whose label can only be explain by one of the concepts — for instance, a positive point from a region only covered by one of the concepts — will be sufficient to determine the target concept. If the concept class instead contained two hundred possible concepts, again described as partially overlapping regions of $\mathbb{R}^2$, then intuitively it would take more examples to determine the correct concept.

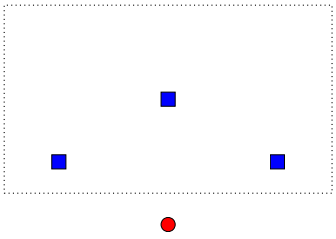
Most interesting concept classes do not, however, contain a finite number of concepts. The simple class of axis-aligned rectangles in $\mathbb{R}^2$, for example, contains an infinite number of concepts, because each rectangle is described by its real-valued corners, and the set of real numbers is infinite. This concept class seems like it must be easier to learn, however, than the concept class of all convex k-gons in $\mathbb{R}^2$, since many more concepts can be represented in the latter class even though both are infinite. It would be useful to have some way to measure the complexity of infinite concept classes in order to quantify this intuition.

The *Vapnik-Chervonenkis dimension* (VC dimension) provides such a measurement using the concept of *shattering* [Vapnik and Chervonenkis, 1971, Vapnik, 1995]. Suppose n points are selected from the instance space on which a concept class $\mathcal{F}$ is defined; there are 2^n possible assignments of 0/1 labels to these points. If all possible such assignments can be represented by a concept in $\mathcal{F}$, then $\mathcal{F}$ is said to shatter the set of points.

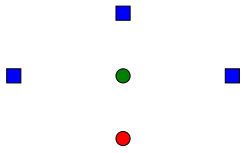
Definition 2.3 (VC dimension) *The VC dimension of a concept class $\mathcal{F}$ $VC(\mathcal{F})$ is the size of the largest set of points that can be shattered by $\mathcal{F}$. If $\mathcal{F}$ can shatter arbitrarily large finite sets of points, then $VC(\mathcal{F}) = \infty$.*

An important part of this definition is that $\mathcal{F}$ does not have to shatter all sets of points of size $VC(\mathcal{F})$, only one possible arrangement of $VC(\mathcal{F})$ points. Thus when showing a lower bound on the VC dimension, one must only show that there exists some set of n points that are shattered by $\mathcal{F}$, while for proving an upper bound, it is necessary to show that that there does not exist such a set.

The VC dimension gives a means of comparing the capacities of two infinite concept classes. For example, the VC dimension of the set of all axis-aligned rectangles is 4. If four points are arranged in a diamond, it is always possible to draw a rectangle that includes only a subset of them. No matter how five points are arranged, however, one point will always be "internal", and there is no way to draw a rectangle



(a) A possible labeling of four points, and the matching axis-aligned rectangle



(b) A possible labeling of five points, with the negative interior point marked in green.

Figure 2.4: VC dimension of the concept class of axis-aligned rectangles illustrated. A concept that perfectly represents the labeling of the four points is given. There is no axis-aligned rectangle concept that matches the labeling of the five points. These five points can be arranged any way, but there will always be an interior point which creates unrepresentable labelings and prevents shattering.

that includes only the four external points. This is illustrated in Figure 2.4. The VC dimension of the set of all convex k-gons, however, is $2k + 1$, which matches the intuition that this concept class is harder to learn than the class of axis-aligned rectangles [Kearns and Vazirani, 1994].

VC dimension is also a convenient way to measure the relative learnability of concept classes. Because PAC learning is searching for a hypothesis that is ϵ-close to the target concept f, the more flexible the concept class, the more complex it is possible for f to be, and therefore the more difficult it is to be sure that h is truly a good approximation of f. More complex concept classes, therefore, will require more examples to get a good approximation, with the following specific complexity bound on the number of examples required for PAC learning with PAC parameters ϵ, δ [Blumer et al., 1989]:

$$n \geq \max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{8VC(\mathcal{F})}{\epsilon}\log\frac{13}{\epsilon}\right) = O\left(\frac{1}{\epsilon}\left(\log\frac{1}{\delta} + VC(\mathcal{F})\log\frac{1}{\epsilon}\right)\right) \tag{2.5}$$

So long as $VC(\mathcal{F})$ is finite, then, $\mathcal{F}$ can be PAC-learned. Intuitively, having an

infinite VC dimension means that a concept class contains infinitely complex concepts, so it makes sense that such concepts would probably be impossible to approximate confidently.

2.2.3 Empirical Risk Minimization

A considerable number of learning algorithms have been proposed, such as the support vector machines of subsection 2.1.1. These learning algorithms, however, are structured around certain assumptions and typically are restricted to a certain set of concept classes. This is inconvenient for proving learning theoretic results, which often generalize to all concept classes that fulfill certain requirements, such as all concept classes of finite VC dimension. *Empirical risk minimization* (ERM) solves this problem by describing a general principle for approximating the target concept within an arbitrary concept class [Vapnik, 1992, Vapnik, 1995].

Although a concept class may technically be infinite, for a particular sample of size n there are 2^n equivalence classes of concepts that predict the same output on the sample. If the labels given by the target concept f are available for every example, and if f is in the concept class $\mathcal{F}$ being searched over, it makes the most sense to select a *consistent hypothesis*, a hypothesis h that matches f on every example. If f is not in the concept class $\mathcal{F}$, then it makes sense to select the hypothesis h that minimizes the number of examples in the sample for which it disagrees with the target, a measurement known as the *empirical risk*:

Definition 2.4 (Empirical Risk) *Given a sample X of size n, a target concept f and a hypothesis concept h, the empirical error, or empirical risk, of h is*

$$\hat{R}(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[h(x_i) \neq f(x_i)]$$

Note that, unlike the generalization error, this is only an estimate of the accuracy

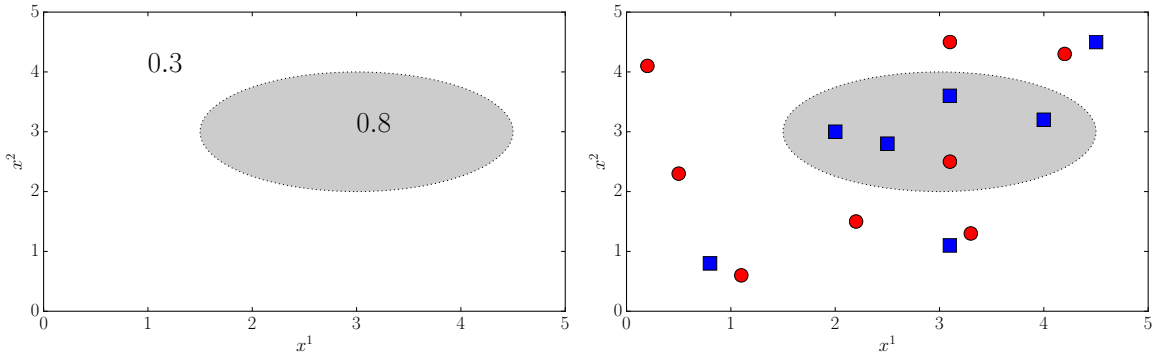
of a hypothesis; if only one example is available, the fact that a hypothesis matches the target on that example says virtually nothing about how accurate of a concept it actually is. Minimizing the empirical risk allows for quick convergence within a concept class to a good hypothesis so long as the VC dimension of the concept class is finite [Vapnik, 1999]. Furthermore, the loss function $\mathbb{1}[h(x_i) \neq f(x_i)]$ in the formula for empirical risk can be replaced with any loss function; indeed, several common algorithms can be replicated by substituting the appropriate loss function.

Empirical risk minimization's requirement of finite VC dimension is the same as in the complexity bound for PAC learning (Equation 2.5). Combined with that complexity bound, ERM can theoretically be used to PAC-learn an arbitrary concept class $\mathcal{F}$, so long as $VC(\mathcal{F})$ is finite. Practically, of course, certain assumptions can often be made about the nature of either the target concept or the dataset that allow for the application of faster, more specialized algorithms, such as support vector machines.

2.2.4 Probabilistic Concepts

Thus far, everything discussed has been in the context of deterministic labeling schemes. For certain problems, however, this may not be the most natural representation. A probabilistic concept, or *p-concept*, is a function $c : \mathcal{X} \mapsto [0, 1]$ that represents the probability that a positive label will be observed on an input x [Kearns and Schapire, 1993]. For example, if $c(x) = 0.8$, then when example x is drawn, there is an 80% chance that it will be given a positive label y and a 20% chance of a negative label. This can be seen in Figure 2.5, where within the circular region, where $c(x) = 0.8$, 80% of the points are given a positive label, while outside the region, $c(x) = 0.3$, so 30% of the points are given a positive label.

Probabilistic concepts are ideal for modeling situations where there is uncertainty over the outcome of an event, but only the outcome is observed. An example is



(a) Example probabilistic concept

(b) Dataset labeled with the p-concept

Figure 2.5: An example dataset constructed according to a probabilistic concept. Positive and negative-labeled instances are represented by blue squares and red circles, respectively.

predicting whether or not it will rain: while only the true or false output is observed (whether or not it rains), this behavior is governed by some invisible probabilistic function. It is also necessary for there to be some sort of structure to the uncertainty, because in general an instance x will not be seen multiple times, so it is impossible to estimate the probabilistic function c by taking the fraction of times x is true; rather, by sampling enough points and noticing trends in the true and false observations, it becomes possible to estimate the structure of c.

With deterministic concepts, there is a simple goal of trying to find a hypothesis that generates the same outputs as the target f. With p-concepts, there are two possible goals. The first is, similarly, to attempt to learn a deterministic *decision rule* that will predict 1/0 labels as accurately as possible. The second is to learn a real-valued *model of probability* that approximates the actual p-concept c as close as possible. In the context of weather prediction, learning a good decision rule means trying to say whether or not it will rain and being right as often as possible, while learning a model of probability means trying to accurately make statements such as "there is an 80% chance of rain."

Although generalization error is typically used to measure how good a determinis-

tic concept is, p-concepts use slightly different measurements of risk. When learning a decision rule, the risk measurement used is the predictive error, which is the probability that a hypothesis h will misclassify a randomly-drawn point, $\mathbb{E}_{x \in D_{\mathcal{X}}}[h(x) \neq b(x)]$, where $b(x)$ is the label assigned by the p-concept c to x. This is similar to the generalization error, except that the label to be matched is now determined probabilistically. When learning a model of probability, there are several equally useful loss functions [Kearns and Schapire, 1993], with the most convenient being the quadratic loss between c and h, which changes the risk measurement to $\mathbb{E}_{x \in D_{\mathcal{X}}}[(h(x) - c(x))^2]$. Using these risk measurements, a similar framework to the PAC learning framework (Definition 2.2) can be defined for p-concepts.

2.2.5 Pseudo-dimension

The VC dimension is a convenient way to measure the learnability of a concept class containing deterministic functions; however, if a concept class contains real-valued probabilistic concepts, a new measurement is necessary. The *pseudo-dimension* of a probabilistic concept class $\mathcal{C}$, $PD(\mathcal{C})$, is defined in exactly the same way as the VC dimension, except for a different definition of shattering. Suppose n points are selected from the instance space. For each point x_i, let c_i^* be the probabilistic value of x_i. The p-concept class $\mathcal{C}$ shatters this set of points if for every one of the 2^n possible assignments of 0/1 labels y to the points there is a concept $c \in \mathcal{C}$ s.t. $c(x_i) \geq c_i^*$ if $y_i = 1$ and $c(x_i) < c_i^*$ if $y_i = 0$ [Haussler, 1992, Kearns and Schapire, 1993]. Essentially, there is some hypothesis p-concept that predicts the label of the point with at least as much confidence as the probabilistic value c_i^*, which is a looser condition than the exact matching of predictions required in the definition of VC dimension. As in VC dimension, $PD(\mathcal{C})$ is the size of the largest set of points that can be shattered in this way, and it is infinite if no such limit exists. If $\mathcal{C}$ only allows for concepts that output 0 or 1, which are essentially deterministic concepts, then the pseudo-dimension

is equal to the VC dimension [Haussler, 1992].

A p-concept class $\mathcal{C}$ with finite pseudo-dimension $PD(\mathcal{C})$ can be PAC-learned, with the following upper bound on the number of examples required [Kearns and Schapire, 1993]:

$$n \geq \frac{64}{\epsilon^2}\left(2PD(\mathcal{C})\log\frac{16e}{\epsilon} + \log\frac{8}{\delta}\right) = O\left(\frac{1}{\epsilon^2}\left(\log\frac{1}{\delta} + PD(\mathcal{C})\log\frac{1}{\epsilon}\right)\right) \tag{2.6}$$

This is identical to the bound for deterministic concept classes given (Equation 2.5), except for the extra factor of $\frac{1}{\epsilon}$ and the substitution of pseudo-dimension for VC dimension. The similarity of their complexity bounds is not the only similarity between pseudo-dimension and VC dimension; for many specific concept classes, the two measurements are closely related. For example, in the case of linear concept classes, the pseudo-dimension is equal to the VC dimension [Kearns and Schapire, 1993].

2.3 Multiple-Instance Learning

In the typical supervised learning scenario, a piece of data is represented by a single vector $x \in \mathcal{X}$, along with its corresponding label $y \in \mathcal{Y}$. This way of representing data, however, may not always be the most convenient or intuitive, and in some scenarios the information required to represent data in this way may not be available. *Multiple-instance learning* is a more general framework for supervised learning; rather than being presented with single instances, the learner is presented with sets of instances, called *bags*, each of which may contain an arbitrary number of instances [Dietterich et al., 1997]. Labels are assigned to bags rather than to the instances themselves, although as in supervised learning it is assumed that the information contained in the instances is what determines the label for the set.

Multiple-instance learning is strictly a more general problem than standard su-

pervised learning, because any supervised learning problem can be described as a multiple-instance learning problem where every bag contains only one instance. Problems involving labeled bags of instances rather than labeled instances, however, are much more naturally represented in the multiple-instance framework than in the supervised framework. Any attempt to convert bags from such a problem to vectors results in a loss of information; for example, concatenating all the instances into one large vector would eliminate the distinction between different instances and impose an artificial ordering on the instances within a particular bag. Such a representation is especially uncomfortable when considering that bags in multiple-instance learning may contain different numbers of instances, while supervised learning requires all instances to be of the same dimensionality.

Multiple-instance learning was originally proposed as a way of representing the 3-Dimensional Quantitative Structure-Activity Relationship (3D-QSAR) problem [Dietterich et al., 1997]. In this problem, a learner attempts to determine which molecules will bind well with a target molecule, indicating a promising molecule for drug design research. The binding strength of a molecule is a function of its shape; however, complicating the issue is the fact that a single molecule can take on any number of shapes simply by rotating its internal bonds. While a molecule may be promising, some or even most of its shapes may not be.

Because the information available about binding strength only extends to the level of individual molecules, rather than individual shapes of molecules, this problem is convenient to represent in the multiple-instance learning framework. Each molecule is represented by a bag of instances, where each instance describes a particular shape it can take on. A bag is labeled positive if the molecule binds well with the target, and negative if it does not. While labels for the individual instances are not available, the label of the bag is a function of the invisible labels of the instances, because a molecule will be labeled positive so long as one of its shapes is able to bind strongly, while it

will be labeled negative only if none of its possible shapes can bind strongly. Because the object to be classified, a molecule, is best represented by a bag of instances, this problem is a natural fit for multiple-instance learning. Other problems that can be represented in the multiple-instance framework include image classification [Maron and Ratan, 1998, Andrews et al., 2003], text classification [Zhou et al., 2009], audio recognition for bird songs [Briggs et al., 2012], and protein sequence identification [Tao et al., 2004]

More formally, let B_i represent a bag containing m_i instance $x_{i1}, ..., x_{im_i}$. B_i is given a label $Y_i = F(B_i)$ according to labeling function $F : \mathcal{X}^* \mapsto \mathcal{Y}$, where $\mathcal{X}^* = \bigcup_{m=1}^{\infty} \mathcal{X}^m$ describes all possible finite sets of instances; that is, all possible bags. A key distinction here is between the instance-level labeling function f, which labels individual instances, and the bag-level labeling function F, which labels bags of instances. It is common to restrict the multiple-instance learning problem by codifying the relationship between f and F.

A large amount of multiple-instance research has focused on the problem when restricted by the *standard MI assumption*, which states that the bag is to be given a true label if at least one of the instances it contains has a true label, while it is given a false label only if all its instances are false. Logically, this means that $F(B_i) = \bigvee_{j=1}^{m_i} f(x_{ij})$, the label of a bag is the disjunction of the labels of its instances. This is illustrated in Figure 2.6, where bag B_1 is given a positive label because it contains two positive instances. Other, looser assumptions about the relationship between f and F have been proposed [Foulds and Frank, 2010], and there are some assumptions about F that circumvent any notion of instance labeling whatsoever, including the generalized multiple-instance learning assumption, which considers F to be based on the geometric locations of the instances in a bag rather than any labeling of those instances [Scott et al., 2003]. The standard MI assumption, however, has a wide range of applications; for example, it applies to 3D-QSAR, where the label of a

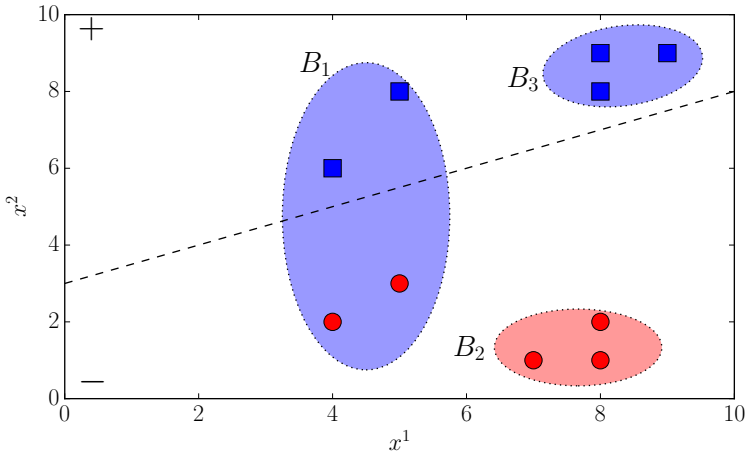


Figure 2.6: Example multiple-instance learning dataset, with the dashed line representing the unknown instance classifier f. B_2 and B_3 contain all negative or positive points, so their labels are 0 and 1, respectively. B_1 contains both positive and negative points; under the standard MI assumption, it would be given a label of 1.

molecule indicates whether any shape of that molecule can bind strongly.

From a theoretical standpoint, multiple-instance learning represents an intermediate point between attribute-value learning and inductive-logic programming [De Raedt, 1998]. Specifically, in attribute-value learning, each example is a propositional conjunction, while in inductive-logic programming, each example is a first-order conjunction. In multiple-instance learning, each example is a disjunction of propositional conjunctions, which is capable of representing every attribute-value problem but also some new problems. Although inductive-logic programming and attribute-value learning are very different problems, studying multiple-instance learning has the added benefit of providing an understanding of how techniques and results can change over the transition between the two.

2.3.1 Generative Models for Multiple-Instance Learning

In supervised learning, it is generally assumed that the dataset is constructed by first sampling each instance independently from the instance distribution $D_{\mathcal{X}}$ and then labeling each instance x_i with the label $f(x_i) = y_i$. In the case of multiple-instance

learning, it is not quite as simple, because the construct of bags must be taken into consideration as well as the method of assigning labels to bags. The assumptions about, and restrictions on, the creation of a dataset are summarized by a *generative model*.

The earliest generative model for multiple-instance learning, the IID r-tuple model, followed straightforwardly from the usual model for supervised learning [Blum and Kalai, 1998]. Each bag is composed of r instances drawn independently from the underlying instance distribution $D_{\mathcal{X}}$, and then given the label that is the disjunction of the labels of its instances. This model considerably restricts the problem by requiring each bag to have the same finite number of instances, $\forall i,\ m_i = r$; furthermore, the assumption that every instance in every bag is drawn from the same distribution ignores the fact that bags themselves represent entities with certain differing properties. In the 3D-QSAR problem described above, bags represent molecules which are represented by instances describing the different shapes they can take on; certainly the distribution of possible shapes will be different for different molecules. The limitations of this model have made theoretical justification of various proposed algorithms for the multiple-instance problem difficult, because while algorithms will typically be designed to take advantage of some assumed structure or other information common to the instances within a particular bag, the IID r-tuple model makes no such guarantee, instead describing the instances within a bag to be completely independent of the bag.

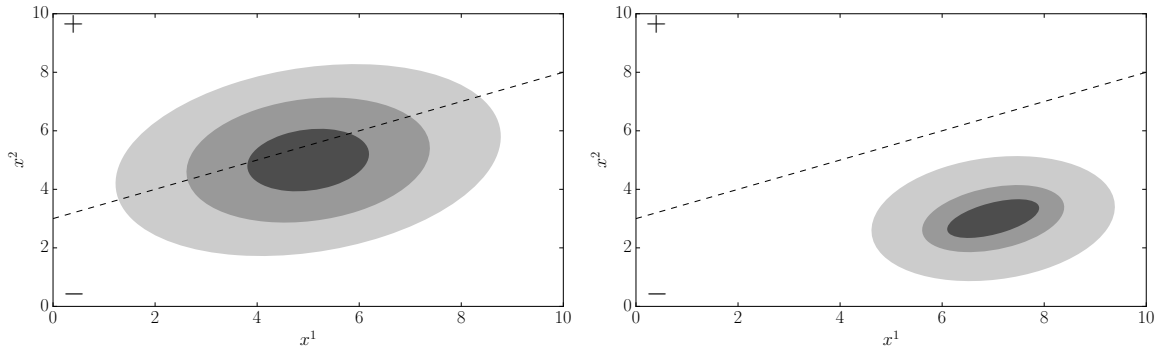
Sabato and Tishby present a model that has loosened some of the restrictions on the IID r-tuple model by instead assuming a distribution over bags $D_{\mathcal{B}}$, with each bag being a set of instances [Sabato and Tishby, 2012]. The bag size is also no longer restricted to r but allowed to vary in the finite range $m_i \in [1, R], R \in \mathbb{N}$. This model allows for an assumption of internal structure within a bag, because the distribution over bags can be restricted to only provide bags whose instances follow some internal

logic; for instance, in 3D-QSAR, one could assume under this model that the bags have been drawn from some distribution in such a way that every instance within a bag represents a possible shape of the same molecule, whereas under the IID r-tuple model such an assumption would not be supported by the model. This generative model, however, restricts the space of possible bags with a finite upper bound; in many scenarios, this is not realistic. In the 3D-QSAR problem, molecules can take on an infinite number of possible shapes, and there is no reason to exclude a bag containing $R + 1$ examples of shapes.

Recently, a generative model has been proposed that maintains the advantages described above while avoiding the disadvantage of limiting the size of possible bags. In this model, called the *bags-as-distributions* model, bags are viewed not as sets of instances but rather as distributions over the space of possible instances, with the representative instances inside a bag being drawn from the distribution particular to that bag [Doran and Ray, 2014]. In this model, $\mathcal{B}$ is the space of probability distributions over $\mathcal{X}$, $D_{\mathcal{B}}$ is a distribution over those distributions, which assigns to each bag a probability $\Pr_{\mathcal{B}}(B)$ of drawing that bag, and each bag is described by its bag-specific distribution $\Pr(x|B)$. The dataset is constructed by first sampling n distributions from $D_{\mathcal{B}}$, as shown in Figure 2.7, labeling each one with its label $Y_i = F(B_i)$, and then for each bag B_i drawing m_i instances from $\Pr(x|B_i)$. While there is no explicit instance distribution under this model, it can still be found by marginalizing out the bag-specific distributions; that is, $D_{\mathcal{X}}$ is described by:

$$\Pr_{\mathcal{X}}(x) = \int_{\mathcal{B}} \Pr(x|B) \mathrm{d}\Pr_{\mathcal{B}}(B) \tag{2.7}$$

One issue in multiple-instance learning, which forms the basis of several hardness proofs, is that if a particular negative instance only appears in positive bags, it is impossible to distinguish from an instance that is actually positive [Auer et al., 1997,



(a) B_1 (b) B_2

Figure 2.7: Two possible bags in the bags-as-distributions generative model. Darker regions indicate areas from which points are more likely to be drawn, and the dashed line indicates the instance-labeling function f. Were these the only two bags, $D_{\mathcal{B}}$ might be $\Pr_{\mathcal{B}}(B_1) = 0.7$, $\Pr_{\mathcal{B}}(B_2) = 0.3$. Because B_1 has a nonzero probability of drawing positive instances, its label $F(B_1) = 1$, even though the sample in the training set may have no positive instances. B_2's domain only includes negative instances, so its label is negative.

Diochnos et al., 2012, Kundakcioglu et al., 2010]. For example, consider the Content-Based Image Retrieval (CBIR) problem formulated for multiple-instance learning, where bags represent images, instances represent segments of an image, and a bag is true if it contains a particular concept, such as a waterfall [Maron and Ratan, 1998]. Given an infinite amount of images, it is expected that a learner will be able to understand that certain patches are characteristic of images of waterfalls; these are the positive instances. Suppose, however, that patches representing trees also only appear in images of waterfalls. While a tree has nothing to do with a waterfall, and is thus a negative instance, the fact that it only appears in positive bags means that it is perfectly reasonable for the learner to think that it represents the positive concept. This makes it impossible to distinguish true instances from these sorts of false instances, and learning becomes impossible.

Under the bags-as-distributions generative model, this could only occur if there were some negative-labeled instance that had a zero probability of appearing in a negative bag, so that it could only appear in positive bags. In order to avoid this

possibility, let γ be the minimum probability that a negative instance appears in a negative bag; the generative model requires that $\gamma > 0$. Thus the full set of constraints on this generative model is as follows:

Definition 2.5 (MI-GEN) *Given an instance distribution $D_{\mathcal{X}}$, instance-labeling function f, and $0 < \gamma \leq 1$, MI-GEN$(D_{\mathcal{X}}, f, \gamma)$ is the set of all bag distribution ($D_{\mathcal{B}}$ with measure $\Pr(B)$) and bag-labeling function (F) pairs satisfying the following conditions:*

1. $\Pr(x) = \int_{\mathcal{B}} \Pr(x|B) \mathrm{d}\Pr(B)$
2. $\forall x, B : f(x) = 1 \wedge F(B) = 0 \Rightarrow \Pr(x|B) = 0$
3. $\forall x : f(x) = 0 \Rightarrow \Pr(F(B) = 1|x) \leq 1 - \gamma$

This generative model allows for arbitrarily large bags in a sample by defining bags as distributions, from which an arbitrary number of samples can be drawn. It is more general than the IID r-tuple model, as that model can be represented as a special case of bags as distributions [Doran, 2015]. The same is true for the Sabato and Tishby model, with the exception that bags as distributions requires $\gamma > 0$, while the Sabato and Tishby model is presented without making this assumption.

