Solving the Multiple-Instance Problem with Axis-Parallel Rectangles

Thomas G. Dietterich
Oregon State University
Corvallis, OR 97331-3202

Richard H. Lathrop
Department of Computer and Information Science
University of California
Irvine, CA 92717

Tomas Lozano-Perez
Arris Pharmaceutical Corporation
385 Oyster Pt. Blvd.
South San Francisco, CA 94080

and MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139

Abstract

The multiple instance problem arises in tasks where the training examples are ambiguous: a single example object may have many alternative feature vectors (instances) that describe it, and yet only one of those feature vectors may be responsible for the observed classification of the object. This paper describes and compares three kinds of algorithms that learn axis-parallel rectangles to solve the multiple-instance problem. Algorithms that ignore the multiple instance problem perform very poorly. An algorithm that directly confronts the multiple instance problem (by attempting to identify which feature vectors are responsible for the observed classifications) performs best, giving 89% correct predictions on a musk-odor prediction task. The paper also illustrates the use of artificial data to debug and compare these algorithms.

1 Introduction

Consider the following learning problem. Suppose there is a keyed lock on the door to the supply room in an office. Each staff member has a key chain containing several keys. One key on each key chain can open the supply room door. For some staff members, their supply room key opens only the supply room door; while for other staff members, their supply room key may open one or more other doors (e.g., their office door, the mail room door, the conference room door).

Suppose you are a lock smith and you are attempting to infer the most general required shape that a key must have in order to open the supply room door. If you knew this required shape, you could predict, by examining any key, whether that key could unlock the door. What makes your lock smith job difficult is that the staff members are uncooperative. Instead of showing you which key on their key chains opens the supply room door, they just hand you their entire key chain and ask you to figure it out for yourself! Furthermore, you are not given access to the supply room door, so you can't try out the individual keys. Instead, you must examine the shapes of all of the keys on the key rings and infer the answer.

We call this kind of learning problem the *multiple instance problem.* It arises in complex applications of machine learning where the learning system has partial or incomplete knowledge about each training example. In traditional supervised learning problems, the learning system is given training examples of the form $\{\langle object_i, result_i \rangle\}$. This situation is depicted in Figure 1(a). Each object is typically represented as a fixed-length vector of attribute values (usually called a "feature vector").

However, as machine learning applications become more complex, the situation shown in Figure 1(b) can arise. Here, the learner has incomplete information about each training example. Rather than knowing that each training example can be represented as a feature vector, the learner only knows that each example can be represented by one of a *set* of *potential feature vectors.* In our lock smith problem, instead of knowing which key (from each key chain) opens the supply room, the learning system only knows that one of the keys on the key chain opens the door.

An early example of this learning situation arose in the Meta-DENDRAL project (Buchanan & Mitchell, 1978; Lindsay, Buchanan, Feigenbaum & Lederberg, 1980). In Meta-DENDRAL, the goal was to learn rules that could predict the behavior of molecules inside a mass spectrometer. A mass spectrometer bombards a molecule with high energy particles, which causes the molecule to break into fragments. These fragments are then analyzed to produce a histogram of their mass-to-charge ratio, which is called a mass spectrum. The main problem in Meta-DENDRAL was to predict which bonds would break. Each molecule is analogous to a key chain, and each bond is analogous to an individual key. By observing several molecules (and the resulting fragments), Meta-DENDRAL was able to formulate a small number of bond-breakage rules that accounted for the observed fragments.

A similar situation arises in explanation-based learning with a "promiscuous" domain theory (Dietterich & Flann, 1988). Given an input example, the domain theory can construct multiple explanations that account for the observed result. The learning task is to examine several training examples and find one explanation that can account for all of the observed results. In this case, each example is like the key chain, and each alternative explanation is like an individual key.

The multiple instance problem also arises in the application domain of drug activity prediction, which is of central importance in this paper. In this domain, the input object is a molecule, and the observed result is a measurement of the degree to which the molecule binds to a target "binding site." A binding site is a cavity or pocket (part of a much larger molecule) into which the input molecule binds. A good drug molecule will bind very tightly to the desired binding site, while a poor drug molecule will not bind well. The variant instances are alternative "conformations" of the molecule—alternative shapes that the molecule can adopt by rotating its bonds. One (or a few) of these shapes actually bind to the binding site and produce the observed result. The other conformations typically have no effect on binding. The learning task is to infer the requirements for the observed drug activity.

This is directly analogous to the lock smith problem. Each molecule is like a key chain. The different shapes that it can adopt (the conformations) are like individual keys. The goal is to infer the most general shape required for binding to the binding site (opening the lock).

Drug activity can be measured in many ways. In some settings, a laboratory assay can measure the kinetics of the binding reaction and determine the binding strength directly. In other settings, activity is measured by its observed biological effect. For example, in the musk odor prediction task described below, activity is measured by human subjects who characterize a chemical compound as "active" or "inactive".

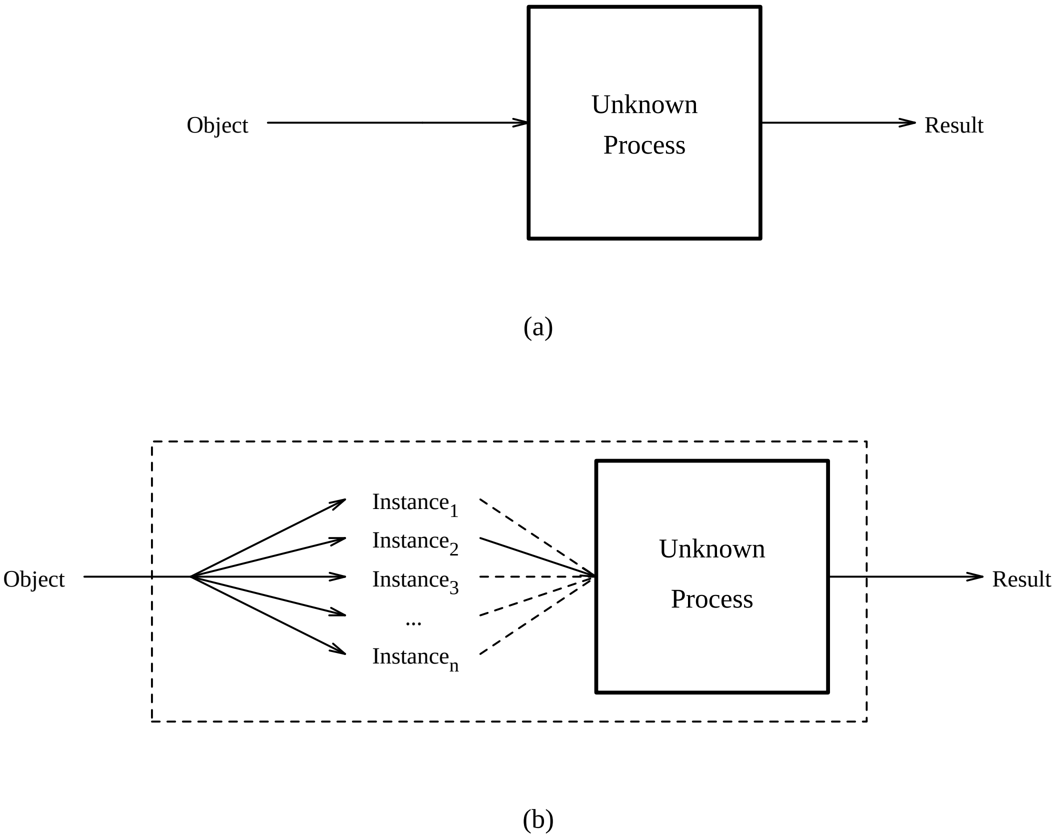

Figure 1: Supervised learning: (a) usual situation and (b) multiple-instance situation.

