Learning from Ambiguity

by

Oded Maron

Sc.B., Brown University (1992)
M.S., Massachusetts Institute of Technology (1994)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1998

Author ....
Department of Electrical Engineering and Computer Science
May 20, 1998

Certified by ....
Tomás Lozano-Pérez
Cecil H. Green Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Learning from Ambiguity

by

Oded Maron

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 1998, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

There are many learning problems for which the examples given by the teacher are ambiguously labeled. In this thesis, we will examine one framework of learning from ambiguous examples known as Multiple-Instance learning. Each example is a bag, consisting of any number of instances. A bag is labeled negative if all instances in it are negative. A bag is labeled positive if at least one instance in it is positive. Because the instances themselves are not labeled, each positive bag is an ambiguous example. We would like to learn a concept which will correctly classify unseen bags.

We have developed a measure called Diverse Density and algorithms for learning from multiple-instance examples. We have applied these techniques to problems in drug design, stock prediction, and image database retrieval. These serve as examples of how to translate the ambiguity in the application domain into bags, as well as successful examples of applying Diverse Density techniques.

Thesis Supervisor: Tomás Lozano-Pérez
Title: Cecil H. Green Professor of Computer Science and Engineering

Learning from Ambiguity

by

Oded Maron

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 1998, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

There are many learning problems for which the examples given by the teacher are ambiguously labeled. In this thesis, we will examine one framework of learning from ambiguous examples known as Multiple-Instance learning. Each example is a bag, consisting of any number of instances. A bag is labeled negative if all instances in it are negative. A bag is labeled positive if at least one instance in it is positive. Because the instances themselves are not labeled, each positive bag is an ambiguous example. We would like to learn a concept which will correctly classify unseen bags.

We have developed a measure called Diverse Density and algorithms for learning from multiple-instance examples. We have applied these techniques to problems in drug design, stock prediction, and image database retrieval. These serve as examples of how to translate the ambiguity in the application domain into bags, as well as successful examples of applying Diverse Density techniques.

Thesis Supervisor: Tomás Lozano-Pérez
Title: Cecil H. Green Professor of Computer Science and Engineering

Acknowledgments

A year ago, I was ready to submit a paper to NIPS. It contained the basic idea of Diverse Density, but it had only been tested on small or artificial problems where the relevant features were known. My advisor, Tomás Lozano-Pérez, said that to make it a good paper we should really implement an algorithm that performed feature scaling and test it on the MUSK dataset. I had been planning to do that eventually, but it was two days before the submission deadline and I was coming down with a cold. Reluctantly, I agreed. We hacked through that night and the following day. Amazingly, the code worked without much tweaking, and the paper got out just in time. The rest of this thesis flowed easily after that. I tell this story not because Tomás is a slave driver — quite the contrary. It was the one time when I needed a push and I got it. During the rest of my graduate career, Tomás gave me the freedom to explore a variety of topics, and the support when I floundered. In other words, he is everything you could ask for in an advisor.

The other members of my committee, Paul Viola and Peter Dayan, were both mentors and collaborators. They provided a wealth of knowledge and inspiration, and focused my energies on making the thesis more rigorous, more readable, and more practical.

Large sections of the thesis are the results of collaboration with a number of people. I enjoyed hours of useful feedback from Charles Isbell. Our discussions paid off in the development of `PWDD`. Aparna Lakshmi Ratan provided the perfect Machine Vision foil to the learning techniques developed here. Our collaboration is detailed in Chapter 6, and I feel that it brought out the best in both of our work.

Throughout most of my graduate career, I consulted for Grantham, Mayo, Van Otterloo & Co. They have been extremely generous in allowing me to experiment with various techniques on their financial data. Chris Darnell and Tom Hancock have supported me with ideas and insights into the world of computational finance.

I tried to attend as many group meetings as I could during my time at the AI Lab, and they have helped to shape my research and refine my critical eye: Tommy

Poggio's learning meetings, Patrick Winston's AI group meeting, Ron Rivest's (and then Anselm Blumer's) Machine Learning reading group, and Paul Viola's Learning and Vision group. I am also grateful to Ed Wang and Lisa Tucker-Kellogg for putting up with my lack of knowledge of biochemistry during our group meetings.

There are many people who gave me advice, a reference, or a sympathetic ear. They shaped the curves of my research trajectory. Eric Grimson helped with the image retrieval application. Pam Lipson shared code and expertise of the problems of natural scenes. Carl de Marcken helped me turn initial intuitions into a working algorithm. Dana Ron provided timely references and advice. Mike Kearns got me to think of bags in different ways. Tom Dietterich was generous with advice and code. I had many constructive discussions with Peter Auer, who helped me understand his algorithm and generously provided me with his code. Discussions with Haym Hirsh helped me put my work in context.

Finally, Holly Yanco was with me through this process, both when it was an adventure and when it was an ordeal. Her contributions are on every page.

This research was supported by the AFOSR ASSERT program, Parent Grant F49620-93-1-0263, and also by ONR's "A Trainable Modular Vision System," grant N00014-95-1-0600.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The field of Machine Learning is concerned with methods that use experience to improve performance. There are a number of reasons why one would want a computer to learn a task instead of programming the task into the computer. One reason is that many problems are simply too complex to program; predicting the behavior of the stock market, determining the three-dimensional structure of a protein from its amino acid sequence, and forecasting the weather are some examples of systems that are not understood well enough to program as an algorithm. Another reason is that some systems need to be adaptable to changing environments that could not have been foreseen by the programmer. For example, a robot that wanders around an office environment might need to learn on-line a new map if a doorway is blocked or a new partition is erected. Finally, one key component of natural intelligence is the ability to learn. If we can build artificial learners, then perhaps we can develop insights about how minds work.

Machine Learning algorithms receive examples from a teacher or from the environment and attempt to learn some structure that will generalize for unseen examples. The majority of work in Machine Learning falls into three learning frameworks: supervised, unsupervised, and reinforcement.

In *supervised learning*, the algorithm attempts to learn a concept from labeled examples that predicts the labels of the training examples correctly and generalizes to produce correct labels on examples outside of the training set. The label can

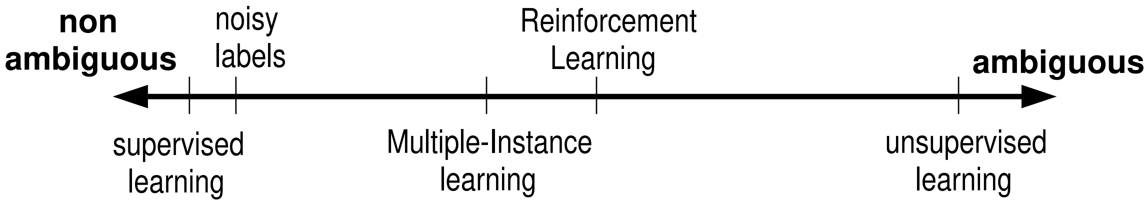


Figure 1-1: An ambiguity spectrum, with learning frameworks of increasingly ambiguous training examples.

be a class, in which case the learning task is called *classification*, or a continuous signal, in which case the task is called *regression.* Some concept representations used to learn from labeled examples include decision trees [Quinlan, 1992], nearest neighbor [Dasarathy, 1991], neural networks [Rumelhart *et al.*, 1986], and Bayesian networks [Pearl, 1988].

In *unsupervised learning*, the examples are not labeled. The algorithm attempts to learn the structure of the underlying source of the examples. Some instances of unsupervised learning are clustering [Cheeseman *et al.*, 1988], Principal Component Analysis [Chatfield and Collins, 1980], and Independent Component Analysis [Bell and Sejnowski, 1995, Comon, 1994].

In *Reinforcement Learning* the goal is to learn a policy, which is a mapping from states to actions. Examples are not labeled with the correct action; instead an occasional reinforcement signal which denotes the utility of some state is received. Reinforcement Learning methods are reviewed in [Sutton and Barto, 1998] and [Kaelbling *et al.*, 1996].

One way to relate these three frameworks is to look at the ambiguity of the training examples by determining how much information the label of an example conveys. In supervised learning, every example is perfectly labeled, so there is no ambiguity. In unsupervised learning, no example is labeled with respect to the desired output, so there is much ambiguity. Figure 1-1 shows a rough picture of the ambiguity spectrum, mapping supervised learning, learning with noisy labels (discussed in Chapter 7), Reinforcement Learning, Multiple-Instance learning, and unsupervised learning. In fact, there are many learning problems in which the examples are neither perfectly labeled nor completely unlabeled. We call these situations *learning from ambiguous* *examples.*

In this thesis, we concentrate on a relatively new framework of learning from ambiguity called *Multiple-Instance learning.* In this framework, each positive example is made up of a collection of instances, at least one of which is positive; each negative example is made of a collection of negative instances. Whether Multiple-Instance learning or Reinforcement Learning is more ambiguous is not clear, and so their order in Figure 1-1 is arbitrary. The contribution of this thesis is threefold:

- We present a new scheme — *Diverse Density* — of learning from Multiple-Instance examples.
- We show that there are many different learning problems that fall into the Multiple-Instance learning framework.
- Finally, we describe architectures for transforming difficult learning problems into Multiple-Instance learning problems, which in turn may be solved with Diverse Density techniques.

Supervised learning has been a dominant and useful framework over the past few decades. Many learning problems with ambiguous labels were forced into the supervised learning framework with the hope that enough examples would overcome any ambiguity or noise. However, a major undercurrent of this thesis is that it is crucial to explicitly tackle the ambiguity inherent in many learning problems. The techniques developed in this thesis should make the process of tackling ambiguity a simple, efficient (both in the number of examples and computation needed), and general one.

1.1 Multiple-Instance learning

Multiple-Instance learning is a framework for learning from ambiguity. Like supervised learning, each example is labeled. Unlike supervised learning, an example is not a simple feature vector, but is actually a collection of instances. We call a collection of instances a *bag*. Each *instance* is described by a vector of features. Each bag can contain a different number of instances. In this thesis, we will deal only with two-class classification problems, so a bag's label is either positive or negative. Extensions to multi-class and regression problems are discussed in Chapter 8.

- A bag is labeled **negative** if all the instances in it are negative.
- A bag is labeled **positive** if at least one of the instances in it is positive.

Note that while there exists some unknown function that can label individual instances, only the bags are actually labeled. Given a training set of labeled bags, a Multiple-Instance learning algorithm attempts to find a concept that correctly predicts the labels on the training set and generalizes to predict the labels for unseen bags. It can also predict the class of individual instances if a bag only contains a single instance.

If there was only one instance per bag, then the Multiple-Instance learning problem would be reduced to regular supervised learning. However, with many instances per bag, the "noise ratio" in a positive bag can be arbitrarily high. Specifically, the Multiple-Instance framework does not prohibit a bag with one true positive and millions of negatives being labeled as a positive bag. Even when supervised learning problems assume noisy labels, the probability of a corrupt label is less than 50%. The Multiple-Instance learning problem therefore can be much more difficult than the noisy supervised learning problem. The tradeoff for learning in a more complex environment is that the concepts learned in this framework will be very simple. We will not be learning concepts such as a neural network.

There are two trivial transformations of a Multiple-Instance learning problem to a supervised learning problem, but neither one is likely to be successful. One transformation is to treat every instance in a positive bag as positive, and every instance in a negative bag as negative. This is likely to fail because the ratio of true positive instances to false positive instances can be arbitrarily high, and the supervised learner would be overwhelmed by incorrectly labeled examples. Another transformation is to make each bag a single example, with all the instances concatenated together to form a single feature vector. This is likely to fail because the "true instance" might be the first feature of one example and the third feature of a different example. In addition, every bag can have a different number of instances, leading to examples with varying numbers of features.

1.2 Multiple-Instance learning applications

The learning framework described above may appear abstract, but in fact, there are a variety of learning problems that can be tackled as Multiple-Instance problems. In this section we summarize the applications which are discussed in detail in Chapters 4, 5, and 6. In all three of these applications, the bags need to be constructed before a Multiple-Instance learning algorithm can be applied.

1.2.1 Drug discovery

In the drug discovery application, one objective is to predict whether a candidate drug molecule will bind strongly to a target protein known to be involved in some disease state. Typically, one has examples of molecules that bind well to the target protein and also those that do not bind well. Just as with a lock and key, shape is the most important factor in determining whether a drug molecule and the target protein will bind. However, drug molecules are flexible, and can adopt a wide range of shapes. A positive example does not convey what shape the molecule took when binding to the target – only that *one* of the shapes that the molecule can take was the right one. However, a negative example means that none of the shapes that the molecule can achieve was bound to the target.

Each molecule is represented by a bag, and the bag's label is positive if the molecule binds well to the target protein. A bag is made up of instances, where each instance represents one shape (or *conformation*) that the molecule can take. The instances are sampled from the large variety of low-energy conformations of the molecule. After being trained on a set of positive and negative bags, the learning algorithm returns a concept which represents constraints on the shape of a molecule that would bind to the target protein. In Chapter 4, we discuss how to limit the number of instances per bag and how to represent shape. In addition, we examine a particular set of molecules (musks) and show results of Diverse Density and other learning algorithms on this task.

1.2.2 Stock prediction

The stock market is one of the most noisy (some would even say chaotic [Trippi, 1995]) and popular domains for Machine Learning techniques. One reason for the apparent high amount of noise in the stock market is that most price changes are caused by unpredictable current events and public sentiment, rather than by fundamental financial and economic reasons. For example, IBM's stock price went up in May 1997 because their Deep Blue system beat Gary Kasparov in chess, an event for which few computerized trading systems were trained[1]. Coca-Cola's stock went up in June 1993 when rumors of people finding syringes in Pepsi cans spread across the country[2]. The price of IPOs with "net" or "web" in the company's name does not always lead to judicious choices. Netscape's stock value tripled in its first four months, only to return to its initial value.[3]

A learning algorithm trained on these stocks will only learn spurious correlations between the stock's behavior and its economic features. However, some stocks do go up and down in value according to some fundamental (*platonic*) aspects of the stock. If a learning algorithm could be trained on these stocks, then the ideal stock description could be found. The ambiguity in this problem is that we do not know which stocks behave according to some fundamental economic features (or when) and which do not. However, if we put enough high-return stocks in a bag, then with high probability at least one of them will be platonically good. Every stock is described

[1] The price of an IBM share went from $161 on May 2, a day before the match started, to $173.50 on May 12, a day after Deep Blue's victory.

[2] The price of a Coca-Cola share went from $20.02 on June 9 when a Seattle couple found a syringe in a Pepsi can, to $21.22 on June 17 before a Colorado woman was caught by a supermarket surveillance camera placing objects into a can of Diet Pepsi.

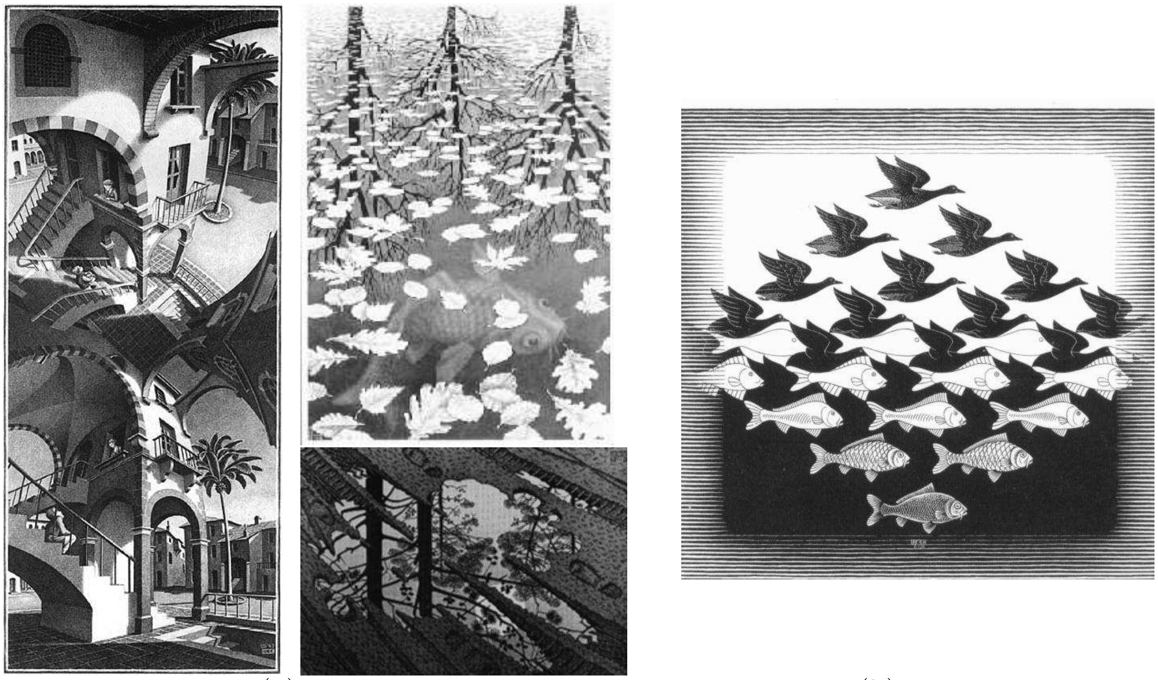
[3] Netscape went from $29.125 as the closing price on its first day of trading (May 12, 1995), to $85.50 on December 5, 1995. It was back at $29.125 on April 29, 1998.

Figure 1-2: What is this image? A rock formation (El Capitan), a river (Merced River), clouds, or trees? This is an example of how images are inherently ambiguous.

using a feature vector, and a collection of stocks (instances) that performed well during a particular month constitute a positive bag. Each bag represents a different month. The fundamental concept, by definition, does not change from month to month. Therefore, there will be a common instance to all the positive bags, much like the common shape of all of the positive molecules in the previous subsection. In Chapter 5, we show results of this approach in learning to select stocks in the US stock market.

1.2.3 Image database retrieval

If a person were asked to describe the image in Figure 1-2 , any of these responses would be considered valid: “a rock formation,” “a river,” “trees,” “clouds,” “a blue blob over a green blob,” etc. A computer, even armed with an object recognizer (still an elusive goal for Machine Vision researchers), could hope to do no better. Most images are inherently ambiguous disseminators of information. Unfortunately, interfaces to image databases normally involve the user giving the system ambiguous queries such as “Give me images like Figure 1-2.” By treating each query as a Multiple-Instance example, we make the ambiguity in each image explicit. In addition, by



(a) (b)

Figure 1-3: Some (a) positive and (b) negative training examples

receiving several positive and negative examples, the system can learn what the user desires. Using the learned concept, the system returns images from the database that are close to that concept.

Let us take a hypothetical example: imagine that the user enters the images in Figure 1-3(a) as positive examples of what they are looking for, and the image in Figure 1-3(b) as a negative example[4]. Let us further suppose that we have an object recognizer such that when given an image, it outputs the set of objects that are in the image. In the three positive images the output could be

{`building, trees, people, windows`}
{`trees, leaves, fish, water`}
{`water, trees, footsteps, tire tracks, mud`}

and for the negative image the output could be

{`fish, geese`}.

This fits the Multiple-Instance framework precisely, where every image is a bag, and each instance in a bag is a particular way of describing the image. In this hypothetical example, we assume that the image can be described by some of the objects in it. In Chapter 6, we assume that the image can be described by some of the subregions in it because object recognition is not currently feasible. A Multiple-Instance learning algorithm can be trained on the bags and determine that the concept is images with trees in them. The database is then searched for other images with trees. In Chapter 6, we describe a similar system, designed to search a database for images of natural scenes such as sunsets and mountains. We explore various bag generators, and show that, at least for images of natural scenes, one does not need the full power of the mythical object recognizer.

1.3 Diverse Density

We now give an intuitive explanation of a technique called Diverse Density for learning from multiple-instance examples. This technique was first introduced in [Maron and Lozano-Pérez, 1998]. A more detailed discussion of the algorithm, a formal derivation, its mathematical assumptions, and its variants are given in Chapters 2 and 3. The applications in the previous section have hinted at the approach taken in this thesis. If we can find what is in common among the positive bags and does not appear in the negative bags, then we have a concept that agrees with the training examples. If we treat bags as sets, the natural operators for this technique are the intersection, union, and difference operators. Specifically, we want

the intersection of the positive bags minus the union of the negative bags.

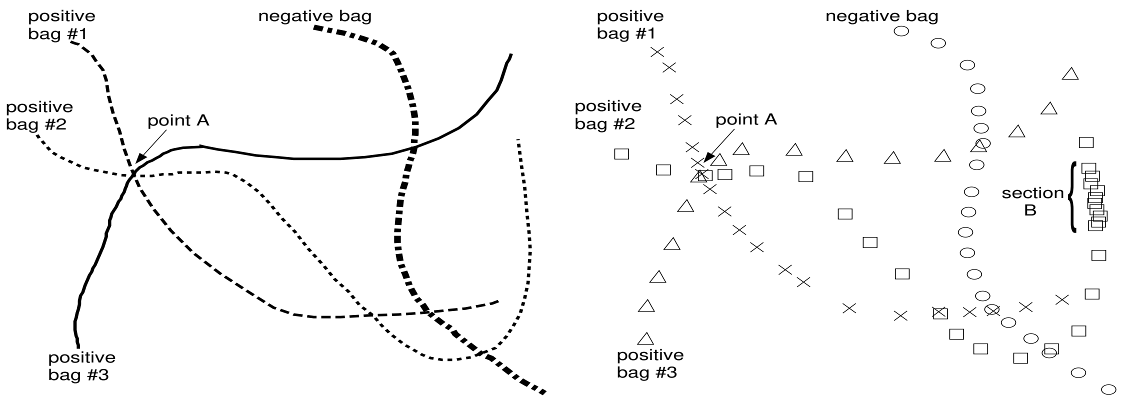
Unfortunately, most real world problems involve noisy information. The features of the instances might be corrupted by noise, some features may be irrelevant or less important than others, some of the labels of the bags might be wrong, or the strict intersection of the positive bags may be empty. We would still like to perform operations such as intersection, but we would like to have a softer (less strict) version of them. In this thesis, we achieve a soft version of intersection by thinking of the instances and bags as coming from some probability distribution. The location of an instance is therefore treated as evidence of the location of the true underlying concept. Chapter 2 describes how evidence from instances within a single bag is combined. It also describes how evidence from multiple bags is combined to form a probability distribution over the space of candidate concepts (each of which is a potential intersection area).

Once soft versions of intersection, union, and difference have been established we can assign every possible concept a measure of "goodness." If we could use a discrete version of intersection, then goodness would be a binary value: 1 if the concept is contained by the intersection of the positive bags minus the union of the negative bags, and 0 otherwise. Using soft versions, the value of goodness is continuous and is indicative of the probability that this concept agrees with the underlying distribution of positive and negative bags. We have coined the term *Diverse Density* for this measure to indicate that it measures not merely a co-occurrence of samples (i.e. intersection of instances), but a co-occurrence of instances from *different* (diverse) positive bags.

To illustrate this technique, we use a simplified example of the drug discovery problem. Assume that two features are sufficient to describe the shape of a molecule. A particular conformation is therefore represented as a point in a 2-dimensional feature space, and all possible conformations of a molecule can be represented as a manifold through this feature space.

If a candidate molecule is labeled positive, we know that in at least one place along the manifold, it took on the right shape to fit into the target protein. If the molecule is labeled negative, we know that none of the conformations along its manifold will allow binding with the target protein. If we assume that there is only one shape that will bind to the target protein, the correct shape is where all positive feature-manifolds intersect without intersecting any negative feature-manifolds. For example, in Figure 1-4(a), the correct shape is point A.

Unfortunately, a multiple-instance bag does not give us complete distribution information, but only some arbitrary sample from that distribution. In fact, in applications other than drug discovery, there is no notion of an underlying continuous



(a)

The different shapes that a molecule can take on are represented as a path. The intersection point of positive paths is where they took on the same shape.

(b)

Samples taken along the paths. Location B is a high density area, but location A is a high Diverse Density area.

Figure 1-4: A motivating example for Diverse Density

manifold. In this case, Figure 1-4(a) becomes Figure 1-4(b). Instead of finding an intersection, we need to find an area where there is both high density of positive points and low density of negative points — a soft intersection. Normally, to find an intersection of a set of samples, one finds an area of high sample density. Section B in Figure 1-4(b) is an area of high density, but that is not the answer we want. Therefore, we are not just looking for high density, but high "Diverse Density." We define Diverse Density at a point to be a measure of how many *different* positive bags have instances near that point and how far the negative instances are from that point. In Chapter 2, we estimate the probability that a bag is near a hypothesized concept. We also show that Diverse Density can be computed as a combination of the probability densities from each bag.

In Chapter 3, we examine several ways of using Diverse Density to generate a concept from multiple-instance examples. One is to find the concept that maximizes Diverse Density by finding locations in the feature space (concepts) that have high Diverse Density. Another approach is to use Diverse Density to find which instances in a positive bag are the "true positive instances." If Diverse Density is computed at every instance, the "true positive instances" will be those instances which have the highest Diverse Density. Once those are found, the problem has been reduced to a simple supervised learning problem. The advantage of this method is that the computationally expensive operation of finding locations in feature space with high Diverse Density has been bypassed.

Chapters 4, 5, and 6 discuss the three applications outlined in Section 1.2. They are representative of situations where Multiple-Instance learning can be used. In these applications, we describe a variety of *bag generators* that transform the problems into the Multiple-Instance framework. In the drug discovery application, the data is sampled from low-energy conformations of the various molecules and converted into bags. In the stock prediction application, highly noisy instances need to be gathered into bags. In the image database retrieval application, each highly ambiguous example (image) needs to be broken up into instances (things the image could be about). In this application, we describe a range of possible bag generators, and show results from experiments using a variety of different generators. These results are taken from [Maron and Lakshmi Ratan, 1998].

In Chapter 7, we discuss other work on Multiple-Instance learning and on learning from ambiguous examples in general. Finally, future work and conclusions are presented in Chapter 8. Appendix A details the computation of Diverse Density and its derivatives. Appendix B gives details of the various experiments described in the thesis.

Chapter 2

Computing Diverse Density

In this chapter we explore different ways to compute Diverse Density. There are two types of variations for computing it. One variation involves different ways of combining evidence from instances in a bag. The other variation is derived from our use of different concept classes. A *concept class* defines the set of possible concepts that can be learned by the algorithm. Diverse Density of a possible concept is a measure of the amount of intersection between the positive bags at that concept. It also measures the amount of negative evidence (instances from negatively labeled bags) at that concept.

2.1 Notation

The training data is presented as positive and negative bags. Positive bag are denoted as

$$B_1^+, \cdots, B_n^+$$

and negative bags as

$$B_1^-, \cdots, B_m^-$$

When the label on the bag does not matter, it will simply be referred to as B_i. A bag may contain any number of instances. The p instances of the i^{th} positive bag are written as

$$B_{i1}^{+}, \cdots, B_{ip}^{+}$$

and similarly for a negative bag. Each instance represents a point in a k-dimensional feature space. The individual feature values of the j^{th} instance of the i^{th} positive bag are

$$B_{ij1}^{+}, \cdots, B_{ijk}^{+}.$$

2.2 Diverse Density as probability

Given some concept class C, consisting of concepts $\{c_t\}$, we define a random variable T, whose domain is the set of concepts. $\Pr(T = t)$ is the probability that the t^{th} concept is correct. We use $\Pr(t)$ as shorthand for that probability. We want to compute the probability that a concept, c_t, is the target concept given the training examples. We call this probability the *Diverse Density* of the t^{th} concept:

$$DD(t) = \Pr(t \mid B_1^{+}, \cdots, B_n^{+}, B_1^{-}, \cdots, B_m^{-}) \tag{2.1}$$

We would like to maximize this probability with respect to t, to find the target concept c_t that is most likely to agree with the data. In Chapter 3, we examine methods for maximizing Diverse Density. In this chapter, we examine how to compute Diverse Density for a particular concept c_t.