2.3.2 Multiple-Instance Learning as a Case of Supervised Learning with One-Sided Noise

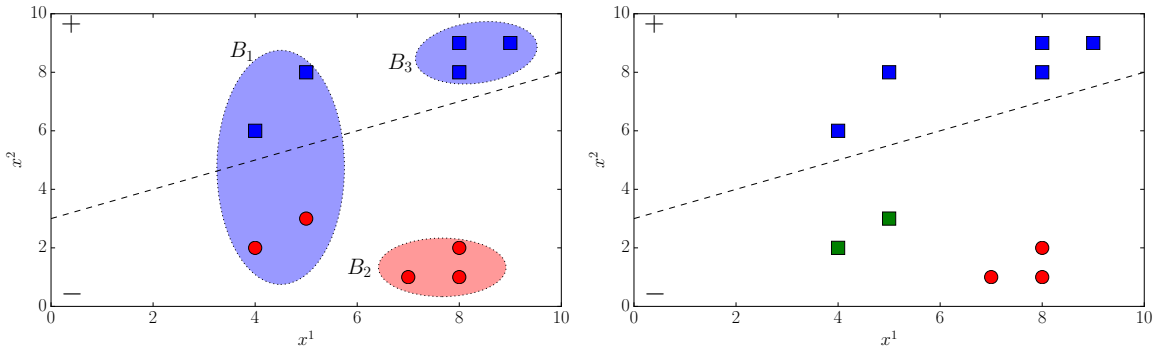
The standard case of supervised learning assumes that all instances in the data set are labeled according to the deterministic function f. In practice, however, the labeling on data may not always be correct, either due to human or technical errors or to ambiguities in the labeling scheme. In this scenario, the dataset is said to be noisy according to a classification noise rate η if a label y_i is flipped with probability η [Kearns and Vazirani, 1994]. In the context of the standard supervised generative model, this adds an extra step, where after an instance x_i is drawn and assigned label

$y_i = f(x_i)$, the pair (x_i, y_i) is included in the dataset with probability $(1 - \eta)$ and the pair $(x_i, \neg y_i)$ is included with probability η. If the noise occurs on both positive and negative instances, it is said to be two-sided; in contrast, one-sided noise describes the scenario when noise only occurs on either positive or negative instances but not both.

Shortly after the multiple-instance learning problem was proposed, it was shown that the problem could be converted to the problem of supervised learning under one-sided noise by assigning each instance the label of its bag [Blum and Kalai, 1998]. Because all the instances in a negative bag must be negative, assigning them a negative label will be correct; however, because a positive bag may contain both positive and negative instances, assigning a positive label to those instances will only be correct some of the time, and some negative instances will be mis-labeled as being positive. Thus if a negative-labeled instance is seen, it is certain that the label is correct, but if a positive-labeled instance is seen, it may actually be a mis-labeling of an instance that is actually negative but happened to be in a positive bag. An illustration of this conversion is given in Figure 2.8.

Under the bags-as-distributions generative model, the noise rate can be explicitly represented as $\eta = 1 - \gamma$, which is the exact noise rate in the case of uniform noise or an upper bound if the noise is allowed to vary by instance. To see this, note that the one-sided noise only applies to negative instances. A negative instance will appear in a negative bag with at least probability γ, which means it will appear in a positive bag with probability at most $1 - \gamma$. Under the labeling-scheme described, it will have its correct (negative) label if it is in a negative bag, and an incorrect (positive) label if it is in a positive bag; therefore, it will be incorrectly labeled with probability at most $1 - \gamma$, so this is the noise rate for one-sided noise.

Subsequent work on this idea of converting a multiple-instance dataset to a supervised dataset with one-sided noise has shown that supervised learning algorithms



(a) Multiple-instance learning problem

(b) Noisy supervised learning problem

Figure 2.8: A multiple-instance learning problem and its corresponding supervised learning problem with one-sided noise. In the second figure, squares are positive-labeled instances and circles are negative-labeled instances. Note that because all the instances in B_1 are given positive labels, the two green squares are labeled incorrectly.

on such a dataset can perform competitively with state-of-the-art multiple-instance learning algorithms operating on the original multiple-instance dataset [Ray and Craven, 2005]. Under the bags-as-distributions model, it is theoretically possible to learn a good ranking over instances (that is, one with high AUC — see section 2.5) using a polynomial number of examples [Doran and Ray, 2014]. While this supervised conversion approach to multiple-instance learning loses information and seems like it must be naive, the fact that it is competitive with more complex algorithms specific to the multiple-instance learning domain in both the classification and ranking tasks leads to the question: are there any other problems in the multiple-instance learning domain for which this idea could prove surprisingly useful?

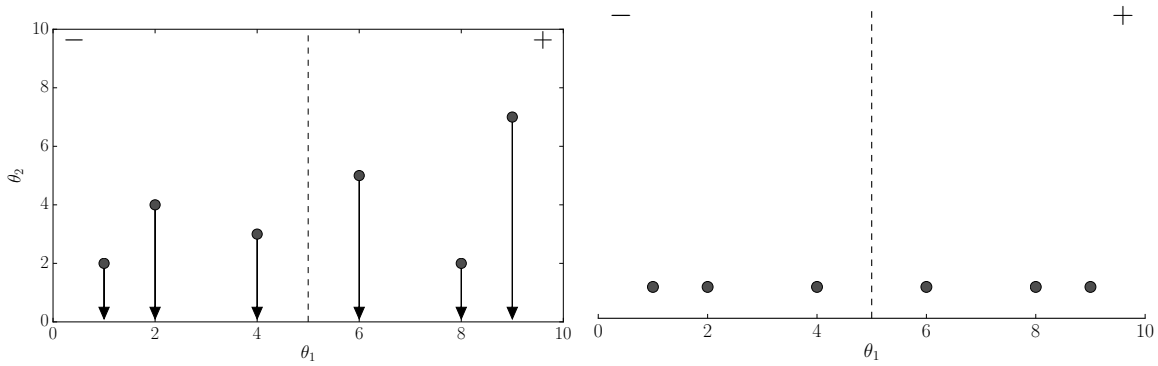
2.4 Dimensionality Reduction

For datasets where the number of features d is high, supervised learning will often perform poorly, a trend commonly referred to as the *curse of dimensionality* [Murphy, 2012]. For instance, classifiers constructed as a function of features may easily overfit the data when the number of features is substantially higher than the number of

instances. Furthermore, suppose all the data is normalized to fit inside a hypersphere of radius 2; because the volume of the data space increases exponentially with the number of features, a high number of features makes it difficult to sample a dataset with sufficient granularity to make trends evident. A high number of noisy, random or useless features can mislead the classifier on even relatively simple concepts by falsely correlating with the label on the sample. Even when a model is found that performs and generalizes well, a high number of features can make that model difficult to interpret in a human-understandable fashion.

Unfortunately, modern problem domains often suffer from this curse. As data collection techniques become more sophisticated and more detailed data becomes widely available, the number of features will only continue to rise. In addition, certain common domains of learning are particularly prone to this curse. In text classification, for instance, datasets are often constructed using the *bag-of-words model*, where a document is represented by a vector where each feature represents the frequency of a particular word. In this model, d is the size of the dictionary used, which may be in the hundreds of thousands or even higher [Guyon and Elisseeff, 2003]. In bioinformatics, analysis of genomes often leads to datasets where d is the number of genes in a particular microarray under analysis, which may number in the thousands [Xing et al., 2001].

Dimensionality reduction techniques attempt to mitigate this problem by substantially reducing the number of features used in the dataset from d to some lower value $d' \ll d$. A new feature set is selected to try to maximize some sort of signal, such as the separability of the positive and negative instances, in the resulting low-dimensional dataset. In addition to avoiding the curse of dimensionality, reducing the size of the dataset can also result in substantial savings of space and faster running times for algorithms whose time complexity is dependent on the dimensionality [Guyon and Elisseeff, 2003]. The new feature set may be composed of a subset of



(a) Dataset before feature selection

(b) Dataset after feature selection

Figure 2.9: Results of feature selection when the label is determined by one feature. Filtering reduces this feature space from R^2 to R^1 by selecting just feature θ_1. Because the label is determined entirely by feature θ_1, the dataset is still perfectly separable.

the original features, through a process called *feature selection*, or by creating new features, which is called *feature construction*.

2.4.1 Feature Selection

Feature selection techniques create a smaller feature set by selecting a subset of the original features. An example of this is shown in Figure 2.9, where feature θ_1 is selected to join the new feature set while feature θ_2 is discarded. The three main categories of feature selection techniques are *filtering* techniques, which act as a pre-processing step before the classifier is constructed and select features independent of their performance on the classifier, *wrapper* techniques, which use performance on the classifier as a method for evaluating a particular set of features [Kohavi and John, 1997], and *embedded* techniques, which include feature selection as part of an optimization equation in the classifier itself; for instance, a classifier may be heavily penalized for each additional feature it uses when constructing a decision boundary [Guyon and Elisseeff, 2003]. Wrapper methods generally outperform filtering techniques [Kohavi and John, 1997], which is not surprising as such methods select a

subset of features specific to the classifier to be evaluated; however, wrapper methods also often take substantially longer than filter methods because the classifier, which may be very sophisticated and have a large running time, must be trained and evaluated many times over. Because wrapper methods evaluate subsets of features to try to find the optimal set, they must be provided with some heuristic method of searching over the space of all feature subsets, a space which has size exponential in the number of features. Even the simple greedy approach of building a feature set by consecutively adding the feature that most increases performance over the current set requires $(d \cdot d')$ classifiers to be trained and evaluated, which may require a prohibitively long running time.

Filtering techniques typically avoid the problems inherent in pairing up with the classifier by using a simpler scoring function to evaluate features or subsets of features. While filtering techniques exist that, like wrapper techniques, search over the exponential space of feature subsets, in this thesis I will focus on *feature ranking* techniques, which evaluate some scoring function on each feature independent of the others, and then sort the features by their scores and select the top d'. This is shown in Algorithm 1. Common scoring functions include mutual information or the correlation of a particular feature with the labels. On the text classification problem, it is possible to outperform an SVM trained on the full feature set by pre-processing the data with a feature ranking filtering scheme that uses one of a number of relatively simple metrics [Forman, 2003]. Example metrics include the mutual information shared by the feature and the labels, $I(\theta; Y) = \sum_{x \in \theta} \sum_{y \in \mathcal{Y}} \Pr(x, y) \log \left(\frac{\Pr(x,y)}{\Pr(x)\Pr(y)} \right)$ [Brown et al., 2012], and the accuracy of a classifier trained on the feature, $\Pr(h(x) = f(x))$.

Feature ranking is appealing for its simplicity and speed; it is typically much faster to evaluate a scoring metric than to train and evaluate a classifier, and because independent features are considered rather than feature subsets, the scoring function must only be evaluated d times. For datasets with very high dimensionality,

Algorithm 1 Feature Ranking

Require: Supervised dataset (X, Y), scoring function $\mathcal{S}$, feature set Θ, new dimensionality d'

1: $V \leftarrow [\,]$
2: **for** feature θ_i in Θ **do**
3: $\quad V[i] \leftarrow (\mathcal{S}(X, Y, \theta_i), \theta_i)$
4: **end for**
5: $V \leftarrow \textsc{SortDecreasing}(S)$
6: **return** $[\theta_i$ for θ_i in $V[1...d']]$

such as text classification datasets, feature ranking may be the only practical option. A drawback of feature ranking, however, is that if multiple features are needed to discover correlation, they may not be selected because all features are considered independently. When this is the case, it may be necessary to use a method that selects features sequentially; for example, features may be added one at a time, with each new feature being selected to maximize the joint mutual information score $JMI(\theta) = \sum_{\theta' \in S} I(\theta\theta'; Y)$ with respect to the set of already-selected features S [Brown et al., 2012]. The running time of these methods tends to be polynomial, rather than linear, in the number of features d, as each new feature added requires re-evaluation of the scoring function for every feature not already selected.

2.4.2 Feature Construction

Where feature selection techniques use a subset of the original features as a feature set, feature construction techniques create a new set of features, often by following a linear projection strategy, wherein each new feature is represented as a linear combination of the d original features. The flexibility of feature construction techniques means that they can often lead to better performance. Unfortunately, they can often be more expensive than feature selection methods both space-wise and time-wise [Guyon and Elisseeff, 2003]. Furthermore, because the dataset no longer uses the original features, the results can be more difficult to interpret. For example, in data

visualization, eliminating all but two or three of the most relevant features is necessary for human comprehension. If the features have some inherent meaning, then when feature selection is used, a visualization may still be human-understandable. When feature construction is used, the individual features may no longer have any understandable meaning, but may convey more information.

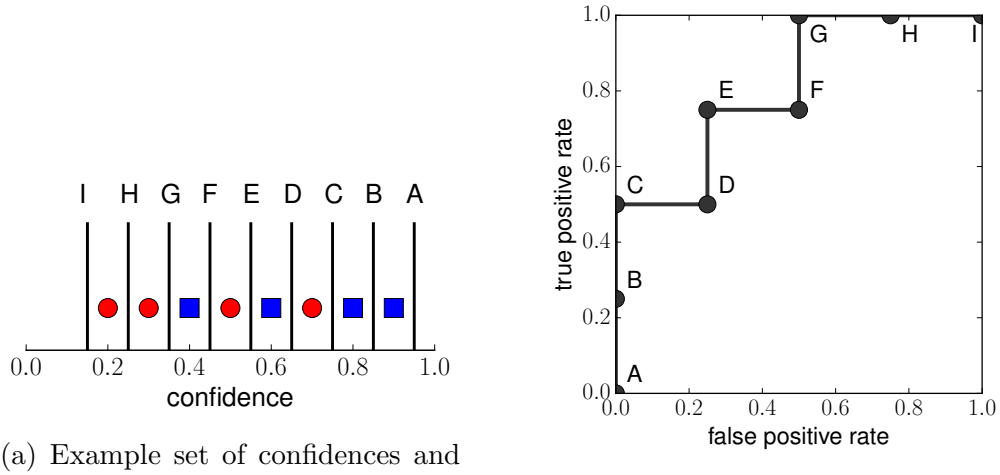
Principal Components Analysis (PCA) is a popular feature construction algorithm. PCA uses eigenvalue decomposition to find the set of orthonormal vectors along which the dataset has maximum variance [Shlens, 2014]. Each vector is a linear combination of the original features. The d' newly-constructed feature vectors that represent the most variability in the dataset are selected. PCA makes the strong assumption that the data distribution is Gaussian, which illustrates another drawback of feature construction methods: they often make strong assumptions about the data, which not only may not hold up but may also not match the assumptions of the classifier subsequently trained on the data.

While PCA does not take the labels into account, *Linear Discriminant Analysis* (LDA) attempts to find a feature space where the positive and negative instances are maximally separable [Murphy, 2012]. It does this by attempting to maximize the variance between the positive and negative instances while minimizing the variance within the two distributions; that is, if two points share the same label, LDA tries to ensure that they remain close together in the new feature space, while if they have different labels, LDA tries to make them far apart in the new space. Unlike PCA, LDA does not assume that the entire dataset is distributed as a Gaussian; instead, it assumes that the positives and negatives are each independently distributed as Gaussians. LDA suffers from the fact that, when used for binary classification, it can only construct a single new dimension [Friedman et al., 2001]; however, the concept of maximizing the between-class variance while minimizing the within-class variance is frequently used in other feature construction algorithms.

2.5 Area Under the ROC Curve

There are several methods available for representing the "goodness" of a particular hypothesis during evaluation. The simplest of these is accuracy, which measures the ratio of instances for which a given hypothesis agrees with the target concept. When a hypothesis only outputs labels, accuracy is a good measurement; however, hypotheses often output not only labels but also a degree of confidence that an instance is positive, which can be taken as a probabilistic output. For instance, if a hypothesis concept is 70% confident in its positive output prediction for a particular instance x_1, that instance can be thought of as being "less positive", in the opinion of the hypothesis concept, than another instance x_2 which it is 99% confident is positive. While the accuracy measurement simply throws this information away, the *Area under the ROC curve* (AUC) measurement takes it into account.

A *Receiver Operating Characteristic* (ROC) curve takes into account two measurements: the true positive rate, $\frac{\#\text{ true positives}}{\#positives}$, which measures the fraction of positives that are correctly labeled by the hypothesis, and the false positive rate, $\frac{\#\text{ false positives}}{\#negatives}$, which measures the fraction of negatives that are incorrectly labeled as positives. Each point on the curve represents the true positive rate vs. the false positive rate for a particular threshold on the instance ranking, where all points above the threshold are given a positive label and all points below the threshold are given a negative label. As the threshold is decrease from 1 to 0, a curve is formed from $(0, 0)$ to $(1, 1)$. This is shown in Figure 2.10; note that as the threshold moves from 1 to 0, whenever true instances (blue squares) are included, the curve moves up, increasing the area underneath it, and whenever false instances (red circles) are included, the curve moves right, decreasing the area underneath it. The Area under the ROC curve measurement, then, intuitively represents the correctness of a ranking, as when more positive instances are ranked higher than negative instances, the curve will move up



(a) Example set of confidences and thresholds

(b) Example ROC curve

Figure 2.10: An example ROC curve, with points marking the false positive rate vs. true positive rate for each threshold.

earlier, and thus will have a higher area underneath it.

One important and useful property of the AUC measurement is that it is equivalent to the probability that a randomly-selected positive instance will be ranked higher by the hypothesis than a randomly-selected negative instance [Hand and Till, 2001, Fawcett, 2006]. For this reason, the AUC measurement is identical to the measurement given by the Wilcoxon-Mann-Whitney test, which compares the value of some measurement on two separate populations; in the case of AUC, the two separate populations are the positive and negative instance distributions [Hanley and McNeil, 1982].

2.6 Related Work

2.6.1 MidLABS

Many algorithms for multiple-instance learning treat the instances in each bag as being IID; however, this can ignore the information revealed by the connections between

Algorithm 2 MidLABS

Require: Multiple-instance dataset (B, Y), new dimensionality d'

1: **for** bag B_i in B **do**
2: $G_i \leftarrow \epsilon$-graph for B_i
3: **end for**
4: $S_b = \sum_{Y_i \neq Y_j} \frac{\sum_{a=1}^{n_i} \sum_{b=1}^{n_j} (x_{ia} - x_{jb})^2}{n_i n_j} + C \frac{\sum_{c=1}^{m_i} \sum_{d=1}^{m_j} (e_{ic} - e_{jd})^2}{n_i^2 n_j^2}$
5: $S_w = \sum_{Y_i = Y_j} \frac{\sum_{a=1}^{n_i} \sum_{b=1}^{n_j} (x_{ia} - x_{jb})^2}{n_i n_j} + C \frac{\sum_{c=1}^{m_i} \sum_{d=1}^{m_j} (e_{ic} - e_{jd})^2}{n_i^2 n_j^2}$
6: Solve generalized eigenvalue problem $S_b w = \lambda S_w w$
7: Sort eigenvectors w by their eigenvalues λ
8: $W \in \mathbb{R}^{d \times d'} \leftarrow$ top d' eigenvectors w of $\frac{w^T S_b w}{w^T S_w w}$
9: **return** projection matrix W

instances in the same bag. Multi-Instance Dimensionality Reduction by Learning a Maximum Bag Margin Subspace, or MidLABS for short, is a method that attempts to find an optimal subspace for the task of classifying bags while taking into account the structural information of the instances in the bag [Ping et al., 2010].

MidLABS is a feature construction method, and it operates on the bag-level, meaning it focuses on creating a feature space conducive to separating bags, rather than separating instances. Like Linear Discriminant Analysis, it tries to find a subspace where bags with the same label are close together and bags with different labels are far apart. Pseudocode for MidLABS is given in Algorithm 2.

In order to take structural information about the instances within each bag into account, MidLABS uses a customized bag distance measurement when calculating the optimal subspace. This bag distance measurement represents bags using ϵ-graphs [Tenenbaum et al., 2000]. In an ϵ-graph for a bag, each instance is treated as a node, and an edge is added if the Euclidean distance between two instances is below than some threshold ϵ. This way of representing bags has been perviously established as a good way to examine the relationships between instances in bags [Zhou et al., 2009]. MidLABS represents bags in this way, and defines a distance metric between bags

based on their ϵ-graphs as follows:

$$Dis(X_i, X_j) = \frac{\sum_{a=1}^{n_i}\sum_{b=1}^{n_j}(w^T x_{ia} - w^T x_{jb})^2}{n_i n_j} + C\frac{\sum_{c=1}^{m_i}\sum_{d=1}^{m_j}(w^T e_{ic} - w^T e_{jd})^2}{n_i^2 n_j^2} \quad (2.8)$$

where w is the linear projection vector representing a particular constructed feature, m_i is the number of edges in the ϵ-graph of bag i, and an edge is represented by an edge vector e that is equal to the difference between the two instances it connects. This distance measurement is similar to the set kernel, a common tool for comparing the closeness of bags, which also sums over all the pairs of instances between two bags [Gartner et al., 2002].

2.6.2 CLFDA

Citation Local Fisher Discriminant Analysis (CLFDA), like MidLABS, is a feature construction method. Unlike MidLABS, however, CLFDA operates at the instance-level, attempting to find a subspace where positive and negative instances are maximally-separable [Kim and Choi, 2010]. While instance labels are not available, CLFDA attempts to estimate the instance labels using the heuristic that if a point from a positive bag is close to a lot of points from negative bags, it is probably a negative point. Pseudocode for CLFDA is given in Algorithm 3.

Specifically, CLFDA defines the R nearest *references* of a point x_i to be the R nearest neighbors of that point in terms of Euclidean distance. It also defines the *citers* of a point to be all the instances that have x_i as one of their C nearest neighbors. If the ratio of instances from negative bags versus instances from positive bags in the citers and references of x_i exceeds some threshold t, then x_i is given a negative label. Otherwise, it is given a positive label. This process is only used on instances from positive bags, as under the standard MI assumption all instances in negative bags have negative labels. The citers are needed because in a negative region

Algorithm 3 CLFDA

Require: Multiple-instance dataset (B, Y), new dimensionality d', parameters C, R, t

$X \leftarrow$instances in B
$G \leftarrow \max(C, R)$-nearest-neighbors graph of X
for $i = 1 \rightarrow n$ **do**
    $C_i \leftarrow C$ nearest citers of instance x_i
    $R_i \leftarrow R$ nearest references of instance x_i
    $n_i \leftarrow$ number of instances from negative bags in $C_i + R_i$
    $p_i \leftarrow$ number of instances from positive bags in $C_i + R_i$
    if $\frac{n_i}{p_i} \geq t$ or x_i is in a negative bag **then**
        instance label $y_i \leftarrow -1$
    else
        $y_i \leftarrow 1$
    end if
end for
$A^B \leftarrow$ between-class affinity matrix
$A^W \leftarrow$ within-class affinity matrix
Between-class scatter matrix $S_b = \frac{1}{2}\sum_{i,j=1}^{n} A_{ij}^B (x_i - x_j)(x_i - x_j)^T$
Within-class scatter matrix $S_w = \frac{1}{2}\sum_{i,j=1}^{n} A_{ij}^W (x_i - x_j)(x_i - x_j)^T$
$W \in \mathbb{R}^{d \times d'} \leftarrow$ top d' eigenvectors w of $\frac{w^T S_b w}{w^T S_w w}$
return projection matrix W

with a high number of false positive points, x_i may have many other false-positive points nearby, but the negative instances in the region will act as citers for the false positives, leading them to be given negative labels.

Once the instance labels have been estimated, a new set of features is constructed using Local Fisher Discriminant Analysis (LFDA), which is a version of Linear Discriminant Analysis (see subsection 2.4.2) modified to handle *multimodal* data, where instances in the same class form several clusters [Sugiyama, 2007]. The central change is that when maximizing between-class variance and minimizing within-class variance, the variance contribution of an individual point pair is weighted by the locality of that point pair, so that data points from different clusters that share the same label are not forced to be close. This has the added effect of allowing for multiple dimensions to be extracted, unlike LDA which is limited to one dimension.

2.6.3 MIDR

Multi-Instance Dimensionality Reduction (MIDR) is also a feature construction method, and like MidLABS, it is a bag-level method [Sun et al., 2010]. Unlike the previous two methods, however, MIDR operates as an embedded method, learning a feature space in conjunction with the Multiple Instance Logistic Regression classifier [Ray and Craven, 2005].

MIDR attempts to find a feature space where a classifier will have good performance. This is measured with an objective function that minimizes the quadratic loss between the confidences assigned to bags and their actual labels after projection into the new feature space, $\min_A \sum_i (P_i - y_i)^2$, where A is the projection matrix and P_i is the confidence of the lower-dimensional bag given by a classifier. In order to enforce orthonormality, it is required that the projection matrix be such that $A^T A = I$, and in order to encourage sparsity, a regularization term $\sum_{s,t} |A_{st}|$ is added to the objective function, with a parameter C_1 controlling the tradeoff between sparsity and classification strength; thus, the optimization problem is:

$$\begin{aligned} &\min_A \sum_i (P_i - y_i)^2 + C_1 \sum_{s,t} |A_{st}| \\ &\text{s.t. } A^T A = I \end{aligned} \tag{2.9}$$

The classifier used to estimate P_i is Multiple-Instance Logistic Regression, which uses a logistic function $P_{ij} = \frac{1}{1+e^{-w \cdot x_{ij} + b}}$ to calculate the confidence of each instance having a positive label, and then assigns as a bag confidence the *softmax* of the instance confidences. The softmax is a smooth parametric approximation of the max function: $P_i = softmax_\alpha(P_{i1}, ...P_{in_i}) = \frac{\sum_j P_{ij} e^{\alpha P_{ij}}}{\sum_j e^{\alpha P_{ij}}}$ [Ray and Craven, 2005]. MIDR operates in a loop, first calculating the parameters w and b for MILR based on the prior projection matrix A, and then calculating a new projection matrix from the optimization problem using the newly-calculated bag confidences P_i. This is

Algorithm 4 MIDR

Require: Supervised dataset (B, Y), new dimensionality d', sparsity parameter C_1

$A \leftarrow$ random orthonormal matrix $\in \mathbb{R}^{d \times d'}$
while No convergence **do**
    Train MILR classifier h using data $A^T B, Y$
    for $B_i \in B$ **do**
        $P_i \leftarrow$ confidence $h(B_i)$
    end for
    Solve the optimization equation to get a new A
end while
return projection matrix A

illustrated in Algorithm 4.

Chapter 3

Multiple-Instance Feature Ranking Using Accuracy as a Scoring Function

In this chapter, I will analyze a feature selection technique for multiple-instance learning. This technique performs filtering via feature ranking, which, as described in subsection 2.4.1, tends to be the fastest approach to feature selection. The scoring function used for ranking will be the predictive power of a feature, as defined by the accuracy of an instance-space classifier operating on that feature, following similar examples in the case of supervised learning [Guyon and Elisseeff, 2003].

The central challenge of learning a good feature ranking under a metric determined by an instance-space evaluation comes from the fact that instance labels are not available in the multiple-instance learning task. In this chapter, I will use the strategy of giving instances the labels of their bags, which creates a supervised dataset with one-sided noise, as described in subsection 2.3.2. While previous work has shown empirically that this strategy can often result in competitive learning algorithms [Ray and Craven, 2005], my theoretical results in this chapter will serve to describe

the factors that lead this strategy to perform successfully in the feature ranking task, and the scenarios in which its limitations outweigh its benefits, which may in the future guide further exploration of the connection between supervised learning and multiple-instance learning.

The information limitations encountered by the technique presented in this chapter are the limited sample size and the one-sided noise from the bag-labeled instances strategy. I will define the ranking of features under the ideal circumstances when these limitations are not present to be the "correct" feature ranking. To measure the quality of the multiple-instance feature selection technique of this chapter, I will ask whether the features selected by it are highly likely to be the same as those that would be selected by a technique that finds the correct feature ranking. In order to generalize to any desired new dimensionality d', I will ask whether the entire ranking can be recovered; however, I will also discuss situations in which only the subset of the most "useful" features may be required, and how the technique may perform on this task.

The theoretical results in this chapter take the approach of analyzing the feature ranking task by representing the labeling concept on the one-dimensional space as a p-concept. The scoring function for the feature is measured with respect to that p-concept. To the best of my knowledge, this is a novel use of p-concepts. In this chapter, I use accuracy as a scoring function, so that each feature is ranked according to its characteristic accuracy, and the features selected are those that, when evaluated independently, seem to be the most helpful in constructing a feature space from which a highly accurate concept can be learned.