The availability of "inactive" molecules is one aspect of the drug activity problem that extends beyond the lock smith problem. For an inactive molecule, we know that *none* of its possible conformations (shapes) can bind to the binding site. In the lock smith problem, this would be analogous to having some key chains from people in other businesses where we know that *none* of the keys on their key chains open the supply room door.

In this paper, we will let M be the set of possible objects m_i. Each object m_i has an observed result, $f(m_i)$. Because our musk data is labeled as "active" or "inactive", we will treat $f(m_i)$ as a binary quantity: $f(m_i) = 1$ for "active" molecules, and $f(m_i) = 0$ for "inactive" molecules. The function f represents the unknown process. The goal of learning is to find a good approximation $\hat{f}$ to f by analyzing a set of training examples drawn from M and labeled by f.

In ordinary supervised learning, we would usually represent each object $m_i \in M$ by a vector of n real-valued features, $V(m_i) \in R^n$. Indeed, most machine learning papers make no distinction between the objects and their feature vectors because of this one-to-one correspondence. A labeled training example thus has the form

$$\langle V(m_i), f(m_i) \rangle.$$

However, in the setting of Figure 1(b), each object m_i may have ν_i variant instances denoted $m_{i,1}, m_{i,2}, \ldots, m_{i,\nu_i}$. Each of these variants will be represented by a (usually) distinct feature vector $V(m_{i,j})$. A complete training example is therefore written as

$$\langle \{V(m_{i,1}), V(m_{i,2}), \ldots, V(m_{i,\nu_i})\}, f(m_i) \rangle.$$

In other words, the representation for each training example is ambiguous, and a machine learning algorithm must overcome this ambiguity. We will assume that the complete set of variants is finite and known to the learning algorithm.

It is a property of all of the domains described above that if the observed result is "positive" (e.g., "active" in drug design, "present" in mass spectrometry), then at least one of the variant instances must have produced that positive result. Furthermore, if the observed result is "negative", then none of the variant instances could have produced a positive result. We can model this by introducing a second function $g(V(m_{i,j}))$ that takes a single variant instance and produces a result. The externally-observed result, $f(m_i)$, can then be defined as follows:

$$f(m_i) = \begin{cases} 1 & \text{if } \exists\, j \;\; g(V(m_{i,j})) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

In short, object m_i is predicted to be a positive example if and only if there exists at least one feature vector for m_i (one variant instance) that is predicted to be positive according to g. This definition allows for the possibility that more than one variant is predicted positive by g.

We will refer to all of the variants of positive examples as positive instances, even though only one of them may have produced the positive result. Similarly, we will refer to all of the variants of a negative example as negative instances.

In the remainder of the paper, the goal of the machine learning algorithm will be to construct an approximation $\hat{g}$ to the internal function g. An hypothesis $\hat{g}$ is consistent with a set of training examples if it classifies every feature vector of every negative example as negative and if it classifies at least one feature vector of every positive example as positive.

We call this learning problem the "multiple instance problem," because each training example is represented by multiple instances (or feature vectors). The goal of this paper is to demonstrate that the multiple-instance problem is an important problem and to compare the effectiveness of three general approaches to solving the problem in the case where an axis-parallel rectangle bias is appropriate.

The paper begins by describing an application domain—drug activity prediction—in which the multiple instance problem arises. We then describe a feature representation for this application for which a good bias would appear to be axis-parallel hyper-rectangles (APRs). We consider three general designs for APR learning algorithms:

- A noise-tolerant "standard" algorithm. The naive APR algorithm just forms the smallest APR that bounds the positive examples. We explore a noise-tolerant version of this algorithm that ignores the multiple-instance problem.

- An "outside-in" algorithm. This algorithm is a variation on the "standard" algorithm. It constructs the smallest APR that bounds all of the positive examples and then shrinks this APR to exclude false positives. The "shrinking process" addresses the multiple-instance problem.

- An "inside-out" algorithm. This algorithm starts with a seed point in feature space and "grows" a rectangle with the goal of finding the smallest rectangle that covers at least one instance of each positive example and no instances of any negative example. We found it necessary to expand the resulting APR (via a statistical technique) in order to get good performance.

Our results show that the "inside-out" algorithm performs much better than either of the others. We will present evidence that this is because the "inside-out" algorithm can identify the relevant features better than the "outside-in" algorithm. The results will also demonstrate that the "standard" algorithm performs much worse than either of the others—this will prove that the multiple instance problem cannot be ignored but instead must be considered during the design of learning algorithms.

To conduct this research and obtain these results, we found it extremely valuable to develop an artificial data set that mimics the behavior of the real data. This was valuable for several reasons:

- The two real data sets under study contained only 92 and 102 examples each. Hence, there was a great danger of overfitting on these data sets. Overfitting could occur during a single run. But an even greater risk was that the entire algorithm development process would overfit the data as we attempted to improve cross-validated accuracy.

- An artificial data set allowed us to develop and debug our algorthms with data for which the "right answer" was known. This substantially improved the efficiency of our research.

- To construct the artificial data set, we were forced to carefully analyze our real data sets in order to understand how they might have been generated. This provided many ideas for new algorithms.

The remainder of the paper is therefore organized as follows. After describing the application domain, we present an analysis of the data sets under study. Based on this analysis, we then describe our artificial data set.

Next, we present each of the three algorithm designs and compare their performance on the artificial data. Runs on the artificial data help determine parameter values for the learning algorithms. Finally, we run the learning algorithms on our two real-world data sets and summarize the results.

2 Drug Activity Prediction

The algorithms described in this paper were motivated by the task of drug activity prediction.

2.1 Background

Most drugs are small molecules that work by binding to much larger protein molecules such as enzymes and cell-surface receptors. The potency of a drug is determined by the degree to which it binds to the larger, target molecule. Drug molecules typically do not bind covalently to target molecules. Instead, they exploit a variety of weak interactions including (a) hydrogen bonds, (b) van der Waals attractions, (c) electrostatic (charge) interactions, and (d) hydrophobic interactions. The "right" molecular shape conforms closely to the shape of the binding site (which enables van der Waals attractions and hydrophobic interactions over large surface areas) and presents electronically active surface atoms near complementary binding site atoms (which enables electrostatic and hydrogen bond interactions). An analogy is often drawn to a lock and key: A key will operate a lock only if its shape is complementary to the shape of the lock.

The goal of drug activity prediction is to predict the activity of new (not-yet synthesized) molecules by analyzing a collection of training examples consisting of previously-synthesized molecules and their observed activities when binding to a binding site of medical interest. By focusing the expensive and time-consuming efforts of chemists on synthesizing the most promising candidate molecules, accurate drug activity predictions could yield large savings in time and money for pharmaceutical companies.