Using Bayes' rule, Formula 2.1 becomes

$$DD(t) = \frac{\Pr(B_1^{+}, \cdots, B_n^{+}, B_1^{-}, \cdots, B_m^{-} \mid t) \Pr(t)}{\Pr(B_1^{+}, \cdots, B_n^{+}, B_1^{-}, \cdots, B_m^{-})}. \tag{2.2}$$

The quantity $\Pr(t)$ represents the prior knowledge of what concept is preferred. In this thesis, we do not use any prior knowledge, and therefore $\Pr(t)$ is constant. The quantity $\Pr(B_1^{+}, \cdots, B_n^{+}, B_1^{-}, \cdots, B_m^{-})$ is also constant with respect to t. Since we are maximizing with respect to t, this quantity can be thought of as a normalizing term and need not be calculated explicitly. Therefore, we are left to compute the likelihood:

$$\Pr(B_1^+, \cdots, B_n^+, B_1^-, \cdots, B_m^- \mid t) \tag{2.3}$$

We can further simplify Formula 2.3 by making the key assumption that all bags are conditionally independent given the true target concept. This allows us to decompose Formula 2.3 into

$$\prod_{1 \leq i \leq n} \Pr(B_i^+ \mid t) \prod_{1 \leq i \leq m} \Pr(B_i^- \mid t) \tag{2.4}$$

2.2.1 Exact generative models

A generative model which describes how each bag was generated from the target concept could be plugged into Formula 2.4 to calculate Diverse Density exactly. We now examine two possible generative models, one for a discrete domain and another for a continuous domain. We do not have generative models for the applications described in Chapters 4, 5 and 6. However, the models examined here hint at possible general approximations discussed in Section 2.3.

A discrete generative model example

Let us assume that all instances are taken from the discrete domain $\{1, \ldots, D\}$. The set of possible concepts is identical, where the meaning of concept c_2 is that any instance with value 2 should be labeled positive, and instances with values other than 2 should be labeled negative.

We describe one possible generative model in this situation. The concept is picked uniformly at random from the D possible concepts. Each bag contains n instances. Positive bags have one instance equal to the concept and the rest are independently and uniformly chosen at random (with replacement) from D possible values. Negative bags have all instances independently chosen from the non-concept domain values. We can now calculate the following quantities:[1]

[1] δ[expression] is defined to be 1 when the experssion is true, and 0 otherwise.

$$\begin{aligned}
\Pr(t) &= 1/D \\
\Pr(B_i^+ \mid t) &= \sum_{1 \le j \le n} \frac{1}{n} \delta[B_{ij}^+ = t] \left(\frac{1}{D}\right)^{n-1} \\
\Pr(B_i^+) &= \left(\frac{1}{D}\right)^n \\
\Pr(t \mid B_i^+) &= \frac{1}{n} \sum_{1 \le j \le n} \delta[B_{ij}^+ = t]
\end{aligned}$$

If this was indeed the way in which bags were generated, then the above equations could be plugged into Formula 2.4. Formula 2.4 would in turn be plugged into Formula 2.2 for a complete definition of Diverse Density. Note that we can also compute the value of $\Pr(t \mid B_i^+)$, which is non-zero only when t is equal to at least one of the instances in the bag. Despite the fact that we do not need to calculate the exact inverse generative model, $\Pr(t \mid B_i^+)$, the calculations in this section motivate approximations of $\Pr(t \mid B_i^+)$ in Section 2.3.

Likewise, the generative model and inverse generative model for negative bags can be calculated as:

$$\begin{aligned}
\Pr(B_i^- \mid t) &= \left(\frac{1}{D-1}\right)^n \prod_{1 \le j \le n} \delta[B_{ij}^- \ne t] \\
\Pr(B_i^-) &= \frac{1}{D} \left(\frac{1}{D-1}\right)^n \sum_{1 \le t \le D} \prod_{1 \le j \le n} \delta[B_{ij}^- \ne t] \\
\Pr(t \mid B_i^-) &= \frac{\prod_{1 \le j \le n} \delta[B_{ij}^- \ne t]}{\sum_{1 \le t \le D} \prod_{1 \le j \le n} \delta[B_{ij}^- \ne t]}
\end{aligned}$$

where $\sum_{1 \le t \le D} \prod_{1 \le j \le n} \delta[B_{ij}^- \ne t]$ is the number of different instances in bag B_i^-. Note that the inverse generative model, $\Pr(t \mid B_i^-)$, is constant everywhere except for values of t which are equal to one of the instances, where it is zero.

A continuous generative model example

Let us assume that each instance is described using a single real number. Once again, there are as many concepts as elements in the domain. We select the true concept to be c_t with probability $G_L(t)$, where $G_L(\cdot)$ is a zero-mean, L-variance Gaussian, and $L \gg 1$ is a large number (in other words, a roughly uniform distribution).

One possible generative model is as follows. Each bag contains n instances. A positive bag contains one instance that was generated from a Gaussian distribution with mean equal to the concept and a variance of 1. The rest of the instances were generated independently from a zero-mean, L-variance Gaussian. We can now calculate the following quantities:

$$
\begin{aligned}
\Pr(t) &= G_L(t) \\
\Pr(B_i^+ \mid t) &= \sum_{1\le j\le n}\left(\frac{1}{n}G_1(B_{ij}^+ - t)\prod_{\substack{1\le l\le n\\ l\neq j}} G_L(B_{il}^+)\right) \\
&= \frac{\prod_{1\le l\le n} G_L(B_{il}^+)}{\prod_{1\le l\le n} G_L(B_{il}^+)} \sum_{1\le j\le n}\left(\frac{1}{n}G_1(B_{ij}^+ - t)\prod_{\substack{1\le l\le n\\ l\neq j}} G_L(B_{il}^+)\right) \\
&= \prod_{1\le j\le n} G_L(B_{ij}^+) \sum_{1\le j\le n}\left(\frac{1}{n}G_1(B_{ij}^+ - t)\prod_{\substack{1\le l\le n\\ l\neq j}} G_L(B_{il}^+)\frac{1}{\prod_{1\le l\le n} G_L(B_{il}^+)}\right) \\
&= \frac{\prod_{1\le j\le n} G_L(B_{ij}^+)}{nL\sqrt{2\pi}} \sum_{1\le j\le n}\left(G_1(B_{ij}^+ - t)\exp\left(\frac{(B_{ij}^+)^2}{2L^2}\right)\right) \\
\Pr(B_i^+) &= \prod_{1\le j\le n} G_L(B_{ij}^+) \\
\Pr(t \mid B_i^+) &= \frac{G_L(t)}{nL\sqrt{2\pi}} \sum_{1\le j\le n}\left(G_1(B_{ij}^+ - t)\exp\left(\frac{(B_{ij}^+)^2}{2L^2}\right)\right)
\end{aligned}
$$

Once again, we can plug the generative model's calculation of $\Pr(B_i^+ \mid t)$ into Formula 2.4 to get a complete definition of Diverse Density. As seen, we can also calculate $\Pr(t \mid B_i^+)$, which turns out to be proportional to a mixture of Gaussians. The

Gaussians are centered at the instances and the mixture coefficients are $\exp\left(\frac{(B_{ij}^{+})^2}{2L^2}\right)$. Note that when L is large, these coefficients are all approximately one. This is similar to the inverse generative model in the discrete example, except that instead of having spikes at the instances, there are Gaussian bumps.

Likewise, let us assume that negative bags are generated by independently picking n instances from $G_L(\cdot)$, except not in the area close to the target t. Namely,

$$\Pr(B_{ij}^{-} \mid t) = \begin{cases} 0 & \text{if } |B_{ij}^{-} - t| < 1 \\ \frac{G_L(B_{ij}^{-})}{1-A(t)} & \text{otherwise} \end{cases}$$

where $A(t) = \int_{t-1}^{t+1} G_L(x)dx$. The generative and inverse generative models can be calculated as follows:

$$\begin{aligned} \Pr(B_i^{-} \mid t) &= \prod_{1\leq j\leq n} \Pr(B_{ij}^{-} \mid t) \\ \Pr(t \mid B_i^{-}) &= \frac{G_L(t) \prod_{1\leq j\leq n} \Pr(B_{ij}^{-} \mid t)}{\Pr(B_i^{-})} \end{aligned}$$

Note that the inverse generative model is only zero where t is close to one of the negative instances.

2.2.2 Defining Diverse Density in terms of $\Pr(t \mid B_i)$

For the applications described in Chapters 4, 5 and 6, we do not know what the bags' generative model is. Instead of estimating a different generative model $\Pr(B_i \mid t)$ for each application, we instead attempt to use a general estimator for $\Pr(t \mid B_i)$. Intuitively, this is an easier quantity to compute because we use the instances as pieces of evidence for potential concept locations. We describe several such estimators in Section 2.3. For now, we derive Diverse Density in terms of $\Pr(t \mid B_i)$. Using Bayes' rule again on each of the terms in Formula 2.4, we get

$$\prod_{1\le i\le n} \frac{\Pr(t \mid B_i^+)\Pr(B_i^+)}{\Pr(t)} \prod_{1\le i\le m} \frac{\Pr(t \mid B_i^-)\Pr(B_i^-)}{\Pr(t)}$$

Inserting this back into Formula 2.2, we get the following definition for Diverse Density:

$$\begin{aligned} DD(t) \;=\; & \left[\prod_{1\le i\le n} \Pr(t \mid B_i^+) \prod_{1\le i\le m} \Pr(t \mid B_i^-)\right] \times \\ & \left[\frac{\prod_{1\le i\le n} \Pr(B_i^+) \prod_{1\le i\le m} \Pr(B_i^-)}{\Pr(B_1^+,\cdots,B_n^+,B_1^-,\cdots,B_m^-)}\right] \times \left[\frac{1}{Pr(t)^{n+m-1}}\right] \end{aligned} \tag{2.5}$$

Once again, the last term in Formula 2.5 is a constant if we assume a uniform prior. The middle term is also constant with respect to t, and disappears if we were to assume that the bags are generated independently. Neither the middle nor the last term in Formula 2.5 needs to be computed if the operation we want to perform is a comparison between different concepts. This is the only operation (also known as Maximum Likelihood) needed when we optimize Diverse Density in Chapter 3. The last term in Formula 2.5 might seem odd because it indicates that concepts with a low prior become more probable with additional examples. However, there is a canceling prior term in each of the $\Pr(t \mid B_i)$ terms, as can be explicitly seen in the continuous generative model example.

The first line of Formula 2.5 agrees with the intuition established in Chapter 1. The multiplication of the terms represents the idea that we want to find a concept that intersects the first positive bag *and* the second positive bag, etc., *and* is not in the first negative bag, *and* is not in the second negative bag, etc. This is a general definition of Diverse Density, but we have yet to specify how to compute $\Pr(t \mid B_i)$. In the following sections, we give several ways of computing this term, each of which gives a different instantiation of Diverse Density. The intuitive idea behind $\Pr(t \mid B_i^+)$ is that we would like it to be high if the i^{th} positive bag is "close" to c_t, and low otherwise.

Similarly, we would like $\Pr(t \mid B_i^-)$ to be high if c_t is "far" from the i^{th} negative bag, and low otherwise. The two exact generative models described earlier indicate that a reasonable estimate of $\Pr(t \mid B_i^+)$ would be a combination of the distances from each instance to the potential concept c_t.

2.3 Ways to estimate $\Pr(t \mid B_i)$

2.3.1 Using noisy-or to estimate a density

One way to estimate the likelihood of a hypothesized concept given a single bag is to use *noisy-or* [Pearl, 1988]. Noisy-or is an idea from Bayesian Networks, where it is used to calculate the probability of a binary event with multiple possible causes. In the noisy-or model, it is assumed that the event can only happen if at least one of the causations occurred. It is also assumed that the probability of any cause failing to trigger the event is independent of any other cause. Therefore, if we denote the event as E and the causes as $C_1, \cdots, C_p$, then

$$\Pr(E \mid C_1, \cdots, C_p) = 1 - \prod_{1 \leq j \leq p} (1 - \Pr(E \mid C_j)),$$

where $\Pr(E \mid C_j)$ is the causal probability of the j^{th} cause triggering event E.

By analogy, the event E corresponds to "c_t is the underlying concept." This event can happen if at least one of the instances in the bag is in c_t, so the causes are represented by instances in the bag. The density distribution analogous to noisy-or is

$$\begin{aligned} \Pr(t \mid B_i^+) &= \Pr(t \mid B_{i1}^+, \cdots, B_{ip}^+) \\ &= \frac{1 - \prod_{1 \leq j \leq p} (1 - \Pr(B_{ij}^+ \in c_t))}{Z} \end{aligned} \tag{2.6}$$

where Z is a normalizing constant that depends on the concept class, and can be computed as $\int_t \Pr(t \mid B_i)\Pr(t)dt$. If even one of the instances in the bag is likely to be in the concept c_t, then $\Pr(t \mid B_i^+)$ will be high. If none of the instances is likely to be in c_t, then $\Pr_c(t \mid B_i^+)$ will be close to zero. Computing $\Pr(B_{ij}^+ \in c_t)$ depends on the concept class, and is discussed in Section 2.4. Negative bags are handled in a similar manner, resulting in

$$\Pr(t \mid B_i^-) = \frac{\prod_{1 \le j \le p} (1 - \Pr(B_{ij}^- \in c_t))}{Z}.$$

2.3.2 Other density estimators

All-or-nothing density estimator

If we assume that there is no noise involved in any of the instances, then one interpretation of $\Pr(t \mid B_i)$ is

$$\Pr(t \mid B_i^+) = \begin{cases} 1/Z & \text{if } \exists j \text{ such that } B_{ij}^+ \in c_t \\ 0 & \text{otherwise} \end{cases}$$
$$\Pr(t \mid B_i^-) = \begin{cases} 0 & \text{if } \exists j \text{ such that } B_{ij}^- \in c_t \\ 1/Z & \text{otherwise} \end{cases} \tag{2.7}$$

This "all-or-nothing" approach means that the distribution of $\Pr(t \mid B_i^+)$ is zero everywhere except for spikes where c_t overlaps positive instances. Likewise, $\Pr(t \mid B_i^-)$ is zero only where c_t overlaps the negative instances. This is similar to the discrete generative model example, except that the size of the spike does not depend on the number of instances in the bag. When this approach is used with Formula 2.5, Diverse Density is computing a strict intersection between the positive bags minus a union of the negative bags. If the data is not noisy and precisely agrees with our assumptions, then this strict approach should work. However, in most real world problems there are noisy measurements of the instances, irrelevant attributes, and perhaps mislabeling of the bags. In that case, strict intersection would return the empty set, but a softer

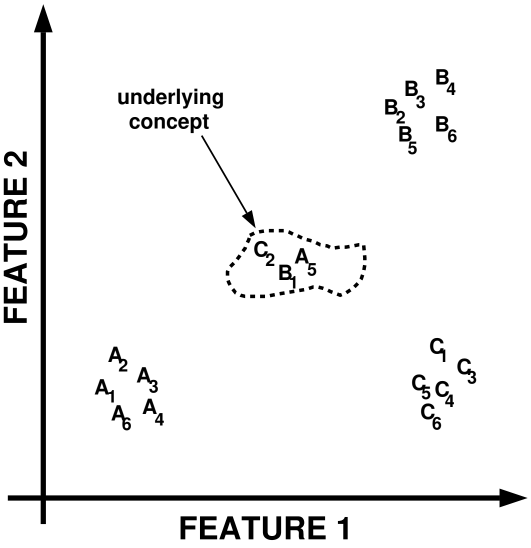


Figure 2-1: There are three bags (A, B, C), each with six instances $(A_1, \cdots, A_6, B_1, \cdots, B_6, C_1, \cdots, C_6)$. Every bag is positive because it has at least one instance within the true concept. A strict intersection of the bags would be empty. An intersection of the estimated densities of the bag would work only if A_5, B_1, and C_2 are not ignored as outliers.

approach (such as noisy-or) would be able to handle some amount of noise.

Most-likely-cause estimator

An approach which is similar to noisy-or, but which requires fewer independence assumptions, is the most-likely-cause method. $\Pr(t \mid B_i)$ is estimated by looking only at the instance in the bag which is most likely to be in the concept c_t. Specifically,

$$\begin{aligned} \Pr(t \mid B_i^+) &= \max_j \{\Pr(B_{ij}^+ \in c_t)\}/Z \\ \Pr(t \mid B_i^-) &= (1 - \max_j \{\Pr(B_{ij}^- \in c_t)\})/Z \end{aligned} \tag{2.8}$$

Using domain knowledge in the estimator

So far, we have tried to make minimal assumptions about how the bag was generated. If domain-specific knowledge is known, other density estimators may be better and more computationally efficient than the ones described here. However, without that knowledge, some density estimators are likely to be very bad choices for computing Diverse Density. For example, suppose that we tried to estimate the density of each of the bags in Figure 2-1 using a single Gaussian for each bag. For bag A, the Gaussian would be centered in the cluster of A's instances, and instance A_5 — the very instance that makes the bag positive — would be ignored as an outlier. The same problem occurs in the other bags, so Diverse Density would fail as an indicator of intersection if this method were used. The lesson to be learned from this example is that without problem-specific knowledge one cannot afford to treat *any* instance as an outlier.

One example of using domain knowledge to create a better evidence combination function occurs in [Keeler *et al.*, 1991a] and [Dayan and Zemel, 1995]. Unlike the noisy-or model, which assumes that there is at least one cause, they assume that there is exactly one cause. They use a sum of likelihood-ratios estimate of event E given a set of causes $C_1, \cdots, C_p$:

$$\Pr(E \mid C_1, \cdots, C_p) = \sum_{1 \leq j \leq p} \frac{\Pr(E \mid C_j)}{\Pr(\overline{E} \mid C_j)}$$

In Multiple-Instance learning, this translates to the assumption that there is *exactly* one instance which is a "true positive instance" in every positive bag.

Section 2.5 compares some of the density estimators for $\Pr(t \mid B_i^+)$. In the next section, we describe the final piece needed to compute Diverse Density: the probability that an instance B_{ij} is in the concept c_t. This probability will depend on the concept class from which the desired concept is to be learned.

2.4 Computing $\Pr(B_{ij} \in c_t)$ for various concept classes

Previous algorithms for Multiple-Instance learning, [Dietterich *et al.*, 1997, Auer, 1997, Long and Tan, 1996], tried to learn from a hypothesis class of axis-parallel hyper-rectangles. This led to concepts that were clearly defined: either the instance was in the rectangle or it was not. Here we attempt to learn concepts that lead to a probabilistic measure of whether an instance is in the concept or not. We assume that instances are corrupted by some noise. However, the farther the instance is from the concept, the less likely it is to be a corrupted part of the concept.

We describe three concept classes of increasing complexity. All three are used in the applications of Chapters 4, 5 and 6. The three classes are also relatively simple concepts; some of the difficulties with more complicated concepts are discussed in Chapter 8. Finally, it is straightforward to measure *distance* between an instance and the concept in all three cases. The probability of an instance being in the concept will depend on that distance.

2.4.1 Single point concept class

The simplest concept class is one where every concept corresponds to a a single point in feature space. This class assumes that given an underlying true concept P, every positive bag has at least one instance that is equal to P corrupted by some Gaussian noise. Every negative has no instances that are equal to P corrupted by some Gaussian noise.

The concept c_t is therefore described as a k-dimensional vector $c_{t_1}, \cdots, c_{t_k}$, where k is the number of dimensions in feature space. We calculate the probability that an instance B_{ij} is in the concept c_t with a Gaussian-like distribution:

$$\Pr(B_{ij} \in c_t) = \exp(-\sum_{1 \leq l \leq k} (B_{ijl} - c_{t_l})^2) \tag{2.9}$$

The closer instance B_{ij} is to the t^{th} concept (which in this case is location t in feature space), the more likely it is to be in that concept.

2.4.2 Single point-and-scaling concept class

Whenever a distance metric is imposed over a feature space (as in Equation 2.9), the question arises whether the scaling of the dimensions is correct. In most real world learning problems, the data has many irrelevant or redundant attributes, and even some of the relevant attributes are more important than others. We would like to learn the best weighting of the attributes so that irrelevant ones are ignored (their weight goes to zero) and important ones are more noticeable stand out (their weight goes up).

We expand our concept class to include feature scaling in the following manner. Each member of the new concept class is denoted as a duple (c_t, c_s). Each concept is described as two k-dimensional vectors, where k is the dimensionality of the feature space. The first vector, denoted $\{c_{t_1}, \cdots, c_{t_k}\}$, is a point in feature space. The second vector, denoted $\{c_{s_1}, \cdots, c_{s_k}\}$, is a weighting of each feature. This concept class assumes that given an underlying true concept P, every positive bag has at least one instance that was generated from a Gaussian centered at P with a covariance of P's distance metric.

We calculate the probability that an instance B_{ij} is in the concept (c_t, c_s) as follows:

$$\Pr(B_{ij} \in (c_t, c_s)) = \exp(-\sum_{1 \le l \le k} (c_{s_l}(B_{ijl} - c_{t_l}))^2) \tag{2.10}$$

The closer (using distance metric c_s) instance B_{ij} is to the location c_t in feature space, the more likely it is to be in that concept. We can think of the two concept classes described so far as being parameterized. The single point concept class is parameterized by a position vector, and the single point-and-scaling concept class is parameterized by both position and scaling vectors. In Chapter 3 we show that the best parameters (i.e. the best concept) are found the same way for both concept classes — by maximizing Diverse Density with respect to those parameters.

This definition of the single point-and-scaling concept class only allows independent scaling of the features. In other words, it only allows stretching of the distance metric along the feature axis. It is possible to scale with a full covariance matrix, thereby allowing a stretch along an arbitrary axis. However, the number of parameters needed to specify c_s is k^2, which results in a large computational expense and the potential for overfitting when searching for the best concept.

2.4.3 Disjunctive point-and-scaling concept class

So far we have assumed that the target concept is a single area in feature space. More complicated concept classes can be formed by allowing a disjunction of d single-point

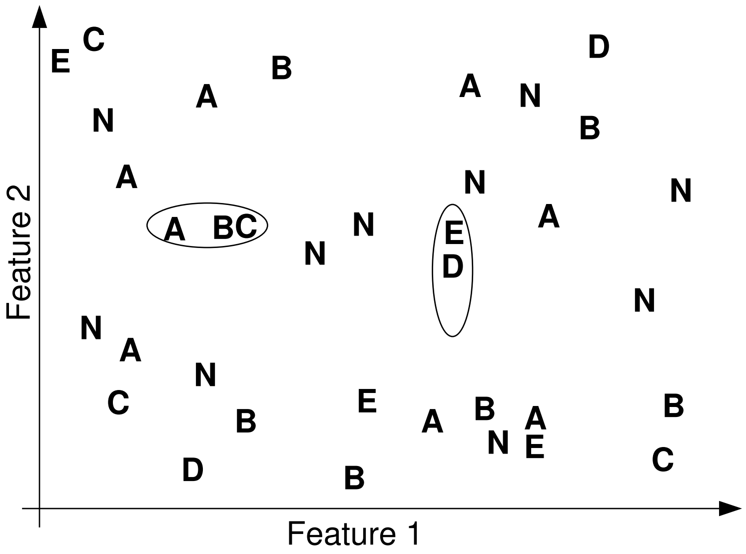


Figure 2-2: Example of a disjunctive concept. There are 5 positive bags, called A, B, C, D and E. An instance is labeled by the name of its bag. Negative instances are marked as N, all from a single negative bag. The 2-disjunct concept shown has different feature weightings in each disjunct. A new bag will be labeled positive if one of its instances falls near either disjunct.

concepts. A bag is positive if at least one of its instances is in concept (c_t^1, c_s^1) or in concept (c_t^2, c_s^2) or ... or in concept (c_t^d, c_s^d). A bag is negative if none of its instances are in any of the d concepts. Figure 2-2 shows an example of some positive bags that agree with a 2-disjunct concept.

There are many ways to estimate the probability that an instance B_{ij} is in the concept $(c_t^1, c_s^1) \vee (c_t^2, c_s^2) \vee \cdots \vee (c_t^d, c_s^d)$. The estimation we chose is to measure the maximum probability over being in any one of the disjuncts. Namely,

$$\Pr(B_{ij} \in (c_t^1, c_s^1) \vee (c_t^2, c_s^2) \vee \cdots \vee (c_t^d, c_s^d)) = \max_{1 \le a \le d} \{\Pr(B_{ij} \in (c_t^a, c_s^a))\} \tag{2.11}$$

where $\Pr(B_{ij} \in (c_t^a, c_s^a))$ is calculated as in Formula 2.10. The disjunctive concept class is parameterized by $2 \times d$ k-dimensional vectors. As we see in Chapter 3, this increase in complexity leads to a more computationally expensive algorithm, but it can also lead to better concepts.

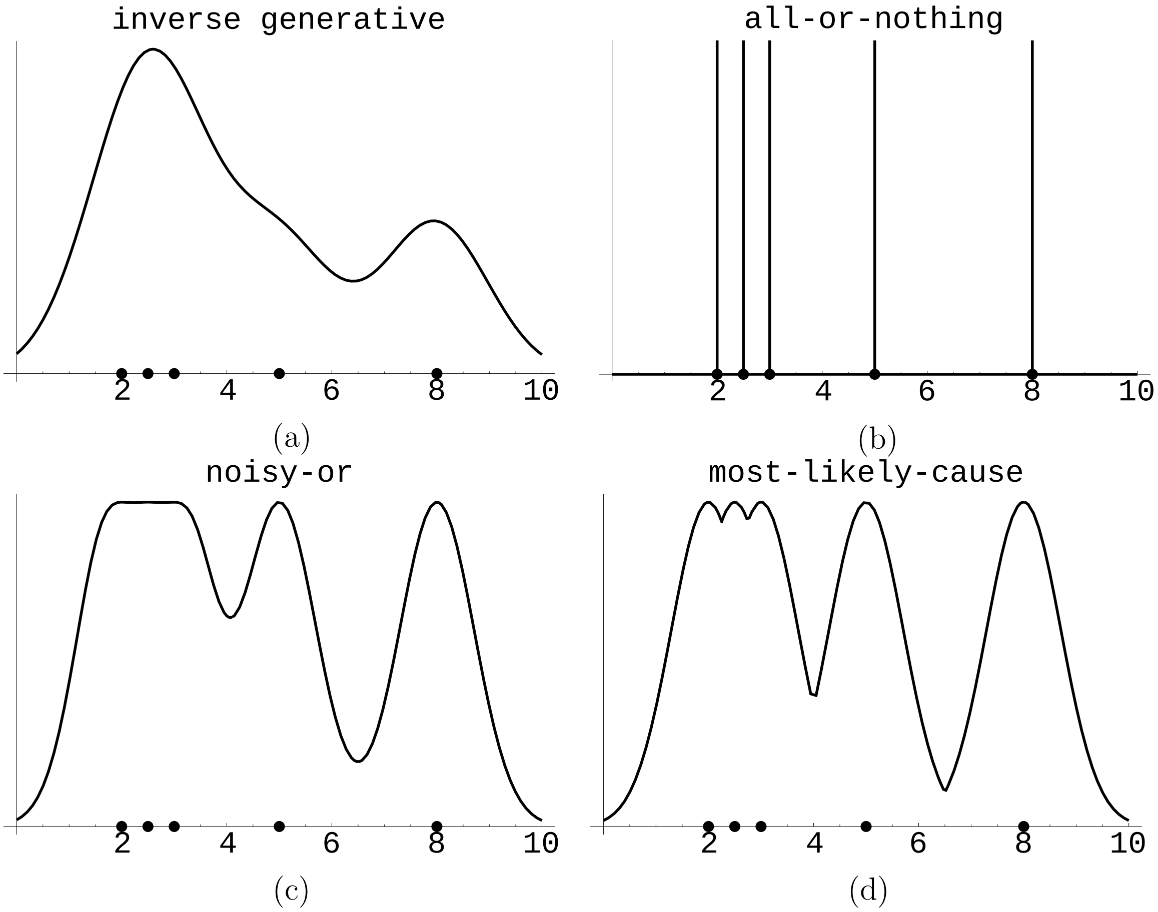


Figure 2-3: Comparison between the (a) continuous inverse generative model from Section 2.2.1, (b) all-or-nothing, (c) noisy-or, and (d) most-likely-cause approaches to approximating $\Pr(t \mid B_i^+)$. In our example, the positive bag has instances at $2.0, 2.5, 3.0, 5.0$, and 8.0 (in a one-dimensional feature space). They are marked as points along the x-axis. We use a single point concept class, so each graph shows the value of $\Pr(t \mid B_i^+)$ for every concept c_t, $t \in [0, 10]$.

2.5 Comparing density estimators

Figures 2-3 shows a comparison between the (a) continuous inverse generative model from Section 2.2.1, (b) all-or-nothing, (c) noisy-or, and (d) most-likely-cause approaches to estimating $\Pr(t \mid B_i^+)$. The positive bag contains five instances, each one described by a single continuous feature. They are marked as points along the x-axis. We use the single point concept class. The graph shows the likelihood of every concept c_t ($t \in [0, 10]$). There are a number of aspects worth noting from these examples:

- The all-or-nothing approximation is extremely restrictive and not immune to noisy data.
- The inverse generative model is based on the assumption that the instances were generated independently. Therefore, the probability that the "true positive instance" is in the cluster of instances around $t = 2.5$ is higher than at the isolated instance at $t = 8$. The noisy-or and most-likely-cause approaches do not know that the instances were generated independently, and therefore give equal credence to the hypotheses $t = 2.5$ and $t = 8$.
- The noisy-or and most-likely-cause approaches appear very similar, and we examine their differences in Figure 2-4. In Figure 2-4(a), a positive bag B_i^+ contains only one instance. If we use the single point concept class, both noisy-or and most-likely-cause compute the same value for $\Pr(t_0 \mid B_i^+)$, the likelihood of concept c_{t_0}. However, if that bag contains 7 instances which are all equidistant from t_0 (as in Figure 2-4(b)), then the likelihood of concept c_{t_0} increases with the noisy-or estimator. It remains constant with the most-likely-cause estimator.