For this analysis, I will assume that the concept in $\mathbb{R}^d$ belongs to a concept class with finite VC dimension, and that the probability of a negative instance appearing in a negative bag is constant over the instance distribution.

3.1 MI-FEAR

In this section I will describe the technique used to discover a feature ranking under multiple-instance learning. The approach described is sufficiently general that it can be adapted to any learning problem.

I propose an algorithm named "Multiple-Instance Feature Ranking" (MI-FEAR) that attempts to learn a feature set that will lead to a high-performing classifier. In order to learn this quickly and effectively, I propose that each feature θ be given, as a score, the performance of a learned hypothesis h_θ operating on just that one feature. The performance may be determined by any evaluation metric $\mathcal{S}$. Once these scores have been assigned, all features are ranked by their characteristic scores, and the d' most accurate features selected as the feature set. Because the learned hypothesis is an instance-classifying hypothesis and requires a set of instance labels to learn from, in order to accommodate the multiple-instance setting, instances will be given the labels of their bags. Pseudocode for this algorithm is given in Algorithm 5.

The algorithm that learns the hypothesis, $\mathcal{A}$, will only ever learn over a one-dimensional space; therefore, it may be completely independent of any learning algorithm used in $\mathbb{R}^d$. Thus, it is possible to use a much quicker and simpler algorithm to evaluate features than will ultimately be used for classification. In this respect, MI-FEAR is similar in motivation to the VS-SSVM wrapper algorithm, which uses a simple linear L1-norm SVM for feature subset evaluation but a standard non-linear SVM for the final learning task [Bi et al., 2003].

This algorithm generalizes to any metric $\mathcal{S}$ that can evaluate the performance of h_θ; in this chapter I will consider accuracy as a possible metric. An important point is that the characteristic accuracy value of the feature that MI-FEAR assigns is based on the performance of h_θ when measured against the noisy labels Y^γ. Because the true instance labels are not available to MI-FEAR, it is impossible to get a

Algorithm 5 MI-FEAR

Require: Multiple-instance dataset (B, L), evaluation metric $\mathcal{S}$, learning algorithm $\mathcal{A}$, new dimensionality d'

1: $X, Y^\gamma \leftarrow$ supervised dataset of bag-labeled instances using (B, L)
2: $V \leftarrow [\,]$
3: **for** feature θ_i in feature set Θ **do**
4: $\quad X^\theta \leftarrow X$ with all features except θ_i discarded
5: $\quad h_\theta \leftarrow$ output of $\mathcal{A}$ when trained on input (X^θ, Y^γ)
6: $\quad V[i] \leftarrow$ (performance of h_θ on input (X^θ, Y^γ) according to $\mathcal{S}$, θ_i)
7: **end for**
8: $V \leftarrow$ SortDecreasing(V)
9: **return** $[\theta_i$ for θ_i in $V[1...d']]$

true measurement of the performance of h_θ, and instead this measurement must be estimated using its performance on the noisy instance labels.

While the dimensionality reduction techniques for multiple-instance learning described in section 2.6 implement feature construction, MI-FEAR implements feature selection, which sacrifices predictive power for speed, simplicity, and interpretability of the new feature space, as described in subsection 2.4.2. If $t_\mathcal{A}$ is the time required for algorithm $\mathcal{A}$ to find a hypothesis on n one-feature instances, and $t_\mathcal{S}$ is the time required to calculate the value of the evaluation metric $\mathcal{S}$ given a hypothesis and n samples, then the time complexity of MI-FEAR is $O\left(d(t_\mathcal{A} + t_\mathcal{S}) + d \log d\right)$, with the $d \log d$ factor coming from the time required to sort the d scores in decreasing order. Note that both $t_\mathcal{A}$ and $t_\mathcal{S}$ will be dependent on n, the number of instances contained in all the bags. The specific dependence will be determined by the algorithm $\mathcal{A}$ and evaluation metric $\mathcal{S}$ selected, but for virtually all practical purposes this dependence will be either linear or a low-order polynomial. This time complexity compares favorably to those of the algorithms in section 2.6; for example, the time complexity of CLFDA is $O(n^2d^2+d^3)$, the time complexity of MidLABS is $O(|B|^2n_{ave}^4d^2+d^3)$, where n_{ave} is the average number of instances per bag, and the time complexity of MIDR is $O\left(t_{in}t_{out}(ndd'^2 + d^2d'^4)\right)$, where t_{in} is the average number of iterations required to calculate a hypothesis for MIDR and t_{out} is the number of outer iterations required

for MIDR to converge [Chai et al., 2014]. Because $t_{\mathcal{A}}$ is calculated for a one-feature input space, it is independent of d; thus, MI-FEAR has the best time complexity in terms of the size of the feature space d.

3.2 A Theoretical Analysis of MI-FEAR with Accuracy

In this section, I will analyze the performance of MI-FEAR when accuracy is used as the evaluation metric $\mathcal{S}$. In order to calculate the characteristic accuracy of an individual feature, I will consider the labeling function on a single feature as a p-concept. Within this model I will describe how the one-sided noise added by the bag-labeled instances strategy for multiple-instance learning changes the p-concept, and study how this affects the accuracy metric learned. I will also study how the restriction of a limited dataset affects the ability to learn an accurate feature ranking. Following this analysis, I will show how these restrictions affect both the ability of MI-FEAR with accuracy to find a correct feature ranking and the number of instances required to find such a ranking.

This section is organized as follows: In subsection 3.2.1, I will show describe how the quality of a ranking can be measured with respect to a correct feature ranking learned under ideal circumstances by measuring the accuracies of the hypotheses learned on each individual feature against the accuracies of the maximum-accuracy concepts on each feature. In subsection 3.2.2, I will illustrate how the labeling function on a single feature can be viewed as a p-concept, and describe what the maximum-accuracy concepts both with and without noise would be. In subsection 3.2.3, I will show how the number of instances provided to MI-FEAR affects its ability to learn an accurate feature ranking, and give a complexity bound that relates the number of instances provided to the quality of the ranking. In subsection 3.2.4, I will show

how the strategy of using bag-labeled instances affects MI-FEAR's ability to learn an accurate feature ranking, and analyze the different factors that control whether or not it possible to learn the ranking under this strategy given an unlimited number of instances. Finally, in subsection 3.2.5, I will prove a complexity bound on the number of instances required to learn a good feature ranking with high probability using MI-FEAR with accuracy.

3.2.1 Determining the Quality of a Feature Ranking

In general, discovering an optimal feature subset is a hard problem; even in the simple case of boolean features, when the concept is a boolean function over a relevant subset of the features, exponential time is required to discover the correct features [Mossel et al., 2003]. For this reason, I make no guarantee that MI-FEAR with accuracy will be able to find the perfect feature subset; instead, I ask whether, in spite of its limitations, it will be able to correctly select the d' features which have the highest characteristic accuracy. The correct features, in this sense, are determined to be the top d' when ranked by their "true" characteristic accuracies.

The *true characteristic accuracy* of a feature θ is defined to be the maximum accuracy possible, over all possible deterministic concepts f_θ defined on θ, when measured against the true instance labels. The concept that gives the maximum accuracy on feature θ $Acc_{f_\theta}(\theta)$ is f^*_θ:

$$f^*_\theta = \arg\max_{f_\theta} Acc_{f_\theta}(\theta) \tag{3.1}$$

MI-FEAR with accuracy will use a learning algorithm $\mathcal{A}$ to learn a deterministic hypothesis concept h_θ for feature θ. It will then measure the accuracy of h_θ over the instance set against the noisy labels Y^γ. This introduces the concept of *perceived characteristic accuracy*. The perceived characteristic accuracy of a feature θ according to a concept f_θ, $Acc^\gamma_{f_\theta}(\theta)$, is the accuracy of f_θ when measured against the instance

labels provided by using bag-labeled instances. The perceived characteristic accuracy may be overly optimistic or overly pessimistic relative to the true characteristic accuracy. For example, suppose a set of instances, 50% of which were labeled 1 and 50% of which were labeled 0, was given 100% noise so that every instance was labeled 1. Then a hypothesis that predicted 1 on every input would achieve a perceived accuracy of 1, while its actual accuracy would only be 0.5. Thus, it optimistically perceives itself to be perfectly accurate. While f_θ^* is the optimal concept with respect to true accuracy, it is not the optimal concept with respect to perceived accuracy. This concept will be written f_θ^γ:

$$f_\theta^\gamma = \arg\max_{f_\theta} Acc_{f_\theta}^\gamma(\theta) \tag{3.2}$$

When all the features are given their true characteristic accuracies $\{Acc_{f_\theta^*}(\theta)\}_{\theta\in\Theta}$, they can be ranked by those accuracies. The goal of MI-FEAR is to find a set of hypotheses $\{h_\theta\}_{\theta\in\Theta}$ such that the perceived characteristic accuracies $\{Acc_{h_\theta}^\gamma(\theta)\}_{\theta\in\Theta}$ are ranked in the same way as the true characteristic accuracies. Let τ be defined as the minimum separation between the true characteristic accuracy measurements of two neighboring features in this ranking, so long as those two measurements are not equal, since for two features that are ranked equally it does not matter which is ranked higher than the other:

$$\tau = \min_{\theta_1,\theta_2 : Acc_{f_{\theta_1}^*}(\theta_1) \neq Acc_{f_{\theta_2}^*}(\theta_2)} \left| Acc_{f_{\theta_1}^*} - Acc_{f_{\theta_2}^*}(\theta_2) \right| \tag{3.3}$$

Further, let the accuracy difference on feature θ, $\Delta_{Acc}(\theta)$, be the difference between the true characteristic accuracy of θ and its perceived characteristic accuracy according to h_θ:

$$\Delta_{Acc}(\theta) = \left| Acc_{f_\theta^*}(\theta) - Acc_{h_\theta}^\gamma(\theta) \right| \tag{3.4}$$

As τ is the smallest possible separation between two features when ranked by true characteristic accuracy, so long as, for every feature θ, the perceived characteristic accuracy of θ is $\frac{\tau}{2}$-close to the true characteristic accuracy of θ, the entire ranking will be exactly preserved, modulo equally-accurate features. Preserving the entire ranking guarantees that no matter what d' is selected, MI-FEAR with accuracy will select the correct feature subset. An upper bound can be placed on the difference $\Delta_{Acc}(\theta)$ between the two accuracies as follows:

$$\Delta_{Acc}(\theta) \leq \left| Acc^{\gamma}_{h_\theta}(\theta) - Acc^{\gamma}_{f^{\gamma}_{\theta}}(\theta) \right| + \left| Acc^{\gamma}_{f^{\gamma}_{\theta}}(\theta) - Acc_{f^{*}_{\theta}}(\theta) \right| \tag{3.5}$$

As subsequent analysis will reveal, the first term in this bound represents the accuracy difference due to approximation using the limited sample available, and can be driven to zero. The second term represents the accuracy difference due to one-sided noise from the use of bag-labeled instances, and has a constant value on feature θ. For convenience, let the second term be $\Delta^{\gamma}_{Acc}(\theta) = \left| Acc^{\gamma}_{f^{\gamma}_{\theta}}(\theta) - Acc_{f^{*}_{\theta}}(\theta) \right|$ and let $\Delta^{\gamma *}_{Acc} = \max_\theta \Delta^{\gamma}_{Acc}(\theta)$. I will show that so long as $\Delta^{\gamma *}_{Acc} < \frac{\tau}{2}$, MI-FEAR with accuracy can learn a correct feature ranking:

Theorem 1 *Let k be the VC dimension of the concept class learned by $\mathcal{A}$. If $\Delta^{\gamma *}_{Acc} < \frac{\tau}{2}$, then with probability at least $1 - \delta$, MI-FEAR with accuracy can learn a correct feature ranking over d features using n examples, where*

$$n = O\left(\frac{1}{\frac{\tau}{2} - \Delta^{\gamma *}_{Acc}} \left(\log \frac{d}{\delta} + k \log \frac{1}{\frac{\tau}{2} - \Delta^{\gamma *}_{Acc}} \right) \right) \tag{3.6}$$

The following sections will serve as a proof of this theorem.

3.2.2 Bayes-optimal Concepts on a Feature Axis

In this subsection I will define terms and functions that will come in useful when analyzing the difference in characteristic accuracy for a particular feature θ. I will also give functions that describe the optimal concept with respect to true accuracy, f^*_θ, and with respect to perceived accuracy, f^γ_θ. In this subsection, z is occasionally used to represent a point on a one-dimensional feature space in the context of projection, where x represents a point in $\mathbb{R}^d$; in all other places, x is used to represent a point on a one-dimensional feature space.

Suppose we are examining a set of n points $X \in \mathbb{R}^d$ distributed according to an instance space distribution $D_\mathcal{X}$. If a single individual feature $\theta \in \Theta$ is selected, then, for every point, all the information contained in the other $d-1$ features is thrown away. In the Euclidean space $\mathbb{R}^d$, this corresponds to *projecting* each of the points onto the axis θ. This is illustrated in Figure 2.9.

In this new one-dimensional space, points that were previously far apart in $\mathbb{R}^d$ may now overlap on the axis. For example, let $x_1 = (0, 3, -8)$ and $x_2 = (9, 3, 5)$. If the second dimension is selected, then both the first and third dimensions are thrown away, and both instances project onto the point (3). Asymptotically, the marginal distribution of the instance space on θ can be defined as the weight of the probability density function $D_\mathcal{X}$ on the value of z for feature θ, integrated with respect to all possible values for the other features:

$$D_{\mathcal{X}|\theta} : \Pr_\theta(z) = \int_{\Theta \setminus \theta} \Pr(x^1, ..., x^{\theta-1}, z, x^{\theta+1}, ..., x^d) \mathrm{dPr}(x) \tag{3.7}$$

In this definition, x^θ indicates the value of feature θ for point x.

This representation can be approximated with a histogram representation, creating an empirical marginal distribution. This allows the projection to be discussed in terms of individual instances or sets of instances rather than probability density func-

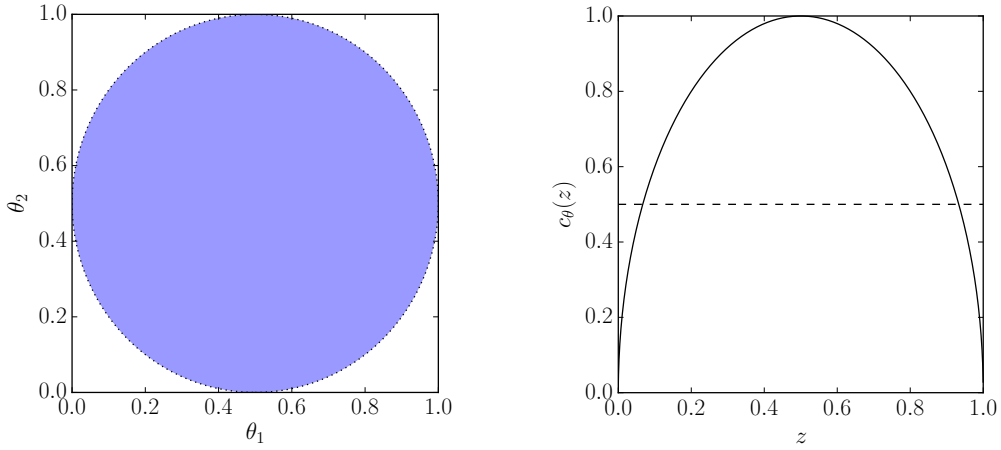
tions. While the granularity of the real space may make observable overlap unlikely on some finite datasets, in the limit of infinite saturation of the instance space there will always be overlap and the asymptotic behavior of the probability density function can be approximated with arbitrary precision. For convenience and clarity, in this analysis, I will use this empirical approximation of the real marginal distribution:

$$D_{\mathcal{X}|\theta} : \Pr_\theta(z) = \int_{\mathcal{X}} \mathbb{1}[x^\theta = z] \mathrm{dPr}(x) \tag{3.8}$$

Because points that project onto the same value on the feature axis may be arbitrarily far apart in $\mathbb{R}^d$, points with different labels may be projected onto the same value z of the feature. For this reason, it is no longer correct to describe points as having a deterministic 0/1 label; rather, it makes more sense to talk about the ratio, under the instance space distribution, of positive points to negative points projected onto z. For example, let f be the target instance-labeling concept in $\mathbb{R}^d$. Using x_1 and x_2 from above, if $f(x_1) = 1$ and $f(x_2) = 0$, then their mutual projection on the θ axis, point $z = 3$, can not be thought of as having a deterministic 0 or 1 label. If $D_{\mathcal{X}}$ assigned a weight of 0.7 to x_1 and a weight of 0.3 to x_2, then the aforementioned ratio would be 0.7/0.3, meaning that if a set of points was drawn according to $D_{\mathcal{X}}$, labeled according to f, and projected onto θ, in the limit of an infinite sample, 70% of the appearances of instance (3) would have a positive label, and 30% would have a negative label.

This scenario, with visible 0/1 labels assigned based on a pointwise probability, corresponds exactly to the p-concepts of subsection 2.2.4. The labeling function on the θ axis, therefore, is described as a p-concept, denoted c_θ. The value of the p-concept on a particular point is:

$$c_\theta(z) = \int_{\mathcal{X}} \mathbb{1}[f(x) = 1] \mathrm{dPr}(x|x^\theta = z) \tag{3.9}$$



(a) A deterministic concept in $\mathbb{R}^2$. Assume $D_{\mathcal{X}}$ is uniform in $[0, 1]^2$.

(b) The p-concept $c_\theta(z)$ after projection onto the axis of θ_1

Figure 3.1: An example of label projection onto one feature. Note how for values of θ_1 closer to 0.5, the circle takes up more vertical space, meaning more points are likely to be labeled positive. This is reflected in the p-concept, in that values of θ_1 closer to 0.5 have a higher value of c_θ.

Figure 3.1 gives an example of a concept in $\mathbb{R}^2$ and its corresponding p-concept when projected onto a single feature axis. It should be noted that the p-concept is determined by both the labeling concept f and the instance distribution $D_{\mathcal{X}}$. If only a small area of the instance space projected onto z is given a positive label by f, but that area is given a heavy weight by $D_{\mathcal{X}}$, then $c_\theta(z)$ will have a higher value than if $D_{\mathcal{X}}$ was uniform.

In subsection 2.4.1, the accuracy of a concept was defined as the probability that the output of that concept matches the output of the labeling concept. Because the labeling concept is no longer deterministic but rather probabilistic, this definition no longer holds. Instead, let $b_{c_\theta}(x)$ be the 0/1 label assigned to a point x according to the p-concept $c_\theta(x)$; accuracy can be defined as the probability that the output of the hypothesis on a point will match the label assigned to that point:

$$Acc_{f_\theta}(\theta) = \Pr\left(f_\theta(x) = b_{c_\theta}(x)\right) \tag{3.10}$$

The output of a deterministic concept $f_\theta(x)$ must always be the same for point x, but $b_{c_\theta}(x)$ may change for multiple instances of point x. For this reason, in general, it is impossible to get an accuracy of 1 using a deterministic concept to predict labels assigned according to a p-concept; however, there is still a maximum-accuracy concept that can be obtained.

In the terminology of Kearns and Schapire [Kearns and Schapire, 1993], the *predictive error* of a concept f_θ on a p-concept c_θ with respect to the marginal distribution $D_{\mathcal{X}|\theta}$, $R_{D_{\mathcal{X}|\theta}}(c_\theta, f_\theta)$, is the probability that f_θ will misclassify a randomly-drawn point, $R_{D_{\mathcal{X}|\theta}}(c_\theta, f_\theta) = \Pr\left(f_\theta(x) \neq b_{c_\theta}(x)\right)$. A *Bayes-optimal concept* for c_θ is a concept f_θ that minimizes $R_{D_{\mathcal{X}|\theta}}(c_\theta, \cdot)$. The definitions of accuracy and predictive error are complementary, so minimizing the predictive error is equivalent to maximizing the accuracy. Bayes-optimality is defined over the set of all possible concepts. Therefore, a Bayes-optimal concept will also be a maximum-accuracy concept, and the true characteristic accuracy of feature θ can be defined as the accuracy of a Bayes-optimal concept for c_θ.

The concept described by the function $\pi_{c_\theta}(x) = \mathbb{1}[c_\theta(x) \geq 0.5]$ is Bayes-optimal [Kearns and Schapire, 1993]. To see this, note that it is always best to predict the label that is most likely to occur on a given point. For example, it is never possible to get more accuracy, asymptotically, by predicting 0 on a point that is more likely to be 1. Because $\pi_{c_\theta}(x)$ is a Bayes-optimal concept, it is also the maximum-accuracy concept; therefore, $f^*_\theta = \pi_{c_\theta}$. This concept is illustrated in Figure 3.2.

As defined in Definition 2.5, γ is the probability that a negative instance appears in a negative bag, $\Pr(F(B) = 1|x^-)$. As described in subsection 2.3.2, when instances are given the labels of their bags, this is equivalent to the probability that a negative instance is given its correct (negative) label, which means that there is one-sided noise on the negative instances with noise rate $\eta = 1 - \gamma$.

$c_\theta(x)$ is the probability, under the true labels, that an instance drawn at point x on

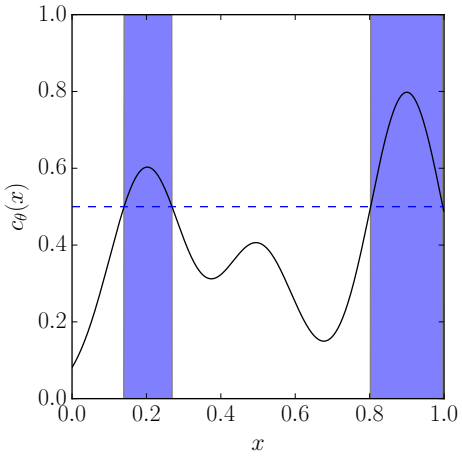


Figure 3.2: Example of a Bayes-optimal concept. Blue regions represent areas where $\pi_{c_\theta}(x) = 1$.

the one-dimensional axis of feature θ is given a positive label. If the one-sided noise is taken into account, then there is still the same probability of a positive instance being drawn, but if a negative instance is drawn then there is a $1 - \gamma$ probability that it will be given a positive label. Because the probability of drawing a negative instance at point x is $1 - c_\theta(x)$, there is an additional $(1 - c_\theta(x))(1 - \gamma)$ probability of seeing a positive instance when the one-sided noise is factored in. This product is always positive, and has the effect, as shown in Figure 3.3, of both lifting and flattening the distribution $c_\theta(x)$ to produce a new noisy p-concept, which I will denote $c^\gamma_\theta(x)$, and whose value at point x can be calculated in closed form as:

$$c^\gamma_\theta(x) = c_\theta(x) + (1 - c_\theta(x))(1 - \gamma) \tag{3.11}$$

The perceived accuracy of a concept on a p-concept c_θ with respect to $D_{\mathcal{X}|\theta}$ is the accuracy of that concept when measured against the noisy labels, which are represented by the p-concept $c^\gamma_\theta(x)$. Therefore, the perceived accuracy of a concept is simply the accuracy of that concept when measured on concept c^γ_θ:

$$Acc^\gamma_{f_\theta}(\theta) = \Pr\left(f_\theta(x) = b_{c^\gamma_\theta}(x)\right) \tag{3.12}$$

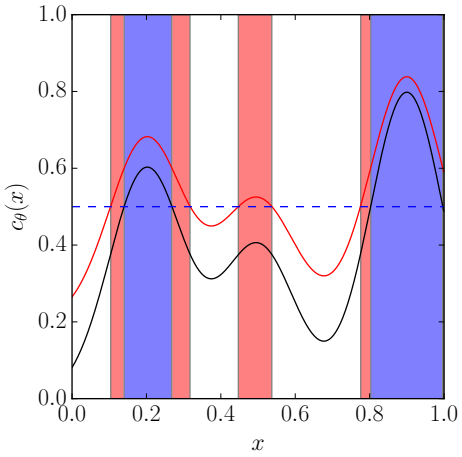


Figure 3.3: $c_\theta(x)$ (black) and $c_\theta^\gamma(x)$ (red) plotted, with $\gamma = 0.8$. Blue regions show where both $\pi_{c_\theta}(x)$ and $\pi_{c_\theta^\gamma}(x) = 1$, and red regions show where only $\pi_{c_\theta^\gamma}(x) = 1$. Notice how $c_\theta^\gamma(x)$ is "lifted" above $c_\theta(x)$. It is also "flattened" in the sense that its rises and falls are less steep due to compression into the space above $c_\theta(x)$.

The Bayes-optimal concept for c_θ^γ is the function $\pi_{c_\theta^\gamma}(x) = \mathbb{1}[c_\theta^\gamma(x) \geq 0.5]$. Just as the Bayes-optimal concept for c_θ maximizes the true accuracy, the Bayes-optimal concept for c_θ^γ maximizes the perceived accuracy; therefore, $f_\theta^\gamma = \pi_{c_\theta^\gamma}$. Note that because $c_\theta^\gamma(x) \geq c_\theta(x)$ at all points, f_θ^γ will predict 1 at all points where f_θ^* predicts 1, but also at some points where f_θ^* predicts 0.

3.2.3 Difference in Characteristic Accuracy Due to Approximation

In this subsection I will place an upper bound on quantity $\left|Acc_{h_\theta}^\gamma(\theta) - Acc_{f_\theta^\gamma}^\gamma(\theta)\right|$. I will explain the scheme by which f_θ^γ is approximated, and show that it is possible to PAC-learn a concept class containing f_θ^γ with an amount of instances linear in its VC dimension, thus allowing the difference in perceived accuracies to be arbitrarily small. I will also explain why it is impossible for the concept class containing f_θ^γ to have infinite VC dimension, and why the Minimum One-sided Disagreement strategy of Simon can not be used for this purpose.

In order for an upper bound to be obtained for the difference in perceived accu-

racies between a hypothesis h_θ and the function f_θ^γ that maximizes the characteristic perceived accuracy of θ, it is necessary for h_θ to be learned from a concept class $\mathcal{C}_\theta$ containing f_θ^γ. If this were not the case, then even in the limit of an optimal hypothesis, the difference in perceived accuracies would still be bounded from above by the difference between the perceived accuracy of f_θ^γ and the perceived accuracy of the most accurate concept in the concept class. It is very difficult to place such an upper bound. For this reason, I assume that there exists a concept class $\mathcal{C}_\theta$ that contains f_θ^γ, and that $\mathcal{C}_\theta$ has finite VC dimension.

That such a concept class exists follows from the fact that f, the target concept in $\mathbb{R}^d$, is assumed to be in a concept class of finite VC dimension. Note that when $\gamma \leq 0.5$, $c_\theta(x) + (1 - c_\theta(x))(1 - \gamma) > 0.5$, so f_θ^γ will always predict 1, while if $\gamma > 0.5$, every point where f_θ^γ predicts 1 must correspond to at least one point in $\mathbb{R}^d$ where f gives a positive label. If f_θ^γ can not be contained in a concept class of finite VC dimension, then the set of points in $\mathbb{R}^d$ that, when projected, cover the points where f_θ^γ predicts 1, must also not be capable of being represented by a function contained in a concept class of finite VC dimension. This is a contradiction, as this set of points is labeled by f, which is contained in a concept class of finite VC dimension. Therefore, there must exist a concept class $\mathcal{C}_\theta$ of finite VC dimension that contains f_θ^γ. Let $k = VC(\mathcal{C}_\theta)$.