An even greater benefit of applying machine learning to drug activity prediction would be to guide the process of drug design. If chemists could obtain a three dimensional model of the requirements for drug activity, they would be able to *design* better drugs. Sometimes the shape of the binding site can be inferred from X-ray crystallography and used to guide drug design (Kuntz, et al., 1992). In many practical cases, however, the shape of the binding site is unknown, and machine learning methods might be able to provide a three dimensional shape hypothesis to support drug design. Such "rational" drug design could cut years off the time required to discover new drugs. It could also result in drugs with higher potency and fewer side-effects.

2.2 The Multiple Instance Problem

The multiple instance problem arises in drug activity prediction because of our choice of representation for the drug molecules. Hence, we must describe why we selected a representation that creates the multiple instance problem.

Because binding strength is largely determined by the *shape* of drug molecules, a good representation must capture the shape of each molecule. Unfortunately, molecules can adopt multiple shapes by rotating some of their internal bonds as shown in Figure 2. Hence, any representation that captures the shape of a molecule will produce multiple feature vectors as the shape changes.

Every combination of angles of the rotatable bonds of a molecule defines a "conformation." Each conformation has a potential energy that is determined by the interactions between the atoms making up the molecule. (This is analogous to the gravitational potential energy between two objects separated by a specified distance.) At ordinary temperatures, the conformation of a molecule in solution is rapidly changing. The probability that the molecule will adopt a particular conformation depends exponentially on the potential energy of that conformation according to the Boltzmann distribution—low-energy conformations are much more probable than high-energy conformations. Hence, in practice, the only conformations in which the molecule is likely to bind to the binding site are conformations of low energy (i.e., within 5 Kcals of the lowest possible energy for any conformation of the molecule).

We make the (standard) assumption that only conformations that correspond to local energy minima are possible candidates for binding. For a molecule with n rotatable bonds, one can usually

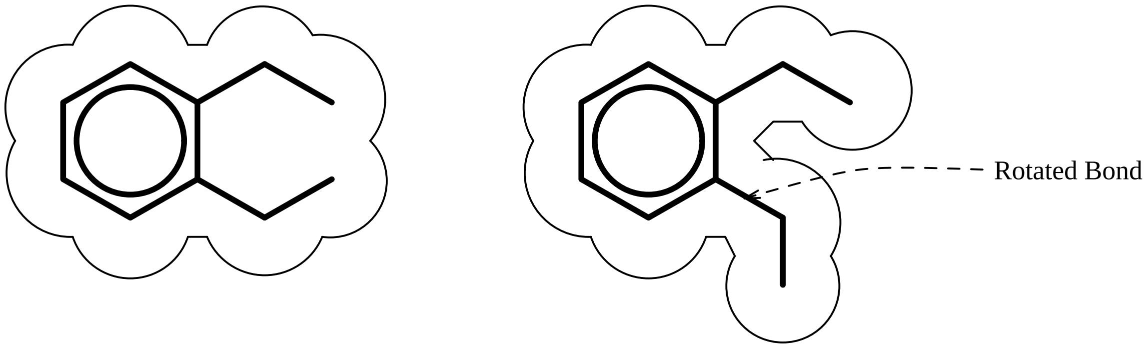

Figure 2: The shape of a molecule changes as it rotates internal bonds. (Thin lines indicate molecular surface.)

expect to find $O(3^n)$ local minimum conformations. Fortunately, only a fraction of these will be of sufficiently low energy. These low energy conformations can be computed by several methods including Monte Carlo search of bond-angle space, systematic bond-angle search, and molecular dynamics (which simulates the motions of the atoms using Newtonian mechanics). In the remainder of this paper, we will restrict our attention to these low energy conformations. Each low energy local minimum conformation will create a distinct feature vector for input to the learning algorithm.

There are some approaches to drug activity prediction that avoid the multiple-instance problem. One approach is to employ a representation that is invariant to changes in bond angles. Previous research in drug activity prediction has attempted to use such representations (e.g., notably methods derived from Hansch & Fujita, 1964 and Hansch & Leo, 1979), and some success has been reported for simple molecules or for families of molecules having large amounts of shared structure (Ståhle & Wold, 1988). For diverse molecules of the type studied in this paper, these methods have not been very successful (Blankley, 1983).

Another approach is to employ a shape-oriented representation, but to attempt to guess in advance which conformation of each molecule is the biologically-active one. The CoMFA method (Cramer, Patterson, & Bunce, 1988) has shown the promise of shape-based representation, but it has difficulty picking the right conformations. Other methods that employ some form of shape representation include Koehler, Rowberg-Schaefer & Hopfinger (1988), Good, So, & Richards (1993), and Vedani, Zginden & Snyder (1993).

A method that confronts the multiple-instance problem directly is the elegant distance-geometry approach of Crippen (Crippen & Havel, 1988). Unfortunately, combinatorial explosions in the search space of their approach limit the complexity of their binding-site hypotheses to constraints on the positions of four or five key atoms. The approach that we describe can learn detailed constraints on the position of the entire molecular surface.

2.3 A ray-based representation for molecular shape

Figure 3 shows the representation that we employed to capture the shape of molecules. We constructed a set of 162 rays emanating from the origin so that they sample space approximately uniformly. To extract features from a molecule, the molecule is placed in a standard position and orientation so that the origin lies inside it. From these 162 rays, 162 feature values are measured. Each feature value is the distance from the origin to the molecule surface. We computed the molecular surface by Connolly's (1983) method with a probe radius of 1.5Å. In addition to these 162

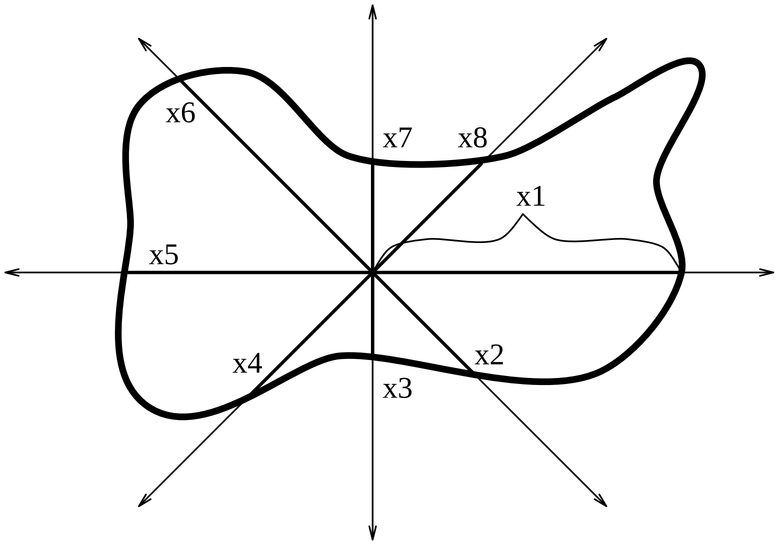

Figure 3: A schematic diagram of the ray-based representation (only 8 rays are shown). Thick curve indicates molecular surface.