2.6 Examples

In this section we present two example data sets. Both are learned using the single point concept class with a noisy-or estimator, and both have a one or two dimensional feature space. This allows us to plot the Diverse Density surface. This plot shows, for every concept c_t, the value of $\prod_{1 \leq i \leq n} \Pr(t \mid B_i^+) \prod_{1 \leq i \leq m} \Pr(t \mid B_i^-)$, which is proportional to $DD(t)$. The peak of this plot corresponds to the most likely concept. We can also plot each term in the above product. In the applications of Chapters 4, 5 and 6 the complexity of the concept class and the feature space will prohibit such plots.

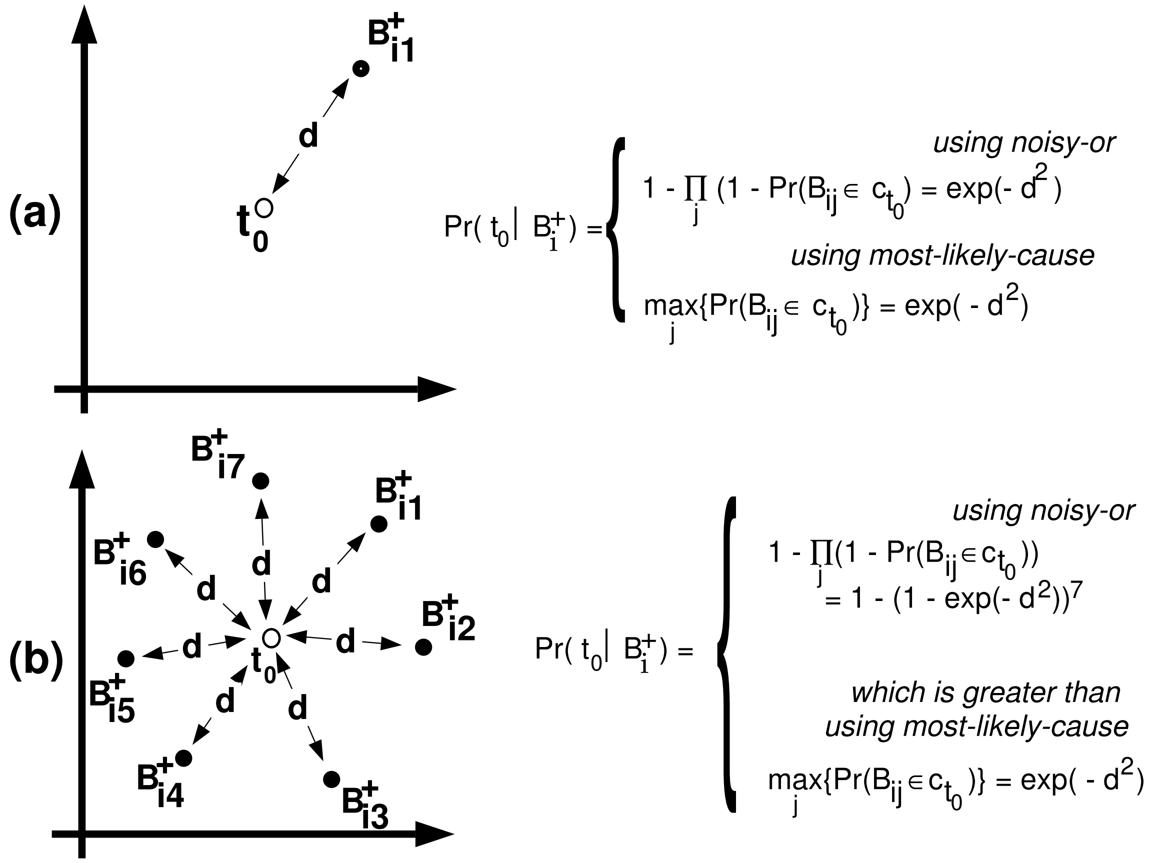


Figure 2-4: Two different situations are examined: (a) a positive bag contains a single instance and we try to estimate $\Pr(t \mid B_i^+)$ at the concept t_0 which is distance d away from B_i^+'s single instance, and (b) a positive bag contains many instances, all of which fall along a circle with radius d and center t_0. We once again try to estimate $\Pr(t \mid B_i^+)$ at t_0. Noisy-or and most-likely-cause are equivalent when there is a single minimally close instance. However, the noisy-or estimate increases as minimally equidistant instances are added, whereas the most-likely-cause estimate remains constant.

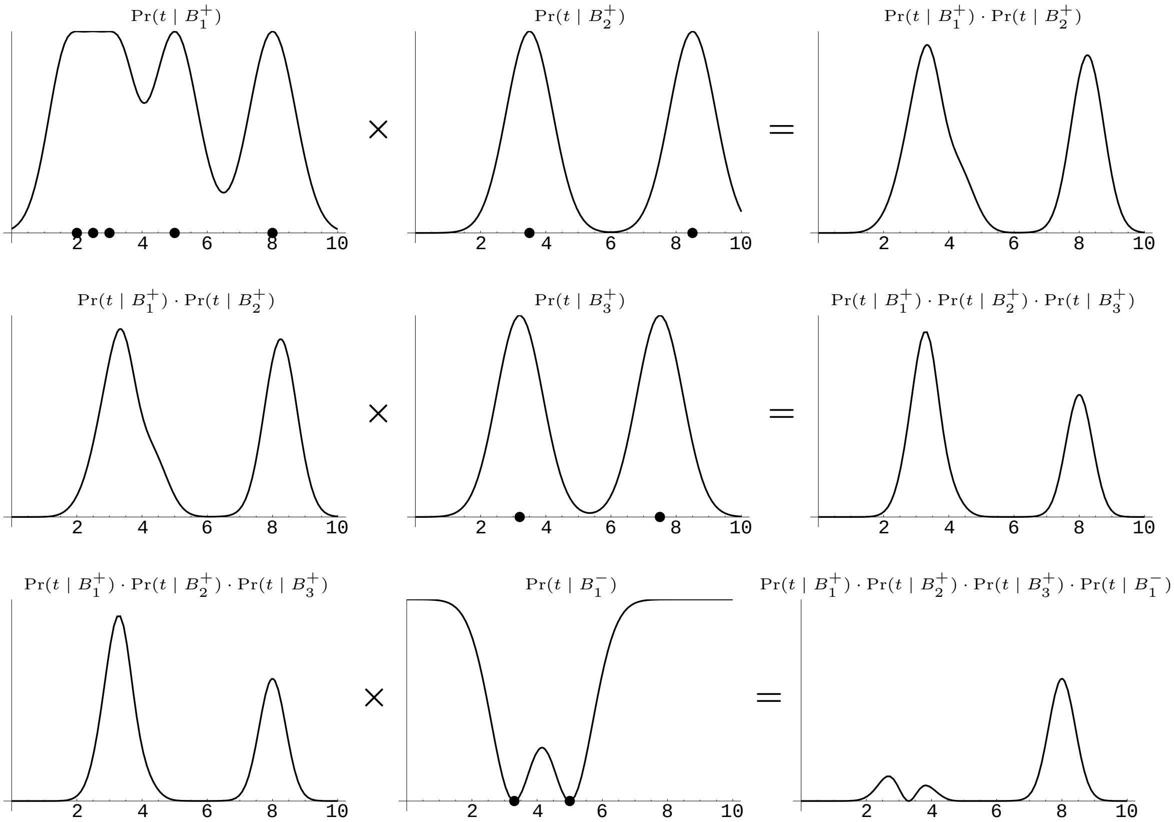


Figure 2-5: Computing $DD(t)$ for every concept c_t ($t \in [0, 10]$) on a simple data set.

2.6.1 Example: A simple data set

We constructed three positive bags (B_1^+, B_2^+, B_3^+) and one negative bag (B_1^-). The feature space is one dimensional. The instances in each bag are shown as dots along the x-axis in Figure 2-5. The concept class is single point, which means that the concept could be at any one feature location. We use noisy-or to compute $\Pr(t \mid B_i)$. Each plot in Figure 2-5 shows the probability of a concept being at location t (where t ranges from 0 to 10), given some evidence.

We can see how the Diverse Density computation progresses as more evidence is combined. The top line in Figure 2-5 shows the Diverse Density surface after combining evidence from the first two positive bags. Good candidate locations for the most likely concept appear to be at $t = 3$ and $t = 8$. The second line shows the surface after combining all three positive bags. The concept at $t = 3$ appears most probable. The third line combines that with evidence from the negative bag. Because there are negative instances close to $t = 3$, the $t = 8$ concept becomes the most likely.

2.6.2 Example: A difficult artificial data set

To test more rigorously how Diverse Density works with a noisy-or estimator and the single point concept class, we generated the following difficult data set. Ten bags were generated, each with 50 instances. Every instance was generated uniformly at random from the space of $[0, 100] \times [0, 100]$. The underlying concept was a 5×5 square in the middle of the 100×100 feature space. A bag was labeled positive if at least one of its instances fell within the square, and negative otherwise. The set of bags is shown in Figure 2-6. The instances from the negative bags are all shown as dots (since it does not matter from which negative bag an instance came). Instances from the i^{th} positive bag are each labeled as i. This is a particularly difficult data set because every bag (both positive and negative) comes from the same underlying distribution.

Figure 2-7(b) shows the Diverse Density surface over the single point concept class $\{c_t : t \in [0, 100] \times [0, 100]\}$. Diverse Density at every point is calculated using the

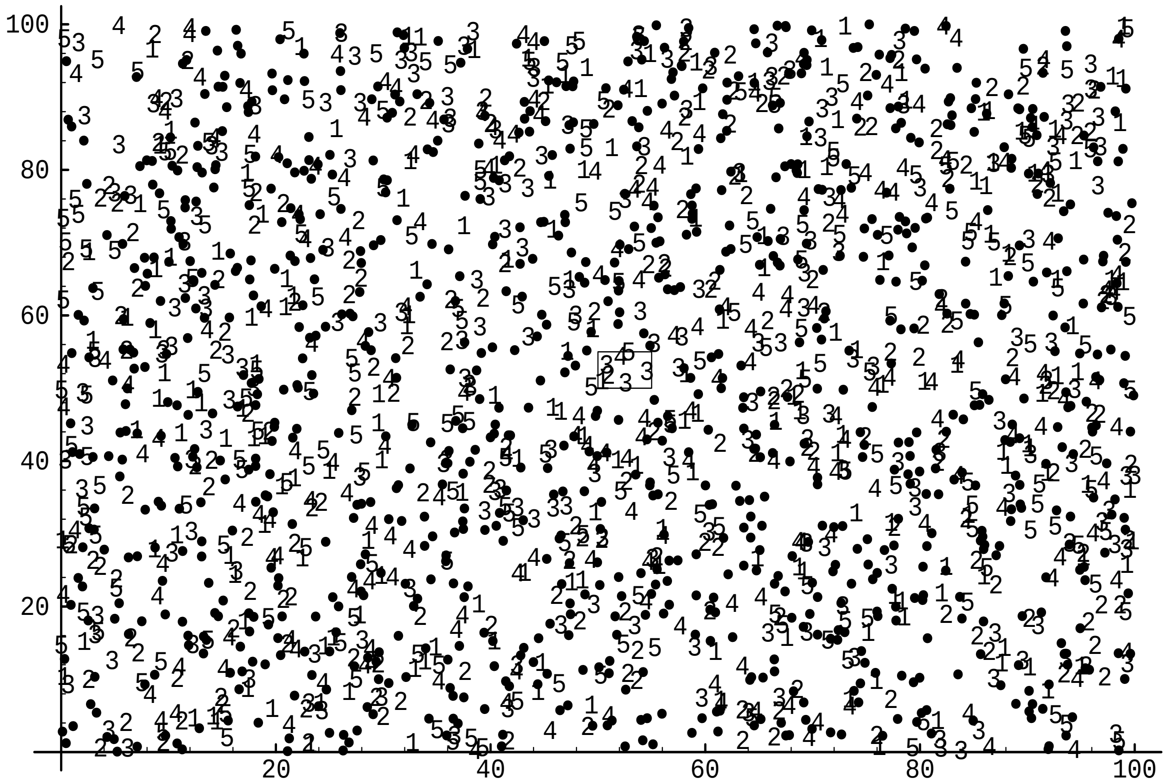
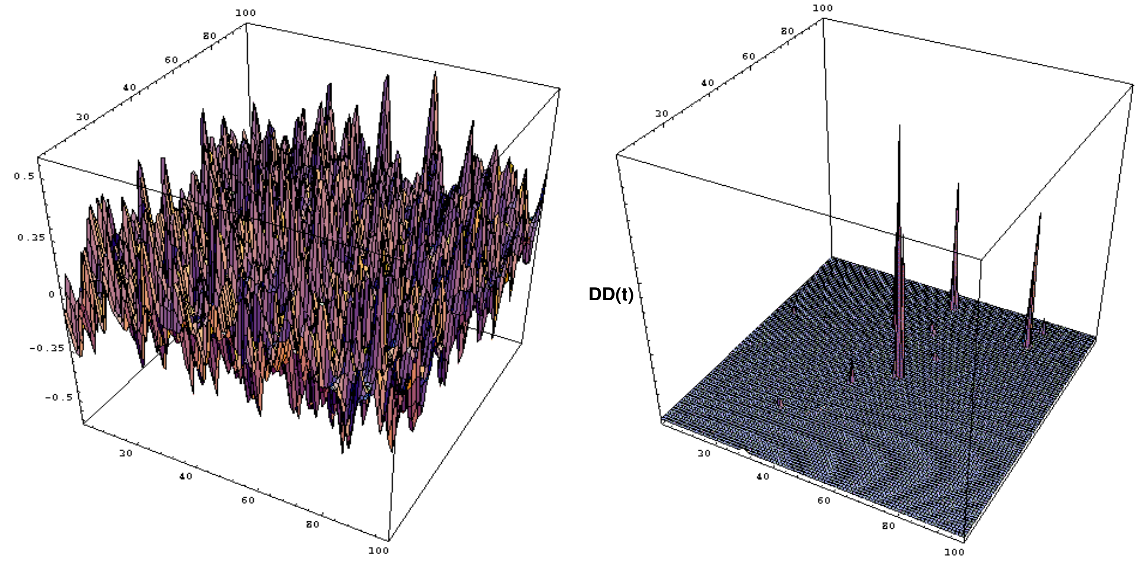


Figure 2-6: Negative and positive bags with all instances drawn from the same distribution. The bags are labeled according to their intersection with the square in the middle. Negative instances are dots, and positive instances are numbered according to which bag they are from. The square (underlying concept) contains at least one instance from every positive bag and no negatives.

noisy-or density estimator. The peak at the middle of feature space falls in the true concept, so an algorithm that returns the concept with maximum Diverse Density as its learned hypothesis (as `maxDD` does in Chapter 3) would perform well on this example.

It is instructive to look at two approaches that would fail on this training set. If we had used an all-or-nothing estimator instead of noisy-or, Diverse Density would be zero everywhere. That is because the strict intersection of the positive bags is empty. If we had tried to solve this problem as if it were a traditional supervised learning problem, then every instance would receive the label of the bag to which it belongs. We can see the surface that this would generate by *adding* (or averaging) the contributions of every bag (instead of multiplying them, as in Diverse Density), with negative bags having a negative contribution. The Kernel Regression [Atkeson



(a) Surface using regular density (b) Surface using Diverse Density

Figure 2-7: Density surfaces over the feature space of the difficult artificial example.

et al., 1997] algorithm[2] can be thought of as performing such a density estimation; it classifies a point in feature space as positive if the total density at that point is greater than zero. Figure 2-7(a) shows this density surface. Not only are there many areas that are mistakenly labeled positive (have density value greater than zero), but the true concept does not stand out. In fact, no supervised learning algorithm will perform well when roughly 49 out of every 50 positive examples are mislabeled.

The surface in Figure 2-7(b) is close to zero almost everywhere because of the exponential nature of Diverse Density. A point that is close to the intersection of n bags will have exponentially higher Diverse Density than a point that is close to the intersection of $n - q$ bags (exponential in q). The smaller peaks in Figure 2-7(b) are the result of chance concentrations of positive instances from different bags and chance sparsity of negative instances. It is possible that a different random collection of instances will have chance concentrations with even higher Diverse Density. However, as the number of training bags increases, the chance of having an accidental concentration with Diverse Density higher than at the true concept becomes

[2]Given a set of input-output training pairs $\{x_i, y_i\}$, kernel regression predicts the output at a new point, p, as a weighted average of all the outputs $\{y_i\}$, where the i^{th} weight is proportional to the zero-mean, σ^2-variance Gaussian $G(\| p - x_i \|)$. The kernel width, σ, indicates how much smoothing is performed. Radial kernels other than Gaussians can also be used.

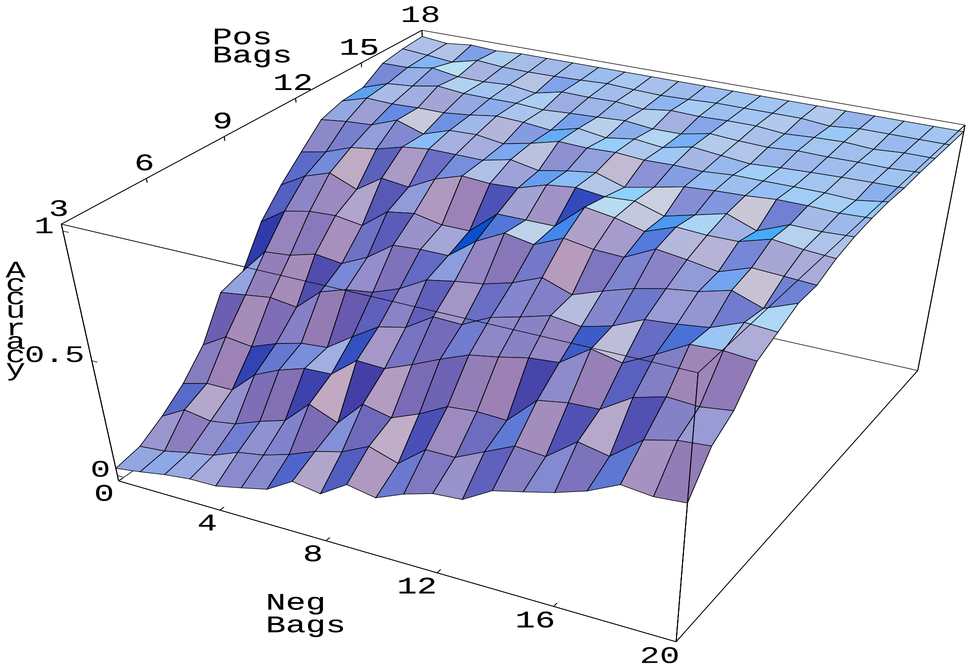


Figure 2-8: Success of Diverse Density vs. number of training bags. Accuracy was computed from 200 tests, where the bags were generated randomly in each test.

lower and lower. Figure 2-8 demonstrates that as the number of positive or negative bags is increased, the chance that the maximum Diverse Density point is within the true concept converges to 1. For every $3 \leq n \leq 18$ positive bags and $0 \leq m \leq 20$ negative bags, 200 tests were run. During each test, 200 instances were generated uniformly at random for each bag (which is a harder problem than the 50-instance bags shown in Figure 2-6). The test was called successful if the maximum Diverse Density point fell within the underlying concept. The small number of examples needed to learn consistently well on this task is compared against the MULTINST algorithm in Chapter 7.

Chapter 3

Learning a concept by using Diverse Density

We present two algorithms for finding a concept from multiple-instance examples using Diverse Density. The first, called `maxDD`, finds a concept that maximizes Diverse Density. The second, called *Pointwise Diverse Density* (`PWDD`), finds the true positive instances in each positive bag by looking for those instances with high Diverse Density. This reduces the multiple-instance learning problem to a supervised learning problem.

3.1 Maximizing Diverse Density

The most straightforward algorithm that uses the Diverse Density measure is one which attempts to find the concept with maximum Diverse Density. Since we assume a uniform prior over concepts, only the first term of Formula 2.5 can be maximized. We call this algorithm `maxDD`, and it can be thought of as performing Maximum Likelihood.

If the size of the concept class is small, then we can simply measure Diverse Density at every concept and pick the largest one. However, in the applications discussed in this thesis, the concept class is at least as big as the entire feature space. Namely, it is continuous and high-dimensional. Finding the maximum point in such a space is a difficult global optimization problem. Most of the functions described in Chapter 2 are smooth, so we can take their derivative with respect to the current concept hypothesis. The only exception is the all-or-nothing estimator[1]. The calculations of the derivatives for various estimators and concept classes are shown in Appendix A.

We can perform a gradient based optimization in concept space, starting at some initial concept, and incrementally changing it in the direction of increasing Diverse Density. This can be most easily thought of with the single point concept class. We search in feature space for the point which is closest to instances from many different positive bags and far from all negative instances. In the next few subsections, we explore how to improve this search technique and its performance on various concept classes. Details of the optimization are given in the appendices.

3.1.1 Learning from a single point concept class using multiple gradient based optimizations

The main problem with the gradient based optimization technique is that it can get stuck in local maxima. In other words, it can find a Diverse Density peak which is not as high as the optimal peak. One popular heuristic to ameliorate this problem is to use multiple optimizations, starting each one at a different random location. However, we can further improve on the multiple restart approach. The Multiple-Instance learning framework provides us with knowledge about possible good starting points.

The highest Diverse Density peak (whose location we want to find) is high precisely because it is close to instances from several different positive bags. Starting a gradient based optimization with an initial concept equal to one of those instances will likely lead to the peak because we are starting close to it. How do we know which instances are near the peak and which are not? We do not, but if we start an optimization at every instance in every positive bag, at least one will begin at a concept which is close to the maximum Diverse Density concept. Notice that the terms "instance" and

[1]The max function is also not smooth, but we will use a softmax function to approximate it (as in Appendix A).

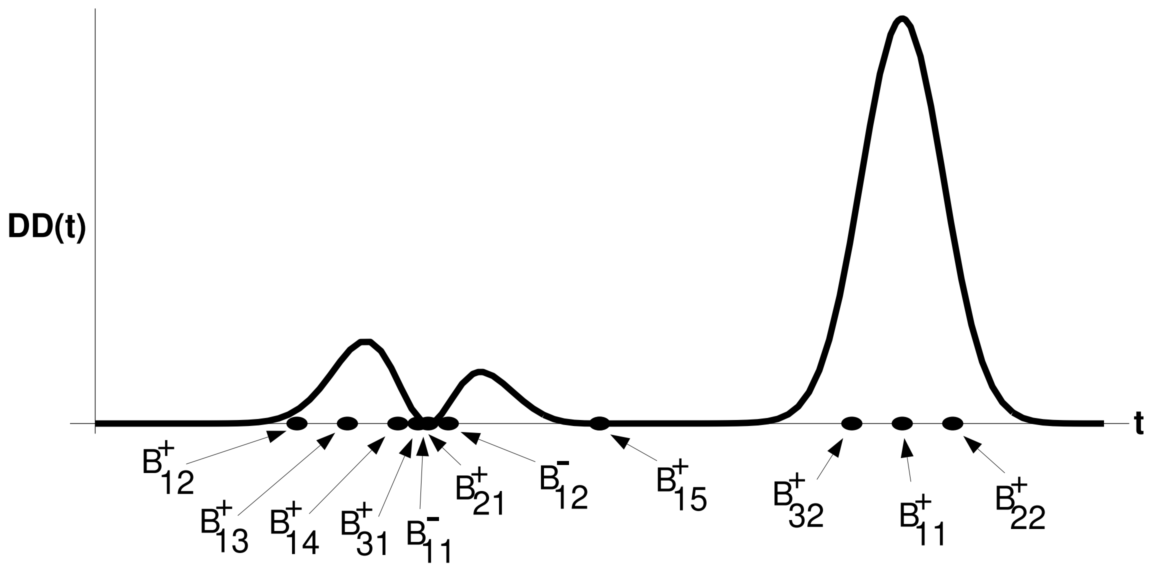


Figure 3-1: Example of potential gradient based optimization starting points for the `maxDD` algorithm. The plot shows Diverse Density for every concept in the single point concept class. The instances are described using only one feature, and are shown as dots along the x-axis. Starting an optimization from every instance in a positive bag will lead to some local minima, but some optimizations will lead to the global maximum. Figure 2-5 shows how the Diverse Density surface plot was derived from the bags.

"concept" are interchanged because there is a one-to-one mapping from a location in feature space (an instance) to a concept.

Figure 3-1 shows the Diverse Density surface derived in Figure 2-5. For every concept c_t, we plot the value of $DD(t)$. The feature space is one dimensional, and the instances are plotted as points along the x-axis. Starting a gradient based optimization at instances $B^+_{12}, B^+_{13}, B^+_{14}, B^+_{31}$ or B^+_{21} will lead to one of the two local maxima. However, starting an optimization at instances B^+_{32}, B^+_{11} or B^+_{22} will lead to the global maximum.

By starting an optimization at every positive instance we are not guaranteed to find the global maximum, but in practice this heuristic has proved very successful. The computational cost can be high if the total number of instances in all positive bags is large. However, we can reuse the same heuristic to further reduce the search. Even if we only use the instances in *one* positive bag, one of them should be close to the Diverse Density peak. Using the difficult artificial dataset of Section 2.6.2, with

5 positive bags, 5 negative bags, and 50 instances per bag, the average number of gradient based optimizations needed before the true best peak was found is 22.75. This is far less than the expected 125 optimizations. Even if the number of positive bags is doubled, the average number of optimizations needed before the eventual best peak was found remains at 22.6.

3.1.2 Learning from a point-and-scaling concept class

Using `maxDD` with a point-and-scaling concept class is not very different than the single point concept class. The concept space now has twice as many dimensions, making it a harder search problem. The multiple gradient based optimization heuristic can still be used, but we no longer have knowledge about starting values for the scaling vector. Arbitrarily, we start all scalings at one. By searching in this new space, we are optimizing for the location and scaling of the concept at the same time. It is possible to have a dual-optimizing algorithm, where we alternate the search for the best location and the best scaling.

The semantics of searching over this concept class are worth more careful study. The scaling vector defines which features are considered important (have high scale value) and which are considered unimportant (have low scale value). The correct scaling of the features at the correct location in feature space will bring instances from different positive bags closer together while keeping negative instances farther from that location; Diverse Density measures how well that is performed. Therefore, searching for the best Diverse Density is equivalent to searching for the relevant features. A schematic example of this is shown in Figure 3-2.