As described in subsection 2.2.3, so long as a concept class has finite VC dimension, it is possible to use empirical risk minimization to PAC-learn that concept class. Furthermore, as described in subsection 3.2.2, minimizing the predictive error with respect to the noisy labels is equivalent to maximizing the perceived accuracy. If ERM is used with $R_{D_{\mathcal{X}|\theta}}(c_\theta^\gamma, h_\theta)$ as the risk, which corresponds to using $\mathbb{1}[h_\theta(x) \neq b_{c_\theta^\gamma}(x)]$ as the loss function, then it will approximate the concept $f_\theta^\gamma(x)$. Therefore, ERM can be used to guarantee that, with probability $\geq 1 - \delta_\theta$, $\left| Acc_{h_\theta}^\gamma(\theta) - Acc_{f_\theta^\gamma}^\gamma(\theta) \right| < \epsilon$ for some ϵ, δ_θ, so long as the number of samples n exceeds the following bound (from

Equation 2.5):

$$n \geq \max\left(\frac{4}{\epsilon}\log\frac{2}{\delta_\theta}, \frac{8k}{\epsilon}\log\frac{13}{\epsilon}\right) = O\left(\frac{1}{\epsilon}\left(\log\frac{1}{\delta_\theta} + k\log\frac{1}{\epsilon}\right)\right) \tag{3.13}$$

It should be noted that while the learning here is ignoring the fact that one-sided noise exists, there exists a concept approximation strategy called "Minimum One-sided Disagreement" that attempts to learn through the one-sided noise [Simon, 2014]. Were it possible to apply Minimum One-sided Disagreement in this scenario, it could be used to directly approximate f^*_θ with h_θ. Minimum One-sided Disagreement selects a function that perfectly classifies the instances whose labels are known to be correct (in this case, those labeled 0). The idea is that so long as $\eta < 1$, in the limit, the negative instances will all reveal themselves through the noise; for example, if the noise rate is 0.999, then when a negative point x is drawn, it will be given a negative label with probability 0.001; however, in the limit of drawing x an infinite number of times, it will be given a negative label at least once, so if a concept is selected that will give x a negative label based solely on that one time it was negative, x will be labeled correctly.

Unfortunately, Minimum One-sided Disagreement can not be used when the original, pre-noise labels are assigned probabilistically according to a p-concept rather than deterministically, because even when a negative instance is seen it can not be assumed that it is best classified as negative. For example, if $c_\theta(x) = 0.8$, there is still a 20% chance that x will have a negative label, but it would be a mistake to classify x as negative, even though the Minimum One-sided Disagreement strategy would do so. Therefore, the strict classification demands of Minimum One-sided Disagreement do not lead to optimality when used against a p-concept, and ERM must be used.

3.2.4 Difference in Characteristic Accuracy Due to One-sided Noise

In this subsection I will place an upper bound on quantity $\Delta^{\gamma}_{Acc}(\theta) = \left| Acc^{\gamma}_{f^{\gamma}_{\theta}}(\theta) - Acc_{f^{*}_{\theta}}(\theta) \right|$. This quantity is nonzero because neither γ nor the correct instance labels are available, which makes it impossible to know what c_θ is, and therefore, what f^*_θ is. For this reason, under the information available, the best approximation of the characteristic accuracy of a feature is $Acc^{\gamma}_{f^{\gamma}_{\theta}}(\theta)$. In this subsection, I will measure how good of an approximation this measurement is, and describe the factors that control the quality of this measurement.

While thus far accuracy has been measured over the entire marginal distribution $D_{\mathcal{X}|\theta}$, it is also possible to break down the accuracy of a concept to the level of individual points. A deterministic concept always predicts the same result for a point, and the p-concept has a constant value on a point, so the pointwise accuracy of the concept is simply the amount of time that it predicts correctly on that point. Consider the concept $f^*_\theta = \mathbb{1}[c_\theta(x) > 0.5]$. It will be rare to see a pointwise accuracy of 1, because both positive and negative instances may be drawn at the same point, with probabilities determined by the p-concept, and the concept will only be correct on instances of one label type. For example, if $c_\theta(x) = 0.1$, $f^*_\theta(x) = 0$, and f^*_θ will have a pointwise accuracy of 0.9 because 90% of the time, negative-labeled points are drawn and f^*_θ predicts correctly, but 10% of the time, positive-labeled points are drawn, and f^*_θ predicts incorrectly. In general, the pointwise accuracy of f^*_θ with respect to c_θ is:

$$Acc^{c_\theta}_{f^{*}_{\theta}}(x) = \begin{cases} 1 - c_\theta(x) & f^*_\theta(x) = 0 \\ c_\theta(x) & f^*_\theta(x) = 1 \end{cases} \tag{3.14}$$

Similarly, the perceived accuracy of f^{γ}_{θ} is with respect to c^{γ}_{θ}, not to c_θ, because the labels that are used to calculate the accuracy come from c^{γ}_{θ}. Therefore, the pointwise

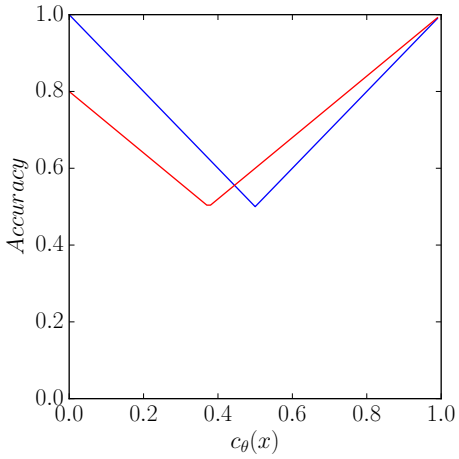


Figure 3.4: Pointwise accuracies for f^*_θ (blue) and f^γ_θ (red), with respect to $c_\theta(x)$, with $\gamma = 0.8$

accuracy of f^γ_θ with respect to c^γ_θ is

$$Acc^{c^\gamma_\theta}_{f^\gamma_\theta}(x) = \begin{cases} 1 - c^\gamma_\theta(x) & f^\gamma_\theta(x) = 0 \\ c^\gamma_\theta(x) & f^\gamma_\theta(x) = 1 \end{cases} \tag{3.15}$$

Figure 3.4 plots $Acc^{c_\theta}_{f^*_\theta}(x)$ against $Acc^{c^\gamma_\theta}_{f^\gamma_\theta}(x)$. If $Acc^{c_\theta}_{f^*_\theta}(x) = Acc^{c^\gamma_\theta}_{f^\gamma_\theta}(x)$, then on point x, f^*_θ and f^γ_θ have the same accuracy, and because the marginal distribution $D_{\mathcal{X}|\theta}$ is independent of noise, making x equally likely in both cases, the contribution of x to $\Delta^\gamma_{Acc}(\theta)$ will be 0. If $Acc^{c_\theta}_{f^*_\theta}(x) \neq Acc^{c^\gamma_\theta}_{f^\gamma_\theta}(x)$, however, then there will be some accuracy difference on the point. The total value of $\Delta^\gamma_{Acc}(\theta)$ can be written as the total difference in pointwise accuracy between f^*_θ and f^γ_θ, integrated over all points, with respect to the marginal distribution:

$$\left| Acc^\gamma_{f^\gamma_\theta}(\theta) - Acc_{f^*_\theta}(\theta) \right| = \left| \int_{\mathcal{X}|\theta} Acc^{c_\theta}_{f^*_\theta}(x) - Acc^{c^\gamma_\theta}_{f^\gamma_\theta}(x) \mathrm{dPr}(x) \right| \tag{3.16}$$

$Acc^{c^\gamma_\theta}_{f^\gamma_\theta}(x)$ is neither always greater than nor always less than $Acc^{c_\theta}_{f^*_\theta}(x)$. Where $Acc^{c^\gamma_\theta}_{f^\gamma_\theta}(x) > Acc^{c_\theta}_{f^*_\theta}(x)$, the noise causes f^γ_θ to believe that it has achieved a greater pointwise accuracy than is actually possible on the point under $c_\theta(x)$. When this

occurs, I say that f_θ^γ is optimistic. Similarly, where $Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x) < Acc_{f_\theta^*}^{c_\theta}(x)$, I say that f_θ^γ is pessimistic. Since optimism and pessimism counteract each other, the actual difference in accuracy due to noise will be the difference between the difference due to optimism and the difference due to pessimism.

In order to aid analysis of $\Delta_{Acc}^\gamma(\theta)$, it is helpful to examine the optimistic and pessimistic cases separately. Let $\mathcal{O}_\theta$ be the difference in accuracy due to optimism, and $\mathcal{P}_\theta$ be the difference in accuracy due to pessimism:

$$\mathcal{O}_\theta = \int_{\mathcal{X}|\theta} \left(Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x) - Acc_{f_\theta^*}^{c_\theta}(x) \right) \mathbb{1}[Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x) > Acc_{f_\theta^*}^{c_\theta}(x)] \mathrm{dPr}(x) \tag{3.17}$$

$$\mathcal{P}_\theta = \int_{\mathcal{X}|\theta} \left(Acc_{f_\theta^*}^{c_\theta}(x) - Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x) \right) \mathbb{1}[Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x) < Acc_{f_\theta^*}^{c_\theta}(x)] \mathrm{dPr}(x) \tag{3.18}$$

The total difference in accuracy due to noise $\Delta_{Acc}^\gamma(\theta) = |\mathcal{O}_\theta - \mathcal{P}_\theta|$.

To give an example, suppose that $D_{\mathcal{X}|\theta}$ places half of the probability mass on point x_1, with $c_\theta(x_1) = 0.4$, and the other half of the probability mass on point x_2 with $c_\theta(x_2) = 0.8$. Further, suppose that $\gamma = 0.8$, so the noise $\eta = 1 - \gamma = 0.2$. Because $c_\theta^\gamma(x) = c_\theta(x) + (1-\gamma)(1 - c_\theta(x))$, $c_\theta^\gamma(x_1) = 0.52$ and $c_\theta^\gamma(x_2) = 0.84$.

At x_1, $f_\theta^*(x_1) = 0$ while $f_\theta^\gamma(x_1) = 1$, so $Acc_{f_\theta^*}^{c_\theta}(x_1) = 1 - c_\theta(x_1) = 0.6$ while $Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x_1) = c_\theta^\gamma(x_1) = 0.52$. At x_2, $f_\theta^*(x_2) = 1$ and $f_\theta^\gamma(x_2) = 1$, so $Acc_{f_\theta^*}^{c_\theta}(x_2) = c_\theta(x_2) = 0.8$ and $Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x) = c_\theta^\gamma(x_2) = 0.84$.

Since $Acc_{f_\theta^*}^{c_\theta}(x_1) > Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x_1)$, f_θ^γ is pessimistic on x_1, and x_1 contributes to $\mathcal{P}_\theta$. On the other hand, $Acc_{f_\theta^*}^{c_\theta}(x_2) < Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x_2)$, so f_θ^γ is optimistic on x_2, meaning x_2 contributes to $\mathcal{O}_\theta$.

Over the entire distribution $D_{\mathcal{X}|\theta}$, $Acc_{f_\theta^*}^{c_\theta} = 0.5(0.6) + 0.5(0.8) = 0.7$. $Acc_{f_\theta^\gamma}^{c_\theta^\gamma} = 0.5(0.52) + 0.5(0.84) = 0.68$. It makes sense that $Acc_{f_\theta^\gamma}^{c_\theta^\gamma} < Acc_{f_\theta^*}^{c_\theta}$, because the difference in accuracy due to optimism $\mathcal{O}_\theta = (0.84 - 0.8)(0.5) = 0.02$, which leads f_θ^γ to have a higher perceived accuracy than f_θ^*'s accuracy, is less than the difference in accuracy due to pessimism $\mathcal{P}_\theta = (0.6 - 0.52)(0.5) = 0.04$, which causes f_θ^* to have a higher

accuracy. The total accuracy difference is $|0.68 - 0.7| = 0.02$, which is equal to $|\mathcal{O}_\theta - \mathcal{P}_\theta| = |0.02 - 0.04|$.

Since $Acc^{c_\theta}_{f^*_\theta}(x)$ and $Acc^{c^\gamma_\theta}_{f^\gamma_\theta}(x)$ are based on $c_\theta(x)$ and $c^\gamma_\theta(x)$, respectively, it makes sense to analyze the contributions to $\mathcal{O}_\theta$ and $\mathcal{P}_\theta$ in the context of the relationship between $c_\theta(x)$ and $c^\gamma_\theta(x)$. I will present several cases of this relationship, with consistent and easily-calculated bounds and contributions within each.

There are three possible cases for the relationship between $c_\theta(x)$ and $c^\gamma_\theta(x)$:

Case A: $c_\theta(x) < 0.5$ and $c^\gamma_\theta(x) < 0.5$

Case B: $c_\theta(x) > 0.5$ and $c^\gamma_\theta(x) > 0.5$

Case C: $c_\theta(x) < 0.5$ and $c^\gamma_\theta(x) > 0.5$

The case where $c_\theta(x) > 0.5$ and $c^\gamma_\theta(x) < 0.5$ need not be considered, because the one-sided noise guarantees that $c^\gamma_\theta(x) \geq c_\theta(x)$ at all points x .

Since $c^\gamma_\theta(x) \geq c_\theta(x)$, case A includes all points where $c^\gamma_\theta(x) < 0.5$:

$$\begin{aligned} c^\gamma_\theta(x) &< 0.5 \\ c_\theta(x) + (1-\gamma)(1 - c_\theta(x)) &< 0.5 \\ 1 + \gamma(c_\theta(x) - 1) &< 0.5 \\ 0.5 &< \gamma(1 - c_\theta(x)) \\ \frac{0.5}{\gamma} &< 1 - c_\theta(x) \\ c_\theta(x) &< 1 - \frac{0.5}{\gamma} \end{aligned} \tag{3.19}$$

Both f^*_θ and f^γ_θ will predict 0, so $Acc^{c_\theta}_{f^*_\theta}(x) = 1 - c_\theta(x)$ and $Acc^{c^\gamma_\theta}_{f^\gamma_\theta}(x) = 1 - c^\gamma_\theta(x)$. $c^\gamma_\theta(x) \geq c_\theta(x)$, so $Acc^{c^\gamma_\theta}_{f^\gamma_\theta}(x) \leq Acc^{c_\theta}_{f^*_\theta}(x)$ and all points that fall under case A will

contribute to $\mathcal{P}_\theta$. The contribution to $\mathcal{P}_\theta$ is:

$$\begin{aligned}(1 - c_\theta(x)) - (1 - (c_\theta(x) + (1-\gamma)(1 - c_\theta(x))))& \\ &= (1 - c_\theta(x)) - (1 - (c_\theta(x) + 1 - \gamma - c_\theta(x) + \gamma c_\theta(x))) \\ &= (1 - c_\theta(x)) - (1 - c_\theta(x) - 1 + \gamma - (\gamma - 1)c_\theta(x)) \\ &= (1 - c_\theta(x)) - (\gamma - \gamma c_\theta(x)) \\ &= (1 - \gamma)(1 - c_\theta(x)) \qquad (3.20)\end{aligned}$$

The boundaries for case B are simpler to determine than those of case A: $c_\theta^\gamma(x) \geq c_\theta(x)$, so case B encompasses all points where $c_\theta(x) \geq 0.5$. Similar to case A, both f_θ^* and f_θ^γ predict 1, so $Acc_{f_\theta^*}^{c_\theta}(x) < Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x)$ at all points that fall under case B; therefore, all such points will contribute to $\mathcal{O}_\theta$. The contribution to $\mathcal{O}_\theta$ is simply $c_\theta^\gamma(x) - c_\theta(x) = (1-\gamma)(1 - c_\theta(x))$, exactly the same as in case A.

Case C is best split up into two separate cases. To illustrate why, suppose $c_\theta(x) = 0.3$, so f_θ^* predicts 0 and $Acc_{f_\theta^*}^{c_\theta}(x) = 0.7$. Now consider $c_\theta^\gamma(x) > 0.5$, where f_θ^γ predicts 1 and $Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x) = c_\theta^\gamma(x)$. If $c_\theta^\gamma(x) = 0.6$, then $Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x) = 0.6 < 0.7$, and x contributes to $\mathcal{P}_\theta$. If, however, $c_\theta^\gamma(x) = 0.9$, then $Acc_{f_\theta^\gamma}^{c_\theta^\gamma}(x) = 0.9 > 0.7$, and x contributes to $\mathcal{O}_\theta$. The threshold on $c_\theta(x)$ between points that contribute to pessimism and points that contribute to optimism occurs at the value of $c_\theta(x)$ where $1 - c_\theta(x) = c_\theta^\gamma(x)$. All points that have values of $c_\theta(x)$ between the cutoff for case A and this threshold will contribute to $\mathcal{P}_\theta$, since $c_\theta^\gamma(x)$ will be on the side of $(1 - c_\theta(x))$ closer to 0.5; therefore, I call this case $\mathrm{C}_\mathcal{P}$. Similarly, all points whose values $c_\theta(x)$ are between this threshold and 0.5 will contribute to $\mathcal{O}_\theta$, so I call this case $\mathrm{C}_\mathcal{O}$.

The threshold on $c_\theta(x)$ is the value at which $1 - c_\theta(x) = c_\theta^\gamma(x)$:

$$\begin{aligned} 1 - c_\theta(x) &= c_\theta(x) + (1-\gamma)(1 - c_\theta(x)) \\ 1 - 2c_\theta(x) &= 1 - \gamma - c_\theta(x) + \gamma c_\theta(x) \\ -c_\theta(x) &= -\gamma + \gamma c_\theta(x) \\ (1+\gamma)c_\theta(x) &= \gamma \\ c_\theta(x) &= \frac{\gamma}{1+\gamma} \end{aligned} \tag{3.21}$$

In case $\mathrm{C}_\mathcal{P}$, $Acc^{c_\theta}_{f^*_\theta}(x) = 1 - c_\theta(x) \geq Acc^{c^\gamma_\theta}_{f^\gamma_\theta}(x) = c^\gamma_\theta(x)$. The contribution to $\mathcal{P}_\theta$ is:

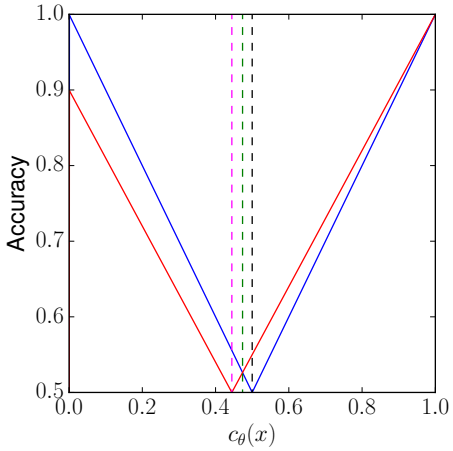
$$\begin{aligned} &1 - c_\theta(x) - c^\gamma_\theta(x) \\ &\qquad = 1 - c_\theta(x) - (c_\theta(x) + (1-\gamma)(1 - c_\theta(x))) \\ &\qquad = 1 - 2c_\theta(x) - 1 + \gamma + c_\theta(x) - \gamma c_\theta(x) \\ &\qquad = -c_\theta(x) + \gamma - \gamma c_\theta(x) \\ &\qquad = \gamma(1 - c_\theta(x)) - c_\theta(x) \end{aligned} \tag{3.22}$$

In case $\mathrm{C}_\mathcal{O}$, the same measurements of accuracy apply, so the contribution to $\mathcal{O}_\theta$ will simply be the negation of the contribution of the points in case $\mathrm{C}_\mathcal{P}$, $c_\theta(x) - \gamma(1 - c_\theta(x))$.

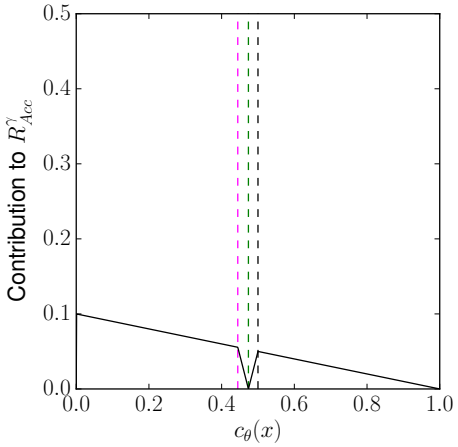
Table 3.1 summarizes the contributions for each case and the set of $c_\theta(x)$ values whose points fall into each case. Some interesting aspects of this analysis are worth pointing out. First, note that all points whose values of $c_\theta(x)$ are less than $\frac{\gamma}{1+\gamma}$

	Domain (c_θ)	Contributes to	Contribution
A	$(0, 1 - \frac{0.5}{\gamma})$	$\mathcal{P}_\theta$	$(1-\gamma)(1 - c_\theta(x))$
$\mathrm{C}_\mathcal{P}$	$(1 - \frac{0.5}{\gamma}, \frac{\gamma}{1+\gamma})$	$\mathcal{P}_\theta$	$\gamma(1 - c_\theta(x)) - c_\theta(x)$
$\mathrm{C}_\mathcal{O}$	$(\frac{\gamma}{1+\gamma}, 0.5)$	$\mathcal{O}_\theta$	$c_\theta(x) - \gamma(1 - c_\theta(x))$
B	$(0.5, 1)$	$\mathcal{O}_\theta$	$(1-\gamma)(1 - c_\theta(x))$

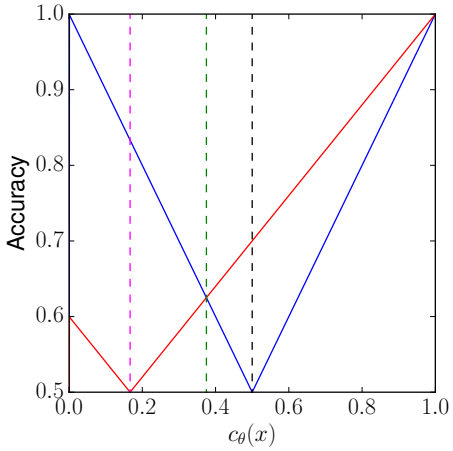
Table 3.1: Summary of the four cases when measuring accuracy



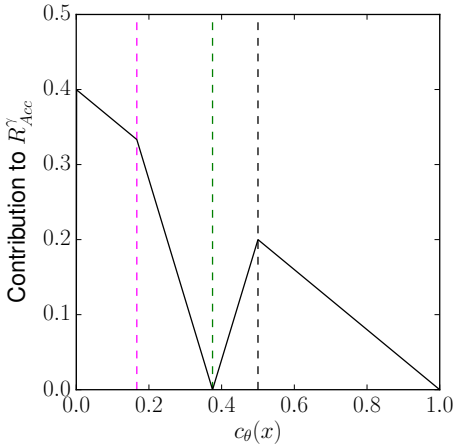
(a) Pointwise accuracies, $\gamma = 0.9$



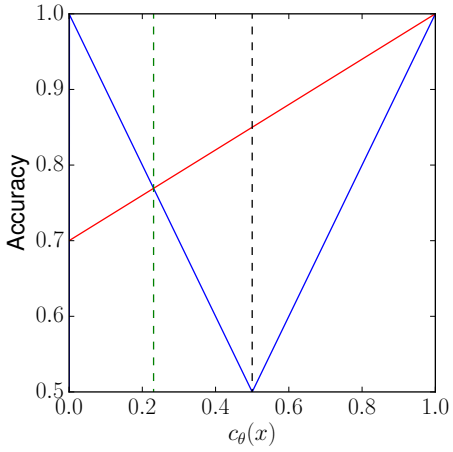
(b) Contribution to R^γ_{Acc}, $\gamma = 0.9$



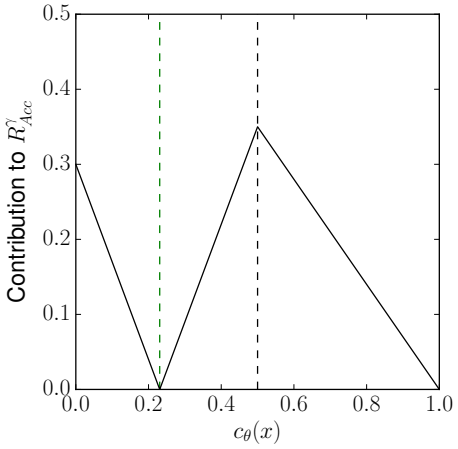
(c) Pointwise accuracies, $\gamma = 0.6$



(d) Contribution to R^γ_{Acc}, $\gamma = 0.6$



(e) Pointwise accuracies, $\gamma = 0.3$



(f) Contribution to R^γ_{Acc}, $\gamma = 0.3$

Figure 3.5: Charts of pointwise accuracy values (left) and absolute contributions to $R^\gamma_{Acc}(\theta)$ (right) by value of $c_\theta(x)$. The dashed lines represent the boundaries between A and $C_\mathcal{P}$ (purple), $C_\mathcal{P}$ and $C_\mathcal{O}$ (green), and $C_\mathcal{O}$ and B (black). On the left, the blue line is $Acc^{c_\theta}_{f_\theta}$ while the red line is $Acc^{c^\gamma_\theta}_{f^\gamma_\theta}$

contribute to $\mathcal{P}_\theta$, while all points with higher values of $c_\theta(x)$ contribute to $\mathcal{O}_\theta$. This is because for all values of $c_\theta(x)$ less than $\frac{\gamma}{1+\gamma}$, $c_\theta^\gamma(x)$ will be closer to 0.5 than $c_\theta(x)$, while for all higher values of $c_\theta(x)$, $c_\theta^\gamma(x)$ will be further away from 0.5.

Observe that the contributions of the cases are intuitive in the following ways. The contributions of case A and case B are both equivalent to the extra positivity provided to $c_\theta^\gamma(x)$ due to noise, or the frequency with which negative-labeled points will be mis-labeled. The contributions of both C-cases are equal to the distance between $c_\theta^\gamma(x)$ and the reflection of $c_\theta(x)$ across 0.5. The contributions of cases A and B get smaller as $c_\theta(x)$ increases, since the noise is only on negative instances. The contributions of the two C cases get larger as $c_\theta(x)$ gets further from $\frac{\gamma}{1+\gamma}$, because this means $c_\theta^\gamma(x)$ will be further from $1 - c_\theta(x)$.

Case A only exists when $\gamma > 0.5$. As γ gets smaller, which causes more noise, case A also gets smaller. Equivalently, Case $\mathrm{C}_\mathcal{P}$'s lower boundary moves further and further down as γ decreases, until it hits 0 for $\gamma < 0.5$. Lowering γ also causes $\mathrm{C}_\mathcal{P}$'s upper boundary to decrease, although the movement of the lower boundary will be faster than the movement of the upper boundary, causing the domain of $\mathrm{C}_\mathcal{P}$ to increase as γ decreases until $\gamma = 0.5$, when the lower bound is fixed at 0 and the size of $\mathrm{C}_\mathcal{P}$ decreases as γ moves to 0. As $\mathrm{C}_\mathcal{P}$'s upper boundary decreases, the size of $\mathrm{C}_\mathcal{O}$ increases, as the upper boundary of its range is the fixed constant 0.5. The set of points in case B never changes as both bounds are constant.

3.2.5 Finding a Correct Feature Ranking

Recall from subsection 3.2.1 that so long as the accuracy difference $\Delta_{Acc}(\theta)$ is less than $\frac{\tau}{2}$ for every θ, MI-FEAR with accuracy will find a correct feature ranking. This quantity is bounded from above by $\left|Acc_{h_\theta}^\gamma(\theta) - Acc_{f_\theta^\gamma}^\gamma(\theta)\right| + \left|Acc_{f_\theta^\gamma}^\gamma(\theta) - Acc_{f_\theta^*}(\theta)\right|$ (Equation 3.5). In subsection 3.2.3, I showed that $\left|Acc_{h_\theta}^\gamma(\theta) - Acc_{f_\theta^\gamma}^\gamma(\theta)\right|$ can be brought below ϵ with probability $\geq 1 - \delta_\theta$. In subsection 3.2.4, I showed that

$\Delta^{\gamma}_{Acc}(\theta) = \left| Acc^{\gamma}_{f^{\gamma}_{\theta}}(\theta) - Acc_{f^{*}_{\theta}}(\theta) \right|$ is constant for θ. It is for this reason that Theorem 1 requires that the maximum value of Δ^{γ}_{Acc} for all features, $\Delta^{\gamma*}_{Acc}$, be less than $\frac{\tau}{2}$.