"shape" features, we also computed four domain-specific features that represented the position of a designated atom (an oxygen atom) on the molecular surface (see below).

To determine the standard position and orientation of each molecule, all of the molecules were aligned to one another via translation and rotation. These alignments were carried out by an ad hoc algorithm that superimposed the atoms in the benzene rings of the molecules[1] and then attempted to place the designated oxygen atom at one of two positions that could support the formation of a hydrogen bond.

This ray-based representation is sufficient for molecules that have a compact, spheroidal shape. Similar representations could easily be constructed for molecules with other shapes such as long columns, curved segments, or even loops. This representation was sufficient for the molecules studied in this paper.

Note that as the shape of a molecule changes from one conformation to another, the distances measured along the rays will change. Hence, different conformations will be represented by distinct feature vectors. Also note that the measured feature values are locally correlated: values measured along adjacent rays will be quite similar. This suggests that the actual number of features required to characterize active molecules—the number of *relevant features*—will probably be substantially less than 162.

The ray-based representation immediately suggests a representation for hypotheses. Let us suppose that the binding site requires that the surface of the molecule be in certain locations. Then, by placing an upper and lower bound along each ray, we can describe the allowed positions of the molecular surface along each ray. For well-separated rays, it is likely that the allowed positions along each ray are independent, because each surface patch of the molecule interacts with a different surface patch on the binding site). Hence, the bounds along the rays correspond to an axis-parallel hyper-rectangle in the 162-dimensional feature space. Figure 4 shows that these

[1]A benzene ring is a planar 6-member ring of carbon atoms with a very strong de-localized bonding structure. It is denoted by a hexagon with alternating single and double bonds. Every one of our molecules contains at least one benzene ring.

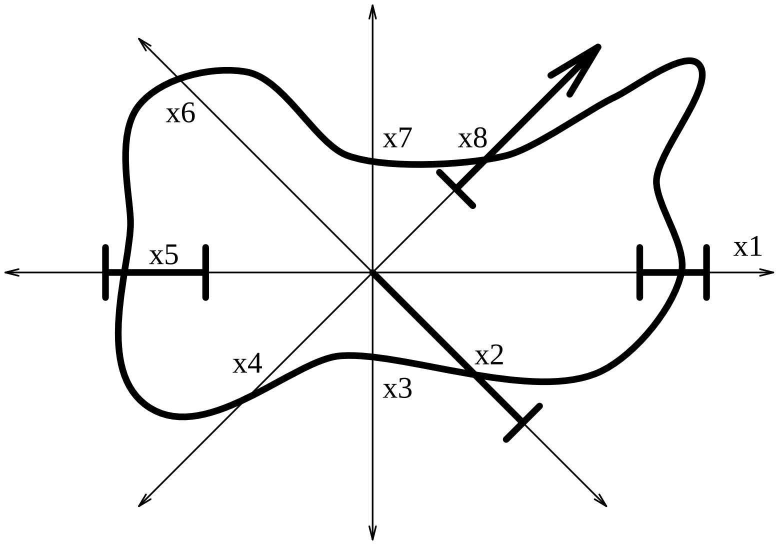

Figure 4: A schematic diagram of a binding hypothesis represented as bounds along the rays. If the molecular surface satisfies the bounds, then the molecule is predicted to be active. Rays x_1 and x_5 have two-sided bounds; x_2 has an upper bound only; x_8 has a lower bound only; and rays x_3, x_4, and x_7 are unconstrained (irrelevant for binding).

bounds can be two-sided or one-sided: they can require that the molecule "stick out" beyond a certain distance (via a lower bound on the ray); they can require that the molecule not "stick out" too far (via an upper bound on the ray). Both of these conditions reflect important domain interactions. A lower bound may require the molecule to extend far enough to make critical van der Waals or hydrophobic interactions with the binding site. An upper bound may prohibit the molecule from colliding with the wall of the binding site.

2.4 Predicting musk strength

To develop our learning methods for drug activity prediction, we chose to study the problem of predicting the strength of synthetic musk molecules. This problem had several attractive aspects: (a) it is non-proprietary, (b) a large number of musk compounds and similar non-musk compounds have been published in the open literature, (c) the identity and shape of the binding site or sites is unknown, and (d) the molecules are similar in size and composition to orally-active drug molecules. The only aspect of the musk problem that is substantially different from typical pharmaceutical problems is that musk strength is measured qualitatively by expert human judges, whereas drug activity is usually measured quantitatively through biochemical assays. This makes the musk problem somewhat more difficult.

We surveyed the literature on musk compounds (Theimer & Davies, 1967; Ham & Jurs, 1985; Narvaez, Lavine & Jurs, 1986; Beets, 1987; Fehr, Galindo, Haubrichs & Perret, 1989; Bersuker, Dimoglo, Yu. Gorachov, Vlad & Pesaro, 1991) and selected two (overlapping) data sets of musk molecules. Because of the subjective nature of the test for musk strength, there is quite a bit of variation from one paper to another. We considered only compounds that appeared in at least 2 publications and for which all published musk judgements agreed. Data set 1 contains 47 musk molecules and 45 similar non-musk molecules. Data set 2 contains 39 musks and 63 non-musks. 72

Table 1: The musk data sets

Data Set	Musks	Non-Musks	Total	Low-energy conformations
1	47	45	92	476
2	39	63	102	6598

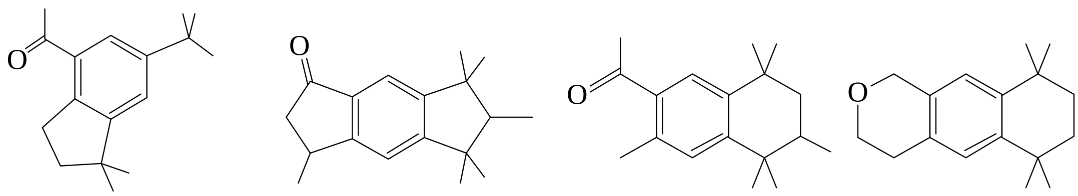

Figure 5: Four musk molecules

of the molecules are shared between the two data sets.

Once the molecules had been identified and their bond graphs entered into the computer, it was necessary to search the space of possible conformations of each molecule to find low-energy minima. For both data sets, we employed the monte carlo search algorithm implemented in the MacroModel program (Chang, Guida & Still, 1989; Still, undated) to minimize the AMBER force field (Weiner, Kollman, Case, Singh et al., 1984; Weiner, Kollman, Nguyen & Case, 1986). For Data Set 1, a subset of the resulting low-energy conformations for each molecule was chosen to maximize the pairwise root-mean-square distances between the atom positions of each pair of conformations. The goal was to obtain a small set of diverse low-energy conformations for each molecule.

For data set 2, we selected the molecules more carefully. We are more certain that the molecules have been properly classified, and the conformational searching was much more thorough. All conformations produced by MacroModel were retained. Despite the larger number of molecules, this second data set has been more difficult for learning algorithms because it contains many more conformations. This is indirect evidence of the difficulties created by the multiple instance problem. Table 1 summarizes the two data sets.