There are two forces that pull the search in different directions. First, as the scales *decline* in value, distances get shorter (see Formula 2.10) and instances from positive bags get closer together, increasing Diverse Density. Second, as scales *increase* in value, distances get longer and instances from negative bags get farther from the concept location, also increasing Diverse Density. This tug and pull can best be seen at an extreme: let us suppose there were no negative bags. What is the scaling that will maximize Diverse Density? Setting every feature scale to zero will bring

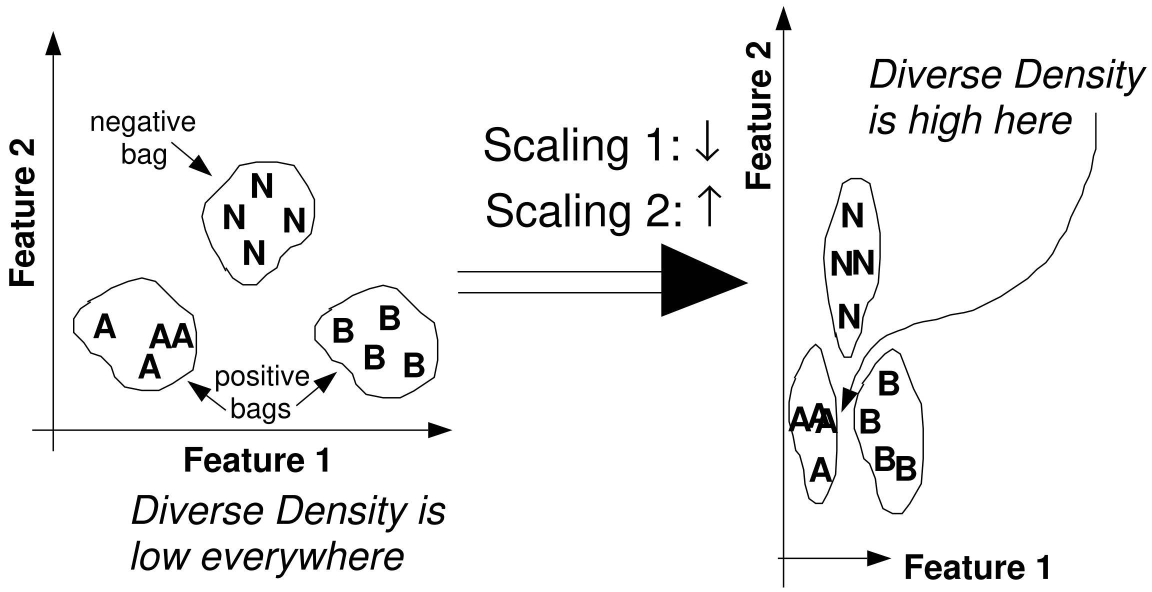


Figure 3-2: A schematic example of the effects of changing feature scales. On the left, Diverse Density has a low value regardless of the single point concept hypothesized. If we decrease Feature 1's scaling and increase Feature 2's, positive instances from different bags are moved closer and negative instances are moved farther away. On the right, we see the scaled feature space, and the Diverse Density between bags A and B is higher than anywhere in the pre-scaled concept space. The gradient on the Diverse Density surface will point toward that change in feature scalings.

all instances from all bags together at a single point, thereby achieving maximum Diverse Density. Negative evidence is essential to discovering the distance metric.

The negative instances can be thought of as girders that keep the distance metric from collapsing. The scalings chosen by `maxDD` will be as big as is required to keep the negative instances away, and no bigger. This gives us a natural bias toward simpler concepts (i.e., concepts that use fewer features). Normally, Machine Learning algorithms need to add a regularizing term (e.g., [Rissanen, 1978], [Girosi *et al.*, 1995]) to prevent overfitting with an overly complicated concept.

3.1.3 Learning disjunctive concepts

There is no conceptual difference between learning a single point-and-scaling concept and learning a disjunction of d of them. However, the computational expense and the size of the search space become much larger. The concept space has $2kd$ dimensions. The multiple gradient based optimization heuristic can be used, but the starting point consists of d instances. Therefore, the number of optimizations performed is increased from N to $\binom{N}{d}$, where N is the total number of instances in positive bags. The search attempts to find d point-and-scaling concepts such that every positive bag is close to at least one of the concepts and every negative instance is far from all concepts.

How many disjuncts are needed?

The choice of the number of disjuncts d in the concept class can be crucial to the performance of `maxDD`. If d is too small then no concept can capture the training data, and the algorithm will likely have to choose some midpoints between the true disjuncts. On the other hand, if d is too big, then it is easy to find good Diverse Density, but it will not necessarily correspond to a concept that classifies unseen examples correctly. For example, if d is chosen to be the number of positive bags, then picking the concept where t^i is an instance from the i^{th} positive bag will result in very good Diverse Density. However, that will not necessarily be a useful concept for generalizing to unseen bags. In fact, Diverse Density can only improve as d is increased, but prediction error on unseen examples will also increase.

The problem of trading off hypothesis complexity against generalization is an old one in the fields of Machine Learning and statistics. In this thesis, there are no hypothesis classes with d greater than two for reasons of computational efficiency. If computation were cheaper then more complex classes could be considered. There are two popular approaches to find the number of disjuncts d.

The first approach uses a validation set. The labeled bags are divided into a training set and a validation set. The algorithm is run repeatedly with the same training set and different values of d. The multi-disjunct concept with the best classification results on the validation set is chosen as the concept with the correct d. For better accuracy, the evaluation of the concept should be done on multiple partitions of the data into training and testing sets. This is known as cross-validation.

A second approach uses ideas from Minimum Description Length [Rissanen, 1978], Regularization theory [Morozov, 1984], PAC learning theory [Valiant, 1984], and Structural Risk Minimization [Vapnik, 1995]. All these theories attempt to capture analytically the tradeoff between complexity of the hypothesis class and ability to generalize.

3.2 Pointwise Diverse Density[2]

The `maxDD` algorithm's goal is to return a concept that maximizes Diverse Density. However, that is not the only possible goal of an algorithm that learns from Multiple-Instance examples. A different goal might be to return the correct label for every instance in the training bags. Instances in negative bags are easy because we know that all of them are negative. For instances in positive bags, the "true positive instances" need to be separated from the "false positive instances." If we had such an algorithm, then we could use it to transform the Multiple-Instance learning problem into a supervised learning problem, for which there is a plethora of solutions.

[2] Material in this section was developed with Charles Isbell.

Pointwise Diverse Density (PWDD) attempts to perform such a transformation. Given a set of positive and negative bags, it returns an instance from each positive bag which is likely to be a "true positive instance." Note that this does not mean that all other instances in positive bags are "false positive instances." During supervised learning, only the instances from negative bags and instances generated by PWDD will be used as training examples.

It is easiest to show how PWDD works with the single point concept class because of the direct mapping from an instance (location in feature space) to a concept. The algorithm behaves as follows:

1. Measure Diverse Density only at the concepts corresponding to instances in positive bags.

2. For each positive bag, return the instance with the highest Diverse Density in that bag.

The algorithm performs no gradient based optimization, and computes Diverse Density exactly N times, where N is the total number of positive instances. This will normally be a great computational saving over maxDD. As a simple example of its behavior, if its input consisted of the bags shown in Figure 3-1, it would select instances B^+_{32}, B^+_{11} and B^+_{22} to return from the three positive bags. Those are indeed the true positives.

The algorithm, though not guaranteed to return true positives, is a useful and efficient heuristic. It succeeds because those instances closest to the true concept will have the highest Diverse Density, and those are the most likely instances to have been generated from the true concept. If we assume a single point concept class and know a true positive from every bag, then simply taking the average of true positives will result in an optimal estimate of the true concept.

When tested on the difficult artificial dataset of Section 2.6.2, PWDD's learning curve behaved exactly like maxDD (Figure 2-8). Percentage of success was measured by the number of returned instances that fell within the true concept. The real advantage of PWDD over maxDD is its computational efficiency. Generating the graph of Figure 2-8 required over 6.7 million runs of either `maxDD` or `PWDD` with anywhere from 600 to 3600 total instances in positive bags. On average, the number of Diverse Density evaluations required for one run of gradient based optimization is 17.2. `PWDD` is therefore 17.2 times faster than `maxDD` on this data. On data with a higher number of dimensions, the improvement is even more marked.

3.2.1 Using `PWDD` to learn a point-and-scaling concept

The discussion so far has assumed that the scaling of feature space is correct. If we want to both find the true positives *and* the best scaling of the features, the `PWDD` technique becomes slightly more complex. The idea is a combination of `maxDD` and `PWDD`: find the feature scalings that maximize Diverse Density in at least one instance in every positive bag. Specifically, do not attempt to find the best scalings for *any* location in feature space, but for *particular* locations: the instances.

There are a number of possible techniques for performing this maximization. We list a few here. We indicate the Diverse Density of the concept corresponding to the instance B^+_{ij} as $DD(B^+_{ij})$.

- For every instance in a positive bag, B^+_{ij}, find the scaling vector $c_s(i,j)$ that maximizes $DD(B^+_{ij})$. The search can be performed using gradient based optimization as in Section 3.1. Pick the scaling that results in the maximum Diverse Density value, and use it to run `PWDD`. This method requires as many optimizations as `maxDD` for the concept class point-and-scaling. However, the search space is of dimensionality k rather than $2k$.

- For every positive bag, B^+_i, find the scaling vector $c_s(i)$ that maximizes the following quantity:

  $$\max_j \{DD(B^+_{ij})\}$$

  Once again, this can be done using gradient based optimization[3], and requires only one optimization per bag. However, the search space can potentially have

[3] To compute the gradient, we actually use softmax as described in Appendix A.

more local maxima. Run `PWDD` using the best scaling vector.

- Find the scaling vector c_s that maximizes the following quantity:

$$\max_i\{\max_j\{DD(B_{ij}^+)\}\}$$

  This requires only one gradient based optimization, but the number of local maxima can be high.

- Perform a two-step optimization process similar to dynamic reposing [Dietterich *et al.*, 1994]. Pick an initial scaling, and find instances with high Diverse Density. Find a scaling (by `maxDD`, for example) to optimize the Diverse Density of all of them. Use this scaling to find instances with high Diverse Density, and repeat.

Thoroughly comparing the merits of each of these methods is a subject of ongoing and future work.

3.3 Computational issues

So far we have discussed ways of using Diverse Density to efficiently learn a concept from multiple-instance examples. The basic operation underlying these methods is the computation of Diverse Density at a particular concept. This basic operation can be expensive as the size of the training set grows.

The calculation of Diverse Density is in many ways similar to the computations needed for a Nearest Neighbor algorithm; the distance from every instance to the hypothesized concept must be calculated. This implies that a Diverse Density calculation requires $O(N)$ space and $O(N)$ time, where N is the total number of instances in all bags. There are techniques for calculating Nearest Neighbor in sub-linear time. Bump trees [Omohundro, 1991] and k-d trees [Preparata and Shamos, 1985] are examples of such techniques, and they can be used to approximate Diverse Density. The main difficulty with using them during learning is that they assume that the distance metric is held constant. If the distance metric changes (as it does when learning with point-and-scaling concept class), the expensive operation of recalculation must be performed on the tree.

A more promising avenue is the parallelization of the Diverse Density calculation. Using the Parallel Problem Server [Husbands and Isbell, 1998], a parallel implementation of `maxDD` and `PWDD` is being developed for document retrieval applications similar to the image database application described in Chapter 6.

Chapter 4

An application to drug discovery

In the following three chapters, we discuss applications of Multiple-Instance learning and investigate how Diverse Density algorithms perform on these tasks. Each chapter will be organized in the following manner: an explanation of the application and its importance, a description of what the bags and instances are and how they were generated, and the results from various experiments on the domain.

4.1 Drug discovery

Given a target protein, the object of "drug discovery" is to find molecules that will bind tightly and selectively to the protein. The most important factor in determining whether a molecule binds to a protein is shape[1]. If we imagine the target protein to be a lock, then we need to find a key (molecule) that will fit into that lock. Performing the physical experiment of combining each candidate molecule with the target protein in a test tube is both expensive and slow, particularly if the number of different candidate molecules is large. If we can use computational techniques to determine how well a given molecule fits into a protein, then we can alleviate the expense of drug discovery and allow for a much larger search space. An even more important consideration is that synthesizing the candidate molecule can be difficult.

[1] Having complementary electrostatics is also important, but secondary for our purposes.

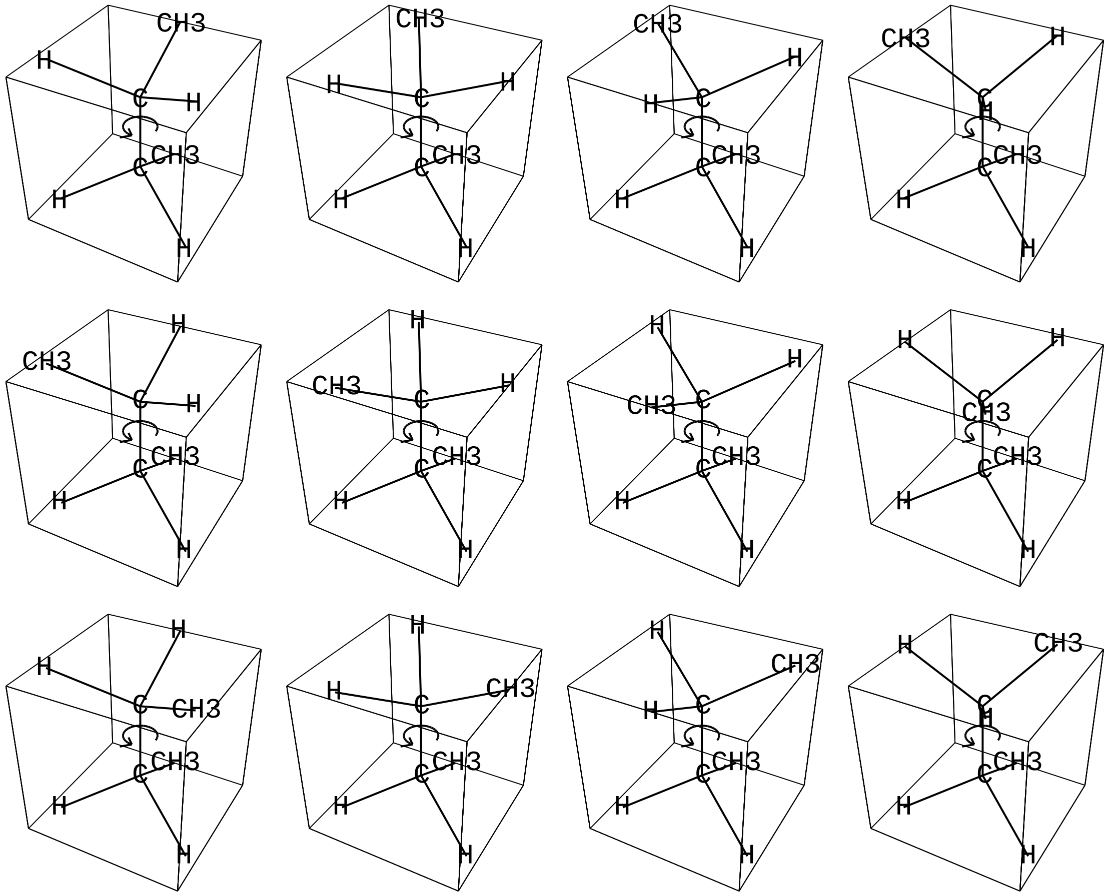


Figure 4-1: An example of the different conformations a single Butane molecule (C_4H_{10}) can take on. In this example, the various conformations occur because the molecule can rotate about the bond between the two central carbon atoms.

If we can model how well the candidate molecule will bind to the protein, then we would know whether to invest the effort needed to synthesize it.

One popular way of finding the right key shape is learning a solution from a series of positive and negative examples. A positive example is a molecule that is known to bind to the target protein, and a negative example is a molecule that does not bind to the protein. The main problem with this approach is that molecules are flexible objects, and can take on many shapes. Figure 4-1 shows an example of various shapes a Butane molecule can take simply by rotating one of its bonds. Therefore, each example is not a single shape, but a collection of shapes. If a molecule is positive, we know that at least one of its possible shapes is the correct one. We do not know

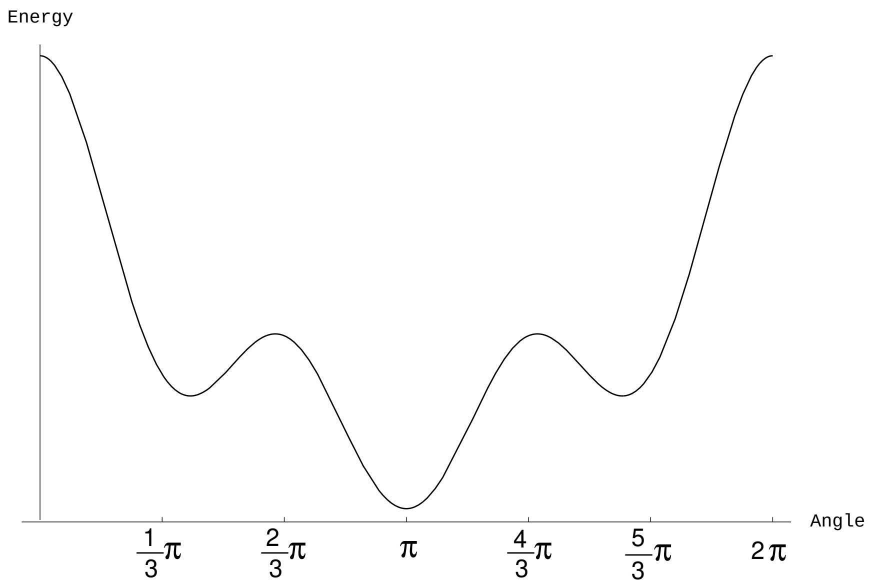


Figure 4-2: The energy function for the Butane molecule C_4H_{10} with respect to the angle of rotation of the bond between the two central carbon atoms. Energy is minimized when the two CH_3 groups are far from each other, and when the opposing hydrogen atoms are also distant.

which of its many shapes are true positives though. If a molecule is negative, we know that none of its shapes is correct. This is a Multiple-Instance learning problem, and we will use Diverse Density techniques to solve it. The learned concept should be able to correctly classify new molecules, and therefore will be a good model for the shape a molecule needs to take in order to bind to the target protein.

4.2 A molecular bag generator

We represent each molecule as a bag. Each instance in a bag is one shape that the molecule takes on. A bag's label is positive if the molecule is known to bind well to the protein, and negative otherwise. The label is generated by measuring the free energy of the binding between the molecule and the protein, and thresholding it. We now describe how the instances were generated from a given molecule, and how each instance is represented.

Given a small molecule, we can simulate essentially all of the conformations (shapes) it can take, as in Figure 4-1. However, many of these conformations are very unlikely to be the correct one. That is because many of these conformations have high potential energy, and the correct binding conformation must have low potential energy. Conformations whose energy is substantially above that of the global energy minimum for that molecule are extremely unlikely. However, it is important to note that the molecule's conformation on binding is the one that minimizes the total free energy of the protein/drug complex. It may be that the binding conformation has a higher potential energy than the most likely conformation of the molecule when found in isolation. Therefore, one cannot simply use the single (or few) conformations whose energy is minimal. This is an important consideration to weigh against the computational advantage of having only a few instances per bag.

The conformations chosen to be placed in a bag are therefore typically sampled from the approximately $O(3^n)$ local minima of the molecule's energy function (where n is the number of rotatable bonds). In addition, conformations whose energy is higher than some bound over the global energy minimum are not sampled at all [Jain *et al.*, 1994]. The bound is chosen so as to accept conformations which may be energetically reachable on binding.

Figure 4-2 shows the energy function of Butane as it goes through the conformations outlined in Figure 4-1. Depending on the value of the energy bound described above, either the conformation(s) corresponding to the one global minimum or to all three local minima would be picked as viable conformations.

There are many possible ways of describing a conformation (instance). In this thesis, we use a ray-representation [Jain *et al.*, 1994, Dietterich *et al.*, 1997] for a conformation. From a central point in the molecule, k rays are drawn out uniformly in three dimensions. Each ray is extended until it hits the surface of the molecule. The length of the k^{th} ray is the value of the k^{th} feature, as in Figure 4-3. This technique gives a straightforward representation of the molecule's shape. However, there are two important problems with the ray-representation. One problem is that it is difficult to represent concave conformations. The second problem is that this representation is not invariant to the position and orientation of the conformation. If all conformations can be aligned to have the same center of rays and the same orientation (as was

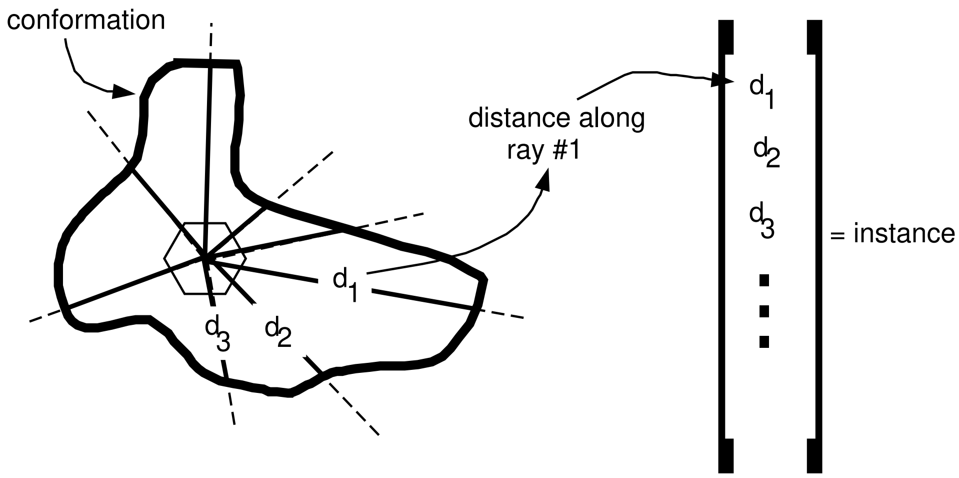


Figure 4-3: An example of using rays to represent a conformation.

done with the MUSK datasets, described in the next section), then this is not a problem. Without such an alignment, the number of instances per bag increases dramatically because each conformation must be represented in every orientation and with varying ray centers. In that case, a preferable representation would be an orientation-independent one, such as the pharmacophore model of [Dammkoehler *et al.*, 1989] or [Leach, 1996]. Another possible representation would be to place many ray-centers throughout 3D space, with only one ray coming out of each one. The distances along each ray would represent the shape of the molecule.

4.3 The MUSK datasets

A publicly available dataset (deposited at [Murphy and Aha, 1996]) was generated as described in the previous section describes positive and negative examples of musks. In this case, the target protein is a putative receptor in the human nose[2], and a molecule binds to the receptor (i.e., is labeled positive) if it smells like a musk. We would like to learn what shape makes a molecule musky (allows us to smell it).

[2]The nose senses smells through a combination of receptors [Kauer, 1991], but for these purposes, this is a reasonable approximation.

data set	number of bags	number positive	number negative	average number instances per bag
MUSK1	92	47	45	5.17
MUSK2	102	39	63	64.69

Table 4.1: Summary descriptions of the two MUSK datasets.

There are two datasets, MUSK1 and MUSK2, with some overlapping molecules. The main difference between the two datasets is that MUSK2 contains molecules that have more possible conformations than MUSK1. Some characteristics of the two datasets are shown in Table 4.1. Each conformation is represented by 162 rays, along with four additional features that specify the location of a unique oxygen atom common to all the molecules, for a total of 166 features. All molecules have been roughly aligned using the COMPASS program [Jain *et al.*, 1994].

4.4 Experiments

We ran `maxDD`, optimizing both location and feature weights (using the point-and-scaling concept class), on MUSK1 and MUSK2. To measure the generalizing capability of the learned concept, we perform several iterations of 10-fold cross validation. In a 10-fold cross validation run, the data set is randomly divided into 10 partitions. The learner is then trained 10 times, with each iteration involving a different combination of 9 partitions as the training set and one partition as the test set. After training, `maxDD` returns a concept which represents the shape (or section of shape) that is most in common to the positive molecules and not to the negative molecules. Specifically, it returns a location in feature space (the length of each ray in the correct shape), and a scaling of that space (which rays are important in distinguishing this shape from negative shapes and which are not). In these experiments, fewer than five of 166 features receive very high weights, and fewer than 30 receive any significant weight.

Unfortunately, a single point in feature space is not a very useful classifier because most molecules will not have a conformation that is described exactly by that point.

Musk Data Set 1	
Algorithm	Accuracy
iterated-discrim APR	92.4
GFS elim-kde APR	91.3
`maxDD`	88.9
MULTINST	76.7
backpropagation	75.0
C4.5	68.5

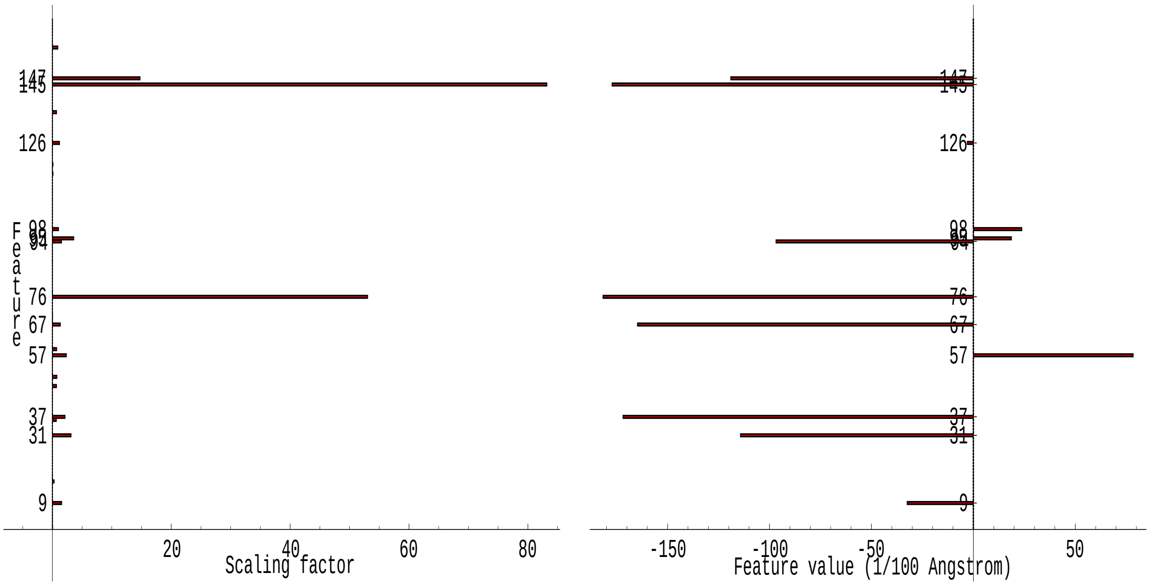
Musk Data Set 2	
Algorithm	Accuracy
iterated-discrim APR	89.2
MULTINST	84.0
`maxDD`	82.5
GFS elim-kde APR	80.4
backpropagation	67.7
C4.5	58.8

Table 4.2: Results of various algorithms on the two MUSK datasets.

We would like to classify a molecule as positive if one of its conformations is *close* to the maximum Diverse Density point, and negative if all of its conformations are far from that point. Distance can be measured using the learned scaling of feature space, but a threshold is needed to determine when a conformation is considered too far from that point. We estimate this threshold by finding the best threshold on the training set. In other words, we find a distance d such that the number of correctly classified training bags is maximized. A bag is classified positive if at least one of its instances is within distance d of the maximum Diverse Density point, and distance is calculated using the learned feature weights.

Results of various algorithms on the MUSK datasets are presented in Table 4.2. The results of the APR algorithms, backpropagation, and C4.5 are taken from [Dietterich *et al.*, 1997]. The results of the MULTINST algorithm are taken from [Auer, 1997]. Backpropagation and C4.5 are popular supervised learning algorithms which use a neural network and a decision tree, respectively, to represent the learned concept. Since supervised learning algorithms cannot learn from Multiple-Instance examples, each instance is presented as a separate example, with a positive label if it came from a positive bag. As expected, traditional supervised learning algorithms do not perform well in this framework.

The details of the MULTINST and APR algorithms are discussed in Chapter 7. The high accuracy of "iterated-discrim APR" on MUSK2 is partly due to the fact that some of its parameters were picked based on experiments on the MUSK1 dataset.



(a) (b)

Figure 4-4: A concept learned by `maxDD` for MUSK1 on one of the cross-validated training partitions. (a) shows the scale $(s_k)^2$ for each feature $k \in \{1, \cdots, 166\}$. The features whose scale is greater than 1 are indicated, and their values are given in (b).

The results of "iterated-discrim APR" can therefore be considered an upper bound on the accuracy possible with these data sets. The success of MULTINST (especially on MUSK2) is surprising because the algorithm assumes that a conformation is independent of the molecule from which it came. Specifically, MULTINST assumes that all instances were generated independently (as discussed further in Chapter 7). This might indicate that the MUSK datasets do not provide an adequate platform to test Multiple-Instance learning algorithms.

Several variations on the learning scheme described above were tried. These included using the most-likely-cause model rather than noisy-or, and learning with a two-disjunct concept class. We have also tried various thresholding methods. None of these methods performed better than the results shown above. However, these minor changes did not result in significant loss of accuracy either, indicating that the basic algorithm is fairly robust to implementation details.

Figure 4-4 shows an example of a concept learned by `maxDD` on the MUSK1 dataset for one of the cross-validated training sets. For each of the 166 features, it shows the scaling associated with that feature. Note that only a few of the features have significant (or relevant) scalings. We also show the value of the feature for those features with a scaling greater than 1. The dominance of features 145 and 76 indicate that the concept learned is almost a decision list; only if those two features cannot discriminate the class of the instance will other features be used. Concepts learned on other cross-validation partitions also use a similar scaling to the one shown here.