If $\Delta^{\gamma*}_{Acc}$ is the largest value of Δ^{γ}_{Acc}, and τ is the smallest separation between two features in the ranking, then so long as $\epsilon < \frac{\tau}{2} - \Delta^{\gamma*}_{Acc}$, $\epsilon + \Delta^{\gamma}_{Acc}(\theta)$ will be less than $\frac{\tau}{2}$ for all θ, guaranteeing a correct ranking. The only issue is that, because PAC learning only guarantees that $\left| Acc^{\gamma}_{h_\theta}(\theta) - Acc^{\gamma}_{f^{\gamma}_{\theta}}(\theta) \right| < \epsilon$ with probability $\geq 1 - \delta_\theta$, a value of δ_θ smaller than the probability bound for the entire ranking is required, because all hypotheses are learned independently. It is necessary to set δ_θ so that it is highly likely that $\left| Acc^{\gamma}_{h_\theta}(\theta) - Acc^{\gamma}_{f^{\gamma}_{\theta}}(\theta) \right| < \epsilon$ for all features θ. Using the union bound:

$$\begin{aligned} \Pr\left[\bigcup_{\theta \in \Theta} \left[\Delta_{Acc}(\theta) \geq \frac{\tau}{2} \right] \right] &\leq \sum_{i=1}^{d} \Pr\left[\Delta_{Acc}(\theta) \geq \frac{\tau}{2} \right] \\ &< \sum_{i=1}^{d} \delta_\theta \\ &= d\delta_\theta \end{aligned} \tag{3.23}$$

That is, in order for the ranking to be correct with probability $\geq \delta$, the probability that any individual $\Delta_{Acc}(\theta) \geq \frac{\tau}{2}$ must be bounded from above by δ. The probability of this being the case is bounded from above by $d\delta_\theta$, so if $\delta = d\delta_\theta$, the ranking will hold with sufficient probability. Because PAC learning is done on individual features, each feature must use $\delta_\theta = \frac{\delta}{d}$.

$\mathcal{A}$ is the same across all features, so the concept class $\mathcal{C}_\theta$ it learns will be the same, meaning the VC dimension of $\mathcal{C}_\theta$, k, will be the same. Substituting these values of $\delta_\theta = \frac{\delta}{d}$ and $\epsilon = \frac{\tau}{2} - \Delta^{\gamma*}_{Acc}$ into Equation 3.13 gives that MI-FEAR can learn a correct feature ranking with probability $\geq 1 - \delta$ using $O\left(\frac{1}{\frac{\tau}{2} - \Delta^{\gamma*}_{Acc}} \left(\log \frac{d}{\delta} + k \log \frac{1}{\frac{\tau}{2} - \Delta^{\gamma*}_{Acc}} \right) \right)$ instances.

3.3 Discussion

The ability of MI-FEAR to recover a good feature ranking depends on whether $\Delta_{Acc}(\theta)$ can be brought below $\frac{\tau}{2}$ for every feature θ. An upper bound of $\epsilon + \Delta^{\gamma}_{Acc}(\theta)$ can be placed on $\Delta_{Acc}(\theta)$. There are a number of factors which affect the tightness of this upper bound and the ability of MI-FEAR to tightly approximate all features. These include the number of features, the confidence δ required, the VC dimension k of the concept class $\mathcal{C}_\theta$, and the distribution of values of c_θ for each feature θ relative to the instance distribution $D_{\mathcal{X}|\theta}$.

The contribution of ϵ to the upper bound depends on the relationship between the number of instances available, which is constant, and the other factors in the complexity bound of Theorem 1. If the VC dimension of the concept class $\mathcal{C}_\theta$ is lower, then ϵ can also be lowered while still maintaining a complexity bound lower than the number of instances available. Similarly, a looser δ requirement causes a looser δ_θ requirement, which allows for a tighter ϵ across all features, a standard tradeoff in PAC-learning.

While the complexity bound may appear to be logarithmic in d, there is actually a hidden linear factor of d, which matches the intuition that at least as many instances as features will be required to learn a feature ranking. Note that ϵ is bounded from above by $\frac{\tau}{2}$, and that with τ defined as the minimum separation between accuracy metrics of two features, if there are d features, then the maximum value of τ is $\frac{1}{d-1} = O\left(\frac{1}{d}\right)$. The complexity bound also includes a linear term $\frac{1}{\epsilon}$, so in fact a linear dependence on the number of features is hidden in this term. Note that this is ignoring the fact that some features may have the same accuracy value; however, this is likely to only be a very small number of features.

Even with an infinite amount of instances, it will not always be possible to recover a proper feature ranking, because $\Delta^{\gamma}_{Acc}(\theta)$ provides a constant offset in accuracy for

each feature θ. This constant offset does not scale with the number of features, so in the limit of a very high number of features, τ may be very small, so it is very unlikely that $\Delta_{Acc}^{\gamma*} < \frac{\tau}{2}$. Even if this is not the case, it is possible to construct pathological features such that $\Delta_{Acc}^{\gamma}(\theta)$ is extremely high; for instance, a feature where $c_\theta(x)$ is always close to 0 and $\gamma = 0.5$ would have $\Delta_{Acc}^{\gamma} \approx 0.5$.

As MI-FEAR is not guaranteed to work in the general case due to Δ_{Acc}^{γ}, it is useful to analyze the factors affecting Δ_{Acc}^{γ} and under what circumstances MI-FEAR may or may not be able to recover a perfect ranking. $\Delta_{Acc}^{\gamma}(\theta)$ depends on γ, which is constant, and on the distribution of values of $c_\theta(x)$ relative to the instance distribution $D_{\mathcal{X}|\theta}$, which is specific to the feature θ. c_θ will take on a range of possible values over the marginal distribution $D_{\mathcal{X}|\theta}$; however, certain values of $c_\theta(x)$ lend themselves more to finding an accurate ranking than others. When aggregated over the entire p-concept, this helps provide an understanding of what sorts of functions c_θ will allow for a good feature ranking, and which will make finding a good feature ranking difficult.

When $\gamma = 1$, $c_\theta^\gamma = c_\theta$ because there is no noise, so $\Delta_{Acc}^{\gamma*} = 0$. When $\gamma = 0$, all instances are given positive labels, and the feature ranking task is essentially pointless (this scenario is not possible due to the $\gamma > 0$ constraint of MI-GEN). Otherwise, the values of $c_\theta(x)$ that contribute the most to $\Delta_{Acc}^{\gamma}(\theta)$ are 0 and 0.5, with 0 contributing more when $\frac{1}{3} < \gamma < 1$ and 0.5 contributing more when $\gamma < \frac{1}{3}$. Intuitively, this is because 0 is the value of $c_\theta(x)$ where f_θ^* has the highest accuracy, and 0.5 the value where f_θ^* has the lowest accuracy, so it makes sense that the noise would have the highest effect there. The values of $c_\theta(x)$ that contribute the least to $\Delta_{Acc}^{\gamma}(\theta)$ are $\frac{\gamma}{1+\gamma}$ and 1, because at these points the pointwise accuracy of f_θ^* equals the pointwise perceived accuracy of f_θ^γ, in the case of 1 because they both predict the same label with perfect pointwise accuracy (as the noise has no effect) and in the case of $\frac{\gamma}{1+\gamma}$ because they predict different labels with equal accuracies. As a generalization, the values of c_θ that contribute the most to $\Delta_{Acc}^{\gamma}(\theta)$ are those around the peaks of 0 and

0.5, and the values that contribute the least are those around the low points of 1 and $\frac{\gamma}{1+\gamma}$, with a lower value of γ leading to a higher $\Delta^{\gamma}_{Acc}(\theta)$, except for values close to 0, where the pointwise contribution to $\Delta^{\gamma}_{Acc}(\theta)$ peaks when $\gamma = 0.5$ before steadily declining as γ decreases.

Aside from the absolute pointwise contribution to $\Delta^{\gamma}_{Acc}(\theta)$ of the distribution of points, the absolute difference between the contributions to $\mathcal{O}_\theta$ and $\mathcal{P}_\theta$, which plays a significant role in mitigating $\Delta^{\gamma}_{Acc}(\theta)$, is also important. When there is a fairly even spread of values of $c_\theta(x)$, the contributions will tend to cancel each other out, leaving a low $\Delta^{\gamma}_{Acc}(\theta)$. As γ decreases, however, the threshold between $\mathcal{P}_\theta$ and $\mathcal{O}_\theta$ shifts to lower values of $c_\theta(x)$, causing more points to contribute to $\mathcal{O}_\theta$ and less to $\mathcal{P}_\theta$. A lower value of γ thus leads to a higher value of $\Delta^{\gamma}_{Acc}(\theta)$ not only because of the higher pointwise contributions but also because of the increase in the number of points for which the perceived accuracy is overly-optimistic, which causes a greater imbalance between optimism and pessimism.

In terms of which features are good to select for inclusion in the final feature space, a "useful" feature can be thought of as one which reveals a lot of information about the labels of the instances. A good proxy for this is whether the values of the p-concept c_θ are close to 0 and 1 most of the time. Conversely, a "useless" feature is one for which the values of the p-concept are close to 0.5 most of the time, revealing little information about the labels. In general, very useful features will have high $Acc_{f^*_\theta}(\theta)$ values, while useless features will have low $Acc_{f^*_\theta}(\theta)$ values, with 0.5 being the lowest possible value given that the accuracy of a Bayes-optimal concept is always at least 0.5.

An interesting fact about this is that, because the values of $c_\theta(x)$ for useful features are very close to 0 and 1, there will be very little optimism and very high pessimism. Similarly, useless features have values of $c_\theta(x)$ around 0.5, so there will be very little pessimism and very high optimism. What this means is that at the extreme ends of

the ranking, Δ^{γ}_{Acc} has the effect of "squeezing" the correct ranking so that the features with the highest true characteristic accuracies will have lower perceived characteristic accuracies, with the reverse being true for those with the lowest true characteristic accuracies.

Because consecutive useless or useful features in the correct ranking may have differences due to noise of the same type, it is worth examining whether any increase in pessimism or optimism may be offset by the corresponding increase or decrease in accuracy, so that at the extreme ends of the ranking, the correct ranking will be preserved. That is, is it ever the case that the true accuracy will increase while the perceived accuracy decreases, or vice versa? Looking at the pointwise accuracy charts in Figure 3.5, note that this occurs in the two intervals of c_θ values corresponding to Case C. Useless features will have a high volume of points whose c_θ values fall under case C, making it likely that the ranking will in fact be highly mixed-up at the lower end. So long as $\gamma > 0.5$, though, useful features will have points that fall primarily in cases A and B.

Let us assume that, for two features θ_1 and θ_2, an equal proportion of points are drawn that fall into cases A and B; that is, for both features, some fraction a of the points are in case A and the remaining b in case B. Let c_1, c_2 be the average values of c_θ in cases A and B, respectively, for θ_1, and c_3, c_4 the same values for θ_2. Then the difference in true characteristic accuracies between the two features is:

$$\begin{aligned}(a(1-c_1)+bc_2)-(a(1-c_3)+bc_4) & \\ &= a - ac_1 + bc_2 - a + ac_3 - bc_4 \\ &= a(c_3 - c_1) + b(c_2 - c_4)\end{aligned}$$

Similarly, the difference in perceived characteristic accuracies between the two features

is:

$$
\begin{aligned}
&(a(1-(c_1+(1-\gamma)(1-c_1)))+b(c_2+(1-\gamma)(1-c_2))) \\
&-(a(1-(c_3+(1-\gamma)(1-c_3)))+b(c_4+(1-\gamma)(1-c_4))) \\
&=(a\gamma(1-c_1)+b(1-\gamma+\gamma c_2))-(a\gamma(1-c_3)+b(1-\gamma+\gamma c_4)) \\
&=\gamma\left(a(c_3-c_1)+b(c_2-c_4)\right)
\end{aligned}
$$

Therefore, so long as two features have the same proportion of points fall under cases A and B, and so long as those constitute all the points, as they often will for highly useful features, any change in true accuracy will be matched by a change in characteristic accuracy scaled by a factor of γ, meaning the ranking can be maintained. This means that if there are features that are extremely useful, not only are they likely to be ranked highly by MI-FEAR with accuracy but they are likely to be ranked in the correct order as well.

In summary, MI-FEAR does not guarantee that the correct feature ranking can be recovered in all cases; however, it will work well in scenarios where the spread of values of $c_\theta(x)$ is fairly even, especially when those values are restricted to cases A and B in consistent proportions across instances. It will also tend to rank useful features highly even if they are not ranked in exactly the correct order. More features, a higher VC dimensions for the one-dimensional concept class, lower γ, and a tighter confidence requirement δ are all factors that increase the difficulty of recovering a correct feature ranking.

Chapter 4

A Theoretical Analysis of MI-FEAR with AUC

In this chapter I analyze the effect of changing the scoring function $\mathcal{S}$ used in MI-FEAR from accuracy to AUC. The primary motivation for this switch comes from the fact that MI-FEAR with accuracy learns a deterministic concept over a real-valued function, which intuitively seems unnatural and introduces several challenges. It is worth asking whether learning a real-valued concept would lead to a better approximation of a true feature ranking. This aligns with the distinction made by Kearns and Schapire between learning a deterministic decision rule and learning a real-valued model of probability as the two possible goals for learning a p-concept (see subsection 2.2.4) [Kearns and Schapire, 1993]; the former goal was covered in the previous chapter, while the latter will be covered in this chapter.

The AUC metric (section 2.5) takes the real-valued output of a concept and uses it to evaluate the quality of that concept. Where accuracy directly measures the ability of a feature to help predict the labels for the instances, AUC measures the ability of the feature to help build a ranking of the instances; however, this also contains lots of information about the labels, so AUC is still a valid way to measure a feature's

predictive capability. Accuracy can be thought of as evaluating the deterministic output of placing a threshold on a ranking of instances by their real-valued outputs, where the most positive instances are given labels of 1 and the rest are given labels of 0; typically this threshold is 0.5, although for many tasks it may not be. This makes AUC an easier metric to learn than accuracy, because AUC can be interpreted as measuring the quality of a ranking while accuracy can be interpreted as measuring the quality of a ranking plus a threshold. MI-FEAR is also, conceptually, ranking the features by how much value they independently have in terms of optimizing the metric $\mathcal{S}$. For many tasks, AUC may be the more appropriate metric; for example, in 3D-QSAR, the desired output in general is a ranking of the molecules in terms of which should be tested first in practice. For this reason, AUC is often used to measure the quality of a learner rather than accuracy, in which case a feature selection algorithm that optimizes for AUC may be more useful than one that optimizes for accuracy.

Because AUC measures the quality of a real-valued concept, the learning algorithm $\mathcal{A}$ must now produce a real-valued concept rather than a deterministic concept. For a concept class of real-valued concepts, the pseudo-dimension (subsection 2.2.5) is an appropriate metric to measure capacity. Thus while I assumed in chapter 3 that the VC dimension of f was finite, which implied that the VC dimension of the concept class learned by $\mathcal{A}$ was finite, in this chapter, I will assume that the pseudo-dimension of that concept class $PD(\mathcal{C}_\theta)$ is some finite value k, and that it contains the concept that maximizes the perceived AUC. The assumption from the previous chapter that the probability of a negative instance appearing in a negative bag is constant over the instance distribution holds in this chapter as well.

This chapter will be organized in the following manner: in section 4.1, I will introduce the characteristic AUC of a feature and provide definitions of many of the terms used in chapter 3 in the context of AUC. I will also state the theorem for feature ranking with AUC as a scoring function to be used in the remainder of the chapter.

In section 4.2, I will describe an optimal concept with respect to AUC, which can be approximated with arbitrary precision, as I demonstrate in section 4.3. In section 4.4, I will describe how the use of bag-labeled instances introduces a difference in AUC between the true characteristic AUC of a feature and its perceived characteristic AUC, and discuss how a good feature ranking is possible to obtain when using AUC. Finally, in section 4.5, I will discuss the factors that control when AUC is a good metric to use for learning a feature ranking, and compare AUC to accuracy as a scoring metric for this purpose.

4.1 AUC of a Probabilistic Concept

As described in section 2.5, AUC measures the probability that a randomly-drawn positive instance will be ranked higher than a randomly-drawn negative instance. Ordinarily, this is simply measured over all possible positive/negative pairs with respect to a probability distribution on those pairs. When points are projected onto a feature axis, however, as described in subsection 3.2.2, the labeling function is a p-concept c_θ rather than a deterministic concept. Therefore, every point x may have both a positive and a negative label, and AUC is slightly more complicated to both compute and conceptualize. For example, if $\Pr_\theta(x) = 0.1$ for some point x, and $c_\theta(x) = 0.7$, then there is an 0.07 probability, over the entire marginal distribution $D_{\mathcal{X}|\theta}$, of seeing an instance at point x with label 1, and an 0.03 probability of seeing an instance at point x with label 0.

	$b_{c_\theta}(x_2) = 1$	$b_{c_\theta}(x_2) = 0$
$b_{c_\theta}(x_1) = 1$	0.27	0.63
$b_{c_\theta}(x_1) = 0$	0.03	0.07

Table 4.1: Probabilities of labeling combinations when $c(x_1) = 0.9$ and $c(x_2) = 0.3$. Note that because AUC concerns only pairs of points where one is positive and the other is negative, the two probabilities of same-labeled points (0.27 and 0.07) are discarded in this analysis.

When formulated as the probability that two different-labeled instances are ranked correctly, AUC is a measurement taken over all pairs of points, rather than all points. For a particular pair of points (x_1, x_2), there will be some probability of a pair where x_1 is positive and x_2 is negative, and a different probability of a pair where x_1 is negative and x_2 is positive. Thus, there are two possible different labelings of instances drawn from the two points, both of which may have some non-zero probability of occurring. For example, suppose $c(x_1) = 0.9$ and $c(x_2) = 0.3$. The probabilities of observing each possible combination of labels for these two instances are given in Table 4.1. Because AUC only concerns pairs where one instance is positive and the other is negative, the two cases where a pair of instances is drawn from x_1, x_2 where both are labeled 1 or both are labeled 0 are ignored. The probability of observing the instances with two different labels is $0.63 + 0.03 = 0.66$. Conditioned on the two instances having different labels, $\Pr(b_{c_\theta}(x_1) = 1, b_{c_\theta}(x_2) = 0) = \frac{0.63}{0.66} \approx 0.95$, and $\Pr(b_{c_\theta}(x_1) = 0, b_{c_\theta}(x_2) = 1) = \frac{0.03}{0.66} \approx 0.05$. Thus, if a hypothesis f_θ were to assign a higher ranking to x_1 than x_2, regardless of what the exact real values were, it would be right on 95% of different-labeled pairs drawn from these two points, and wrong on 5% of such pairs. That probability that it is right, 95%, is the *pairwise AUC* for f_θ on (x_1, x_2) with respect to c_θ, $Auc_{f_\theta}^{c_\theta}(x_1, x_2)$. Assuming, without loss of generality, that x_1 is the point ranked higher by f_θ, it can be written in closed form as follows:

$$Auc_{f_\theta}^{c_\theta}(x_1, x_2) = \frac{c_\theta(x_1)(1 - c_\theta(x_2))}{c_\theta(x_1)(1 - c_\theta(x_2)) + c_\theta(x_2)(1 - c_\theta(x_1))} \tag{4.1}$$

This is simply the probability that a pair of instances x_1, x_2 is drawn where x_1 is positive and x_2 is negative, thus making f_θ's ranking of x_1 above x_2 correct, conditioned on the probability of drawing a pair of different-labeled instances from x_1, x_2. This can also be written as $Auc_{f_\theta}^{c_\theta}(x_1, x_2) = \Pr(b_{c_\theta}(x_1) = 1 \wedge b_{c_\theta}(x_2) = 0 | b_{c_\theta}(x_1) \neq b_{c_\theta}(x_2))$. The AUC of concept f_θ is this probability integrated over all possible in-

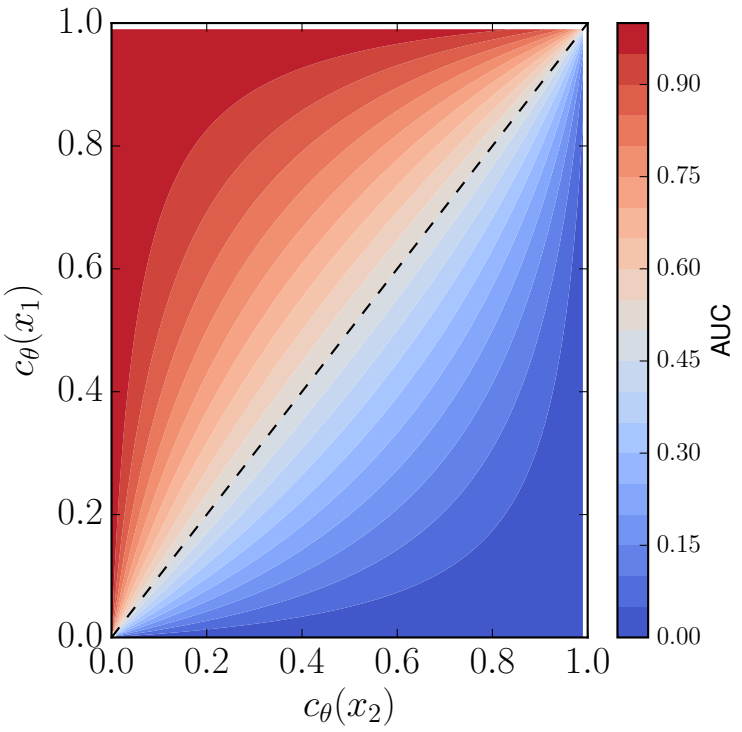


Figure 4.1: Pairwise AUC $Auc_{f_\theta}^{c_\theta}(x_1, x_2)$ of a function f_θ that ranks x_1 higher than x_2

stance pairs, with respect to the marginal distribution $D_{\mathcal{X}|\theta}$:

$$Auc_{f_\theta}(\theta) = \int_{\mathcal{X}|\theta} \int_{\mathcal{X}|\theta} \Pr(b_{c_\theta}(x_1) = 1 \wedge b_{c_\theta}(x_2) = 0 | b_{c_\theta}(x_1) \neq b_{c_\theta}(x_2)) \\ \mathbb{1}[f_\theta(x_1) \geq f_\theta(x_2)] \mathrm{dPr}(x_1) \mathrm{dPr}(x_2) \qquad (4.2)$$

In this formula, probabilities are taken with respect to the marginal distribution, and the indicator function ensures that, because a pair of instances may be encountered in either order, only the order where x_1 is ranked higher than x_2 is considered. A contour plot of AUC based on the values of $c_\theta(x_1)$ and $c_\theta(x_2)$ is given in Figure 4.1.

Many of the definitions and ideas introduced in subsection 3.2.1 can be easily modified to accommodate AUC instead of accuracy as a scoring function. The true characteristic AUC of a feature θ is the maximum AUC possible, over all possible real-valued concepts defined on θ, when measured against the true instance labels,

and the concept that maximizes $Auc_{f_\theta}(\theta)$ will be written f^*_θ. Similarly, the perceived characteristic AUC $Auc^\gamma_{f_\theta}(\theta)$ of a feature θ is its maximum-possible AUC when the instance labels provided by using bag-labeled instances are used, and the concept that maximizes this will be written f^γ_θ:

$$f^*_\theta = \arg\max_{f_\theta} Auc_{f_\theta}(\theta) \tag{4.3}$$

$$f^\gamma_\theta = \arg\max_{f_\theta} Auc^\gamma_{f_\theta}(\theta) \tag{4.4}$$

As in subsection 3.2.1, when features are ranked by their true characteristic AUCs, τ is the minimum difference between any two consecutive features in the true characteristic accuracy ranking. For a particular feature θ, the difference between its true characteristic AUC and its perceived characteristic AUC according to h_θ will be written $\Delta_{Auc}(\theta)$, which can be bounded from above by the sum of the difference in perceived AUCs between h_θ and f^γ_θ due to approximation and the difference between the perceived AUC of f^γ_θ and the AUC of f^*_θ:

$$\Delta_{Auc}(\theta) \leq \left| Auc^\gamma_{h_\theta}(\theta) - Auc^\gamma_{f^\gamma_\theta}(\theta) \right| + \left| Auc^\gamma_{f^\gamma_\theta}(\theta) - Auc_{f^*_\theta}(\theta) \right| \tag{4.5}$$

The second term will be written $\Delta^\gamma_{Auc}(\theta)$, and its maximum value over all features θ written $\Delta^{\gamma*}_{Auc}$. So long as $\Delta^{\gamma*}_{Auc} < \frac{\tau}{2}$, MI-FEAR with AUC can learn a correct feature ranking:

Theorem 2 *Let k be the pseudo-dimension of the concept class learned by $\mathcal{A}$ and $M^* = \max_\theta \mathbb{E}_{(x_1,x_2)\in(\theta\times\theta):c^\gamma_\theta(x_1)\neq c^\gamma_\theta(x_2)}\left[(c^\gamma_\theta(x_1)(1-c^\gamma_\theta(x_2)) + c^\gamma_\theta(x_2)(1-c^\gamma_\theta(x_1)))^{-1}\right]$. If $\Delta^{\gamma*}_{Auc} < \frac{\tau}{2}$, then with probability at least $1-\delta$, MI-FEAR with AUC can learn a correct*

feature ranking over d features using n examples, where

$$n = O\left(\frac{M^{*2}}{\left(\frac{\tau}{2} - \Delta^{\gamma*}_{Auc}\right)^2}\left(\log\frac{d}{\delta} + k\log\frac{M^*}{\left(\frac{\tau}{2} - \Delta^{\gamma*}_{Auc}\right)}\right)\right) \tag{4.6}$$

4.2 A Maximum-AUC Concept on a Feature Axis

As stated in the introduction to this chapter, I am assuming that f^{γ}_{θ} is contained in a concept class $\mathcal{C}_\theta$ of finite pseudo-dimension, and that this concept class is learned by the learning algorithm $\mathcal{A}$. The question remains, however, what exactly f^{γ}_{θ} is, and, similarly, what f^{*}_{θ} is. In this section I will show that the real-valued concept that maximizes AUC with respect to a p-concept is the concept that models that p-concept exactly.

Suppose there is some point x where $f^*_\theta(x) \neq c_\theta(x)$. This may lead to a ranking where $c_\theta(x_1) > c_\theta(x_2)$, but $f^*_\theta(x_1) < f^*_\theta(x_2)$. For example, maybe $c_\theta(x_1) = 0.9$ and $c_\theta(x_2) = 0.88$, but $f^*_\theta(x_1) = 0.87$ while $f^*_\theta(x_2) = c_\theta(x_2)$. If this is the case, then there will be a new pairwise AUC $Auc^{c_\theta}_{f^*_\theta}(x_2, x_1)$. The difference between this and the pairwise AUC of x_1 and x_2 if they had been ranked in the same way as c_θ, which would be the case if $f^*_\theta = c_\theta$, is as follows:

$$\begin{aligned}
&Auc^{c_\theta}_{c_\theta}(x_1, x_2) - Auc^{c_\theta}_{f^*_\theta}(x_2, x_1) \\
&\quad= \frac{c_\theta(x_1)(1 - c_\theta(x_2))}{c_\theta(x_1)(1 - c_\theta(x_2)) + c_\theta(x_2)(1 - c_\theta(x_1))} - \frac{c_\theta(x_2)(1 - c_\theta(x_1))}{c_\theta(x_2)(1 - c_\theta(x_1)) + c_\theta(x_1)(1 - c_\theta(x_2))} \\
&\quad= \frac{c_\theta(x_1)(1 - c_\theta(x_2)) - c_\theta(x_2)(1 - c_\theta(x_1))}{c_\theta(x_1)(1 - c_\theta(x_2)) + c_\theta(x_2)(1 - c_\theta(x_1))} \\
&\quad= \frac{c_\theta(x_1) - c_\theta(x_1)c_\theta(x_2) - c_\theta(x_2) + c_\theta(x_1)c_\theta(x_2)}{c_\theta(x_1)(1 - c_\theta(x_2)) + c_\theta(x_2)(1 - c_\theta(x_1))} \\
&\quad= \frac{c_\theta(x_1) - c_\theta(x_2)}{c_\theta(x_1)(1 - c_\theta(x_2)) + c_\theta(x_2)(1 - c_\theta(x_1))}
\end{aligned} \tag{4.7}$$

As the denominator is positive and $c_\theta(x_1) > c_\theta(x_2)$, this is a positive quantity; there-

fore, a function that ranks a pair of points differently from c_θ will always achieve a lower pairwise AUC on those points than it would have if it had ranked them in the same way as c_θ. Therefore, $f^*_\theta = c_\theta$ is the concept that maximizes true AUC. Similarly, $f^\gamma_\theta = c^\gamma_\theta$ is the function that maximizes perceived AUC.