Figure 5 shows four molecules from our training data set. The molecules are made up entirely of carbon and hydrogen atoms with the exception of one oxygen. Previous authors have concluded that the oxygen is critical for musk strength. Hence, we added four oxygen features to the 162 shape features. Three of these features measure the X-, Y-, and Z- displacements of the oxygen atom from a designated point in space. The fourth feature measures the Euclidean distance between the oxygen atom and the designated point. These features were chosen so that the axis-parallel rectangle method of representing binding requirements could still be employed.

2.5 Related Work

The research reported in this paper grew out of initial efforts at Arris Pharmaceutical Corporation to apply machine learning to drug activity prediction. Subsequent work at Arris has produced an APR-like neural network algorithm, called COMPASS (Jain, Koile, Bauer, Chapman, 1994; Jain, Dietterich, Lathrop, Chapman, Critchlow, Bauer, Webster & Lozano-Perez, 1994), that improves upon the algorithms reported here. One advantage of COMPASS is that it is more robust to errors in the initial alignment of the molecules—during the learning process, COMPASS automatically optimizes the relative alignment of the molecules. Another advantage is that COMPASS can handle

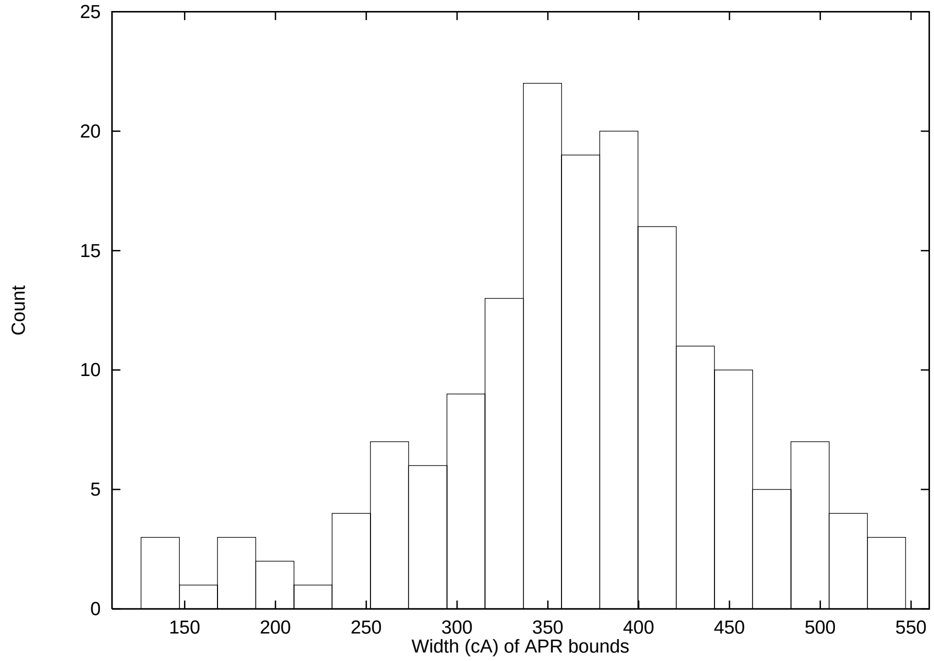

Figure 6: Histogram of the width of the bounds of an APR that tightly contains all feature vectors in Musk Data Set 1

activity prediction tasks in which the activities are continuous quantities, while the work reported here can make only active/inactive classifications.

The primary contribution of the work reported here is that it demonstrates the critical importance of solving the multiple instance problem, and it shows how to solve this problem for hypotheses represented as axis-parallel rectangles. Axis-parallel rectangles are generally easier to interpret than neural networks, and we expect that there will be new applications for the algorithms described in this paper.

3 Data Analysis of Musk Data Set 1

In any application problem, it is important to analyze the data to assess what biases might be appropriate and to gain other insights that can help guide the choice of learning algorithms. Hence, we performed a fairly detailed analysis of the first musk data set, which is presented in this section.

An additional motive for data analysis was to help us design an artificial data set with properties similar to the musk data, but where we could specify the "right answer." This has been extremely helpful during algorithm debugging and sensitivity testing. It also was critical in helping us reduce "overfitting" to the real musk data set. Artificial data sets have also proven useful in the development of DNA sequence assembly algorithms (Engle & Burks, 1993; Soderlund & Burks, 1994) and in the comparison of learning algorithms (Aha, 1992).

We began by constructing a hyper-rectangle that tightly contains all of the data. We then computed the width of the bounds along each ray and plotted the histogram shown in Figure 6. All feature values are measured in hundredths of angstroms (Å)—centiangstroms (cÅ). Note that there is substantial variation (e.g., at least 2Å and typically 3.5Å) along nearly every ray. Only a

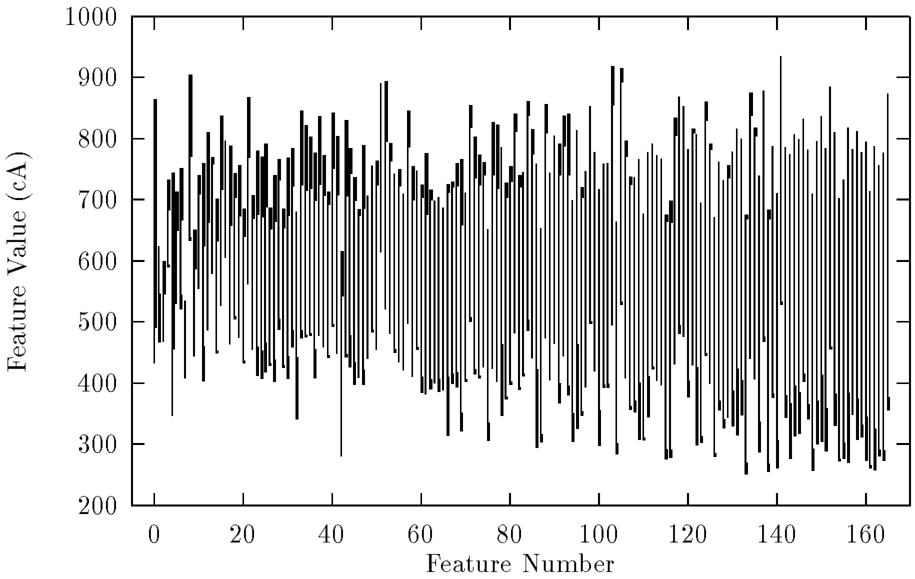

Figure 7: Lower and upper bounds for an APR enclosing only the positive feature vectors (thin vertical lines) and for an APR enclosing all feature vectors (thick vertical lines). Features are sorted by width of the positive-only APR.

few rays have tight bounds, and many of these correspond to regions of the musk molecule above and below the benzene ring where there is essentially no variation in the data set.

Figure 7 shows the actual lower and upper bounds for an APR enclosing all of the data and an APR enclosing only the positive feature vectors. All variant instances of each molecule are included. The features are sorted in ascending order of the width of the positive-only APR bounds. Note that there is only a small separation between the positive-only APR and the all-data APR. This suggests that bounds on any single ray will not eliminate very many negative molecules.

Figure 8 shows an enlarged view of the left end of Figure 7 where the positive-only bounds are tight. Notice that the positive-only upper bounds give better separation from the all-data upper bounds in this region. This suggests that features with narrow bounds are better at discriminating between positive and negative examples than features that have wide bounds.