Chapter 5

An application to stock prediction

In this chapter, we discuss a novel application of Multiple-Instance learning to predicting the value of US market stocks. This is an important application because many people are interested in having a model of the behavior of stocks so that they will know when to sell a stock (when the model predicts its price will decline) and when to buy (when the model predicts its price will increase). Using a good predictive model, one can become very rich very quickly. It is also an important application because it is an extremely difficult problem to find a good predictive model, and the current (and past) state of the stock gives very little information as to its future.

Traditional Machine Learning techniques attempt to learn a predictive model by learning from positive examples (stocks that have performed well) and negative examples (stocks whose value has fallen) in the past. Each example is comprised of a feature vector that may include the stock's past value fluctuations, financial indicators specific to the stock, its sector, or the market at large, and any other information that might be relevant. For a classification task, an example is labeled positive if the stock's value rises in the future[1], while for a regression task an example is labeled with its actual value change. A supervised learner attempts to find a simple concept that predicts the labels on its training set well, and that will also predict the labels of unseen future examples.

[1]How far into the future is a task-dependent parameter that can very from a minute to a month to years.

There are a number of difficulties that make this a hard learning problem, but the main one is that most fluctuations in the price of a stock occur for reasons that are not economically fundamental: world events, public whims, and other unpredictable phenomena. The three examples given in the introductory chapter of this thesis about Coca Cola, IBM, and Netscape are not exceptions; they are typical of the ways that most stock prices behave. This is not to say that all stocks behave for spurious reasons, because at every time period some stocks do increase in value for fundamental economic reasons. However, we do not know which ones do so because we do not know what these fundamental economic factors are. To learn these factors, we use Multiple-Instance learning instead of supervised learning.

5.1 Generating bags as collections of stocks

We take an instance to be a particular stock X at time T. The stock X is described as a point in k-dimensional feature space. The features used to describe the stock are ones that could potentially be used to describe the unknown fundamental economic factors (the concept). We also know the change in value of the stock between time T and time $T+1$. Supervised learning treats all instances whose values increase as positive examples. Multiple-Instance learning treats a collection of instances whose value increases as a positive example.

A positive bag is created from a collection of n stocks from time T which increase in value at time $T+1$. The bag is labeled positive because we assume that at least one of the n increases occurred for fundamental reasons. A negative bag is created from a collection of m stocks from time T which decrease in value at time $T+1$. The bag is labeled negative because we assume that all of the m decreases occurred for fundamental reasons. Both of these assumptions are difficult to guarantee in practice, but we will attempt to ensure them. One positive and one negative bag are created for each time period T in the training set.

There are two problems with the bag generator described above. One problem is that perhaps none of the n instances in a positive bag are true positives. The second problem is that perhaps some of the fundamentally good stocks actually decreased in value for some spurious reasons, and these might be included in a bag as negative instances when they are, in fact, true positives. We present heuristic methods for coping with these problems. In addition, we do not calculate Diverse Density using an all-or-nothing rule, but instead use a softer method such as noisy-or (as in Chapter 2). Therefore, our Diverse Density calculation will be able to tolerate some amount of noise in cases where our heuristics fail.

To handle the possibility of having no true positive instance in the bag, we note that as n approaches infinity, the chance that at least one of the stocks in the bag is fundamentally good approaches 1. However, the more instances there are in positive bags, the harder it is to learn the concept. The solution is to find a value for n such that at least one of the stocks is fundamentally good with high probability. We present one way to estimate n, but note that the assumptions made here for the bag generator are separate from the assumptions made for the bag learner mechanism.

Assume that each stock that goes up in value has some probability p of being a fundamentally good stock. In addition, assume that each stock that goes up at some time period has the same independent probability of being fundamentally good. The chance of a stock not being fundamentally good is therefore $1 - p$, and the chance of n stocks not being fundamentally good is $(1 - p)^n$ (by the independence assumption). The probability of at least one fundamentally good stock in a positive bag is $1 - (1 - p)^n$. Given a confidence probability $1 - \delta$, we would like the chance of generating l valid positive bags (one for every time period) to be $(1-(1-p)^n)^l > 1-\delta$.

A graph of how many instances are needed per bag as p is varied is shown in Figure 5-1. The other two parameters are held at $l = 120$ and $\delta = 0.01$. This derivation is only a coarse approximation of the true number of instances needed per bag. Usually the value of p is not known and the probability that a stock is fundamentally good is not independent across stocks or time.

Generating a bag of negative examples is also not a simple task because a stock whose value decreased is not necessarily a fundamentally bad stock by the same arguments as above. One solution is not to generate any negative bags. This means

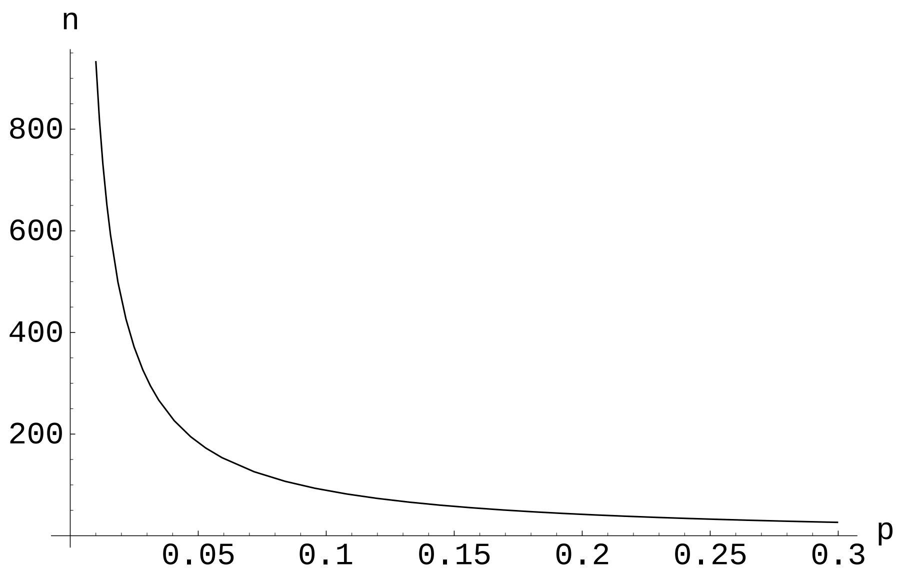


Figure 5-1: A plot of n, the number of instances per positive bag, versus p, the probability that at least one stock fundamentally good. The other two parameters are held at $l = 120$ and $\delta = 0.01$.

that we cannot perform a search for the best feature weights, only for the best feature location (as explained in Chapter 3). Another solution is to have an expert carefully pick out stocks that are fairly certain to not be fundamentally good. The solution we chose is to pick very few negative instances (10 per time period) which have the worst performance in that time period. A stock which performed so abysmally is unlikely to be fundamentally good.

Finally, we note that this approach to generating bags from extremely noisy labeled examples is not a magical cure for learning from any collection of noisy examples by converting it to bags. The main reason that we will be able to learn anything from this data is that the cause of the noise (public whims, world events, etc.) changes from one time period to another, while one of the causes of stocks performing well (some fundamental economic factors) is constant. Figure 5-2 gives a conceptual view of how the distribution of false positives changes over time, while the distribution of true positives remains constant. If all instances were drawn from the same distribution, learning would be more difficult.

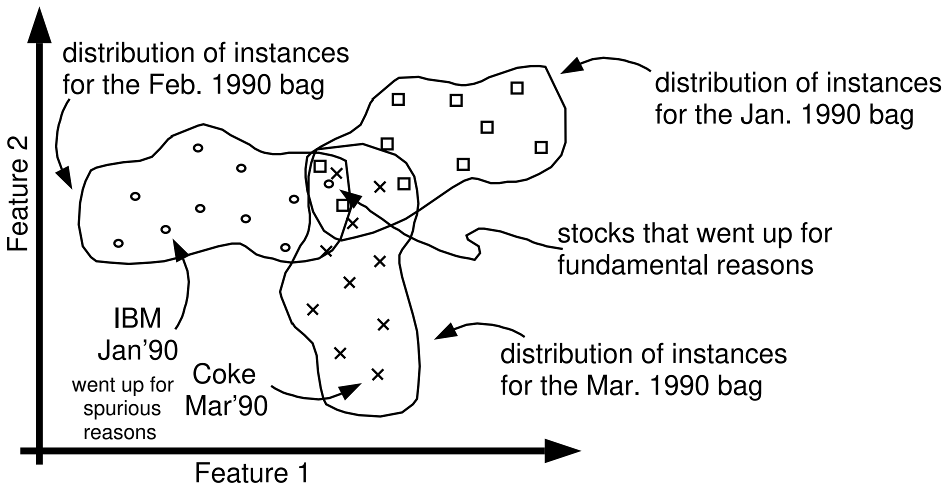


Figure 5-2: A sketch of the Multiple-Instance stock prediction problem. Three positive bags are shown, each one containing stocks that performed well in a given month. The key to distinguishing noisy instances (stocks that performed well because of spurious events) from true positives (stocks that performed well because of fundamental economic reasons) is that the distribution of noisy instances changes from one month to another as the whims of the public change.

5.2 Experiments with financial data

In this section, we report results of Multiple-Instance learning algorithms on the stock prediction task using real financial data. The data was obtained from Grantham, Mayo, Van Otterloo & Co. (GMO), a Boston investment firm, and consists of monthly information about 600 companies in the US market between 1979 and 1995. Each company was described using 17 attributes that included such aspects as the stock's size, cyclicality, momentum, and trailing returns. The learning algorithm was trained on data from 10 years, and tested on data from the following year. This procedure was repeated for a shifting 10-year window.

Bags were generated as described in the previous section. Each month, the 100 stocks with the largest positive change in value were placed in a positive bag. The 10 stocks with the largest negative change in value were placed in a negative bag. Therefore, each training set was made of 120 positive and 120 negative bags. Each test set included all 600 stocks during each of the 12 months of the test year.

Unlike the drug discovery task, the goal here is not to classify stocks as fundamentally good or not, but instead to rank stocks by a prediction of future performance. The concept learned by the system can be thought of as a description of an ideal stock. Stocks that are close to that concept using the learned scales on the features will likely go up for fundamental reasons. It is hard to claim something about the stocks that are far from the concept, but at very least, they are less than ideal.

We measure performance as a *decile run*: distances from the learned concept to all stocks in the test period are calculated, and the test stocks are sorted by those distances. We then calculate the average change in value (average return) of the stocks in each decile (10%) of the test stocks. The top decile (those stocks closest to the learned concept, or the ideal stock) should have high return, and lower deciles should have lower return. The decile run indicates the quality of a classifier without enforcing a particular policy. However, we can assume that some version of the following policy will be used: buy stocks in the top few deciles and sell ones in the bottom few deciles. A good decile run takes the shape of a line running from extremely negative average return at the bottom decile to extremely positive average return at the top decile.

Figure 5-3 shows four decile runs. One is for GMO's predictor, which has been carefully developed over several years. A second one is a decile run of a point-and-scaling concept learned using `maxDD`. A third one is a decile run of a two-disjunct concept learned using `maxDD`; the algorithm attempts to find two ideal stocks, so distance is calculated as the minimum distance to either ideal stock. Finally, we show a decile run of the concept learned by the MULTINST algorithm (discussed in more detail in Chapter 7). Because MULTINST learns a hyper-rectangle, it needed to be slightly modified in order to return a continuous ranking (rather than a binary classification). Given a stock to predict, we calculate the multiplier by which the learned hyper-rectangle needs to be scaled in order to contain that stock. The larger the multiplier, the further the stock is from ideal. Each decile run is actually an average of the seven decile runs on test sets from 1989 through 1995.

As can be seen from the figure, the single or disjunctive concepts found by `maxDD` perform better than GMO's predictor (lower average return in the bottom deciles and higher average return in the top deciles). The main exception to that occurs in

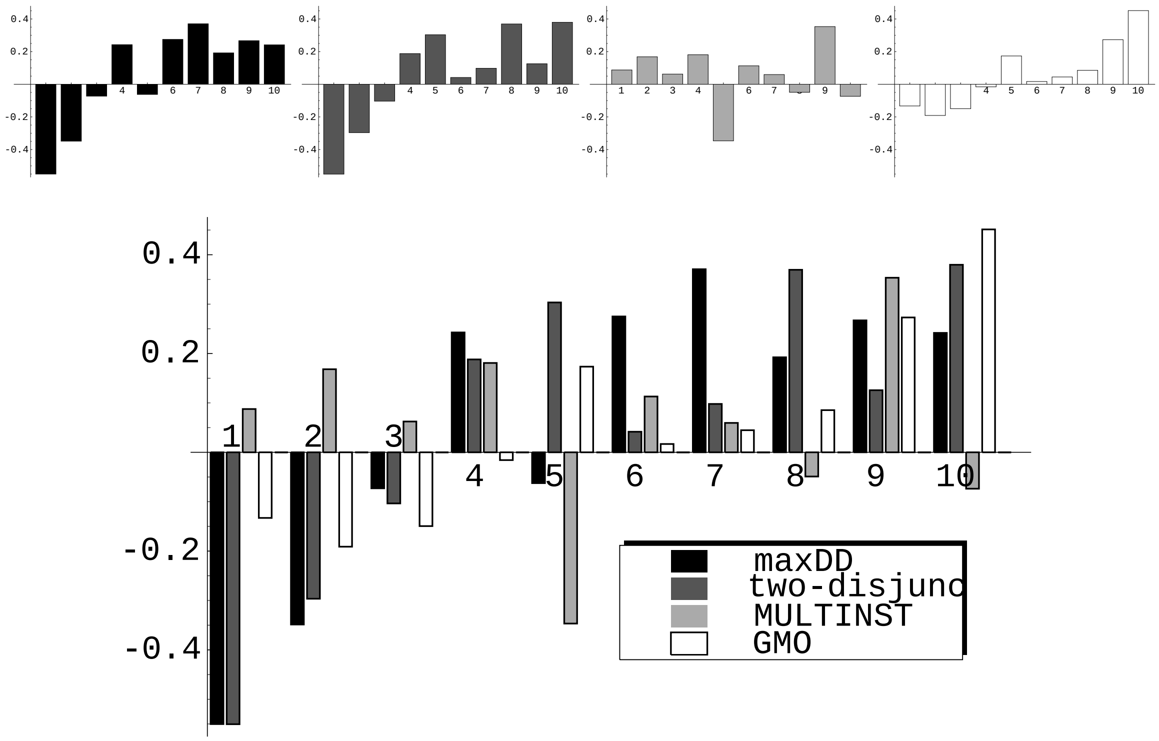
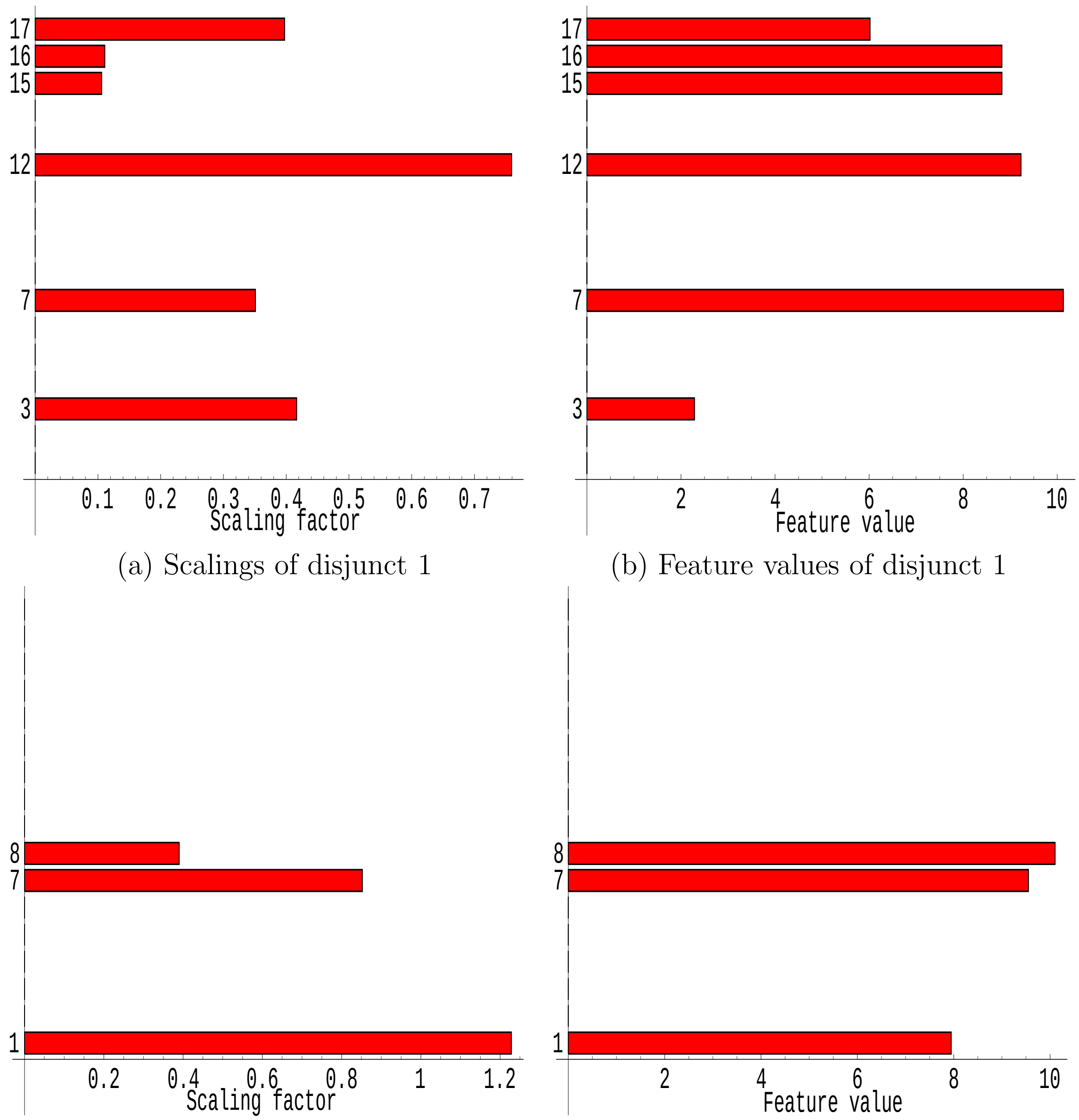


Figure 5-3: For each of four concepts, the decile run is plotted. The black bars indicate the decilized return of a single concept, dark gray bars indicate the decilized return of a two-disjunct concept, light gray bars indicate the decilized return of MULTINST's concept, and the white bars indicate the decilized return of a GMO predictor.

the top decile, where the GMO predictor's ranking of the top 10% of the stocks is more accurate than the ranking generated by distance to the `maxDD` concept. This is only a concern if the desired policy is "buy the stocks in the top decile." However, if the policy is "buy the stocks in the top 5 deciles and sell the ones in the bottom 5 deciles" or some variant thereof, then the concept learned from Multiple-Instance examples should outperform GMO's predictor. Finally, we see that the MULTINST concept generates a decile run which is at best random, and at worst has a negative correlation with return.

Figure 5-4 gives an example of a disjunctive concept learned by `maxDD` on one of the 10-year training partitions. For each disjunct, we show the scalings for each of the 17 features, and also the feature values for the relevant features (those with scaling greater than 0.1). We can see that each disjunct covers a different part of the space and uses different scalings. A new stock will be considered fundamentally good if it is close to either one of the disjuncts.

The values of the features in the data set range from 1.0 to 11.0, and each feature is uniformly distributed in that range. It is interesting to note that most of the feature values learned by the algorithm are not at the extremes of the value range. Instead, they are mostly in the 8–10 and 2–4 range. The majority of high-return stocks lie at the extreme ranges of the features, but so do many low-return stocks. `maxDD` finds a more consistent (though less flashy) type of stock at non-extreme feature values.



(a) Scalings of disjunct 1

(b) Feature values of disjunct 1

(c) Scalings of disjunct 2

(d) Feature values of disjunct 2

Figure 5-4: A disjunctive concept learned by `maxDD` on one of the 10-year training partitions. (a) shows the scale $(s_k^1)^2$ for each feature $k \in \{1, \cdots, 17\}$. The features whose scales is greater than 0.1 are indicated, and their values are given in (b). The second disjunct is shown in (c) — the values of the scales $(s_k^2)^2$ and (d) — the values of the features t_k^2 which have significant scaling values.

Chapter 6

An application to image database retrieval[1]

In the past few years, the growing number of digital image and video libraries has led to the need for flexible, automated content-based image retrieval systems which can efficiently retrieve images from a database that are similar to a user's query. Because the desired set of images can vary greatly, we also want to provide a way for the user to explore and refine the query by having the system bring up examples to be judged by the user.

6.1 Previous work

One of the most popular global techniques for indexing of images is color-histogramming which measures the overall distribution of colors in the image. While histograms are useful because they are relatively insensitive to position and orientation changes, they do not capture the spatial relationships of color regions and thus have limited discriminating power. Many of the existing image-querying systems work on entire images or in user-specified regions by using distribution of color, texture and structural properties. The QBIC system [Flickner *et al.*, 1995] and the Virage system [Bach *et*

[1]This chapter describes joint work with Aparna Lakshmi Ratan [Maron and Lakshmi Ratan, 1998].

al., 1996] are examples of such systems. Some recent systems that try to incorporate some spatial information into their color feature sets include [Smith and Chang, 1996, Huang *et al.*, 1997, Belongie *et al.*, 1998]. Most of these techniques require the user to specify the salient regions in the query image.

All of the methods described above use the absolute color and texture properties of the whole image or fixed regions in the image and some incorporate spatial relations between connected regions having similar color properties. If we consider the domain of natural scene classification, these absolute properties can vary between images of the same class.

Recent work ([Lipson *et al.*, 1997]) in scene classification illustrates that predefined flexible templates that describe the relative color and spatial properties in the image can be used effectively for this task. The flexible templates constructed by Lipson [Lipson *et al.*, 1997] encode the scene classes as a set of image patches and qualitative relationships between those patches. Each image patch has properties in the color and luminance channels. These templates describe the color relationship (relative changes in the R,G,B channels), luminance relationship (relative changes in the luminance channel) and spatial relationship between two image patches. These flexible templates were hand-crafted for a variety of scene classes and could be used to classify natural scenes of fields, waterfalls and snowy mountains efficiently and reliably. For example, the following concept might be designed for the snowy-mountain class: "If the image contains a blue blob which is above a blob with more red, blue, and green (i.e. is white) and which is above a brown blob, then it is a mountain." In this chapter, we would like to *learn* such concepts for natural images given a small set of positive and negative examples.

All of the systems described above require users to specify precisely the salient regions and templates that they want. Minka and Picard [Minka and Picard, 1996] introduced a learning component in their system. They use positive and negative examples to learn which members of a set of image groupings (similarity measures) should be used. The image groupings occur within and across images and are based on color and texture cues. However, their system still requires the user to label various parts of the scene. Our system only receives a label for the entire image and automatically extracts the relevant parts of the scene.

6.2 Image database retrieval as a multiple-instance learning problem

We treat the image database retrieval task as a learning problem. The user has some concept in mind, and supplies positive and negative examples of images that fit or do not fit that concept. After our system learns an approximation of the user's intended concept, we can test the approximation's accuracy by retrieving images from the database that are similar to the concept. If the user approves of these images, we are done. If not, the user can provide more examples.

The learning problem is best treated as a Multiple-Instance learning problem instead of as a supervised learning problem. Each example image given by the user is actually an ambiguous example. We do not know what it is about the image that makes the user label it as positive (as in Figure 1-2). We transform each image into a bag of instances, where each instance represents one way of describing what the image is about. In this chapter, each instance corresponds to one or two subregions of the image. However, subregions are only one way of describing what the image could be about.

The system described in this chapter finds subregions in common between positively labeled images that do not occur in negatively labeled images. Once the concept is learned, images from the database can be classified by whether they contain a subregion which is similar to the concept. In the rest of the chapter, we discuss various ways of generating subregions from an image. We then describe a variety of experiments that use different bag generators, different training schemes, and various concept classes. Our results show that Multiple-Instance learning is successful in this domain, that very simple bag generators suffice, and that interactive training helps.

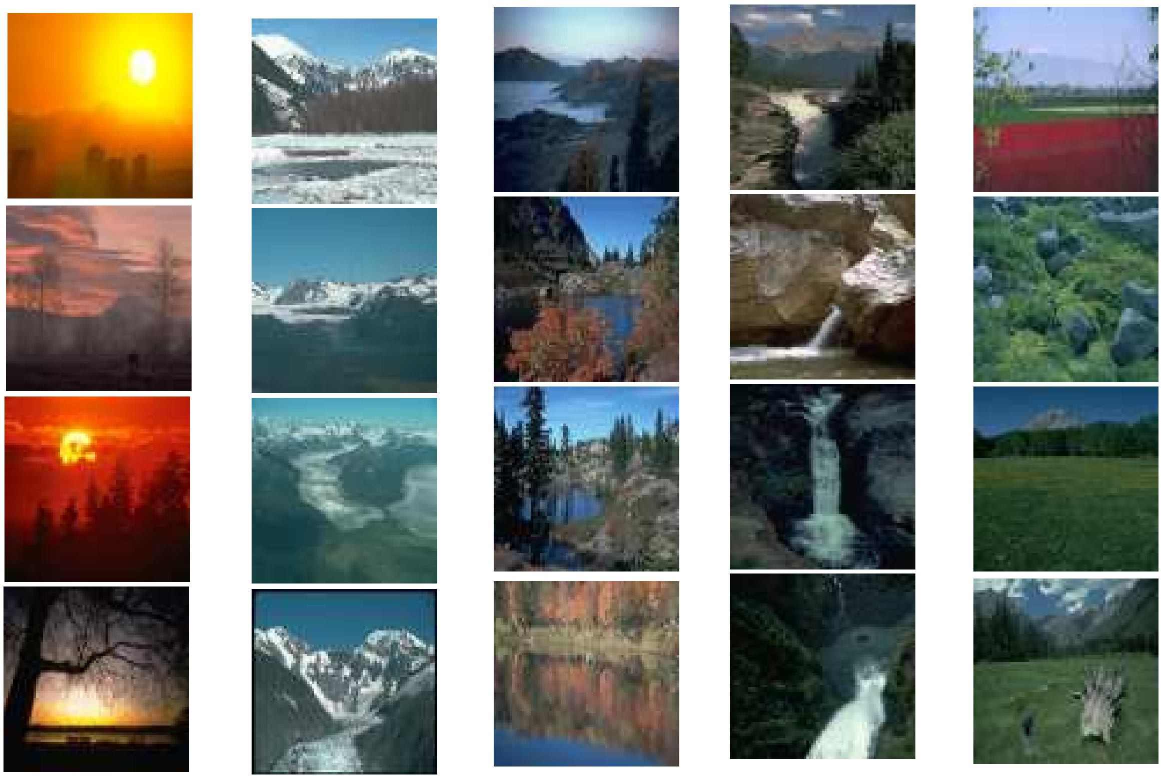


Figure 6-1: Examples of the five image types used from the COREL database.

6.3 Generating a bag from an image

The images that we use are natural scenes such as waterfalls, mountains, fields, lakes, and sunsets. Examples are shown in Figure 6-1. In general, these images do not contain many clearly defined edges. This is a significant difference from many image databases, and allows us to represent images (and subimages) in a very coarse manner. The images were smoothed using a Gaussian filter and subsampled to 8×8.

We experimented with five different bag generators, as described below. Each description details how an instance is represented, and a bag contains all such possible instances. Figure 6-2 schematically describes the types of instances.

row An instance is the row's mean color and the color differences in the rows above and below it. Specifically, an instance j is a vector $\{x_1, \cdots, x_9\}$, where $\{x_1, x_2, x_3\}$ are the mean RGB values of row j, $\{x_4, x_5, x_6\}$ are the mean RGB values of row $j+1$ minus the mean RGB values of row j , and $\{x_7, x_8, x_9\}$ are the mean RGB values of row $j-1$ minus the mean RGB values of row j.