As discussed in subsection 2.2.4, when estimating a model of probability, it is often most convenient to use the quadratic loss as a loss function; however, it has also been shown that minimizing $\mathbb{E}_{x \in D_{\mathcal{X}|\theta}}\left[|h_\theta(x) - f^\gamma_\theta(x)|\right]$ is equivalently solvable, with identical time complexity. For the purposes of this discussion, this definition of loss is more convenient, so empirical risk minimization will minimize $\mathbb{E}_{x \in D_{\mathcal{X}|\theta}}[|h_\theta(x) - f^\gamma_\theta(x)|]$. subsection 2.2.5 gives that ERM can guarantee that $\mathbb{E}_{x \in D_{\mathcal{X}|\theta}}[(h_\theta(x) - f^\gamma_\theta(x))^2] < \epsilon$ with probability $\geq 1 - \delta_\theta$ for some ϵ, δ_θ so long as the concept class $\mathcal{C}_\theta$ has finite pseudo-dimension (in this chapter, assumed to be k) and the number of samples n exceeds the following bound:

$$n \geq \frac{64}{\epsilon^2}\left(2k \log \frac{16e}{\epsilon} + \log \frac{8}{\delta_\theta}\right) = O\left(\frac{1}{\epsilon^2}\left(\log \frac{1}{\delta_\theta} + k \log \frac{1}{\epsilon}\right)\right) \tag{4.8}$$

4.3 Difference in Characteristic AUC Due to Approximation

In this section I will place an upper bound on the quantity $\left|Auc^\gamma_{h_\theta}(\theta) - Auc^\gamma_{f^\gamma_\theta}(\theta)\right|$. In subsection 3.2.3, a similar bound was simply given as ϵ, because ERM could be used to minimize predictive risk, thereby maximizing accuracy. In this chapter, I use ERM to minimize the expected loss; however, the fact that expected loss is less than ϵ does not necessarily imply that the difference in perceived AUCs between h_θ and f^γ_θ is less than ϵ.

From section 4.2, $\mathbb{E}[|h_\theta(x) - f^\gamma_\theta(x)|] < \epsilon$ with probability $\geq 1 - \delta_\theta$. Using Markov's

inequality, this implies:

$$\Pr\left[|h_\theta(x) - f_\theta^\gamma(x)| > t\right] < \frac{\epsilon}{t} \tag{4.9}$$

for some t. Recall that AUC is the probability that a randomly-selected positive/negative pair of points is ranked correctly. An upper bound on the difference $|Auc_{h_\theta}^\gamma(\theta) - Auc_{f_\theta^\gamma}^\gamma(\theta)|$ can be found by considering that for a particular instance pair, the pairwise perceived AUCs of h_θ and f_θ^γ will only be different if h_θ and f_θ^γ rank the two points differently. Taking this into consideration, the AUC difference can be bounded from above by the probability that f_θ^γ and h_θ rank a randomly-drawn positive/negative pair differently, multiplied by the absolute pairwise AUC difference between f_θ^γ and h_θ integrated over all point pairs that the two rank differently:

$$\left|Auc_{h_\theta}^\gamma(\theta) - Auc_{f_\theta^\gamma}^\gamma(\theta)\right| < \tag{4.10}$$
$$\int_{\mathcal{X}|\theta}\int_{\mathcal{X}|\theta} \mathbb{1}\left[f_\theta^\gamma(x_1) > f_\theta^\gamma(x_2) \wedge h_\theta(x_1) < h_\theta(x_2)\right] \left|Auc_{h_\theta}^{c_\theta^\gamma}(x_2, x_1) - Auc_{f_\theta^\gamma}^{c_\theta^\gamma}(x_1, x_2)\right| \mathrm{dPr}(x_1)\mathrm{dPr}(x_2)$$

As shown in section 4.2, because $f_\theta^\gamma = c_\theta^\gamma$, h_θ will always have a pairwise AUC lower than that of f_θ^γ, and the AUC difference between the two will be:

$$\left|Auc_{h_\theta}^{c_\theta^\gamma}(x_2, x_1) - Auc_{f_\theta^\gamma}^{c_\theta^\gamma}(x_1, x_2)\right| = \frac{c_\theta^\gamma(x_1) - c_\theta^\gamma(x_2)}{c_\theta^\gamma(x_1)(1 - c_\theta^\gamma(x_2)) + c_\theta^\gamma(x_2)(1 - c_\theta^\gamma(x_1))} \tag{4.11}$$

Suppose there exists some different-labeled pair x_1, x_2 so that $f_\theta^\gamma(x_1) > f_\theta^\gamma(x_2)$. Let $\omega_{(x_1,x_2)} = f_\theta^\gamma(x_1) - f_\theta^\gamma(x_2) > 0$ be the margin on the number line between the two values assigned two the two points by f_θ^γ. For convenience, I will refer to this as simply ω when the context is clear. Consider the distance on the number line between $h_\theta(x)$ and $f_\theta^\gamma(x)$ on an instance x. ω is the minimum distance that must be covered by the sum of those distances for x_1 and x_2.

This can be thought of as the minimum sum of the distances over the two instance

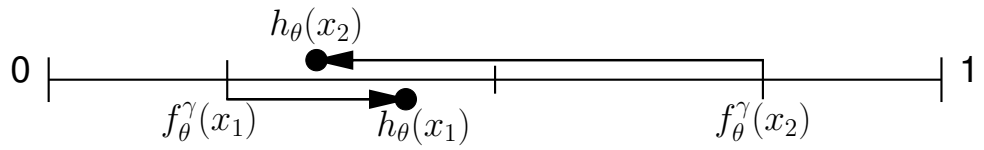


Figure 4.2: The outputs of h_θ on two points, relative to the outputs of f_θ^γ on the same points, have moved towards each other far enough to criss-cross in-between the two f_θ^γ values.

such that, in the worst case, h_θ's outputs would criss-cross on the number line when each $h_\theta(x)$ starts from the value $f_\theta^\gamma(x)$ and moves towards the opposite point's value. This is illustrated in Figure 4.2. The fact that ω is the minimum such difference can be expressed as:

$$f_\theta^\gamma(x_1) > f_\theta^\gamma(x_2) \wedge h_\theta(x_1) < h_\theta(x_2) \Rightarrow |f_\theta^\gamma(x_1) - h_\theta(x_1)| + |f_\theta^\gamma(x_2) - h_\theta(x_2)| > \omega \quad (4.12)$$

In order for the sum of the differences to exceed the margin, it is necessary for one of the differences to exceed $\frac{\omega}{2}$. This is equivalent to saying that the output of h_θ on at least one of the points in the pair must be far enough from the output of f_θ^γ on that point to cross over the midpoint between the outputs of f_θ^γ for the two points in the pair:

$$|f_\theta^\gamma(x_1) - h_\theta(x_1)| + |f_\theta^\gamma(x_2) - h_\theta(x_2)| > \omega \Rightarrow \quad (4.13)$$
$$\left(|f_\theta^\gamma(x_1) - h_\theta(x_1)| > \frac{\omega}{2}\right) \vee \left(|f_\theta^\gamma(x_2) - h_\theta(x_2)| > \frac{\omega}{2}\right)$$

Using the union bound:

$$\begin{aligned}
&\Pr(|f_\theta^\gamma(x_1) - h_\theta(x_1)| + |f_\theta^\gamma(x_2) - h_\theta(x-2)| \geq \omega) \\
&\qquad \leq \Pr\left(|f_\theta^\gamma(x_1) - h_\theta(x_1)| > \frac{\omega}{2} \vee |f_\theta^\gamma(x_2) - h_\theta(x_2)| > \frac{\omega}{2}\right) \\
&\qquad \leq \Pr\left(|f_\theta^\gamma(x_1) - h_\theta(x_1)| > \frac{\omega}{2}\right) + \Pr\left(|f_\theta^\gamma(x_2) - h_\theta(x_2)| > \frac{\omega}{2}\right) \\
&\qquad \leq 2\Pr\left(|f_\theta^\gamma(x) - h_\theta(x)| > \frac{\omega}{2}\right) \qquad (4.14)
\end{aligned}$$

Applying Equation 4.9, with $\frac{\omega}{2}$ substituted for t, this gives that the probability that f_θ^γ and h_θ rank a pair of instances x_1, x_2 differently is bounded from above by $\frac{4\epsilon}{\omega}$. Using the fact that $f_\theta^\gamma = c_\theta^\gamma$, this is $\frac{4\epsilon}{c_\theta^\gamma(x_1) - c_\theta^\gamma(x_2)}$. Plugging this into Equation 4.10, the $c_\theta^\gamma(x_1) - c_\theta^\gamma(x_2)$ terms cancel out, leaving:

$$\begin{aligned}
&\left|Auc_{h_\theta}^\gamma(\theta) - Auc_{f_\theta^\gamma}^\gamma(\theta)\right| < \qquad (4.15) \\
&\qquad \int_{\mathcal{X}|\theta}\int_{\mathcal{X}|\theta} \frac{4\epsilon}{c_\theta^\gamma(x_1)(1 - c_\theta^\gamma(x_2)) + c_\theta^\gamma(x_2)(1 - c_\theta^\gamma(x_1))} \mathrm{dPr}(x_1)\mathrm{dPr}(x_2)
\end{aligned}$$

The term $c_\theta^\gamma(x_1)(1 - c_\theta^\gamma(x_2)) + c_\theta^\gamma(x_2)(1 - c_\theta^\gamma(x_1))$ takes the form of a hyperbolic paraboloid that reaches 0 only when $c_\theta^\gamma(x_1)$ and $c_\theta^\gamma(x_2)$ both equal either 0 or 1, as shown in Figure 4.3. Such points are safe to ignore, however, as it does not make sense to talk about "ranking" two points with equal measures, and the AUC difference will be 0 even if h_θ and f_θ^γ rank the two points differently. Therefore, there will never be a case of dividing by 0, and this AUC difference is well-defined, and can be driven to 0 as $\epsilon \to 0$.

Let

$$\begin{aligned}
M_\theta &= \int_{\mathcal{X}|\theta}\int_{\mathcal{X}|\theta} \frac{1}{c_\theta^\gamma(x_1)(1 - c_\theta^\gamma(x_2)) + c_\theta^\gamma(x_2)(1 - c_\theta^\gamma(x_1))} \mathrm{dPr}(x_1)\mathrm{dPr}(x_2) \qquad (4.16) \\
&= \mathbb{E}_{(x_1,x_2)\in(\theta\times\theta):c_\theta^\gamma(x_1)\neq c_\theta^\gamma(x_2)}\left[(c_\theta^\gamma(x_1)(1 - c_\theta^\gamma(x_2)) + c_\theta^\gamma(x_2)(1 - c_\theta^\gamma(x_1)))^{-1}\right]
\end{aligned}$$

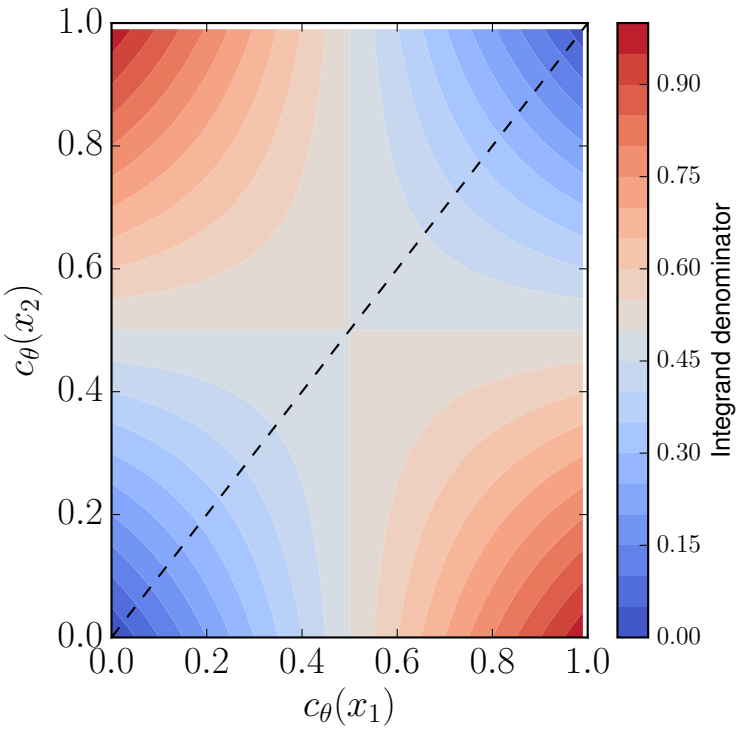


Figure 4.3: Pairwise values for the denominator in Equation 4.15, $c_\theta^\gamma(x_1)(1-c_\theta^\gamma(x_2))+c_\theta^\gamma(x_2)(1-c_\theta^\gamma(x_1))$

Then $\left|Auc_{h_\theta}^\gamma(\theta) - Auc_{f_\theta^\gamma}^\gamma(\theta)\right| < 4\epsilon M_\theta$, and in order to get $\left|Auc_{h_\theta}^\gamma(\theta) - Auc_{f_\theta^\gamma}^\gamma(\theta)\right| < \xi$ for some ξ, the value of ϵ required is $\frac{\xi}{4M_\theta}$. I assume here that M_θ is well-defined and can not go to infinity.

In Equation 2.6, I gave a complexity bound on the number of samples required to PAC-learn a probabilistic concept class. In order to get $\left|Auc_{h_\theta}^\gamma(\theta) - Auc_{f_\theta^\gamma}^\gamma(\theta)\right| < \xi$ with probability $\geq 1-\delta_\theta$, if $\mathcal{C}_\theta$ has pseudo-dimension $PD(\mathcal{C}_\theta) = k$, the number of samples required is:

$$n \geq \frac{1024M_\theta^2}{\xi^2}\left(2k\log\frac{256M_\theta e}{\xi} + \log\frac{8}{\delta_\theta}\right) = O\left(\frac{M_\theta^2}{\xi^2}\left(\log\frac{1}{\delta_\theta} + k\log\frac{M_\theta}{\xi}\right)\right) \quad (4.17)$$

4.4 Difference in Characteristic AUC Due to One-sided Noise

In this section, I will place an upper bound on the difference in characteristic AUC due to one-sided noise, $\Delta^{\gamma}_{Auc}(\theta) = \left| Auc^{\gamma}_{f^{\gamma}_{\theta}}(\theta) - Auc_{f^{*}_{\theta}}(\theta) \right|$. While every instance pair will be ranked in the same way by both f^{*}_{θ} and f^{γ}_{θ}, there will still be some difference in AUC due to the fact that the true instance labels are not available, meaning that the AUC of f^{γ}_{θ} must be calculated with respect to the noisy instance labels. I will then use the results of this section to show that it is possible, in many circumstances, to learn a perfect ranking with MI-FEAR when using AUC as a scoring function, thus proving Theorem 2.

To see that f^{γ}_{θ} will always rank a pair of instances the same as f^{*}_{θ}, note that $f^{\gamma}_{\theta}(x) = c^{\gamma}_{\theta}(x)$, $f^{*}_{\theta}(x) = c_{\theta}(x)$, and $c^{\gamma}_{\theta}(x) = c_{\theta}(x) + (1-\gamma)(1-c_{\theta}(x))$. Let x_1, x_2 be a pair such that $f^{*}_{\theta}(x_1) > f^{*}_{\theta}(x_2)$. Then it must be the case that $c_{\theta}(x_1) = c_{\theta}(x_2) + \nu$ for some ν. In this case, $c^{\gamma}_{\theta}(x_1)$ can be written as:

$$
\begin{aligned}
c^{\gamma}_{\theta}(x_1) &= c_{\theta}(x_1) + (1-\gamma)(1-c_{\theta}(x_1)) \\
&= (c_{\theta}(x_2) + \nu) + (1-\gamma)(1-(c_{\theta}(x_2)+\nu)) \\
&= c_{\theta}(x_2) + (1-\gamma)(1-c_{\theta}(x_2)) + \nu(1-(1-\gamma)) \\
&= c^{\gamma}_{\theta}(x_2) + \nu\gamma \geq c^{\gamma}_{\theta}(x_2)
\end{aligned}
$$

Because $f^{\gamma}_{\theta} = c^{\gamma}_{\theta}$, $f^{*}_{\theta}(x_1) > f^{*}_{\theta}(x_2) \Rightarrow f^{\gamma}_{\theta}(x_1) > f^{\gamma}_{\theta}(x_2)$. Therefore, the AUC difference between f^{γ}_{θ} and f^{*}_{θ} on a particular point pair x_1, x_2, assuming without loss of generality

that $c_\theta(x_1) > c_\theta(x_2)$, can be written as follows:

$$\begin{aligned} &Auc^{c_\theta^\gamma}_{f_\theta^\gamma}(x_1, x_2) - Auc^{c_\theta}_{f_\theta^*}(x_1, x_2) \\ &= \frac{c_\theta(x_1)(1 - c_\theta(x_2))}{c_\theta(x_1)(1 - c_\theta(x_2)) + c_\theta(x_2)(1 - c_\theta(x_1))} - \frac{c_\theta^\gamma(x_1)(1 - c_\theta^\gamma(x_2))}{c_\theta^\gamma(x_1)(1 - c_\theta^\gamma(x_2)) + c_\theta^\gamma(x_2)(1 - c_\theta^\gamma(x_1))} \end{aligned} \tag{4.18}$$

Unfortunately, this does not simplify to any easily-understandable form. For the purposes of analysis, however, contour plots representing the value of the pairwise difference in AUCs by the values of $c_\theta(x_1), c_\theta(x_2)$ are presented in Figure 4.4. The total difference in AUC between the two functions Δ_θ^γ is simply this quantity integrated over all possible point pairs with respect to the marginal distribution:

$$\left| Auc^\gamma_{f_\theta^\gamma}(\theta) - Auc_{f_\theta^*}(\theta) \right| = \left| \int_{\mathcal{X}|\theta} \int_{\mathcal{X}|\theta} Auc^{c_\theta^\gamma}_{f_\theta^\gamma}(x_1, x_2) - Auc^{c_\theta}_{f_\theta^*}(x_1, x_2) \mathrm{dPr}(x_1) \mathrm{dPr}(x_2) \right| \tag{4.19}$$

It should be noted that in Figure 4.4, because of the assumption that $c_\theta(x_1) > c_\theta(x_2)$, only values above the diagonal are to be considered. One fact that immediately jumps out is that every contour above the diagonal is negative, and the diagonal is always 0, which makes sense as it represents the line where $c_\theta(x_1) = c_\theta(x_2)$. This means that there is no need to consider notions of positivity and negativity as in subsection 3.2.4, because the perceived characteristic AUC of a feature will always be lower than its true characteristic AUC.

This difference in AUC based on the noise has an affect similar to the difference in accuracy based on noise of subsection 3.2.4; thus, $\Delta^{\gamma*}_{Auc}$ can be defined as the maximum such difference, and because the difference based on approximation ξ can be driven to 0 as $\epsilon \to 0$, it will be possible to get a good ranking so long as $\frac{\tau}{2} > \Delta^{\gamma*}_{Auc}$. ERM is used to PAC-learn f_θ^γ for each feature, so just as in subsection 3.2.5, the necessary value for the PAC parameter δ_θ to ensure that the entire ranking holds with probability $\geq 1 - \delta$ for some δ is $\frac{\delta}{d}$.

ϵ should be such so that even in the worst case of PAC learning, $\Delta^{\gamma*}_{Auc} + \xi < \frac{\tau}{2}$ with high probability. Let

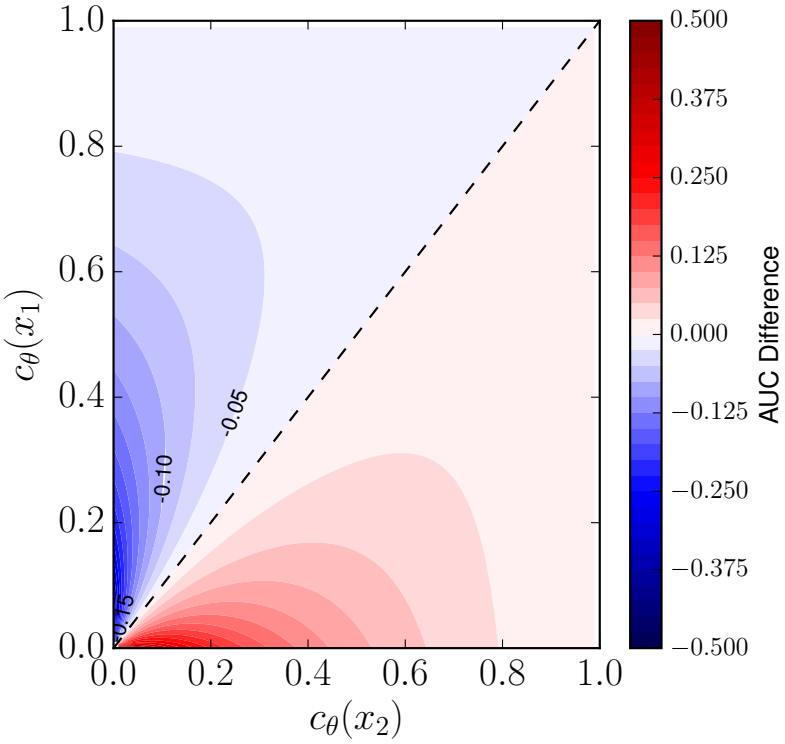
$$M^* = \max_\theta M_\theta = \max_\theta \mathbb{E}_{(x_1,x_2)\in(\theta\times\theta):c^\gamma_\theta(x_1)\neq c^\gamma_\theta(x_2)} \left[(c^\gamma_\theta(x_1)(1-c^\gamma_\theta(x_2)) + c^\gamma_\theta(x_2)(1-c^\gamma_\theta(x_1)))^{-1} \right] \tag{4.20}$$

As $\epsilon = \frac{\xi}{4M_\theta}$, the value of ϵ required for $\xi = \frac{\tau}{2} - \Delta^{\gamma*}_{Auc}$, the maximum possible value of ξ, is $\frac{\frac{\tau}{2}-\Delta^{\gamma*}_{Auc}}{4M_\theta}$ for some feature θ. The maximum value of ϵ that can be used while still being small enough for all features θ is $\epsilon = \frac{\frac{\tau}{2}-\Delta^{\gamma*}_{Auc}}{4M^*}$. Substituting these values of ϵ and δ_θ into Equation 4.17 gives that, so long as $\Delta^{\gamma*}_{Auc} < \frac{\tau}{2}$, MI-FEAR can learn a correct feature ranking with probability $\geq 1-\delta$ using $O\left(\frac{M^{*2}}{\left(\frac{\tau}{2}-\Delta^{\gamma*}_{Auc}\right)^2}\left(\log\frac{d}{\delta} + k\log\frac{M^*}{\left(\frac{\tau}{2}-\Delta^{\gamma*}_{Auc}\right)}\right)\right)$ instances.

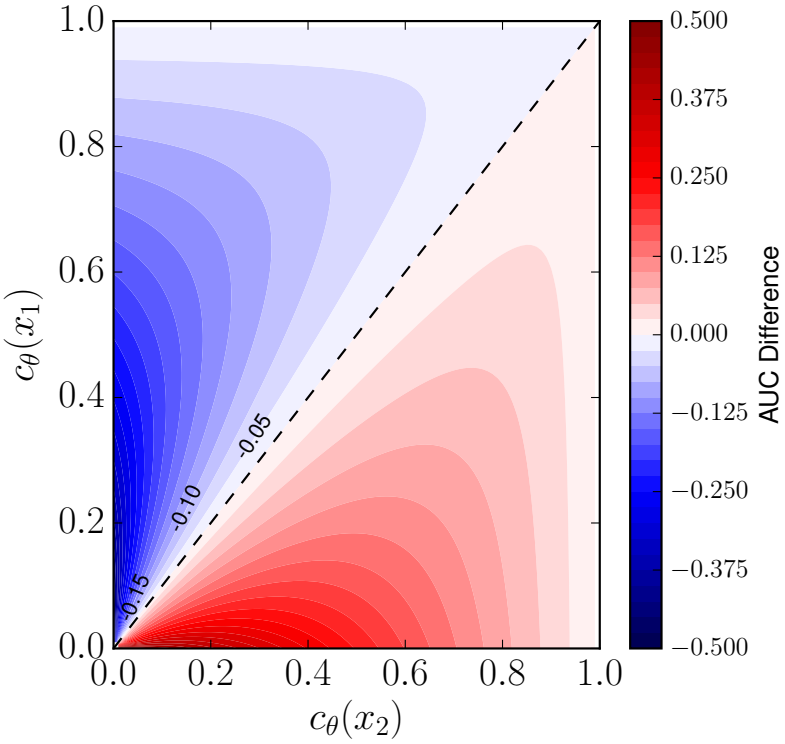
4.5 Discussion

Just as for MI-FEAR with accuracy, a true ranking is not guaranteed in the general case, as $\Delta^\gamma_{Acc}(\theta)$ does not scale with d. Examining the contour plots of Figure 4.4, however, reveals some interesting facts about the nature of $\Delta^\gamma_{Acc}(\theta)$. In Figure 4.4, the contours for -0.05, -0.10, and -0.15 are labeled. As γ decreases, the contours move further and further towards $(1.0, 1.0)$, which indicates, naturally, that the absolute pairwise AUC difference is greater with more noise. The point pairs that cause the most trouble are those in which one value of $c_\theta(x)$ is close to 0, while the other value is slightly higher, but not close to 1. In general, pairs of points whose values of $c_\theta(x)$ are very close to each other have pairwise $\Delta^\gamma_{Acc}(\theta)$ close to 0, so long as $c_\theta(x)$ is not close to 0. Furthermore, any point pair in which one of the values of $c_\theta(x)$ in the pair is close to 1 will have a low value of $\Delta^\gamma_{Acc}(\theta)$.