Figure 9 shows a kernel density estimate of the positive and negative feature vectors along feature 12, which is typical of other features. A kernel density estimate is constructed by placing a small gaussian at each observed feature value and then summing those gaussians to construct a probability distribution (Silverman, 1986). We used gaussians with a standard deviation of 10cÅ.

From Figure 9 we can see that the distribution of positive and negative examples is very similar. It appears that we could separate a few negatives from the positives by placing a bound at 330. Note also that there is a large central peak. Most features have one, two, or at most three such central peaks, and the peaks of the positive and negative densities nearly always coincide. Hence, we can see that this is a very difficult learning problem.

To get a crude idea of how hard it will be to exclude all of the negative instances, we can compute the number of bounds along which each negative instance lies outside the positive-only APR. Figure 10 shows a histogram of the number of bounds that can exclude each negative. Note that several negative instances cannot be excluded at all by the positive-only APR. However, for

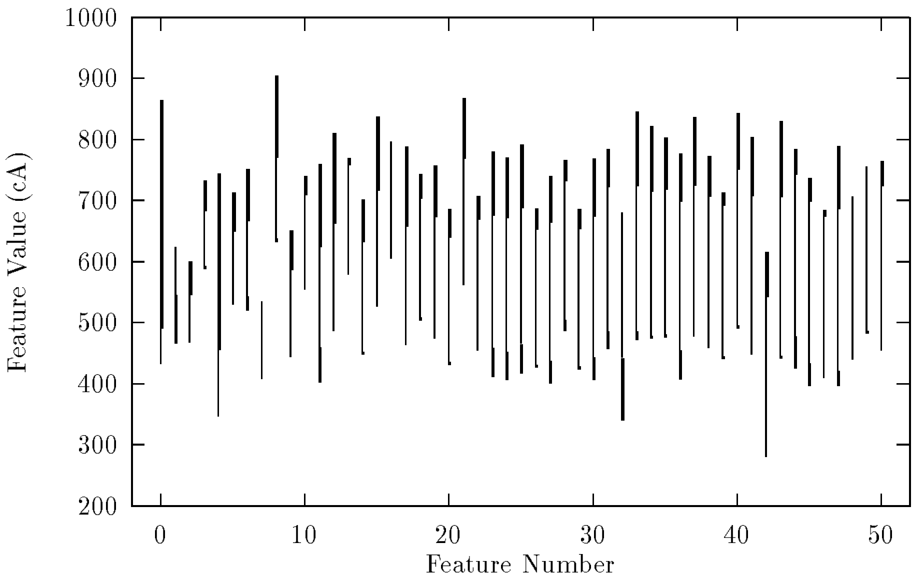

Figure 8: Expanded view of the 50 tightest bounds from Figure 7.

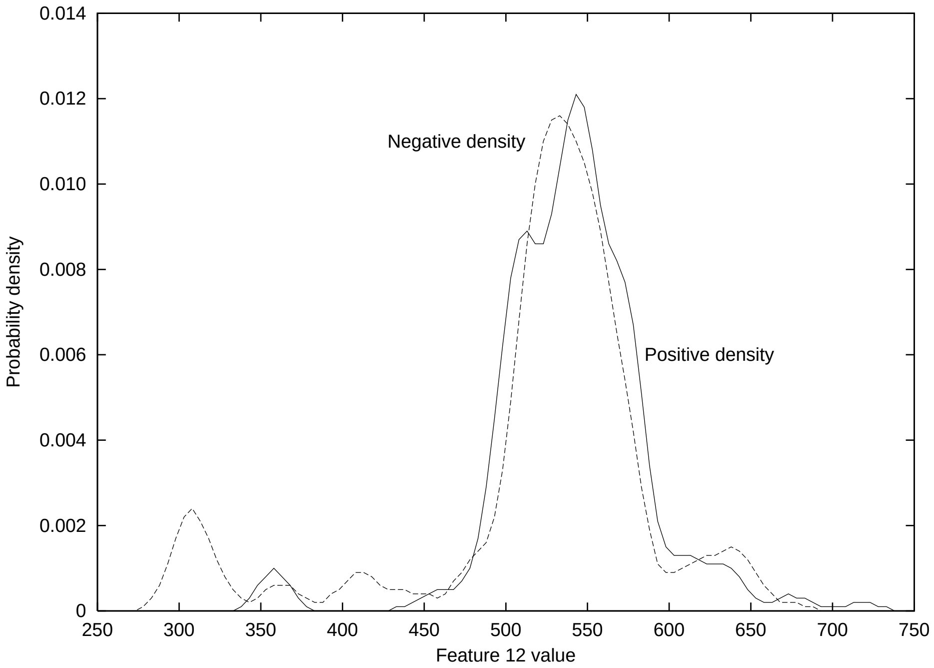

Figure 9: Kernel density estimate for positive and negative feature vectors along feature 12 in Musk Data Set 1

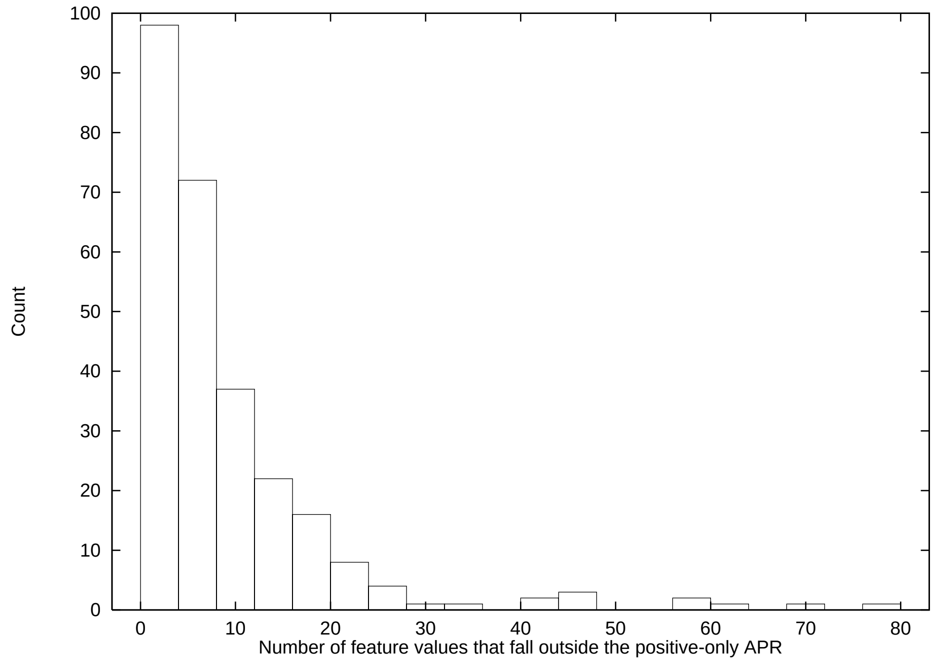

Figure 10: Histogram of the number of bounds along which a negative instance is excluded by an APR that covers all positive instances.

those that can be excluded, the number of excluding bounds is distributed roughly exponentially with mean 8.6.