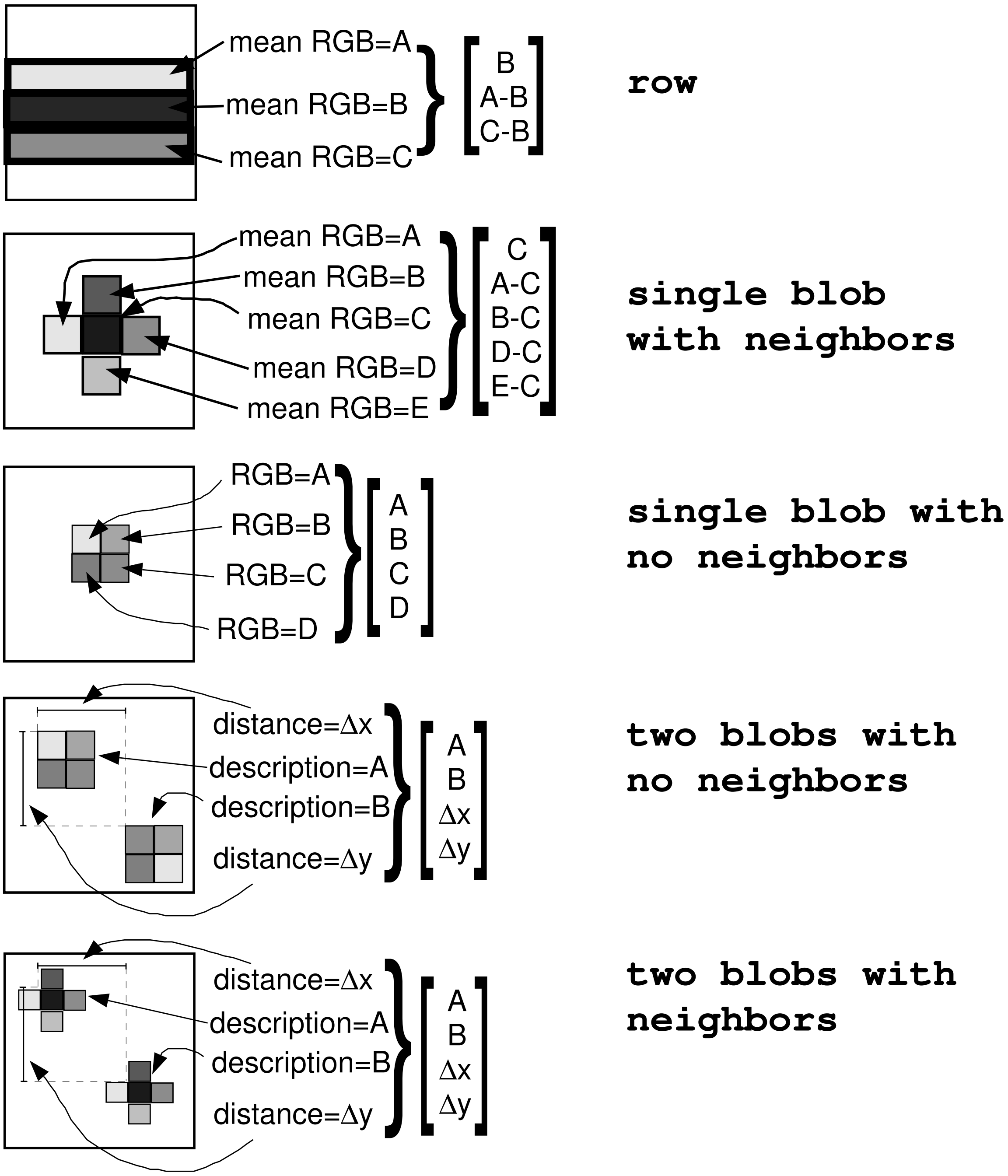


Figure 6-2: Types of instances produced by various bag generators

single blob with neighbors An instance is the mean color of a 2×2 blob and the color differences from its 4 neighboring blobs. Specifically, an instance (i, j) is a vector $\{x_1, \cdots, x_{15}\}$, where $\{x_1, x_2, x_3\}$ are the mean RGB values of the 2×2 blob centered at (i, j), $\{x_4, x_5, x_6\}$ are the mean RGB values of the 2×2 blob centered at $(i + 2, j)$ minus the mean RGB values of the 2×2 blob centered at (i, j), $\{x_7, x_8, x_9\}$ are the mean RGB values of the 2×2 blob centered at $(i, j+2)$ minus the mean RGB values of the 2×2 blob centered at (i, j), $\{x_{10}, x_{11}, x_{12}\}$ are the mean RGB values of the 2×2 blob centered at $(i-2, j)$ minus the mean RGB values of the 2×2 blob centered at (i, j), and $\{x_{13}, x_{14}, x_{15}\}$ are the mean RGB values of the 2×2 blob centered at $(i, j - 2)$ minus the mean RGB values of the 2×2 blob centered at (i, j).

single blob with no neighbors An instance is the color of each of the pixels in a 2×2 blob. Specifically, an instance (i, j) is a vector $\{x_1, \cdots, x_{12}\}$, where $\{x_1, x_2, x_3\}$ are the RGB values of the pixel at (i, j), $\{x_4, x_5, x_6\}$ are the RGB values of the pixel at $(i - 1, j)$, $\{x_7, x_8, x_9\}$ are the RGB values of the pixel at $(i - 1, j - 1)$, and $\{x_{10}, x_{11}, x_{12}\}$ are the RGB values of the pixel at $(i, j - 1)$.

two blob with neighbors An instance is two descriptions of two `single blob with neighbors` and their relative spatial relationship (whether the second blob is above or below, and whether it is to the left or right, of the first blob). Specifically, an instance (i, j, k, l) is a vector $\{x_1, \cdots, x_{32}\}$, where $\{x_1, \cdots, x_{15}\}$ is a `single blob with neighbors` description of instance (i, j), $\{x_{16}, \cdots, x_{31}\}$ is a `single blob with neighbors` description of instance (l, m), $x_{31} = i - k$, and $x_{32} = j - l$.

two blob with no neighbors An instance is the mean color of two descriptions of two `single blob with no neighbors` and their relative spatial relationship. Specifically, an instance (i, j, k, l) is a vector $\{x_1, \cdots, x_{26}\}$, where $\{x_1, \cdots, x_{12}\}$ is a `single blob with no neighbors` description of instance (i, j), $\{x_{13}, \cdots, x_{24}\}$ is a `single blob with no neighbors` description of instance (l, m), $x_{25} = i - k$, and $x_{26} = j - l$.

6.4 Experiments

In this section, we discuss four different types of results from running the system: one is that Multiple-Instance learning is applicable to this domain. A second result is that one does not need very complicated hypothesis classes to learn concepts from the natural image domain. We also compare the performance of various hypotheses, including the global histogram method and a hand-crafted classifier. Finally, we show how user interaction works to improve the classifier.

6.4.1 Experimental setup

We tried to learn three different *image types*: waterfall, mountain, and field. For training and testing we used 500 natural images from the COREL library and the labels given by COREL. These included 100 images from each of the following image types: waterfalls, fields, mountains, sunsets and lakes. Examples of these images are shown in Figure 6-1. We also used a larger test set of 2600 natural images from various image types in the COREL database for additional testing.

We created a *potential training set* of 100 images that consisted of 20 randomly chosen images from each of the five image types. This left us with a *small test set* consisting of the remaining 400 images, 80 from each of the five image types. We separated the potential training set from the testing set to insure that results of using various training schemes and hypothesis classes could be compared fairly. Finally the *large test set* contained 2600 images of nature scenes from a large variety of classes.

For a given image type, we created an *initial training set* by picking five positive examples of the concept and five negative examples, all from the potential training set. A bag generator and concept class were chosen, and the concept was learned using `maxDD`. After the concept was learned from these examples, the unused 90 images in the potential training set were sorted by distance from the learned concept[2]. We did not use all 100 images to train because a typical user of the system would not be

[2] An image/bag's distance from the concept is the minimum distance of any of the image's subregions/instances from the concept.

expected to give 100 examples.

This sorted list can be used to simulate what a user would select as further refining examples. Specifically, the most egregious false positives (the non-concept images at the beginning of the sorted list) and the most egregious false negatives (the concept images at the end of the sorted list) would likely be picked by the user as additional negative and positive examples. For example, if the image type is waterfalls, then non-waterfall images that are close to the learned concept would be considered false positives. Waterfall images that are far from the learned concept would be considered false negatives.

We used four different *training schemes*: `initial` is simply using the initial five positives and five negative examples. `+5fp` adds the five most egregious false positives. `+10fp` repeats the `+5fp` scheme twice. `+3fp+2fn` adds 3 false positives and 2 false negatives.

Because we did not know which of the bag generator's features were relevant, we used the point-and-scaling concept class for each of the five concept types shown in Figure 6-2. We also tried learning 2-disjunct concepts for the `single blob with neighbors` and `single blob with no neighbors` bag generators. This means that a total of 7 different *concept types* could be learned for any given data set.

6.4.2 Precision and recall graphs

A standard way of displaying results in image and text database retrieval is through precision and recall graphs. Let us suppose that there are N objects in our database, G of which we want to retrieve because they are in the correct class. In addition, let us suppose that the classifier we have learned has ranked all N objects, hopefully putting objects of the correct class at the front of the list, and objects that do not belong to the correct class at the end. Finally, let us call $G(n)$ the number of objects in the correct class out of the first n objects in the ranked list. Note that $G(0) = 0$ and $G(N) = G$.

Precision is defined as the ratio of the number of correct objects to the number of objects retrieved: $G(n)/n$. Recall is defined as the ratio of the number of correct objects retrieved to the total number of correct objects in the test set: $G(n)/G$. Plotting precision against n is known as a precision graph, plotting recall against n is known as a recall graph, and plotting precision against recall is known as a precision-recall graph.

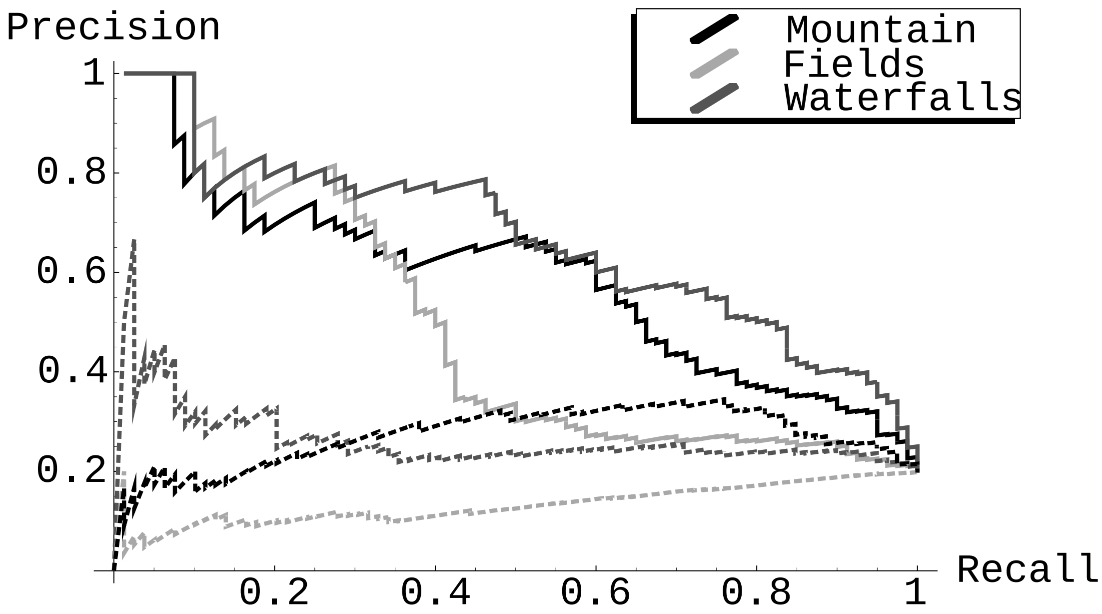
For example, in Figure 6-3, the waterfall precision-recall curve has recall 0.5 with precision of about 0.7, which means that in order to retrieve 40 of the 80 waterfalls, 30% of the images retrieved are not waterfalls. We show both precision-recall and recall curves for the following reasons. The beginning of the precision-recall is of interest to applications where only the top few objects are of importance. On the other hand, the middle of the recall curve is of interest to applications where correct classification of a large percentage of the database is important.

6.4.3 Results

In this section we show results of testing the various concept types, training schemes, and image types against the small test set and the larger one. The small test set does not intersect the potential training set, and therefore is an accurate measurement the generalization of the learned concepts. The large test set does intersect with the potential training set, but is meant to show how the system scales to larger, more varied image databases. We ran our tests under every combination of four training schemes, seven concept types, and three image types.

Learning a concept took anywhere from a few seconds to learn a `row` concept to a few days for the disjunctive concept classes. The more complicated hypotheses take longer to learn because there is a higher number of features and the number of instances per bag is large. In addition, searching for the best concept involves starting a gradient based optimization at every pair of instances (for the disjunctive concept class). Because this is a prototype, we did not try to optimize the running time; however, a more intelligent method for generating instances (for example, a rough segmentation using connected components) will reduce both the number of instances and the running time by orders of magnitude.

In Figure 6-3, we show the performance of the best concept and training method



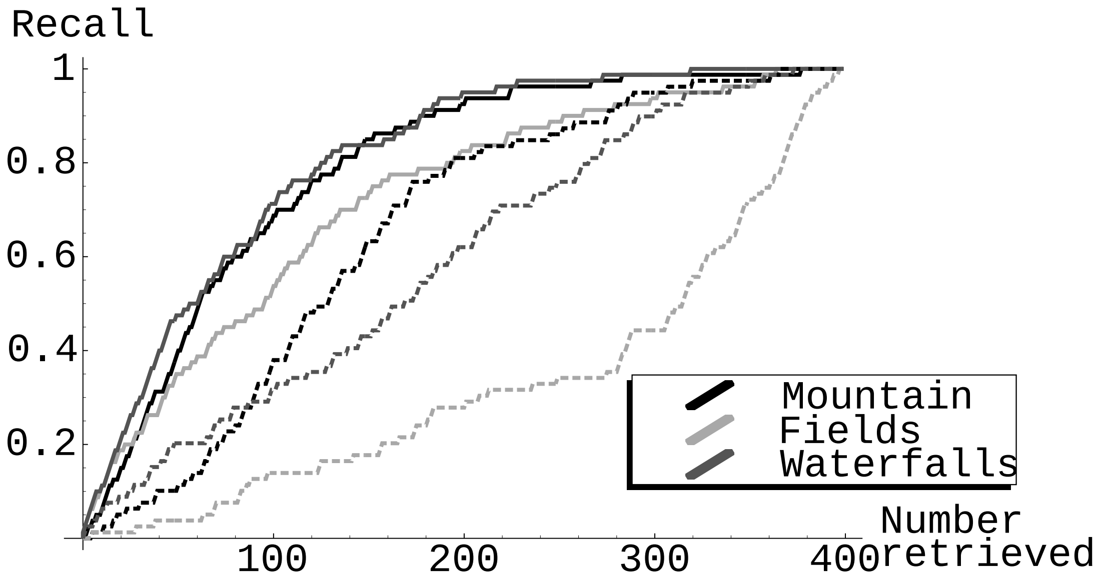
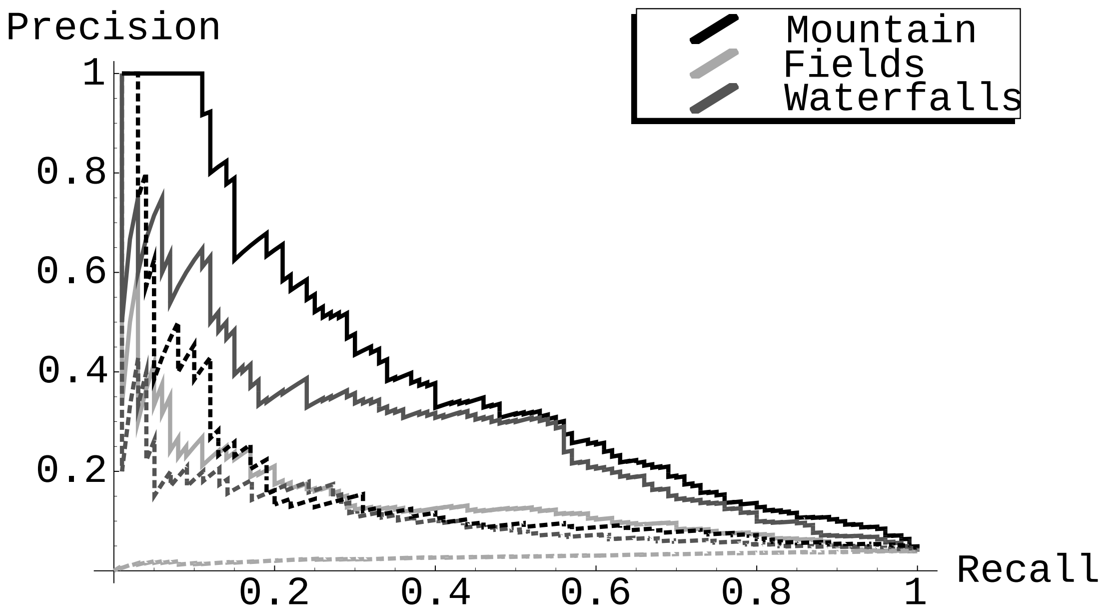


Figure 6-3: The best curves for each image type. Dashed curves are the global histogram's performance, and solid curves show the learned concept's performance. Both use a small test set of 400 images.



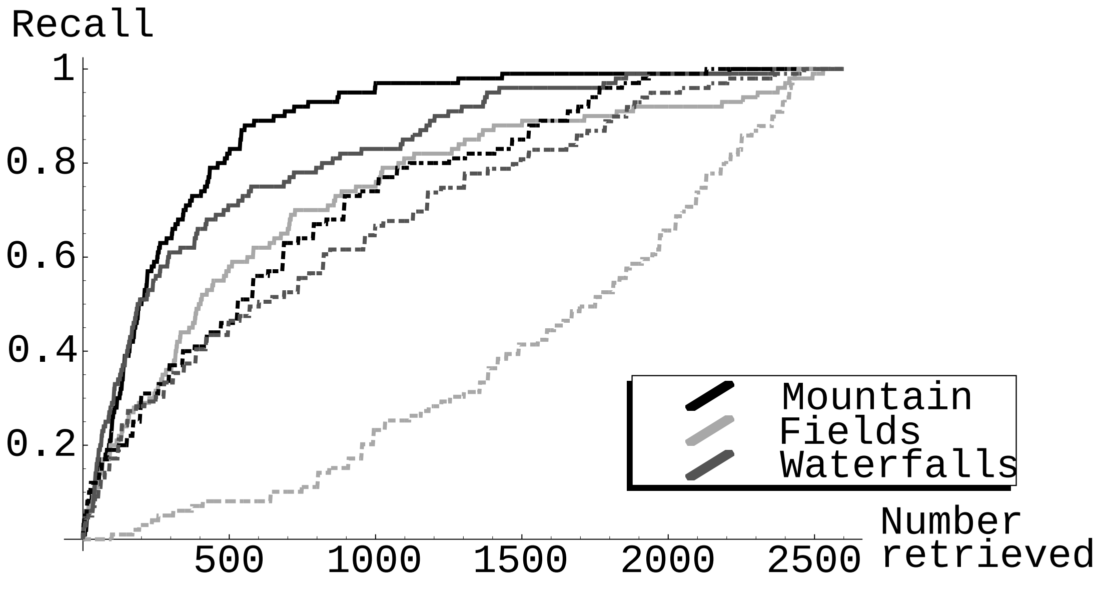


Figure 6-4: The best curves for each image type. Dashed curves are the global histogram's performance, and solid curves show the learned concept's performance. Both use a large test set of 2600 images.

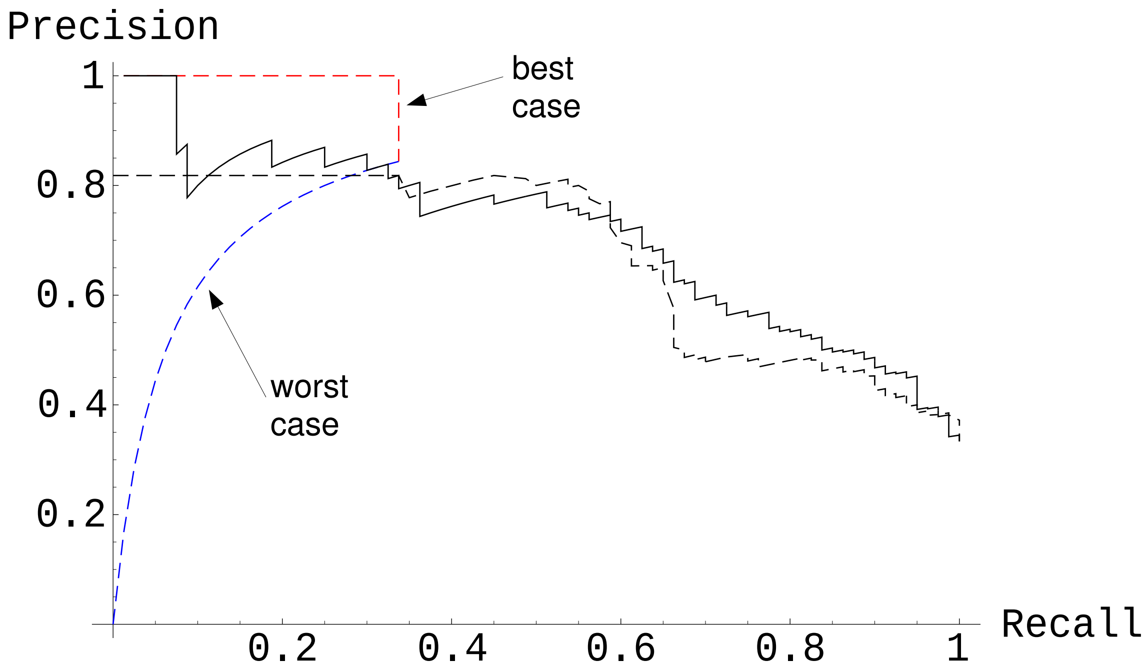
on each image type. The dashed lines show the performance of the global histogram method. The solid lines in the precision-recall graph show the performance of `single blob with neighbors` with `+10fp` for waterfalls, `row` with `+10fp` for fields, and `disjunctive blob with no neighbors` with `+10fp` for mountains. The solid lines in the recall curve show the performance of the `single blob with neighbors` with `+10fp` for waterfalls, `single blob with neighbors` with `+3fp+2fn` for fields, and `row` with `+3fp+2fn` for mountains. Figure 6-4 shows the behavior of the same concepts on the larger database.

As can be seen from these graphs, the global histogram method performs extremely poorly on this data, generating a ranking of the data which is usually random. The learned concepts perform well, and for the most part generalize well to larger databases.

The global histogram method is the straw man of the image database retrieval community. A more worthy adversary for a concept learned from examples is one which was hand-crafted based on domain knowledge of a particular image type. Figure 6-5 shows that the performance of the learned mountain concept is competitive with a hand-crafted mountain template from [Lipson *et al.*, 1997]. Lipson's classifier was modified to give a ranking of each image, rather than its class. The test set consists of 80 mountains, 80 fields, and 80 waterfalls. It is disjoint from the training set.

The learned concept is shown as a solid curve, while the hand-crafted concept is shown as a dashed curve. The hand-crafted model's precision-recall curve is flat at approximately 84% because the first 32 images all receive the same score, and 27 of them are mountains. We also show the curves if we were to retrieve the 27 mountains first (best case) or after the first five false positives (worst case). The learned concept is impressively competitive with the hand-crafted classifier.

In addition to finding specific classifiers, we can also use our experiments to compare the various training schemes and the various concept types. In Figure 6-6, we show the recall and precision-recall curves for each of the four training schemes. Each curve is the result of averaging over all three image types and all seven concept types.



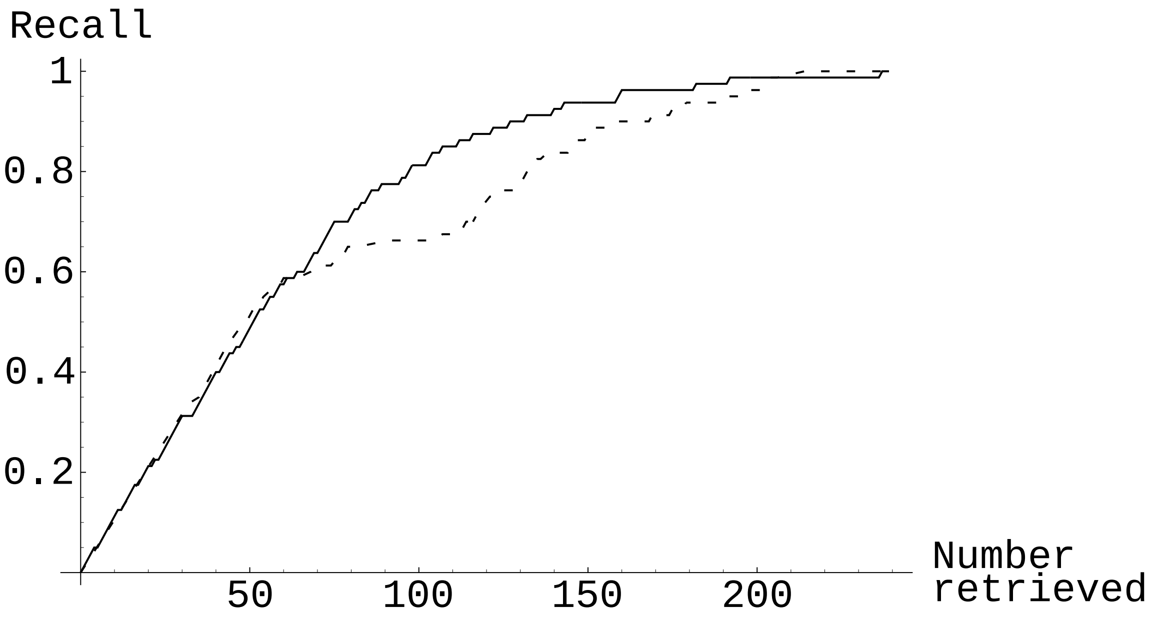


Figure 6-5: Comparison of learned concept (solid curves) with hand-crafted templates (dashed curves) for the mountain concept on 240 images from the small test set. The top and bottom dashed precision-recall curves indicate the best case and worst case curves for the first 32 images retrieved by the hand-crafted template which all have the same score.