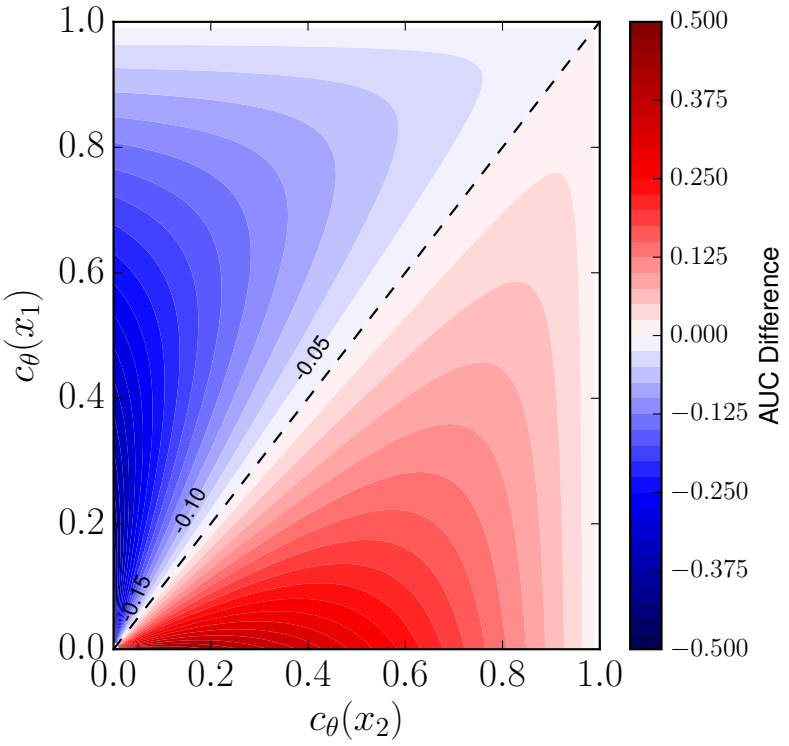
The hyperbolic paraboloid of Figure 4.3 does not affect $\Delta^\gamma_{Auc}(\theta)$; however, low values of this will contribute to a high value of M_θ, which may contribute to a high value of M^*, increasing the sample complexity. Here, extreme values of $c_\theta(x)$ will



(a) Point-pair-wise AUC difference with $\gamma = 0.9$



(b) Point-pair-wise AUC difference with $\gamma = 0.6$



(c) Point-pair-wise AUC difference with $\gamma = 0.3$

Figure 4.4: Contour plots of contribution to $\Delta^{\gamma}_{Auc}(\theta)$ for point pairs when $c^{\gamma}_{\theta}(x_1) > c^{\gamma}_{\theta}(x_2)$. Note that as $c_\theta(x_1) > c_\theta(x_2) \Leftrightarrow c^{\gamma}_{\theta}(x_1) > c^{\gamma}_{\theta}(x_2)$, only values above the diagonal are relevant.

lead to a higher value of M_θ, because the minimums for this contour plot are close to $(0,0)$ and $(1,1)$. While a maximum is at $(1,0)$, having a c_θ plot with lots of $(1,0)$ pairs will also mean that there are a lot of $(0,0)$ and $(1,1)$ pairs. With respect to the sample complexity, the best c_θ plot would be one that has a lot of values close to 0.5.

Recall, from section 3.3, that a useless feature is one that has lots of values close to 0.5, while a useful feature is one that has lots of values close to 0 and 1, thus revealing lots of information about the label. Useful features and useful features had high and low values accuracy, respectively; from Figure 4.1, it is evident that this distinction holds in the case of AUC, so that highly-useful features should be at the top of the correct AUC ranking, while highly-useless features will be at the bottom. As Figure 4.3 revealed, the highly useful features will require more instances to get a good approximation of the true characteristic AUC value. Both useful and useless features, however, will have fairly low values of $\Delta^{\gamma}_{Auc}(\theta)$. To see this, note that for useful features there are three possible pairings: $(0,0), (1,0)$, and $(1,1)$, or values close to these. Two of those three points correspond to the regions of the contour space that have the lowest values of $\Delta^{\gamma}_{Auc}(\theta)$. For useless features, all point pairs will be close to $(0.5, 0.5)$; as discussed above, point pairs with values of $c_\theta(x)$ close to each other will have very low values of Δ^{γ}_{θ}, so long as $c_\theta(x)$ is far from 0, which in the case of useless features it is.

In general, when using AUC as a metric rather than accuracy, there are a wider range of acceptable values for c_θ that will not lead to a terribly high value for the difference in characteristic AUC due to noise. Furthermore, the AUC metric is good for separating highly-useless features out of the ranking as well as identifying useful features, while the accuracy metric has trouble ranking useless features properly due to the fact that a lot of c_θ values will fall into the two C cases. Both accuracy and AUC metrics perform well on datasets that have a high rate of positive points and on datasets with a wide range of possible values, as in the case of AUC the regions that

contribute heavily to $\Delta^{\gamma}_{Auc}(\theta)$ cover only a small region of the space where $c_\theta(x_2)$ is close to 0.

The best possible ranking that can be learned using MI-FEAR with AUC will probably be closer to the correct ranking than when using accuracy; however, the drawback to using AUC is that the sample complexity includes not only an extra $\frac{1}{\left(\frac{\tau}{2}-\Delta^{\gamma*}_{Auc}\right)}$ term, but also the term M^{*2}, which may be quite high for scenarios in which there is at least one feature that is very useful. Therefore, even though the best possible ranking may be quite good, learning that ranking will be more difficult. If not many instances are available, accuracy may be the better metric to use.

Chapter 5

Empirical Evaluation

In this section, I will provide empirical results from using MI-FEAR with both Accuracy and AUC in practice. I will compare MI-FEAR to several other dimensionality reduction algorithms in terms of both performance and runtime. Performance on both the bag-level and instance-level tasks will be measured using both accuracy and AUC as final evaluation metrics (not to be confused with the use of these metrics in MI-FEAR). Additionally, I will compare the ranking of features from MI-FEAR to an approximation of the "correct" ranking that uses the true instance labels, which are available for the datasets selected.

I test several hypotheses in this section. The first is that MI-FEAR will be competitive with other dimensionality reduction algorithms, with competitiveness defined as statistical indistinguishability with respect to the AUC. Furthermore, I will test the hypothesis that the application of MI-FEAR can result in a performance improvement by comparing its performance to that of the same classifier when trained on the original feature set. Another hypothesis I will test is based on the runtime of these algorithms; because MI-FEAR is a feature selection algorithm, it is expected to have a much faster runtime than the feature construction methods to which it will be compared. Finally, I hypothesize that when the feature ranking output by MI-FEAR

is inspected, it will confirm the theoretical results, with better feature rankings being recovered on datasets that are estimated to better fit the conditions under which a good feature ranking may be found.

The experiments in this section were run in parallel across a number of servers belonging to Case Western Reserve University, including the High Performance Computing Resource in the Core Facility for Advanced Research Computing. All of the dimensionality reduction algorithms presented in this section were implemented manually, with the exception of Principal Components Analysis, for which I used the implementation available in the scikit-learn Python library [Pedregosa et al., 2011]. In implementing this algorithms I made extensive use of SciPy [Jones et al., 01] and NumPy [Ascher et al., 2001] in addition to scikit-learn.

5.1 MI-FEAR Implementation Details

When describing MI-FEAR from a theoretical perspective in chapter 3, I stated that the optimal deterministic concept f_θ^γ could be captured in a concept class $\mathcal{C}_\theta$ of finite VC dimension. I made a similar statement about the optimal real-valued concept of chapter 4. While these abstract, non-specific concept classes work for the purposes of a theoretical discussion, an implementation of MI-FEAR requires more concrete concept classes. The introduction of specific concept classes also allows for the introduction of learning algorithms specifically designed to learn those concept classes, which are likely to perform better then Empirical Risk Minimization, which fails to take advantage of any knowledge about the nature of the concept class. In this section I describe and justify the use of a deterministic concept class for the accuracy-based MI-FEAR and a probabilistic class for the AUC-based MI-FEAR, and give learning algorithms capable of learning each concept class. For both learning algorithms I will give the time complexities $t_\mathcal{A}$ and $t_\mathcal{S}$.

5.1.1 Union of Intervals

In general, a deterministic concept can be thought of as covering patches of the instance space where instances are to be labeled 1, with instances lying outside the concept being labeled 0. When the concept is only defined on a one-dimensional axis, this corresponds to covering intervals on the number line. Let k be a maximum number of intervals defined on feature θ. For some set of k pairs $\{(l_i, r_i)\}^k$, the concept of a union of intervals over this set is:

$$h(x) = \mathbb{1}\left[x \in \bigcup_{i=1}^{k} [l_i, r_i)\right] \tag{5.1}$$

Let each l_i be a point where the function $c_\theta^\gamma(x)$ goes from being less than 0.5 to greater than or equal to 0.5, and r_i be the closest point to the right of l_i where $c_\theta^\gamma(x)$ goes back below 0.5. Then each interval (l_i, r_i) will represent an interval where $c_\theta^\gamma(x)$ is greater than or equal to 0.5. This is exactly the function $\pi_{c_\theta^\gamma}$ which describes the concept f_θ^γ. So long as the number of intervals where $c_\theta^\gamma(x)$ is greater than or equal to 0.5 is less than or equal to k, then, $f_\theta^\gamma \in \mathcal{C}_\theta$.

In practice, the number of intervals k will always be finite, because the empirical dataset is finite; therefore, this is a reasonable concept class to use. A decision tree learner is capable of learning concepts that are described as unions of intervals. To implement accuracy-based MI-FEAR, I used the scikit-learn library, which includes a decision-tree learning algorithm whose time complexity is $t_\mathcal{A} = O(dn \log n)$, which in the one-feature case is simply $O(n \log n)$ [Pedregosa et al., 2011]. Therefore, since $t_\mathcal{S}$ is trivially $O(n)$, applying the time complexity equation for MI-FEAR from section 3.1 gives that the running time of this implementation of MI-FEAR with accuracy is $O(d(n \log n + \log d))$.

5.1.2 Nadaraya-Watson Kernel-Weighted Average

As explained in section 4.1, the best real-valued concept f_θ^γ with respect to AUC is the concept that exactly matches c_θ^γ. A concept class is therefore required that contains a concept matching a real-valued function curve. The Nadaraya-Watson kernel-weighted average function approximates the real value of a function at a particular point x_i with an average over all real values taken by the function over its domain, with the contribution of each point x_j weighted by a normalized kernel function between x_i and x_j [Friedman et al., 2001]. The RBF kernel (Equation 2.3) is a natural choice because it defines a smooth function around each point that decreases with distance from the point. Therefore, the values given by the function for points far away from x_i will be given a small weight, while the values from points close to x_i will be heavily-weighted. The function to estimate an output value for a real-valued function c_θ at point x is:

$$h(x) = \frac{\sum_{i=1}^{N} k(x, x_i) c_\theta(x_i)}{\sum_{i=1}^{N} k(x, x_i)} \tag{5.2}$$

In the limit of an infinite number of instances and a small kernel width, it is possible to get an arbitrarily close approximation to the function c_θ. When instances are labeled according to a p-concept, even if $c(x_i)$ is not available, in the limit of an infinite number of instances, the number of positive and negative instances drawn from point x will approach $c_\theta(x)$, and thus the average will approach the real value being approximated as well. For these reasons, I use the concept class of all Nadaraya-Watson kernel weighted average functions defined over θ. My implementation of the Nadaraya-Watson kernel-weighted average function using the RBF kernel requires time $O(n^2 d)$, which is simply $O(n^2)$ for the one-feature case. While methods exist to speed up the calculation time for AUC, the most basic calculation method requires time $t_{\mathcal{S}} = O(n \log n)$ [Bouckaert, 2006]. Thus the time complexity of MI-FEAR with

AUC is $O(d(n^2 + \log d))$, which is longer then the accuracy case.

5.2 Experimental Methodology

In order to evaluate the performance of feature selection, I used single-instance learning (SIL), which converts multiple-instance datasets to supervised datasets by giving instances the labels of their bags, and then learning a standard supervised classifier on the resulting dataset [Doran and Ray, 2014]. This is an instance-based learning method, which corresponds with MI-FEAR being an instance-based feature selection method. It has previously been shown that SIL performs well when used to learn high-AUC instance-labeling concepts [Doran and Ray, 2014], which implies that it can also learn high-AUC bag-labeling concepts, a result which has also been shown [Ray and Craven, 2005]. As a classifier, I used a support vector machine (see subsection 2.1.1), making use of the scikit-learn implementation [Pedregosa et al., 2011].

In order to perform parameter selection, I used 10-fold stratified cross-validation and 5-fold inner cross-validation, with a random search method over all parameters. Every fold is trained on 250 different parameter sets. I used the RBF kernel for the support vector machine, which requires a parameter γ, which I selected from the range $\gamma \in [10^{-6}, 10]$. I also used L_2-norm regularization, which requires a parameter C, which I selected from the range $C \in [10^{-3}, 10^5]$. On all datasets, I used a certain percentage of the original number of dimensions, rounded down to the nearest integer. This percentage was selected as a parameter from the range $\frac{d'}{d} \in [5, 95]$ for MI-FEAR. Because the feature construction methods combine the information from multiple features to create new feature spaces, they are expected to require less features to perform well; thus, for these methods I searched in the range $[5, 50]$. 250 parameter sets were used, rather than the more standard 125 for random search over three parameters, in order to provide more granularity in the number of dimensions selected.

I compared the performance of MI-FEAR with the MidLABS (subsection 2.6.1) and CLFDA (subsection 2.6.2) multiple-instance dimensionality reduction algorithms. MidLABS uses two parameters specific to the dimensionality reduction algorithm: the parameter ϵ as the cutoff for the ϵ-graphs and the parameter C to control the weight tradeoff between the contributions of nodes and edges to the distance between two bags. Following the example of the original authors, I set C to 1 for the experiments; for ϵ, I searched over a parameter $e.p. \in [5, 50]$, and set ϵ such that $e.p.\%$ of the instance pairs over all the bags had a pairwise Euclidean distance below ϵ. CLFDA has three parameters: The number of citers C, the number of references R, and the ratio threshold t. As the original authors of the algorithm suggest, I set $t = 1$. I set both C and R to 5, the upper end of the range of values suggested by the original authors. I implemented both of these algorithms in Python, and will make the code available shortly. I also wrote an implementation of the MIDR algorithm, along with the MILR classifier (subsection 2.6.3), but this algorithm proved to be prohibitively slow for any practical use and was not included in the experiments. In the implementation of MI-FEAR with accuracy, I used a parameter of 5 as the maximum depth for the decision tree, with information gain as the splitting criterion. For the RBF kernel in the implementation of MI-FEAR with AUC, I used a value of 100 for the kernel width parameter γ. I also compared MI-FEAR to principal components analysis (subsection 2.4.2) calculated over all the instances in all the bags, for which I used the scikit-learn implementation [Pedregosa et al., 2011], which is non-parametric.

The AUC metric was used to evaluate all classifiers. In addition, for each algorithm/dataset pair, I calculated the average amount of time required to perform dimensionality reduction to 10%, 25%, and 50% of the original features on the entire dataset, with $e.p.$ for MidLABS set to 20 and all other parameters fixed as in the experiments. In the case of MI-FEAR with AUC, the bulk of the time lay in

Figure 5.1: Two images from the "Checkered Scarf vs. Data Mining Book" SIVAL dataset (SIVAL 4). Both of these images contain the data mining book concept; however, the book is at a different angle, in a different environment, and under different lighting, make the task of recognizing the book more difficult.

the calculation of the kernel matrix, which can be calculated independently for each dataset, and used on every parameter set, resulting in a substantial savings in time across all parameter sets. The times presented in Table 5.3 for MI-FEAR with AUC are based on the use of a pre-calculated Gram matrix; when the time to calculate the Gram matrix is included, the algorithm takes on average 43 seconds.

In order to compare the different dimensionality reduction algorithms, I use a Friedman test on each pair of classifiers, with $\alpha = 0.001$, to reject the null hypothesis that the dimensionality reduction algorithms are equivalent. Provided that this hypothesis is rejected, I then use a Nemenyi test to determine if two classifiers are statistically significantly different based on their ranks over the ten datasets, with a significance level of $\alpha = 0.05$.

5.3 Datasets

I evaluated these methods on ten spatially independent, variable area, and lighting (SIVAL) datasets [Rahmani et al., 2005]. In these datasets, objects are photographed against a collection of different backgrounds. In each image, the angle, location, and

No.	Dataset	Bags	Instances	γ
1	Apple vs. Coke Can	120	3789	0.530
2	Blue Scrunge vs. Ajax Orange	120	3780	0.533
3	Cardboard Box vs. Candle with Holder	120	3791	0.570
4	Checkered Scarf vs. Data Mining Book	120	3811	0.679
5	Dirty Work Gloves vs. Dirty Running Shoe	120	3801	0.572
6	Fabric Softener Box vs. Glazed Wood Pot	120	3796	0.615
7	Julie's Pot vs. Rap Book	120	3793	0.579
8	Striped Notebook vs. Green Tea Box	119	3776	0.584
9	WD-40 Can vs. Large Spoon	120	3786	0.588
10	Wood Rolling Pin vs. Translucent Bowl	120	3778	0.535

Table 5.1: Information about SIVAL Datasets. For brevity, the indices of the left column will be used to refer to these datasets in subsequent tables. γ is estimated from the data available.

size of the object may vary considerably. The goal is to succeed on the content-based image retrieval (CBIR) task of learning which images match a particular category. Two example images from a SIVAL dataset are given in Figure 5.1.

Each image is represented using a bag, which is given a label of 1 if the bag contains the object in question and 0 otherwise. The image is segmented up into 31 or 32 instances according to the ACCIO! algorithm [Rahmani et al., 2005], which first describes each pixel using three color features and three texture features, then uses the Improved Hierarchical Segmentation algorithm [Zhang et al., 2005] over this representation to find segments, and finally represents each segment with six features representing the average of the three color and texture features over all pixels in the segment, and twenty-four features representing the same information for the closest segments in each cardinal direction. Each bag, then, contains 31-32 instances in $\mathbb{R}^{30}$.

The SIVAL dataset has been augmented so that each instance is given a label; while these instance labels are not available during the learning process, they can be used to evaluate the performance of the SIL algorithm [Settles et al., 2008]. The original SIVAL dataset contains 60 images for each object and twenty-five possible objects, for a total of 1500 images in the one-vs-all CBIR task. For efficiency, I used

Dataset	MI-FEAR w/Accuracy	MI-FEAR w/AUC	PCA	MidLABS	CLFDA	No reduction
SIVAL 1	0.637	0.614	0.630	0.717	0.815	0.758
SIVAL 2	0.618	0.613	0.633	0.814	0.830	0.676
SIVAL 3	0.764	0.714	0.788	0.614	0.794	0.647
SIVAL 4	0.959	0.946	0.930	0.932	0.951	0.954
SIVAL 5	0.570	0.652	0.551	0.574	0.699	0.619
SIVAL 6	0.881	0.890	0.823	0.944	0.900	0.895
SIVAL 7	0.854	0.880	0.859	0.936	0.911	0.868
SIVAL 8	0.844	0.860	0.780	0.807	0.818	0.882
SIVAL 9	0.944	0.962	0.947	0.933	0.963	0.965
SIVAL 10	0.704	0.682	0.664	0.681	0.723	0.566

Table 5.2: AUC measurements for the instance-level ranking task

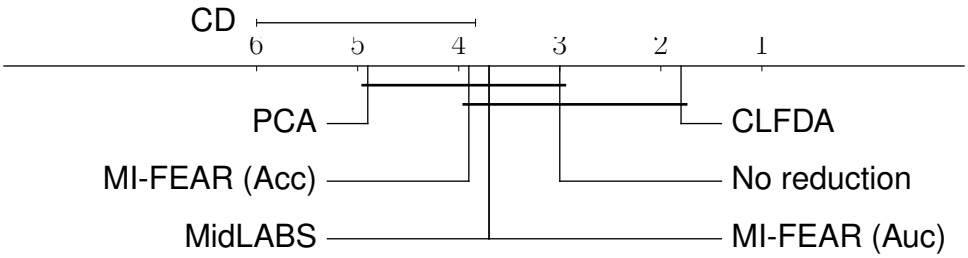


Figure 5.2: A critical-difference diagram of MI-FEAR, the three feature construction algorithms, and the case of no reduction, using AUC on the instance-labeling task. Two classifiers are statistically significantly different if the difference between their average ranks exceeds the critical difference measurement. The horizontal lines connecting algorithms indicate statistical indistinguishability.

a set of datasets where objects have been paired against each other for a one-vs-one task, so that each dataset contains 120 bags. For each dataset, I used the real instance labels, which I had available for these datasets, to estimate the parameter γ with the fraction of negative instances that appear in negative bags in the dataset. These estimates are shown in Table 5.1.

Dataset	MI-FEAR w/Accuracy	MI-FEAR w/AUC	PCA	MidLABS	CLFDA
SIVAL 1	0.599	4.297	0.0215	39.506	54.533
SIVAL 2	0.594	4.136	0.0212	43.000	67.453
SIVAL 3	0.604	4.196	0.0216	57.901	49.133
SIVAL 4	0.608	4.264	0.0214	41.122	70.459
SIVAL 5	0.597	4.256	0.0214	55.776	73.256
SIVAL 6	0.592	4.153	0.0212	58.920	59.850
SIVAL 7	0.591	4.195	0.0213	51.854	128.957
SIVAL 8	0.585	4.190	0.0213	49.989	63.798
SIVAL 9	0.588	4.126	0.0212	34.940	67.941
SIVAL 10	0.587	4.101	0.0211	56.026	70.665

Table 5.3: Time, in seconds, required for each dimensionality reduction to find a new feature space. The times required to find a feature space with 10, 25, and 50% of the original dimensions are averaged. Pre-computed kernel matrices are used in the kernel regression procedure for MI-FEAR with AUC.

5.4 Experimental Results

The results of these experiments, when AUC is measured on the instance-labeling task, are given in Table 5.2. The critical difference diagram of Figure 5.2 shows that both MI-FEAR results are statistically indistinguishable from both of the other dimensionality reduction algorithms for multiple-instance learning. All dimensionality reduction algorithms are statistically indistinguishable from the results when no reduction is performed. In terms of AUC, both variations of MI-FEAR are competitive with MidLABS: MI-FEAR with accuracy outperforms MidLABS on five datasets, while MI-FEAR with AUC outperforms MidLABS on six datasets. MI-FEAR is also a viable dimensionality reduction technique in that in spite of removing a considerable amount of information from the datasets, it can lead to better performance than when that information is still included. In three of the ten datasets, using MI-FEAR with accuracy results in a better performance than when no dimensionality is used at all, while this is true on four datasets for MI-FEAR with AUC.

While the MI-FEAR variations are also statistically indistinguishable from CLFDA,

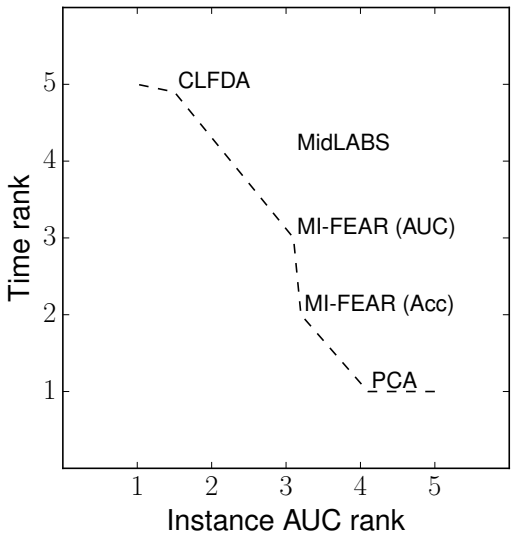


Figure 5.3: Pareto curve of time vs. AUC on the instance-labeling task

CLFDA typically does slightly better, outperforming MI-FEAR with accuracy on eight datasets and MI-FEAR with AUC on nine. Indeed, CLFDA is the highest-performing dimensionality reduction algorithm on six of the ten datasets. The problem with CLFDA, however, is the amount of time required for it to find a good representation. The times, in seconds, that each dimensionality reduction algorithm takes to find a new feature space, calculated as described in section 5.2, are shown in Table 5.3. When ranked by average time taken to compute the representation, the algorithms are in exactly the reverse order of their performance in terms of AUC, with the exception of MidLABS, which has an average ranking of its performance equal to that of MI-FEAR with AUC, but takes longer. This is illustrated in Figure 5.3, where all algorithms except MidLABS are on the Pareto frontier when time and performance are taken into account.

Table 5.4 shows what size the different algorithms choose to reduce the dataset to. MI-FEAR tends to create a larger dataset than the other algorithms. Intuitively, this is because MI-FEAR is a feature selection algorithm, while the others are feature construction algorithms and can capture more information in fewer features. While feature selection techniques are cutting out features, all three feature construction

techniques create a new dataset with information from all of the original features.

There are a number of factors that contribute to the performance of MI-FEAR relative to other dimensionality reduction algorithms, including the value of γ, the degree of correlation between features, the structure of the target concept, and whether the conditions for other dimensionality reduction algorithms to perform well are satisfied. Table 5.5 focuses specifically on MI-FEAR's ability to rank features, independent from other considerations that play a role in the performance of the final classifier. While the limited dataset available makes calculating the correct ranking impossible, the correct ranking can be approximated by providing the true instance labels to the same MI-FEAR algorithm, replacing the noisy instance labels. The ranks and AUC values of each of the thirty features in this approximation of the correct ranking are also shown. I examined the ranking for SIVAL 1, which has the lowest estimated γ value of all the datasets ($\gamma \approx 0.530$), and SIVAL 4, which has by far the highest γ value ($\gamma \approx 0.679$).

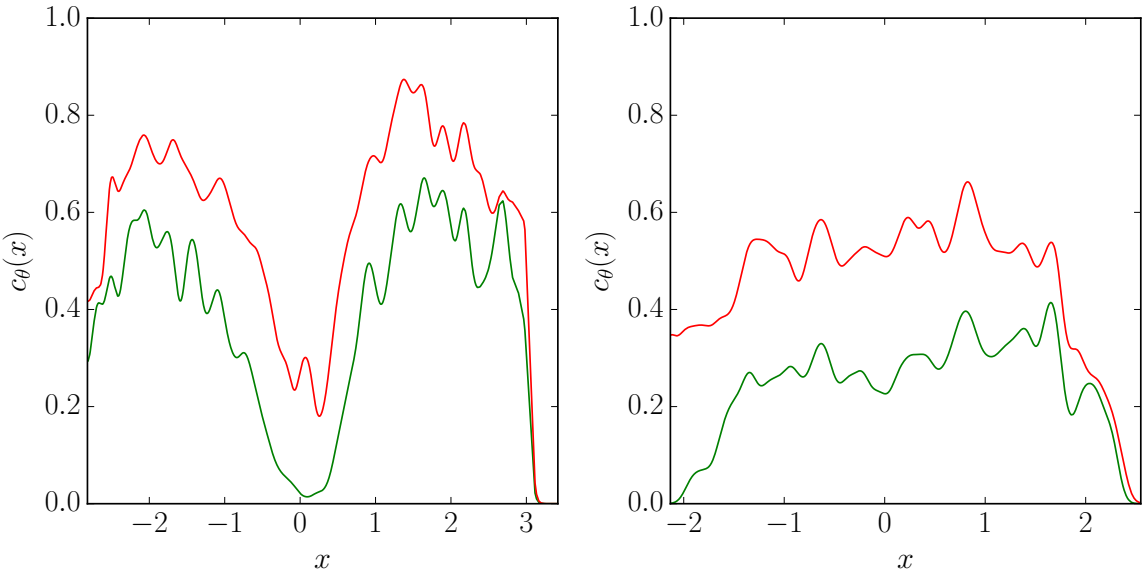
This table shows the degree to which γ has an effect on MI-FEAR's ability to find a good ranking. The ranking with a low value of γ has just as many of both sets of features in its top ten selected features as in the bottom, which indicates that it has not done a very good job of separating useful and useless features. The ranking with a high value of γ, on the other hand, not only agrees with the correct ranking for 9 of the top 10 best features, but also correctly identifies the ten worst features.