In our analysis so far, we have ignored the multiple instance problem. Figure 11 is a display that helps us visualize the multiple instances of each molecule along one ray. From this display, we can see that most molecules exhibit a wide range of feature values (e.g., molecules 9 and 13). Furthermore, we can see that bounds in the neighborhood of 480 and 560 would cover at least one instance of every positive molecule and exclude all instances of several negative molecules. More precisely, we can guarantee that at least one instance of every positive molecule will be included in the bounds along dimension d if we set the lower bound lb_d and the upper bound ub_d to be

$$lb_d = \min_{i \in \text{ positive molecules}} \left[\max_j V_d(m_{i,j}) \right]$$

$$ub_d = \max_{i \in \text{ positive molecules}} \left[\min_j V_d(m_{i,j}) \right],$$

where j ranges over the variant instances of molecule m_i and $V_d(m_{i,j})$ is the value of feature d for variant instance $m_{i,j}$. We will call these bounds the minimax bounds for dimension d. If we construct minimax bounds along all dimensions for either musk data set, the resulting "minimax APR" does not include *any* positive molecules, unfortunately. This is because different instances of each molecule are chosen along different dimensions. However, we can prove the following lemma:

Lemma. *Any APR that covers at least one instance of all positive molecules must contain the minimax APR.*

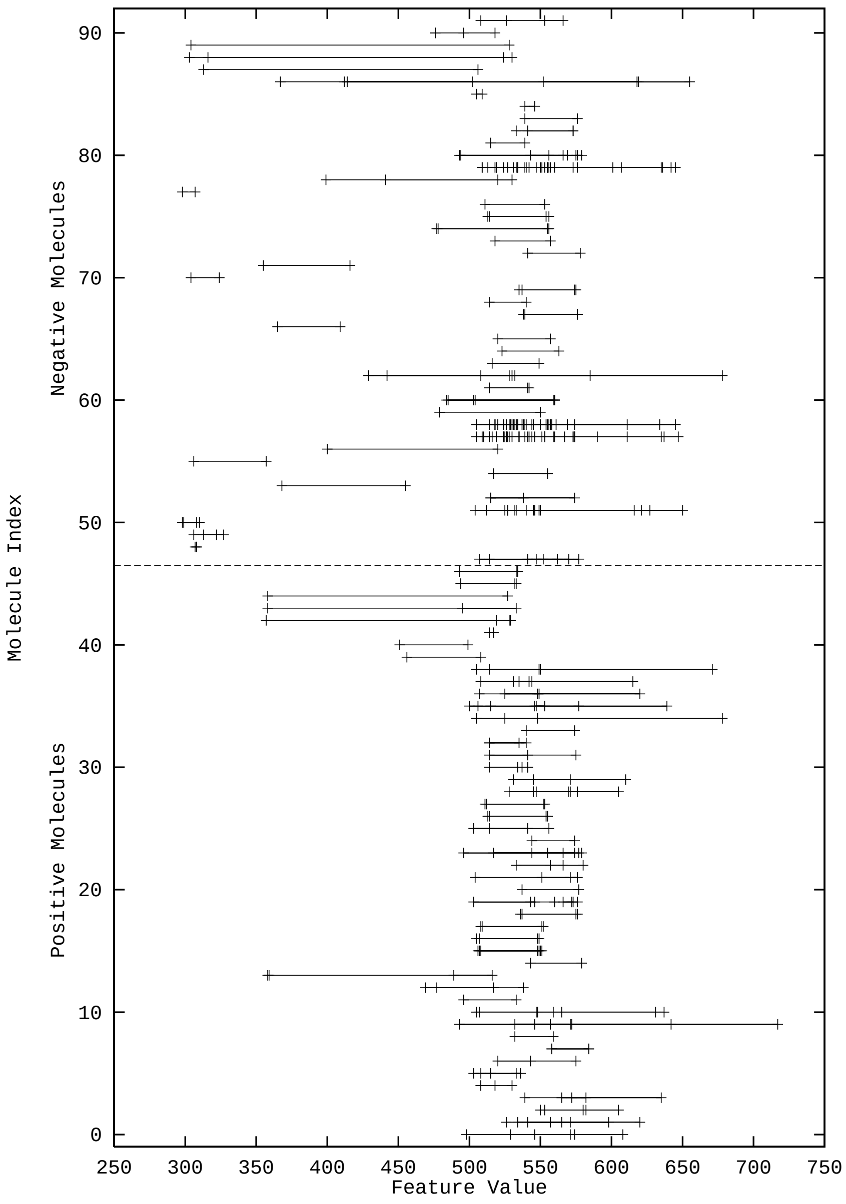

Figure 11: Display of multiple instances along feature 12, Musk Data Set 1. The horizontal axis shows the value of feature 12 for each instance. The vertical axis is an arbitrary numbering of the molecules such that all positive molecules appear below all negative molecules. Each molecule is represented by a horizontal line with vertical ticks. The ticks mark the feature values for each of the multiple feature vectors representing a single molecule. The horizontal line joins these ticks. The horizontal dashed line separates the negatives from the positives.

Proof. Suppose there was an APR that covered at least one instance of all positive molecules, but whose upper bound was less than the minimax bound along feature d. This is impossible, because by definition, there exists at least one positive molecule m for which the minimax upper bound is equal to the smallest value of feature d for any instance of that molecule. Any smaller value would not cover any instances of molecule m. The same argument applies to lower bounds. QED.

We can summarize this data analysis as follows. The distributions of positive and negative feature values are very similar, and they are (very) approximately gaussian with rather long tails. However, if we explicitly consider the multiple instance problem, we can construct fairly tight bounds that exclude many negative instances. If we take the positive-only APR as a crude approximation to the true APR, we can conclude (from Figure 10) that most negative instances are excluded along relatively few dimensions (mean 8.6).

4 An artificial data set

Based on the data analysis of the preceding section, we constructed an artificial data set as follows. First, we chose the artificial "correct" APR by forcing the first 40 features to have two-sided bounds and forcing the remaining 126 features to be irrelevant. We then applied this APR to generate random feature values, which were used to replace the feature values of Musk Data Set 1. Hence, the number of molecules and feature vectors in the artificial data set is the same as in Musk Data Set 1.

More precisely, the artificial APR, denoted $fakeAPR$, was constructed as follows. Let $allAPR$ be the all-data APR that exactly includes all of the feature vectors in Musk Data Set 1. To set the bounds for feature d (on each of the first 40 features), we first chose the width of the bounds by taking a random fraction (uniformly between 0.15 and 0.35) of the width of $allAPR$ along feature d. We then positioned an interval of this width uniformly randomly within the bounds of $allAPR$ along feature d.

The process of constructing artificial feature vectors involved first constructing two gaussian probability distributions for each feature d—one for positive feature vectors and one for negative feature vectors. The two distributions had identical means, but the standard deviation of the negative gaussian was 10% larger than the standard deviation of the positive gaussian. The standard deviation of the positive gaussian was chosen to be 0.25 times the width of $fakeAPR$ (along the first 40 features) and 0.25 times the width of $allAPR$ (along the remaining 126 features). The mean was chosen (uniformly randomly) to lie at least one standard deviation inside $fakeAPR$ (along the first 40 features) or $allAPR$ (along the remaining 126 features).