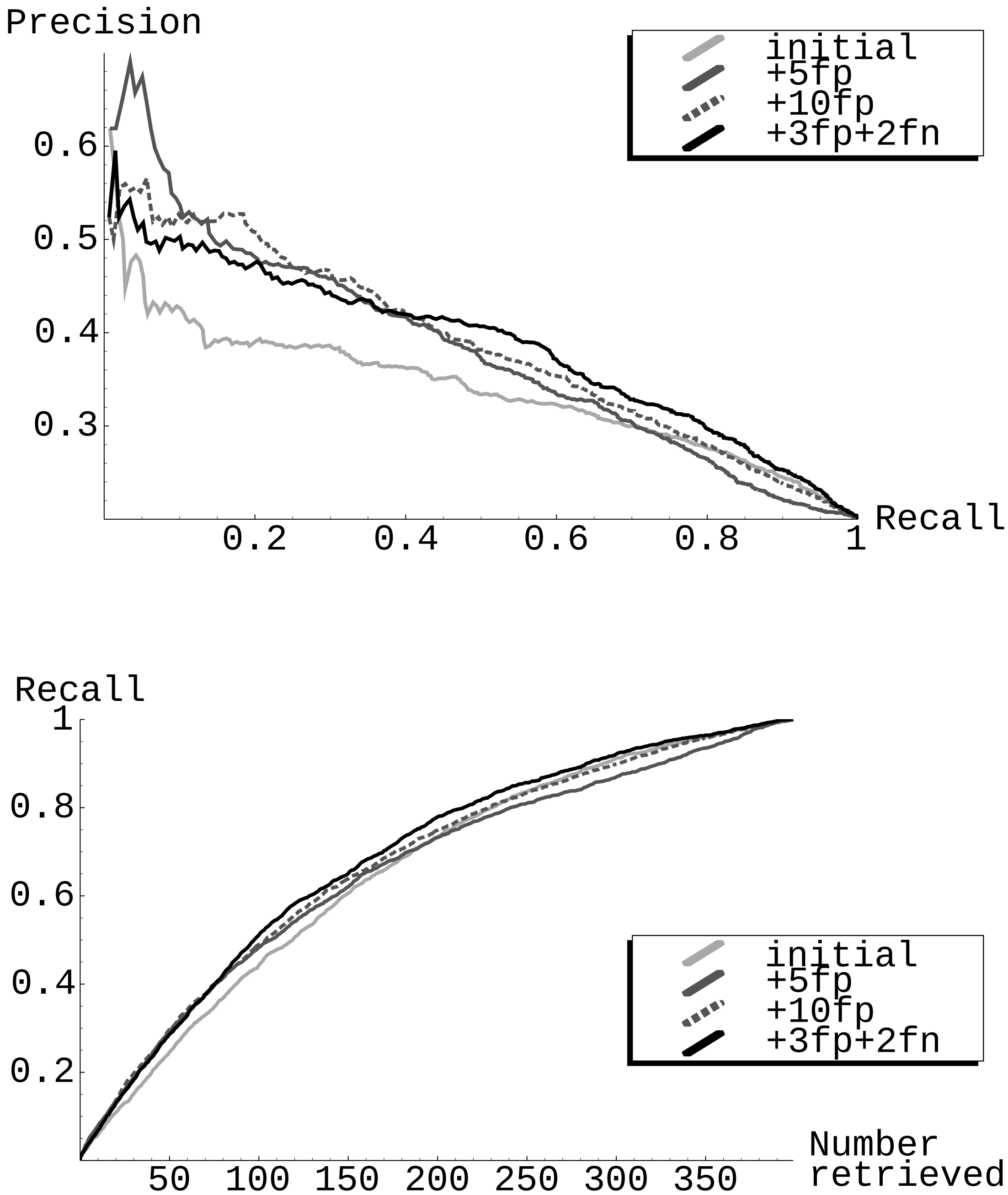
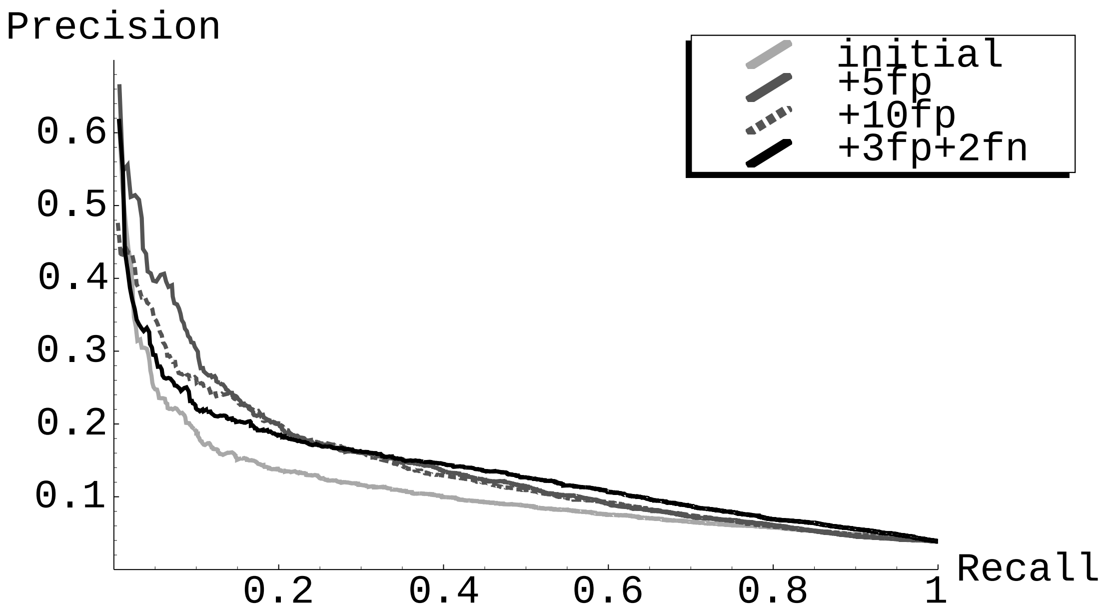


Figure 6-6: Different training schemes, averaged over image type and concept type. A small test set is used.



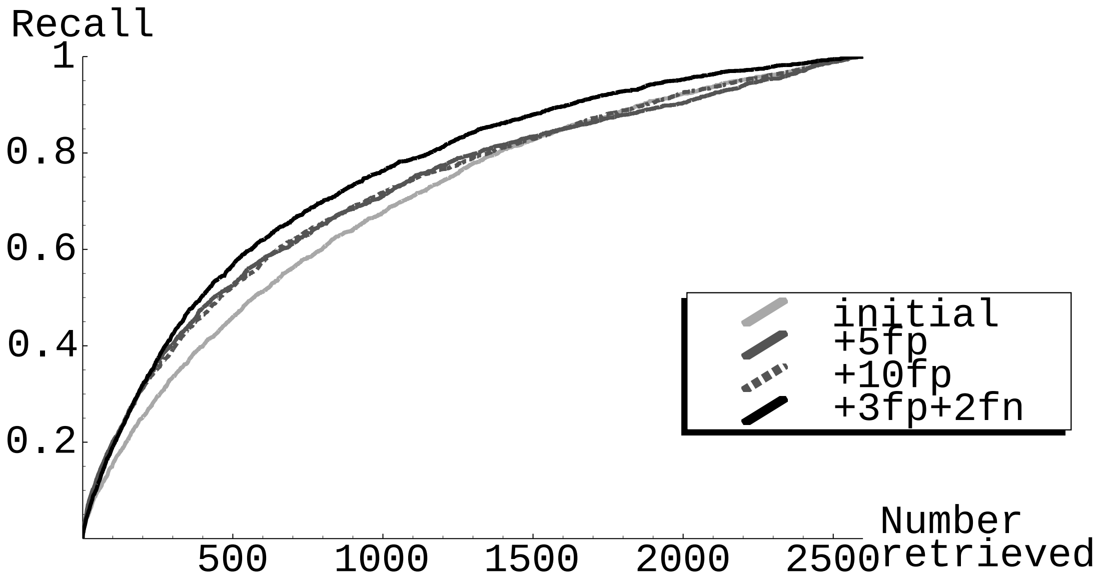


Figure 6-7: Different training schemes, averaged over image type and concept type. A large test set is used.

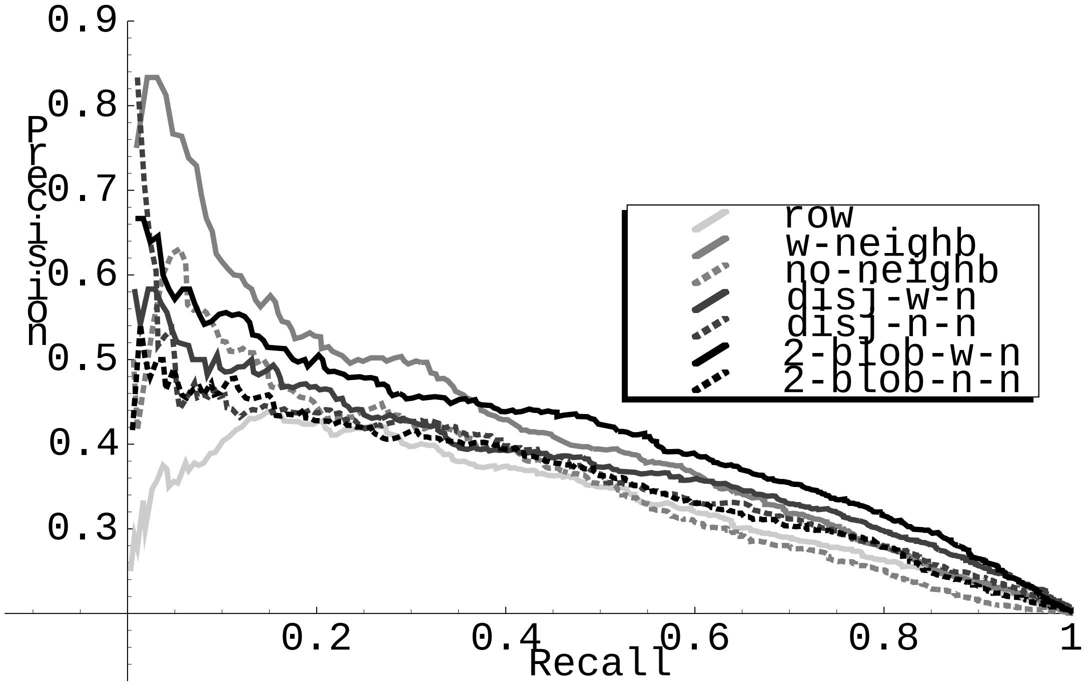
We see that performance improves with user interaction. Any training scheme where there is an iteration of feedback from the user results in markedly better performance. This behavior continues for the larger test set as well, shown in Figure 6-7.

In Figure 6-8, we show the precision-recall and recall curves for each of the seven concept types averaged over all image types and all training schemes on the small test set. Figure 6-9 shows the same curves for the large test set. We see that the single blob with neighbors hypothesis has good precision. We also see that the more complicated hypothesis classes (i.e. the disjunctive concepts and the two-blob concepts) tend to have better recall curves.

In Figure 6-10, we show a snapshot of the system in action. The system is trained using training scheme `+10fp` for the waterfall image type. It has learned a waterfall concept using the `single blob with neighbors` bag generator. The learned waterfall concept is that somewhere in the image there is a blob

- whose left neighbor is less blue, and
- whose own blue value is 0.5 (where RGB values are in the $[0, 1]^3$ cube), and
- whose neighbor below has the same blue value, and
- whose neighbor above has the same red value, and
- whose green value is 0.55, and
- whose neighbor above has the same blue value, and
- whose red value is 0.47.

These properties are weighted in the order given, and any other features were found to be irrelevant. To rate an image from the database, we assign it the minimum distance of one of its instances to the learned concept, where the distance metric uses the learned scaling to account for the importance of the relevant features. As we can see in the figure, this simple learned concept is able to retrieve a wide variety of waterfall scenes.



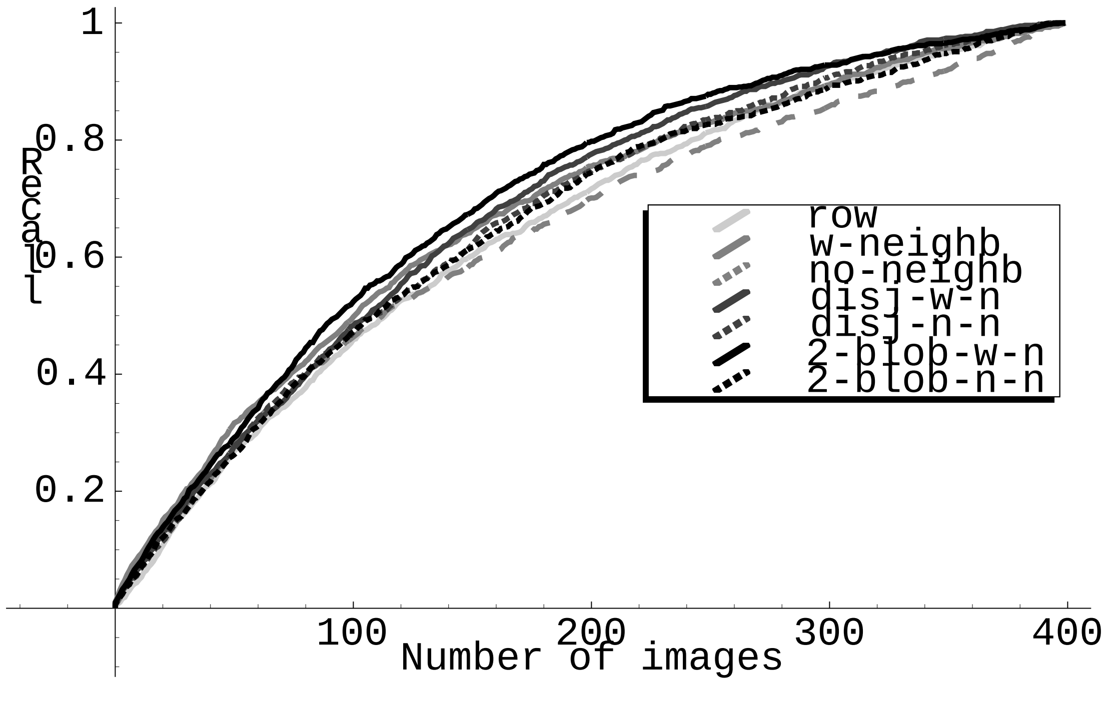


Figure 6-8: Different concept types averaged over image type and training scheme, using a small test set.

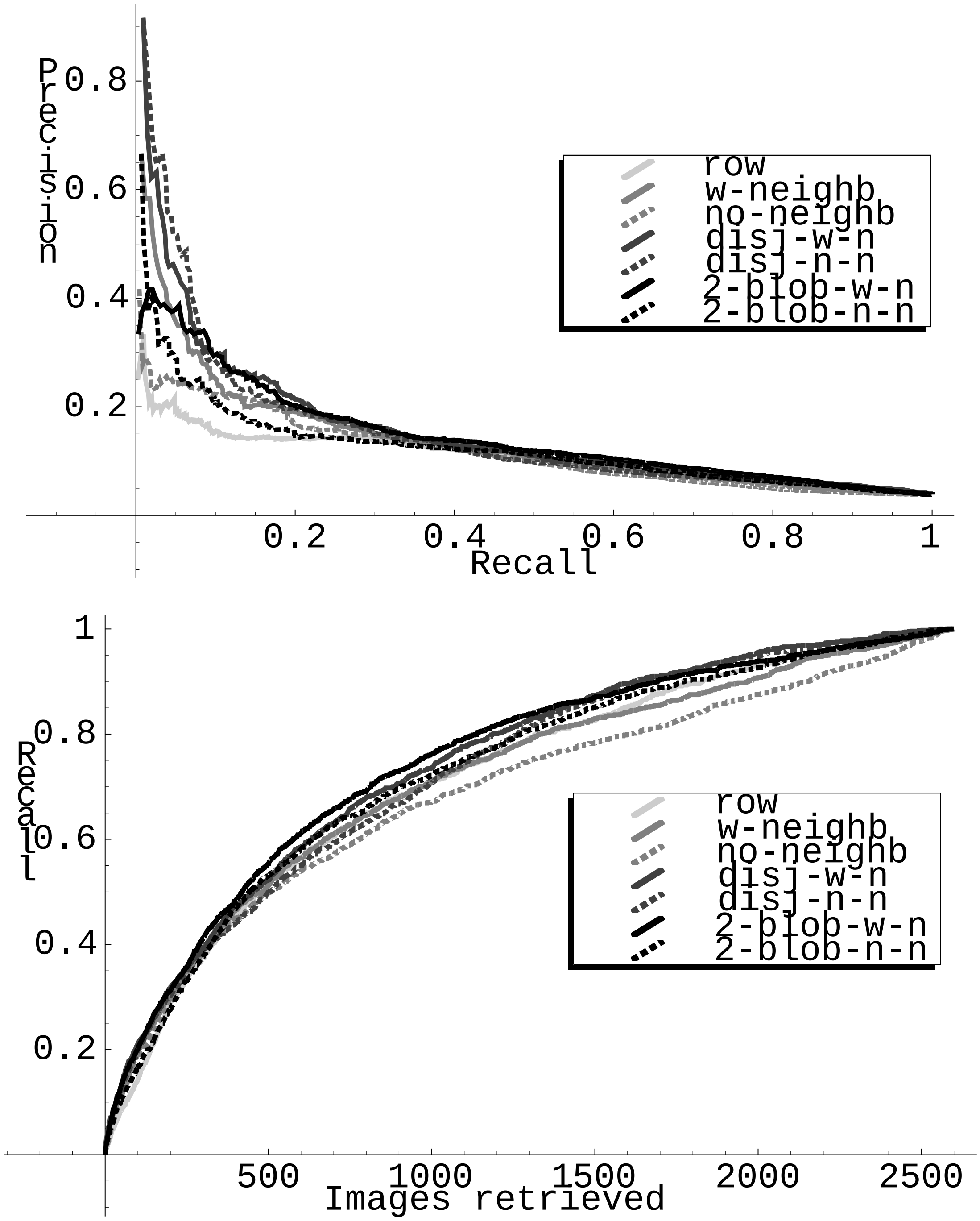


Figure 6-9: Different concept types averaged over image type and training scheme, using a large test set with 2600 images.

Figure 6-10: Results for the waterfall concept using the `single blob with neighbors` concept with `+10fp`. Top row: Initial training set–5 positive and 5 negative examples. Second Row: Additional false positives. Last three rows: Top 30 matches retrieved from the large test set. The red squares indicate where the closest instance to the learned concept is located.

The top 20 images in the figure are the training set. The first 10 images are the initial positive and negative examples used in training. The next 10 images are the false positives added. The last 30 images are images from the large dataset which are closest to the learned concept.

6.5 Conclusions

We can draw several conclusions from the experiments in this chapter.

- Multiple-Instance learning by maximizing Diverse Density can be used to learn a classifier for images of natural scenes. The classifier's performance is competitive with hand-crafted models, and much better than a global histogram approach. Supervised learning approaches would be hard pressed to match `maxDD`'s results on as few training examples.

- Extremely simple concepts that capture color relations in low resolution images can be used effectively in the domain of natural scene classification.
- User interaction (adding false positives and false negatives) over multiple iterations can improve the performance of the classifier.

The architecture of separating the bag generation mechanism from the learning mechanism is conducive to future extensions of this work. A better bag generator (e.g., one that generates coherent subregions of various sizes rather than in a cookie-cutter fashion) will lead to easier learning, since the number of instances per bag will decrease. Likewise, a better learning mechanism (e.g., one that can handle more instances per bag) will allow us to experiment with more general bag generators. This architecture also suggests a change to the underlying goal of Machine Vision. Instead of attempting to identify the object(s) in the image, its goal should now be to identify a set of possible objects, trying to minimize the size of the set while at the same time maximizing the probability that one of the objects is the right one.

Chapter 7

Related work

Multiple-Instance learning is a new framework for learning, so the number of directly related papers is small. We examine the short history of the area and discuss each of the papers published to date. The more general idea of explicitly dealing with ambiguous examples has a longer history, and in the second section of this chapter we look at other approaches to learning from ambiguity.

7.1 Previous work on Multiple-Instance learning

The area of learning from multiple-instance examples has only recently begun to be fully explored. The first paper to state and name the Multiple-Instance framework was [Dietterich *et al.*, 1997]. This paper was concerned with developing an algorithm to tackle the drug discovery problem. Our setup of the drug discovery problem, namely the ray representation of molecular shape, is taken from that paper. Dietterich et al. assumed that the underlying concept (the shape to be learned) can be described with an axis-parallel rectangle (APR). The algorithms that they developed can be thought of as greedily maximizing a discrete version of Diverse Density. In their algorithms, a hyper-rectangle is shrunk and expanded to maximize the number of positive points from different bags that are contained within the rectangle while minimizing the number of negative points within the rectangle. The algorithms were tested on the musk data set, and results are shown in Chapter 4.

To give a specific example of one of their algorithms, we summarize the behavior of their best algorithm: `iterated-discrim APR`. It involves three procedures — **grow**, **discrim**, and **expand**. The **grow** procedure starts with seed APR that covers only one instance from a positive bag. It is grown until the smallest APR that covers at least one instance from every positive bag is found. The **discrim** procedure takes as input the APR generated by the **grow** procedure, and selects those features that are most important in discriminating negative from positive instances. The **grow** procedure is then repeated, this time building an APR only using the features selected by **discrim**. The algorithm iterates between **grow** and **discrim** several times. Finally, the **expand** procedure is applied on the resulting APR. It estimates the density of instances along each feature of the APR, and expands the bounds of the APR so that the estimated probability of excluding a positive instance is ϵ.

Although Multiple-Instance problems have been encountered before (for example, in Meta-Dendral [Buchanan and Mitchell, 1978]), they were transformed into a traditional supervised learning problem. Dietterich et al.'s paper is important because it was the first to give an algorithm specifically for learning from Multiple-Instance examples, and also because it achieved impressive results on the musk data set. However, because the algorithm was designed with the drug discovery problem in mind, it is not clear that it will generalize well to other problems. In fact, it likely represents an upper bound on performance for the musk data set because some of the parameters of the algorithm were trained on MUSK1 and tested on MUSK2.

Because Multiple-Instance learning is a neatly stated problem, it soon attracted the attention of theoreticians. Long and Tan [Long and Tan, 1996] showed that it is possible to PAC-learn [Valiant, 1984] an axis-parallel concept from Multiple-Instance examples, but gave an algorithm with a very high bound on its running time. Specifically, for their algorithm to return a concept that has error ϵ with probability $1-\delta$, it would need $O(\frac{d^5 n^{12}}{\epsilon^{20}} \log^2 \frac{nd}{\epsilon\delta})$ time (where d is the number of dimensions and n is the number of points per bag). In addition, the d dimensions were assumed to be independent. Even worse, Long and Tan assumed that each instance was generated independently, independent of its bag. This assumption clearly does not hold for any

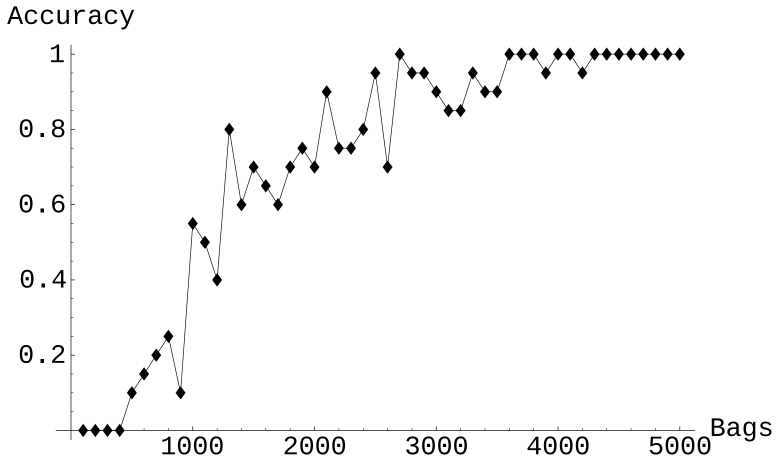


Figure 7-1: The accuracy of MULTINST on the artificial dataset of Section 2.6.2. Each bag contains 50 instances, and each accuracy measurement is based on 20 randomly generated datasets.

reasonable application of Multiple-Instance learning: the shape of a molecule does depend on which molecule generated it, and the same stocks do not go up every month.

Auer, Long, and their colleagues soon published two interesting developments. [Auer, 1997] gave a new algorithm that improved on Long and Tan's bounds and did not require the features to be independent. The MULTINST algorithm learned an axis-parallel rectangle that had error of ϵ with probability $1 - \delta$ using $O(\frac{d^2n^2}{\epsilon^2} \log \frac{d}{\delta})$ examples (bags) in time $O(dl \log l)$ where l is the number of examples. A simplified overview of the algorithm is given in Section 7.1.1. As shown in Chapter 4, MULTINST performed remarkably well on the MUSK datasets. However, as shown in Figure 5-3, it had less success in the stock prediction domain. The artificial dataset described in Section 2.6.2 actually obeys the assumptions made by MULTINST, since every instance is generated independently. It is also an example of the worst-case behavior of the algorithm. Figure 7-1 shows the percentage of times MULTINST learns the correct concept (the 5×5 square) as the number of training bags grows. Each bag contains 50 instances. The percentage of success was calculated from 20 runs on randomly generated datasets. When compared to Figure 2-8, we can see that the number of examples needed by MULTINST are two orders of magnitude greater than the number of examples needed by `maxDD`. In addition, there are 200 instances per bag in the experiments of Figure 2-8, making it a harder task.

The other development was less fortunate: [Auer *et al.*, 1996] showed that Long and Tan's independence assumption is necessary for any efficiently PAC-learnable algorithm. Specifically, they showed that if there is an algorithm to efficiently PAC-learn from multiple-instance examples that are distributed arbitrarily, then it is also possible to efficiently PAC-learn DNF formulas. It is generally assumed that PAC-learning DNF formulas is hard.

[Blum and Kalai, 1998] affirm Auer's proofs. In addition, they show that learning from multiple-instance examples can be reduced to learning with one-sided noise, two-sided noise, and learning in the statistical query model [Kearns and Vazirani, 1994]. They also slightly improve Auer et al.'s sample bounds to $\tilde{O}(\frac{d^2 n}{\epsilon^2})$[1].

To summarize, despite the youth of the field, there have been a number of successful applications and a number of strong theoretical results. However, there is still a large gap between the theoretical claims that learning from multiple-instance examples is hard and the empirical successes of multiple-instance algorithms. In the worst case, `maxDD` would not find the global Diverse Density maximum over concept space in polynomial time. However, in most cases examined in this thesis, `maxDD` found the best concept or one which performed well.

7.1.1 Overview of the MULTINST algorithm

We describe a simplified version of the MULTINST algorithm, based on [Auer *et al.*, 1996]. We assume there is a distribution D over a d-dimensional feature space. Every instance is drawn independently from D. Each contains exactly r instances, and is labeled positive if at least one of the instances falls within the target APR. The target APR is denoted BOX $= \prod_{1 \le k \le d} [a_i, b_i]$, where $\{a_i\}$ is the "bottom left" corner and $\{b_i\}$ is the "top right" corner.

Without loss of generality, we will only try to learn the "top right" corner of the

[1] $\tilde{O}(\cdot)$ is the same as $O(\cdot)$ except it ignores the log terms.

concept. We want to find an approximation $\hat{b}$ to BOX, so that the chance of a true positive instance falling outside $\hat{b}$ is small. For every feature k, we want to find $\hat{b}_k \leq b_k$ such that $\Pr(B_{ij} \in BOX$ and $B_{ijk} > \hat{b}_k)$ is small. We define the function $\beta_k(x)$ to be the probability (over distribution D) that an instance B_{ij} is a true positive (falls within BOX) and $B_{ijk} > x$.

We show that it is possible to compute $\beta_k(x)$ from quantities that can be approximated from the training data. We define the following quantities, where q and $\alpha_k(x)$ are probabilities over distribution D, and $\phi_k(x)$ is defined over distribution D^r.

$$
\begin{aligned}
q &= \Pr(B_{ij} \notin BOX) \\
\alpha_k(x) &= \Pr(B_{ijk} > x) \\
\phi_k(x) &= \Pr(B_i \text{ is negative and } B_{i1k} > x) \\
&= (\alpha_k(x) - \beta_k(x)) \cdot q^{r-1}
\end{aligned}
$$

The last equation can be interpreted as follows: the first term is the probability that instance B_{i1} is negative (does not fall in BOX) and has $B_{i1k} > x$. The second term is the probability that the other $r-1$ instances $(B_{i2}, \cdots, B_{ir})$ are negative. Note that all of these quantities (except $\beta_k(x)$) can be approximated from the data. Therefore, we can estimate

$$\beta_k(x) = \alpha_k(x) - \frac{\phi_k(x)}{q^{r-1}} \tag{7.1}$$

By definition, $\beta_k(x)$ is 0 when $x \leq b_k$. If we find the largest X such that $\beta_k(X)$ is 0, then $\hat{b}_k = X$.

7.2 Previous work on ambiguous examples

As stated in Chapter 1, there is a spectrum of ambiguity in the field of Machine Learning, ranging from supervised learning which has no ambiguity to unsupervised learning in which no example is labeled. Multiple-Instance learning falls in the middle of the spectrum, as does Reinforcement Learning [Sutton and Barto, 1998, Kaelbling *et al.*, 1996]. In Reinforcement Learning, the teacher (or the world) provides an occasional reinforcement signal, but the specific value of being at some state (or which action to take) is usually not given. In this section, we discuss other points along the spectrum, some of which involve ambiguity about a collection of examples (bags), and some which involve noisy labels on individual examples:

- Learning from noisy examples
- Norton's modeling of uncertainty and Hirsh's bounded inconsistency
- Time Delay Neural Network (TDNN), and Integrated Segmentation and Recognition (ISR)
- Dynamic reposing
- Learning from Automata
- Meta-DENDRAL

Learning from noisy examples is perhaps the closest point in the spectrum to the extremum of perfect supervised learning. The noise is added either to the value of the label or to the features used to describe the example. Normally, it is assumed that the noise is Gaussian. However, models where the noise is brought about by a malicious adversary have also been explored [Decatur, 1994]. Because the noise is usually Gaussian, most algorithms assume that with enough examples, the noise can be washed out. Other algorithms take a more proactive approach; for example, k-Nearest-Neighbor and other such variants [Atkeson *et al.*, 1997] smooth out noise through voting schemes. [Kearns and Vazirani, 1994] show that PAC-learning from noisy examples can be reduced to the standard PAC model.

[Norton, 1994] shows that ambiguity can arise not only from an uncertain example, but also from an uncertain representation. Norton attempts to learn DNA promoter sequences from examples. However, there is no such thing as a universal description of a promoter sequence. Therefore, each example agrees with many different structures. Norton uses background knowledge to assign probabilities to structures, combines the evidence from the training example, and picks the most likely concept.

Norton's work is an outgrowth of Hirsh's bounded inconsistency learner [Hirsh, 1990]. The bounded inconsistency model assumes that every example is made up of some true source example corrupted by a noise model. In addition, given a noisy example, one can efficiently describe the version space [Mitchell, 1978] that covers all concepts that agree with the possible true sources of that example. Each example is therefore represented by an "expanded" version space, and the intersection of all the version spaces contains concepts that are consistent with the training data. This is similar to the idea behind Diverse Density, where we find the intersection of the positive bags.