Dataset	MI-FEAR w/Accuracy	MI-FEAR w/AUC	PCA	MidLABS	CLFDA
SIVAL 1	60.2	68.3	34.4	11.7	35.4
SIVAL 2	53.9	66.2	43.0	36.3	48.3
SIVAL 3	88.1	46.4	47.0	19.0	33.9
SIVAL 4	64.9	63.9	43.8	25.9	33.1
SIVAL 5	56.1	77.0	34.8	20.4	44.4
SIVAL 6	87.2	81.8	36.2	30.8	40.8
SIVAL 7	65.8	91.0	48.2	19.4	41.8
SIVAL 8	63.5	39.4	42.9	37.0	48.0
SIVAL 9	54.4	61.5	26.9	35.5	29.6
SIVAL 10	84.5	77.7	40.6	28.1	41.2

Table 5.4: Median percentage of features selected or constructed in the optimal parameter sets for the ten outer folds

SIVAL 1				SIVAL 4			
MI-FEAR		Correct		MI-FEAR		Correct	
Index	AUC	Index	AUC	Index	AUC	Index	AUC
16	0.659	**4**	0.803	**12**	0.749	**4**	0.937
4	0.658	**3**	0.801	**3**	0.745	**5**	0.930
7	0.657	**13**	0.790	**4**	0.744	**3**	0.929
26	0.656	**5**	0.786	**5**	0.744	**8**	0.829
14	0.656	**7**	0.784	**8**	0.740	**14**	0.826
8	0.655	**19**	0.783	**14**	0.740	**7**	0.825
25	0.655	**25**	0.774	**6**	0.737	**12**	0.822
17	0.653	**2**	0.761	**20**	0.737	**26**	0.822
22	0.653	**0**	0.752	18	0.737	**6**	0.822
13	0.652	**18**	0.725	**7**	0.737	**20**	0.818
20	0.652	23	0.718	13	0.733	18	0.817
28	0.649	24	0.712	19	0.732	13	0.815
23	0.648	20	0.704	**26**	0.729	19	0.815
19	0.645	29	0.703	24	0.727	25	0.812
10	0.643	26	0.693	25	0.717	24	0.810
29	0.639	14	0.692	11	0.672	23	0.788
15	0.635	16	0.690	23	0.667	11	0.784
21	0.633	11	0.689	17	0.663	29	0.778
3	0.633	6	0.684	10	0.663	17	0.773
11	0.632	9	0.681	29	0.661	10	0.773
5	0.628	**17**	0.680	**16**	0.652	**22**	0.768
27	0.627	**22**	0.677	**22**	0.651	**16**	0.757
9	0.626	**8**	0.674	**28**	0.644	**28**	0.753
12	0.603	**1**	0.673	**2**	0.639	**9**	0.732
24	0.597	**12**	0.673	**9**	0.638	**27**	0.730
6	0.596	**28**	0.671	**27**	0.636	**15**	0.725
18	0.594	**10**	0.668	**15**	0.634	**21**	0.725
0	0.579	**21**	0.663	**21**	0.629	**2**	0.703
1	0.570	**15**	0.653	**0**	0.609	**1**	0.646
2	0.568	**27**	0.649	**1**	0.591	**0**	0.614

Table 5.5: Features selected by MI-FEAR with AUC on SIVAL 1, which has $\gamma \approx 0.530$, and SIVAL 4, which has $\gamma \approx 0.679$. In each feature ranking, the top ten features according to the correct ranking are marked in green, and the bottom ten in red.

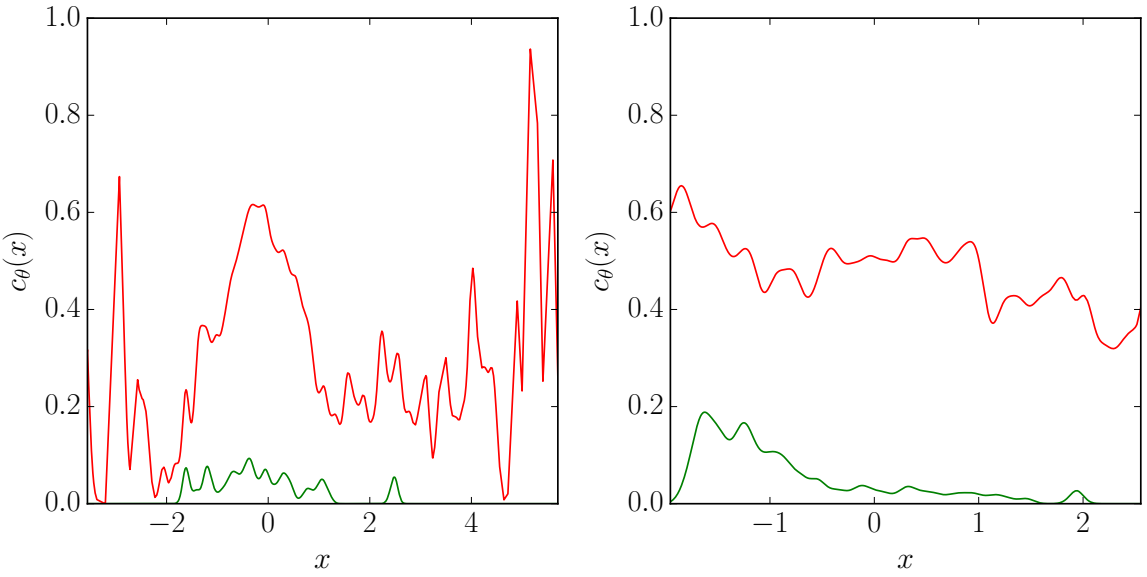


(a) Feature 12 on the SIVAL 4 dataset

(b) Feature 0 on the SIVAL 4 dataset

Figure 5.4: $c_\theta(x)$ (green) and $c^\gamma_\theta(x)$ (red) approximated, using the same process as used for the MI-FEAR with AUC experiments, for two features on the SIVAL 4 dataset.

In both of these rankings, the AUC values for the features in the learned ranking are well below their corresponding values in the correct ranking. This is likely due to the fact that not only is γ fairly low on both datasets, but the two datasets also feature a significantly higher rate of negative instances than positive instances, meaning that a considerable number of instance pairs will fall into the region of the contour plots in Figure 4.4 that has the highest negative contribution to $\Delta^\gamma_{Auc}(\theta)$. The estimations of c_θ and c^γ_θ output by Nadaraya-Watson kernel regression estimation for feature 12 of SIVAL 4, which is ranked the highest by MI-FEAR, and feature 0, which not only is ranked the lowest in the correct ranking but is also the feature for which MI-FEAR's estimation of its characteristic AUC differs the least from the its true characteristic AUC. The estimation of the correct $c_\theta(x)$ curve for feature 12 reveals a large range of values for c_θ, which means that a lot of points will fall far from the diagonal of Figure 4.1, thus giving the feature a high true characteristic AUC. The considerable number of points with values of c_θ close to 0 explains the large difference



(a) Feature 16 on the SIVAL 0 dataset

(b) Feature 2 on the SIVAL 0 dataset

Figure 5.5: $c_\theta(x)$ (green) and $c_\theta^\gamma(x)$ (red) approximated, using the same process as used for the MI-FEAR with AUC experiments, for two features on the SIVAL 0 dataset.

in AUC estimates, as point pairs containing one of these points will contribute highly to $\Delta_{Auc}^\gamma(\theta)$. The plot reveals that c_θ^γ is able to match the curves of c_θ fairly well. On feature 0, c_θ^γ does an excellent job of matching c_θ's pattern; furthermore, note that while feature 12 has many c_θ values close to 0, feature 0 has a lot of values close to 0.3, and is also very flat, meaning that, according to Figure 4.4, most of the point pairs will contribute very little to $\Delta_{Auc}^\gamma(\theta)$.

Figure 5.5, similarly, plots c_θ and c_θ^γ for feature 16 of SIVAL 1, which is the highest ranked by MI-FEAR, and feature 2, which, while having one of the highest true characteristic accuracies, is ranked last by MI-FEAR. This dataset clearly has much more negativity, which causes lots of values of c_θ to be close to 0, leading to a lot of point pairs that have high contributions to $\Delta_{Auc}^\gamma(\theta)$. Coupled with the low value of γ, which causes c_θ^γ to be much further above c_θ for SIVAL 0 than for SIVAL 4, it is clear why the AUC estimates are so poor. This can be seen most dramatically in feature 2, where the flatness of the c_θ curve, coupled with values close to 0, means that

Dataset	MI-FEAR w/Accuracy	MI-FEAR w/AUC	PCA	MidLABS	CLFDA	No reduction
SIVAL 1	0.656	0.558	0.575	0.556	0.689	0.626
SIVAL 2	0.810	0.760	0.639	0.639	0.852	0.785
SIVAL 3	0.783	0.736	0.861	0.621	0.815	0.657
SIVAL 4	0.994	0.916	0.906	0.815	0.949	0.985
SIVAL 5	0.652	0.819	0.577	0.561	0.753	0.648
SIVAL 6	0.744	0.798	0.731	0.845	0.873	0.792
SIVAL 7	0.788	0.943	0.873	0.906	0.907	0.874
SIVAL 8	0.924	0.893	0.901	0.620	0.838	0.819
SIVAL 9	0.962	0.975	0.987	0.910	0.991	0.981
SIVAL 10	0.756	0.795	0.672	0.596	0.740	0.601

Table 5.6: AUC measurements for the bag-level ranking task

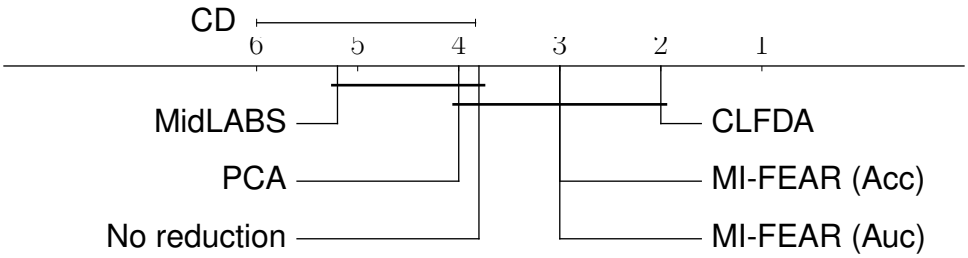


Figure 5.6: A critical-difference diagram of MI-FEAR, the three feature construction algorithms, and the case of no reduction, using AUC on the bag-labeling task.

it has a fairly high AUC score; c_θ^γ is able to match that flatness, but unfortunately, the noise lifts it so much that it ends up having lots of point pairs close to (0.5, 0.5), which are the point pairs that, as Figure 4.1 reveals, have the lowest AUC scores! This feature, then, represents a very difficult scenario for MI-FEAR with AUC, in that it would be tough to design a c_θ curve that results in a higher difference between the perceived and true characteristic AUCs. These charts and their accompanying differences in AUC confirm the theoretical analysis of chapter 4.

While Table 5.2 showed results for the instance-labeling task, where the goal is to determine the correct labels for each individual instance as in a supervised learning setting, Table 5.6 shows results for the bag-labeling task, where the goal is to determine the correct labels for the bags. As described in section 5.2, the

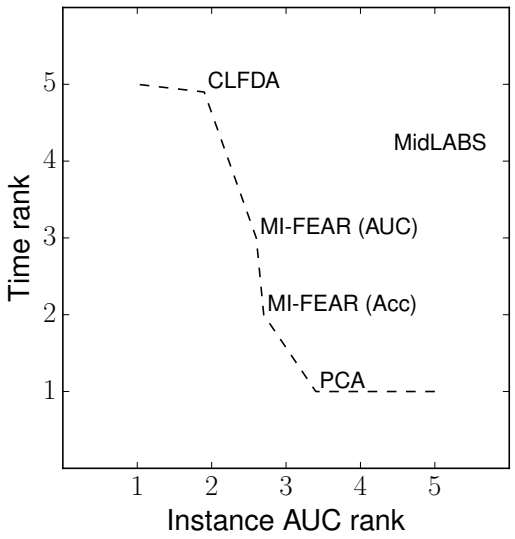


Figure 5.7: Pareto curve of time vs. AUC on the bag-labeling task

learning algorithm used to find a bag-level classifier is SIL. Both variations of MI-FEAR perform very well here; using MI-FEAR with accuracy rather than the original dataset resulted in an improvement on seven of the ten datasets, while the same was true on five of the ten datasets for MI-FEAR with AUC. While CLFDA was again the best, it was outperformed by both MI-FEAR variations on three datasets each, and, as in the instance-level case, is statistically indistinguishable from both variations.

Perhaps surprising here is the poor performance of MidLABS, which is the only multiple-instance dimensionality reduction algorithm that operates at the bag level. MidLABS is in fact statistically inferior to CLFDA and both MI-FEAR variations. This poor performance may be the result of MidLABS' biases when constructing a new feature space being inconsistent with either the nature of the SIVAL datasets or the assumptions made by the SIL algorithm. As in the instance-labeling task, MidLABS is again the only algorithm not on the Pareto curve, as shown in Figure 5.7.

Chapter 6

Conclusion

In this work I have presented a framework for studying feature selection in the context of multiple-instance learning. To do this, I have used probabilistic concepts to describe the labeling function on an individual feature. I have shown the effect that the use of bag-labeled instances has on a probabilistic concept, and analyzed the ways in which the resulting one-sided noise affects the accuracy and AUC of deterministic and real-valued concepts, respectively. I have presented the MI-FEAR approach to feature selection, which ranks features by the perceived value of a scoring function measuring a learned hypothesis on each feature. Using the aforementioned framework and analysis, I have demonstrated that it is theoretically possible to learn an optimal feature ranking using MI-FEAR in certain scenarios, and described the factors that control when this is possible and, when it is not possible, when the resulting feature ranking may still be good. The experimental results show that MI-FEAR is competitive with other dimensionality reduction algorithms while also being much faster than those algorithms, and can find a feature set that leads to better performance over the original set of features.

One aspect of the theoretical results is that some of the parameters used in the complexity bounds of Theorem 1 and Theorem 2 are a function of c_θ, which is un-

known in practice. Therefore, an important direction for future work is to prove corollaries of these results where specific assumptions are made about the nature of c_θ, so that concrete sample complexity bounds can be found. In the current case, domain knowledge must be used to determine whether or not MI-FEAR is likely to perform well.

Much of the analysis in this thesis is based on the idea that the p-concepts resulting from the use of bag-labeled instances will be treated as though they are ordinary concepts, with the learner ignoring the fact that there is one-sided noise. There exist theoretical methods to place an upper bound on noise [Kearns and Vazirani, 1994]. These methods could be integrated into the MI-FEAR analysis by allowing for some knowledge about gamma to be used to attempt to estimate $c_\theta(x)$ rather than using $c_\theta^\gamma(x)$. Were it possible to make a reasonable guess as to what $c_\theta(x)$ may be, then it would also be possible to infer f_θ^*, which could be directly approximate, thus removing the Δ^γ factor in the upper bound on the characteristic accuracy difference for a feature. Similarly, while the minimum one-sided disagreement strategy is not, in its current theoretical state, capable of being used to learn a noisy concept from examples that are not deterministically labeled, it is possible that this strategy could be revised to work in this context, thus also allowing for f_θ^* to be directly approximated. It should be noted, however, that both of these strategies will likely maintain the assumption that γ is constant; a relaxation of this assumption is left for future work.

In addition to attempting to directly approximate f_θ^*, there are two other opportunities for improvement of these theoretical results. The first comes from the fact that the amount of positivity in the instance space will be consistent across all features; that is, simply selecting a different feature will not change the ratio of positive instances over the feature axis, only the distribution of that positivity on the feature axis. Mathematically, this means that the integrated value of the p-concept with respect to the marginal instance distribution for each feature should be the same, which

may be used to eliminate many worst-case scenarios. The second is inspired by the empirical results. As shown in Table 5.5, in practice a feature ranking may hold even if the characteristic perceived scores are nowhere near the true characteristic scores. Additionally, a ranking may still be considered good if the features are in approximately the correct order, even if the order is not exactly correct. For these reasons, it may be possible to loosen the requirement in both theorems that the difference in evaluation metrics for a feature must be less than $\frac{\tau}{2}$ to instead require that the rankings are sufficiently similar, as measured by one of a number of statistical tests that exist for this purpose.

Finally, MI-FEAR is a very simple feature selection algorithm; while this is useful for the purposes of analysis, there are a number of options for further exploration in the field of feature selection for multiple-instance learning. The first is that, while feature ranking is very fast, its assumption of feature independence may not hold, and cases where signal is revealed only by the interaction of multiple features, such as when the concept is an XOR function, will lead to poor results. A sequential feature selection algorithm, which adds features one at a time, may prove more capable in these situations. The second is that while bag-labeled instances may, as I have shown, be a surprisingly valid method for assigning instance labels in the multiple-instance learning problem, a more advanced label-assigning methodology, such as that used by CLFDA, may lead to better performance. Lastly, MI-FEAR is designed specifically for the instance-level classification task when the standard MI assumption is used. For the bag-level learning task, an algorithm specifically engineered to consider instances as part of a bag rather than considering the independently may achieve stronger results. For cases of multiple-instance learning where the standard MI assumption does not hold, an algorithm that is designed to match the assumptions of those cases will likely perform better as well.

In conclusion, I believe that this work enhances the current understanding of

both multiple-instance learning and feature selection. In this thesis I have shown that, when used in the context of the feature selection problem, the performance of the bag-labeled instances strategy can be determined by the value of γ and the structure of the labeling concept relative to the instance distribution. I also describe the MI-FEAR algorithm, the quality of which I demonstrate both theoretically and empirically with two different scoring functions. In addition to these results, I believe that the idea of analyzing feature selection by measuring the scoring function over a p-concept defined based on the projection of the labeling concept provides a valuable perspective from which to analyze future approaches. In addition to the results and framework provided in this thesis, the opportunities for future research described in this conclusion provide several avenues for further development of the academic understanding of multiple-instance learning.

Bibliography

[Andrews et al., 2003] Andrews, S., Tsochantaridis, I., and Hofmann, T. (2003). Support Vector Machines for Multiple-instance Learning. In *Advances in Neural Information Processing Systems 15*, pages 561–568. MIT Press.

[Ascher et al., 2001] Ascher, D., Dubois, P. F., Hinsen, K., Hugunin, J., and Oliphant, T. (2001). *Numerical Python*. Lawrence Livermore National Laboratory, Livermore, CA. `http://numpy.scipy.org/`.

[Auer et al., 1997] Auer, P., Long, P. M., and Srinivasan, A. (1997). Approximating Hyper-rectangles: Learning and Pseudo-random Sets. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 314–323, New York, NY, USA. ACM.

[Bi et al., 2003] Bi, J., Bennett, K., Embrechts, M., Breneman, C., and Song, M. (2003). Dimensionality Reduction via Sparse Support Vector Machines. *J. Mach. Learn. Res.*, 3:1229–1243.

[Blum and Kalai, 1998] Blum, A. and Kalai, A. (1998). A Note on Learning from Multiple-Instance Examples. *Mach. Learn.*, 30(1):23–29.

[Blumer et al., 1989] Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis Dimension. *J. ACM*, 36(4):929–965.

[Botvinnik, 1983] Botvinnik, M. M. (1983). *Solving Inexact Search Problems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Bouckaert, 2006] Bouckaert, R. R. (2006). Efficient AUC Learning Curve Calculation. In *Proceedings of the 19th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, AI'06, pages 181–191, Berlin, Heidelberg. Springer-Verlag.

[Briggs et al., 2012] Briggs, F., Fern, X. Z., and Raich, R. (2012). Rank-loss Support Instance Machines for MIML Instance Annotation. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 534–542, New York, NY, USA. ACM.

[Brown et al., 2012] Brown, G., Pocock, A., Zhao, M.-J., and Luján, M. (2012). Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection. *J. Mach. Learn. Res.*, 13:27–66.

[Campbell et al., 2002] Campbell, M., Hoane, Jr., A. J., and Hsu, F.-h. (2002). Deep Blue. *Artif. Intell.*, 134(1-2):57–83.

[Chai et al., 2014] Chai, J., Ding, X., Chen, H., and Li, T. (2014). Multiple-instance Discriminant Analysis. *Pattern Recognition*, 47(7):2517–2531.

[De Raedt, 1998] De Raedt, L. (1998). Attribute-Value Learning Versus Inductive Logic Programming: The Missing Links (Extended Abstract). In *Proceedings of the 8th International Workshop on Inductive Logic Programming*, ILP '98, pages 1–8, London, UK, UK. Springer-Verlag.

[Dietterich et al., 1997] Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. (1997). Solving the Multiple Instance Problem with Axis-parallel Rectangles. *Artif. Intell.*, 89(1-2):31–71.

[Diochnos et al., 2012] Diochnos, D. I., Sloan, R. H., and Turán, G. (2012). On Multiple-instance Learning of Halfspaces. *Inf. Process. Lett.*, 112(23):933–936.

[Doran, 2015] Doran, G. (2015). *Multiple-Instance Learning From Distributions*. PhD thesis, Case Western Reserve University.

[Doran and Ray, 2014] Doran, G. and Ray, S. (2014). Learning Instance Concepts from Multiple-Instance Data with Bags as Distributions. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[Fawcett, 2006] Fawcett, T. (2006). An Introduction to ROC Analysis. *Pattern Recogn. Lett.*, 27(8):861–874.

[Forman, 2003] Forman, G. (2003). An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *J. Mach. Learn. Res.*, 3:1289–1305.

[Foulds and Frank, 2010] Foulds, J. and Frank, E. (2010). A Review of Multi-Instance Learning Assumptions.

[Friedman et al., 2001] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The Elements of Statistical Learning*, volume 1. Springer series in statistics Springer, Berlin.

[Gartner et al., 2002] Gartner, T., Flach, P. A., Kowalczyk, A., and Smola, A. J. (2002). Multi-Instance Kernels. In *In Proc. 19th International Conf. on Machine Learning*, pages 179–186. Morgan Kaufmann.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.*, 3:1157–1182.

[Hand and Till, 2001] Hand, D. J. and Till, R. J. (2001). A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. *Mach. Learn.*, 45(2):171–186.

[Hanley and McNeil, 1982] Hanley, J. A. and McNeil, B. J. (1982). The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143(1):29–36.

[Haussler, 1992] Haussler, D. (1992). Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications. *Inf. Comput.*, 100(1):78–150.

[Jones et al., 01] Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open Source Scientific Tools for Python. [Online; accessed 2015-08-04].

[Kearns and Schapire, 1993] Kearns, M. J. and Schapire, R. E. (1993). Efficient Distribution-free Learning of Probabilistic Concepts. In *Journal of Computer and System Sciences*, pages 382–391. IEEE Computer Society Press.

[Kearns and Vazirani, 1994] Kearns, M. J. and Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, USA.

[Kim and Choi, 2010] Kim, S. and Choi, S. (2010). Local Dimensionality Reduction for Multiple Instance Learning. In *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop on*, pages 13–18. IEEE.

[Kohavi and John, 1997] Kohavi, R. and John, G. H. (1997). Wrappers for Feature Subset Selection. *Artif. Intell.*, 97(1-2):273–324.

[Kundakcioglu et al., 2010] Kundakcioglu, O. E., Seref, O., and Pardalos, P. M. (2010). Multiple Instance Learning via Margin Maximization. *Appl. Numer. Math.*, 60(4):358–369.

[Leonard et al., 2008] Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., Koch, O., Kuwata, Y., Moore, D., Olson, E., Peters, S., Teo, J., Truax, R., Walter, M., Barrett, D.,

Epstein, A., Maheloni, K., Moyer, K., Jones, T., Buckley, R., Antone, M., Galejs, R., Krishnamurthy, S., and Williams, J. (2008). A Perception-driven Autonomous Urban Vehicle. *J. Field Robot.*, 25(10):727–774.

[Maron and Ratan, 1998] Maron, O. and Ratan, A. L. (1998). Multiple-Instance Learning for Natural Scene Classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 341–349, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Mohri et al., 2012] Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. The MIT Press.

[Mossel et al., 2003] Mossel, E., O'Donnell, R., and Servedio, R. P. (2003). Learning Juntas. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 206–212, New York, NY, USA. ACM.

[Murphy, 2012] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830.

[Ping et al., 2010] Ping, W., Xu, Y., Ren, K., Chi, C.-H., and Shen, F. (2010). Non-iid Multi-instance Dimensionality Reduction by Learning a Maximum Bag Margin Subspace. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

[Rahmani et al., 2005] Rahmani, R., Goldman, S. A., Zhang, H., Krettek, J., and Fritts, J. E. (2005). Localized Content Based Image Retrieval. In *Proceedings of the*

7th ACM SIGMM International Workshop on Multimedia Information Retrieval, MIR '05, pages 227–236, New York, NY, USA. ACM.

[Ray and Craven, 2005] Ray, S. and Craven, M. (2005). Supervised Versus Multiple Instance Learning: An Empirical Comparison. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 697–704, New York, NY, USA. ACM.

[Sabato and Tishby, 2012] Sabato, S. and Tishby, N. (2012). Multi-instance Learning with Any Hypothesis Class. *J. Mach. Learn. Res.*, 13(1):2999–3039.

[Scott et al., 2003] Scott, S., Zhang, J., and Brown, J. (2003). On Generalized Multiple-Instance Learning. Technical report, International Journal of Computational Intelligence and Applications.

[Settles et al., 2008] Settles, B., Craven, M., and Ray, S. (2008). Multiple-instance Active Learning. In *Advances in neural information processing systems*, pages 1289–1296.

[Shlens, 2014] Shlens, J. (2014). A Tutorial on Principal Component Analysis. *CoRR*, abs/1404.1100.

[Simon, 2014] Simon, H. U. (2014). PAC-learning in the Presence of One-sided Classification Noise. *Annals of Mathematics and Artificial Intelligence*, 71(4):283–300.

[Sugiyama, 2007] Sugiyama, M. (2007). Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis. *J. Mach. Learn. Res.*, 8:1027–1061.

[Sun et al., 2010] Sun, Y.-Y., Ng, M. K., and Zhou, Z.-H. (2010). Multi-Instance Dimensionality Reduction. In *AAAI 2010*.

[Tao et al., 2004] Tao, Q., Scott, S., Vinodchandran, N. V., and Osugi, T. T. (2004). SVM-based Generalized Multiple-instance Learning via Approximate Box Counting. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 101–, New York, NY, USA. ACM.

[Tenenbaum et al., 2000] Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323.

[Tsuda and Scholkopf, 2004] Tsuda, K. and Scholkopf, B. (2004). A Primer on Kernel Methods. In *in Kernel Methods in Computational*, pages 35–70. MIT Press.

[Valiant, 1984] Valiant, L. G. (1984). A Theory of the Learnable. *Commun. ACM*, 27(11):1134–1142.

[Vapnik, 1982] Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Vapnik, 1992] Vapnik, V. (1992). Principles of Risk Minimization for Learning Theory. In *Advances in neural information processing systems*, pages 831–838.

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer.

[Vapnik, 1999] Vapnik, V. N. (1999). An Overview of Statistical Learning Theory. *Trans. Neur. Netw.*, 10(5):988–999.

[Vapnik and Chervonenkis, 1971] Vapnik, V. N. and Chervonenkis, A. Y. (1971). On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability & Its Applications*, 16(2):264–280.

[Xing et al., 2001] Xing, E. P., Jordan, M. I., and Karp, R. M. (2001). Feature Selection for High-dimensional Genomic Microarray Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 601–608, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Zhang et al., 2005] Zhang, H., Fritts, J., and Goldman, S. (2005). An Improved Fine-grain Hierarchical Method of Image Segmentation. *Washington University CSE Technical Report*.

[Zhou et al., 2009] Zhou, Z.-H., Sun, Y.-Y., and Li, Y.-F. (2009). Multi-instance Learning by Treating Instances As non-I.I.D. Samples. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1249–1256, New York, NY, USA. ACM.