A Time Delay Neural Network [Waibel *et al.*, 1989, Lang, 1989] is an architecture that allows the network to find time-invariant patterns in the examples. Its original motivation was to build a classifier to distinguish spoken letters. Each example is labeled, but it is not known when the speaker begins uttering the letter and for how long the enunciation lasts. If these details were known about each example, then the learning task would be simple.

An extension of TDNN is the Integrated Segmentation and Recognition system [Keeler *et al.*, 1991a]. The ISR network is given an image with one or more handwritten characters, a label that identifies the characters that appear, but not the pose or location of the characters. This is similar to the image database retrieval problem addressed in Chapter 6. There are two main differences between our approach and the ISR system. First, they assume that there is only one occurrence of each type of character in the image, whereas there is no such limitation in the Multiple-Instance learning framework. This assumption leads to their method of combining evidence through a summation of likelihood ratios (as discussed in Section 2.3.2). In [Keeler *et al.*, 1991b], they show that the assumption can be relaxed by using noisy-or to combine evidence (as in Chapter 2 and [Saund, 1995]). In a different context, [Dayan and Zemel, 1995] show that both summation of likelihood ratios and noisy-or can be suitable models of combining evidence. The other main difference is that the ISR architecture forces the bag generator to be part of the learning mechanism of the network, while we construct the learner and bag generator separately.

Dynamic reposing [Jain *et al.*, 1994, Dietterich *et al.*, 1994] is an algorithm for handling examples that are ambiguous both in pose (like TDNN and ISR) and in containing multiple instances. It was applied to drug discovery problems as described in Chapter 4. The algorithm alternates between optimizing for the best pose and finding the true positive in each bag. As would be expected, local maxima present a major difficulty for the algorithm.

A classic Machine Learning problem is learning a Deterministic Finite-State Automata (DFA) from a series of strings which are labeled according to whether they are accepted by the DFA or not. [Cohen, 1996] examines the dual of this problem: given a sequence of DFAs, each labeled positive or negative, find a string that is accepted by positive examples and not by negative ones. This is an ambiguous learning problem because a labeled example (DFA) represents a variety of different strings. Unfortunately, Cohen shows that it is hard to PAC-learn from these examples, even if the alphabet size is limited to three characters.

The Meta-DENDRAL program [Buchanan and Mitchell, 1978] receives training pairs of molecules and their mass-abundance curve. Its goal is to learn how molecules disintegrate during mass-spectrometry — which bonds are broken and which atoms migrate. Since every location along the mass-abundance curve can be explained by many different hypotheses, the examples are inherently ambiguous. Meta-DENDRAL generates many possible interpretations for each peak in the curve, and attempts to learn production rules that cover as many interpretations as possible.

In summary, there have been a number of applications where highly ambiguous examples have been encountered and a number of different algorithms have been proposed to handle the ambiguity. However, there has been no unified framework for learning from ambiguous examples.

Chapter 8

Conclusion

8.1 Future Work

There are many exciting avenues for extending the work presented in this thesis. We list some of them, including general future research directions, algorithmic enhancements, and extending specific applications.

- Figure 1-1 shows some of the approaches taken to learn from ambiguity. We are only at the beginning of the process of categorizing the large variety of ambiguity that occurs in learning problems and creating algorithms to learn from ambiguous data. Multiple-Instance learning handles one type of ambiguity, Reinforcement Learning another, and unsupervised learning yet another. Is there some unifying learning algorithm that performs well on many types of ambiguity, or will we continue to fill in the ambiguity spectrum of Figure 1-1 in a piecemeal fashion?

- Another question for future investigation is whether we can learn more complicated hypothesis classes than hyper-rectangles (as in [Dietterich *et al.*, 1997] and [Auer, 1997]) or points with scaled features (as in this work). So far, the main tradeoff for learning from extremely ambiguous examples, as opposed to clearly labeled examples, has been that the learned concepts have been very simple, as opposed to concepts such as trees or neural networks. To examine why

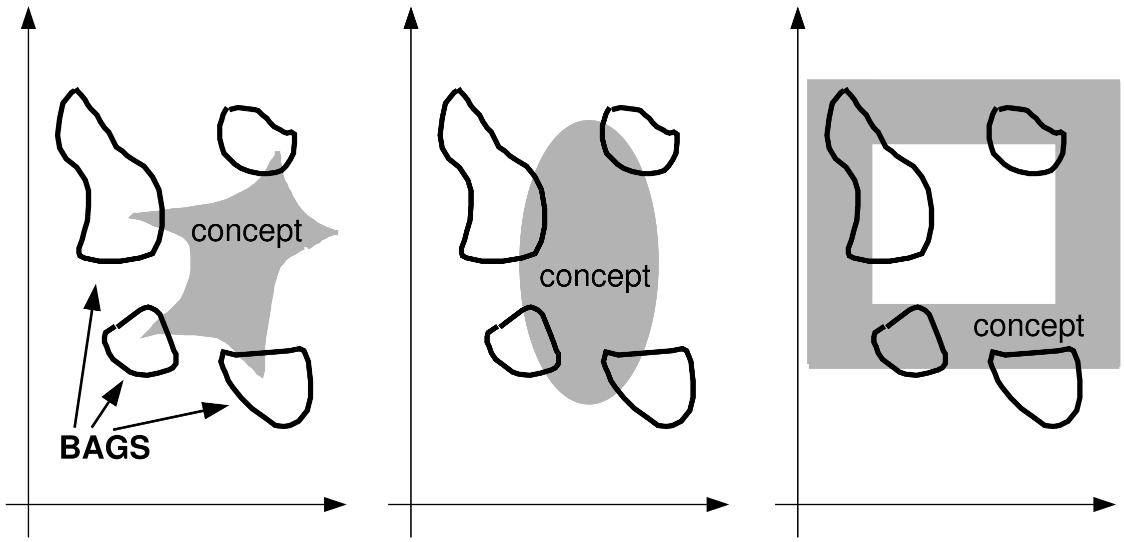


Figure 8-1: Given four positive bags, we see three different complicated concepts that agree with the examples. How are we to choose between them?

this might be a difficult task, let us take an extremely simple situation: instances are described using two attributes, both of which are known to be equally important. Furthermore, every bag contains an infinite number of points and is described merely by its boundary (or boundaries). Finally, let us assume that there are only positive bags. Let the hypothesis class be any two dimensional shape. Clearly, any hypothesis that intersects all the bags in any manner could be a valid concept. Some examples of valid concepts are shown in Figure 8-1. The number of valid concepts is huge, and even negative bags will not cause that number to drop significantly. It is not clear how to choose one shape over another because it is difficult to impose a simplicity metric on this space.

- It would also be interesting to apply Multiple-Instance learning to learning problems that are not a two-class classification problem, such as multi-class or regression applications. Multi-class problems could be transformed into many two-class problems, but perhaps there are more elegant ways of handling the ambiguity of "at least one of the elements in this bag belongs to class X." Regression problems appear to be harder to pose as Multiple-Instance learning problems, but we can state some minimal-regression problems. For example, for each bag you are given the minimum value attained by any instance in the bag. The goal of the learning algorithm is to find the concept that minimizes that value.

In fact, the drug discovery application is best stated this way. Every molecule is labeled with the binding energy required to bind it to the target protein. Molecules that were labeled simply as negative are now labeled with various high energy values, and molecules that were labeled simply as positive are now labeled with various low energy values (since at least one of the conformations of that molecule has very low binding energy with the protein). The learning task is to find a conformation which will minimize the energy needed for binding.

- One of the main concerns about current Multiple-Instance learners that use Diverse Density is that it takes a long time to find the maximum Diverse Density. Techniques such as `PWDD` help, but as with any large optimization task, the search space is infested with local maxima. A possible improvement is to perform a branch-and-bound search. Given a Diverse Density value for some concept, we can try to eliminate (without search) large parts of the search space as not containing potentially better Diverse Densities.

- Another difficulty with the Diverse Density framework is that it is very aggressive in pruning irrelevant features (see Section 3.1.2). Most Machine Learning techniques require a regularizer to prevent overfitting. In this case, however, it seems that an anti-regularizer to prevent under-fitting might be needed. Most domains do not provide enough negative examples for `maxDD` to maintain many features with a non-zero scaling. More work is needed to determine whether adding a term to Diverse Density which will penalize scalings from going to zero is necessary or advisable.

- An exciting area for future research is the development of better bag generators for the applications described in this thesis and also for new applications. For example, in the image database retrieval domain, the current bag generator places all image subregions of some predetermined size into a bag. If we can segment the image into components, such that every component's pixel set is roughly uniform in color or texture, then those components would make better instances than arbitrary subregions. In addition, this should greatly reduce the number of instances per bag, allowing faster learning and more accurate concepts.

- As seen in this thesis, Multiple-Instance learning can be used to tackle old problems with a new tool. One application which seems like an interesting candidate for this tool is document classification and retrieval. The document classification and retrieval task is similar to the image database retrieval application discussed in Chapter 6. The user selects several documents as positive and negative examples, and desires to see more documents like the positive ones and unlike the negative ones. Like images, documents are ambiguous objects, and can pertain to many different topics. A good bag generator for this task would be able to return all topics that a given document is about.

8.2 Summary

This thesis has presented a new technique for learning from multiple-instance examples. Maximizing Diverse Density is an appealing technique because of its simplicity, its clear mathematical assumptions, its graceful degradation when those assumptions fail to hold, and its performance on a wide variety of datasets. This work has also shown that the Multiple-Instance learning framework is useful for more than just the drug discovery problem. Its successful application to stock prediction and image database retrieval will hopefully help it become part of the arsenal of techniques used for data mining and by Machine Learning researchers. We have also shown a variety of bag generators, from one which collects many noisy instances into a bag (in Chapter 5) to one which breaks a large ambiguous example into many possible explanatory instances (in Chapter 6).

The architecture advocated in this thesis, separating the learning mechanism from the bag generator, allows and encourages the introduction of domain knowledge independently of the learning algorithm. Knowledge of which conformations are more likely than others is translated into the instances selected for a bag. Knowledge of current events can be used to eliminate a high-return stock as a candidate from the bag of possible fundamental stocks. Biases such as pose invariance (in the form of position invariance) should be a part of bag generators for images.

Finally, this thesis takes a step away from the assumption that enough training data will overcome any amount of noise, and moves toward a system that treats ambiguity explicitly. The problems tackled here would either be impossible for a supervised learner (drug discovery and the difficult artificial dataset) or extremely difficult (stock prediction and image classification). By using the Multiple-Instance learning framework, we are explicit about what we know about each bag (at least one of the instances is in the concept), and what we do not know (which instance). Diverse Density's success derives from its ability to learn from ambiguity.

Appendix A

Fun with logarithms and derivatives

A.1 Soft min and max

The softmax function is defined as follows:

$$\begin{aligned}\text{softmax}(x_1,\ldots,x_n) &= \sum_{1\le i\le n} \frac{x_i \cdot e^{\alpha x_i}}{\sum\limits_{1\le j\le n} e^{\alpha x_j}} \\ &= \sum_{1\le i\le n} \frac{x_i}{\sum\limits_{1\le j\le n} e^{\alpha(x_j - x_i)}}\end{aligned}$$

The parameter α determines how closely softmax approximates max. As α approaches ∞, softmax's behavior approaches max. When $\alpha = 0$, it calculates the mean. As α approaches $-\infty$, softmax's behavior approaches min.

In this thesis, the main reason for using softmax over max is that we can easily take the derivative of softmax. If each x_i is some function of t, then the derivative of softmax with respect to t is

$$\frac{\partial \text{softmax}(x_1, \ldots, x_n)}{\partial t} =$$

$$\sum_{1 \le i \le n} \frac{\left(\frac{\partial x_i}{\partial t} \sum_{1 \le j \le n} e^{\alpha(x_j - x_i)} \right) - \left(x_i \sum_{1 \le j \le n} e^{\alpha(x_j - x_i)} \cdot \alpha\left(\frac{\partial x_j}{\partial t} - \frac{\partial x_i}{\partial t}\right) \right)}{\left(\sum_{1 \le j \le n} e^{\alpha(x_j - x_i)} \right)^2} \tag{A.1}$$

A.2 Computing with very small probabilities

As stated in Chapter 2, we are only concerned with optimizing the first line of Formula 2.5, the definition of Diverse Density. We will refer to it as

$$\widehat{DD}(t) = \prod_{1 \le i \le n} \Pr(t \mid B_i^+) \prod_{1 \le i \le m} \Pr(t \mid B_i^-)$$

If the number of bags is large, then many probabilities need to be multiplied, usually resulting in a number which is too small to represent on a computer. We therefore compute $-\log \widehat{DD}(t)$, which is known as the negative log-likelihood. We look for concepts with low negative log-likelihood (the perfect concept would have 0 negative log-likelihood), which is equivalent to maximizing Diverse Density.

$$-\log \widehat{DD}(t) = -\left[\log \sum_{1 \le i \le n} \Pr(t \mid B_i^+) + \log \sum_{1 \le i \le m} \Pr(t \mid B_i^-) \right]$$

A.3 Derivative of Diverse Density

As stated in Chapter 3, we optimize Diverse Density over concept space by gradient based optimization. In this section, we derive the gradient of $-\log \widehat{DD}(t)$ for the noisy-or model and various concept classes.

$$\frac{\partial(-\log \widehat{DD}(t))}{\partial t} = -\left[\sum_{1 \le i \le n} \frac{1}{\Pr(t \mid B_i^+)} \times \frac{\partial \Pr(t \mid B_i^+)}{\partial t} + \sum_{1 \le i \le m} \frac{1}{\Pr(t \mid B_i^-)} \times \frac{\partial \Pr(t \mid B_i^-)}{\partial t} \right]$$

If we use noisy-or to model $\Pr(t \mid B_i^+)$, then

$$\Pr(t \mid B_i^+) = 1 - \prod_{1 \le j \le p} (1 - \Pr(B_{ij}^+ \in c_t))$$

and its derivative with respect to t is

$$\frac{\partial \Pr(t \mid B_i^+)}{\partial t} = \prod_{1 \le j \le p} (1 - \Pr(B_{ij}^+ \in c_t)) \times \left(\sum_{1 \le j \le p} \frac{1}{1 - \Pr(B_{ij}^+ \in c_t)} \times \frac{\partial \Pr(B_{ij}^+ \in c_t)}{\partial t} \right)$$

Likewise, if the bag was negative, the derivative is

$$\frac{\partial \Pr(t \mid B_i^-)}{\partial t} = - \prod_{1 \le j \le p} (1 - \Pr(B_{ij}^- \in c_t)) \times \left(\sum_{1 \le j \le p} \frac{1}{1 - \Pr(B_{ij}^- \in c_t)} \times \frac{\partial \Pr(B_{ij}^- \in c_t)}{\partial t} \right)$$

If we had used a most-likely-cause model instead of noisy-or, the derivative would be calculated as in Formula A.1, with $x_j = \Pr(B_{ij} \in c_t)$.
Using the single point concept class, we define

$$\Pr(B_{ij} \in c_t) = \exp(- \sum_{1 \le l \le k} (B_{ijl} - c_{t_l})^2)$$

as in Formula 2.9. c_{t_l} is the value of the l^{th} feature of concept c_t. To find the gradient, we measure the derivative of $\Pr(B_{ij} \in c_t)$ with respect to each feature t_l.

$$\frac{\partial \Pr(B_{ij} \in c_t)}{\partial t_l} = \exp(- \sum_{1 \le l \le k} (B_{ijl} - c_{t_l})^2) \times 2(B_{ijl} - c_{t_l})$$

Using the point-and-scaling concept class, we define

$$\Pr(B_{ij} \in (c_t, c_s)) = \exp(- \sum_{1 \le l \le k} (c_{s_l}(B_{ijl} - c_{t_l}))^2)$$

as in Formula 2.10. c_{t_l} and c_{s_l} are the values of the l^{th} feature and scaling, respectively, of concept (c_t, c_s). To find the gradient, we measure the derivative of $\Pr(B_{ij} \in (c_t, c_s))$ with respect to each feature t_l and each scaling s_l.

$$\frac{\partial \Pr(B_{ij} \in (c_t, c_s))}{\partial t_l} = \exp(-\sum_{1 \leq l \leq k} (c_{s_l}(B_{ijl} - c_{t_l}))^2) \times 2(B_{ijl} - c_{t_l})c_{s_l}^2$$

$$\frac{\partial \Pr(B_{ij} \in (c_t, c_s))}{\partial s_l} = -\exp(-\sum_{1 \leq l \leq k} (c_{s_l}(B_{ijl} - c_{t_l}))^2) \times 2c_{s_l}(B_{ijl} - c_{t_l})^2$$

Appendix B

Experimental details

This appendix describes some of the details of the algorithms and experiments in this thesis so that they may be replicated easily.

Diverse Density calculations

Diverse Density was calculated as described in Appendix A, which alleviated most of the floating point problems involved in handling probabilities. However, occasionally the algorithm was faced with the prospect of computing the logarithm of zero. Arbitrarily, we defined that to be $\log(10^{-7})$. For example, this bounds the maximum contribution of a negative instance (or a positive bag with all distant instances) to be approximately 16.

Gradient based optimization

We use a two-step gradient ascent routine. The routine first performs performs line searches (`lnsrch` in [Press *et al.*, 1992]) along the gradient direction using a loose convergence criterion. After the first step converges, the second step performs a quasi-newton search (`dfpmin` in [Press *et al.*, 1992]) from that point. The number of iterations allowed for each optimization was at least two times the number of dimensions of the search space.

Details of the difficult artificial dataset

The generation of bags was described in Chapter 2. One way of cheating in the positive bag generation process is to randomly pick $p-1$ instances from $[0,100] \times [0,100]$ and then one instance from $[50,55] \times [50,55]$. We generated the positive bags honestly; we picked sets of p instances randomly, discarding each set that did not have at least one instance within the concept. The cheating method biases the average number of "true positive instances" in a bag to be high.

The scalings used in the experiments was 1.0 for both features. The Gaussian in Formula 2.9 has a standard deviation of 1.0 for instances from positive bags and 5.0 for instances from negative bags. The results in Chapter 2 do not significantly change when the standard deviations for positive and negative instances are equal, except that negative bags have less noticeable contribution to accuracy.

Details of the musk experiments

The initial scaling of each feature was one. However, the data was preprocessed by dividing each feature value by 100. This was done so that the gradient based optimizations would not begin at a flat area of Diverse Density space.

Details of the stock experiments

The initial scaling of each feature was one. Each of the 17 features only takes on values from 1 to 11. For single point-and-scaling concepts, we began a gradient based optimization from every instance in every positive bag. For disjunctive concepts, we randomly selected a pair of instances (each from a different positive bag) as a starting disjunctive location for gradient based optimization. This was done repeatedly until no gradient based optimization improved the Diverse Density found. Exhaustively starting at every pair of instances would have been computationally infeasible.

Details of the image experiments

The images were taken from the COREL database, and were labeled according to COREL's classifications. The RGB values were normalized to be in the $[0, 1]^3$ cube. For each gradient based optimization, the initial scaling vector was uniform at 1. However, for some experiments, the initial scaling vector forced all instances to be very far away from each other. This resulted in a flat gradient, and no search was performed. In those cases, the initial scaling vector was decreased to be 0.1 for each feature.

We note that the best scaling (i.e., the one which maximizes Diverse Density) is independent of the initial scaling vector. Using an initial scaling vector with small scalings simply allowed us to escape a local plateau in Diverse Density space.

For single point-and-scaling concepts, we began a gradient based optimization from every instance in every positive bag. When it was computationally feasible, learning disjunctive concepts involved an optimization from every pair of positive instances. When it was not computationally feasible, we performed optimizations starting from randomly selected pairs.

Bibliography

[Atkeson *et al.*, 1997] C. G. Atkeson, A. W. Moore, and S. A. Schaal. Locally weighted learning. In David W. Aha, editor, *Lazy Learning*. Kluwer Academic Publishers, 1997.

[Auer *et al.*, 1996] Peter Auer, Phil M. Long, and A. Srinivasan. Approximating hyper-rectangles: learning and pseudorandom sets. In *Proceedings of the 1996 Conference on Computational Learning Theory*, 1996.

[Auer, 1997] Peter Auer. On learning from multi-instance examples: Empirical evalutaion of a theoretical approach. In *Proceedings of the 14th International Conference on Machine Learning*, 1997.

[Bach *et al.*, 1996] J.R. Bach, C.Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R.C. Jain, and C. Shu. Virage image search engine: an open framework for image management. In *Symposium on Electronic Imaging: Science and Technology - Storage and Retrieval of Image and Video Databases*, volume 4, pages 76–87, 1996.

[Bell and Sejnowski, 1995] A. Bell and T. Sejnowski. An information-maximization approach to blind source seperation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

[Belongie *et al.*, 1998] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color- and texture based image segmentation using em and its application to content-based image retrieval. In *International Conference on Computer Vision*, 1998.

[Blum and Kalai, 1998] A. Blum and A. Kalai. A note on learning from multiple-instance examples. *To appear in Machine Learning*, 1998.

[Buchanan and Mitchell, 1978] B. G. Buchanan and T. M. Mitchell. Model-directed learning of production rules. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern-Directed Inference Systems*. Academic Press, 1978.

[Chatfield and Collins, 1980] Christopher Chatfield and Alexander J. Collins. *Introduction to Multivariate Analysis*. Chapman and Hall, 1980.

[Cheeseman *et al.*, 1988] Peter Cheeseman, James Kelly, Matthew Self, John Stutz, Will Taylor, and Don Freeman. Autoclass: A bayesian classification system. In *Proceedings of the Fifth International Workshop on Machine Learning*. Morgan Kaufmann, 1988.

[Cohen, 1996] William W. Cohen. The dual dfa learning problem: Hardness results for programming by demonstration and learning first-order representations. In *Proceedings of the 1996 Conference on Computational Learning Theory*, 1996.

[Comon, 1994] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36:287–314, 1994.

[Dammkoehler *et al.*, 1989] R. A. Dammkoehler, S. F. Karasek, E. F. B. Shands, and G. R. Marshall. Constrained search of conformational hyperspace. *Journal of Computer-Aided Molecular Design*, 3:3–21, 1989.

[Dasarathy, 1991] B. V. Dasarathy. *Nearest Neighbor Norms: NN Patern Classifaction Techniques*. IEEE Computer Society Press, 1991.

[Dayan and Zemel, 1995] Peter Dayan and Richard S. Zemel. Competition and multiple cause models. *Neural Computation*, 7:565–579, 1995.

[Decatur, 1994] Scott Decatur. Statistical queries and faulty pac oracles. In *Proceeding of the Sixth Annual ACM Workshop on Computational Learning Theory*, pages 262–268. ACM Press, July 1994.

[Dietterich *et al.*, 1994] T. G. Dietterich, A. Jain, R. Lathrop, and T. Lozano-Pérez. A comparison of dynamic reposing and tangent distance for drug activity prediction. In *Advances in Neural Information Processing Systems 6*. Morgan Kauffman, 1994.

[Dietterich *et al.*, 1997] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence Journal*, 89, 1997.

[Flickner *et al.*, 1995] M. Flickner, Harpreet S. Sawhney, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker. Query by image and video content: The qbic system. *IEEE Computer*, 28:23–32, 1995.

[Girosi *et al.*, 1995] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.

[Hirsh, 1990] Haym Hirsh. *Incremental version-space merging: a general framework for concept learning*. Kluwer Academic, 1990.

[Huang *et al.*, 1997] J. Huang, S. Ravikumar, M. Mitra, W. Zhu, and R. Zabih. Image indexing using color correlograms. In *Computer Vision and Pattern Recognition*, 1997.

[Husbands and Isbell, 1998] P. Husbands and C. Isbell. The parallel problems server: A client-server model for large scale scientific computation. In *Proceedings of the Third International Conference on Vector and Parallel Processing*, 1998.

[Jain *et al.*, 1994] A. N. Jain, T. G. Dietterich, R. H. Lathrop, D. Chapman, R. E. Critchlow Jr., B. E. Bauer, T. A. Webster, and T. Lozano-Pérez. Compass: A shape-based machine learning tool for drug design. *Journal of Computer-Aided Molecular Design*, 8:635–652, 1994.

[Kaelbling *et al.*, 1996] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 1996.

[Kauer, 1991] J. S. Kauer. Contributions of topography and parallel processing to odor coding in the vertebrate olfactory pathway. *Trends in Neurosciences*, 14(2):79–85, 1991.

[Kearns and Vazirani, 1994] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, 1994.

[Keeler *et al.*, 1991a] James D. Keeler, David E. Rumelhart, and Wee-Kheng Leow. Integrated segmentation and recognition of hand-printed numerals. In *Advances in Neural Information Processing Systems 3*. Morgan Kauffman, 1991.

[Keeler *et al.*, 1991b] James D. Keeler, David E. Rumelhart, and Wee-Kheng Leow. Integrated segmentation and recognition of hand-printed numerals. Technical report, MCC Technical Report ACT-NN-010-91, 1991.

[Lang, 1989] Kevin J. Lang. *A Time-Delay Neural Network Architecture for Speech Recognition*. Phd dissertation, Carnegie Mellon University, School of Computer Science, 1989.

[Leach, 1996] Andrew R. Leach. *Molecular Modeling: principles and applications*. Longman, Harlow, England, 1996.

[Lipson *et al.*, 1997] P. Lipson, E. Grimson, and P. Sinha. Context and configuration based scene classification. In *Computer Vision and Pattern Recognition*, 1997.

[Long and Tan, 1996] P. M. Long and L. Tan. Pac-learning axis alligned rectangles with respect to product distributions from multiple-instance examples. In *Proceedings of the 1996 Conference on Computational Learning Theory*, 1996.

[Maron and Lakshmi Ratan, 1998] O. Maron and A. Lakshmi Ratan. Multiple-instance learning for natural scene classification. In *Machine Learning: Proceedings of the 15th International Conference*, 1998.

[Maron and Lozano-Pérez, 1998] O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems 10*. MIT Press, 1998.

[Minka and Picard, 1996] T. Minka and R. Picard. Interactive learning using a society of models. In *Computer Vision and Pattern Recognition*, 1996.

[Mitchell, 1978] Tom M. Mitchell. *Version spaces: an approach to concept learning.* Phd dissertation, Stanford University, Dept. of Electrical Engineering, 1978.

[Morozov, 1984] V. A. Morozov. *Methods for solving incorrectly posed problems.* Springer Verlag, 1984.

[Murphy and Aha, 1996] P. M. Murphy and D. W. Aha. Uci repository of machine learning databases. for more information contact ml-repository@ics.uci.edu, 1996.

[Norton, 1994] Steven W. Norton. Learning to recognize promoter sequences in *e. coli* by modeling uncertainty in the training data. In *Proceedings of the Twelfth National Conference on Artificial Intelligence.* MIT Press, 1994.

[Omohundro, 1991] S. M. Omohundro. Bumptrees for efficient function, constraint, and classification learning. In Lippmann, Moody, and Touretzky, editors, *Advances in Neural Information Processing Systems 3*, San Mateo, CA, 1991. Morgan Kaufmann.

[Pearl, 1988] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, 1988.

[Preparata and Shamos, 1985] F. P. Preparata and M. I. Shamos. *Computational geometry: an introduction.* Springer-Verlag, 1985.

[Press *et al.*, 1992] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: the art of scientific computing.* Cambridge University Press, New York, second edition, 1992.

[Quinlan, 1992] J. R. Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann, 1992.

[Rissanen, 1978] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[Rumelhart *et al.*, 1986] David E. Rumelhart, James L. McClelland, and the PDP research Group. *Parallel distributed processing: explorations in the microstructure of cognition*. MIT Press, 1986.

[Saund, 1995] Eric Saund. A multiple cause mixture model for unsupervised learning. *Neural Computation*, 7:51–71, 1995.

[Smith and Chang, 1996] J. Smith and S. Chang. Visualseek: a fully automated content-based image query system. In *Proceedings of the ACM International Conference on Multimedia*. Morgan Kaufmann, 1996.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press/Bradford Books, Cambridge, MA, 1998.

[Trippi, 1995] Robert R. Trippi. *Chaos & Nonlinear Dynamics in the Financial Markets: Theory, Evidence, and Applications*. McGraw-Hill/Irwin, 1995.

[Valiant, 1984] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[Vapnik, 1995] Vladimir Naumovich Vapnik. *The nature of statistical learning theory*. Springer, 1995.

[Waibel *et al.*, 1989] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3), March 1989.