Note that because the gaussian distribution has non-zero probability everywhere, it is possible for a positive feature vector generated at random by this procedure to lie outside $fakeAPR$ and for a negative feature vector to lie inside $fakeAPR$. Hence, additional steps were taken to ensure that the artificial data was consistent with $fakeAPR$. When generating the feature vectors for a positive molecule m_i, we repeatedly generated features for the first feature vector of m_i until all generated feature values lay within the bounds of $fakeAPR$. This ensured that at least one instance of every positive molecule satisfied $fakeAPR$. When generating features vectors for a negative molecule m_i, we first determined which bounds of $fakeAPR$ would be violated (by sampling at random from an exponential distribution fitted to the data in Figure 10). We then ensured (by repeated random generation) that the feature values generated along those violated bounds did in fact lie outside $fakeAPR$.

Figure 12 shows a gaussian kernel density estimate of the positive and negative probability densities along feature 12 in the artificial data set. Note that there is a small amount of negative

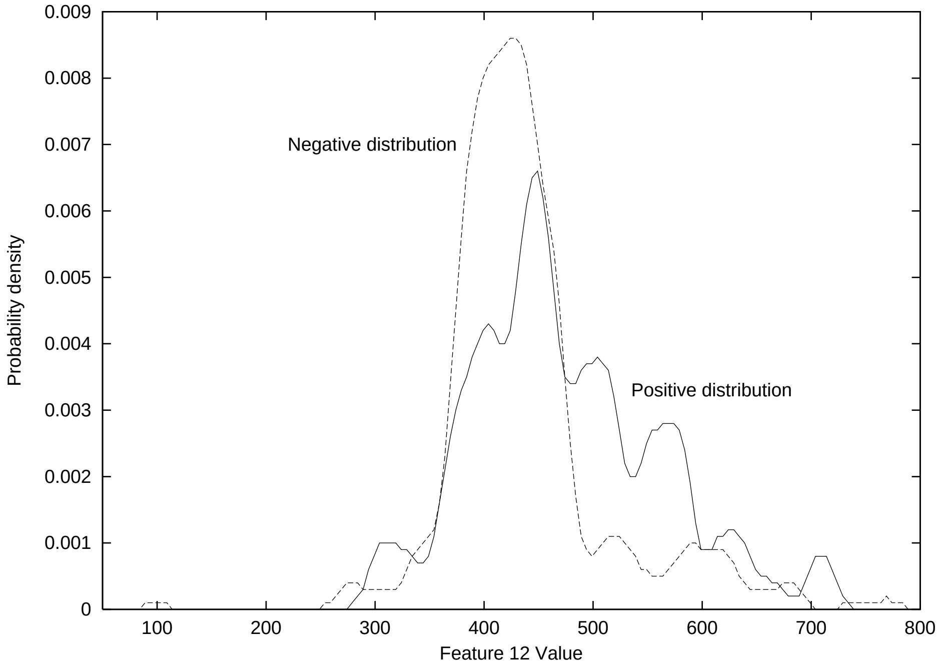

Figure 12: Artificial Data Set. Positive and negative probability densities along feature 12

density beyond the positive density and that the estimated distributions have nearly identical means. It is not surprising that from data like this it is quite difficult to discriminate the positive molecules from the negative molecules.

After constructing this artificial data set, we used it to guide the development and testing of a large variety of learning algorithms, some of which are described below. Since the artificial data set was developed from Musk Data Set 1, what is the risk that we have over-fitted the musk data sets by using this artificial data set? The answer can be determined by examining the ways in which the artificial data set is an accurate replication of the musk data sets. Certainly, the number of features, the range of feature values, and the number of conformations of each molecule are faithfully reproduced by the artificial data set. Hence, there might be some opportunities here for over-fitting to Musk Data Set 1. However, for Musk Data Set 2, the range of feature values and the number of conformations of each molecule are very different, so there is little chance of overfitting to that data set.

The feature values in the artificial data set are entirely different from either musk data set. In particular, the correlations among features are not captured at all by the artificial data set. In the real data set, the features are highly redundant, because the molecular surface does not change too much between adjacent rays. In the artificial data set, each feature value is chosen independently, so there is no correlation. Furthermore, in the musk data sets, four of the features describe the important oxygen atom, whereas in the artificial data set, there are no special features relating to the oxygen atom. For the artificial data set, we know that there exists a low-dimensional axis-parallel rectangle consistent with the data; for the real musk data sets, there could very well be no such APR.

We believe, therefore, that there is little risk of overfitting (particularly for Musk Data Set 2) from doing too many experiments with the artificial data set.

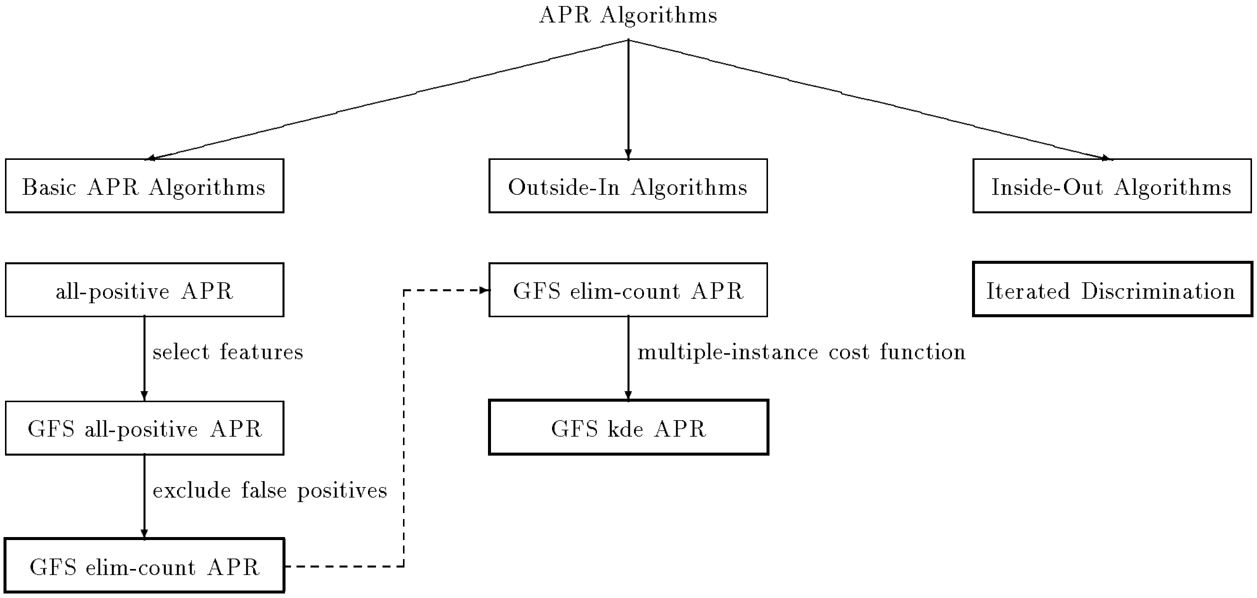

Figure 13: Relationships among the various axis-parallel rectangle algorithms described in this paper. Boxes with heavier lines mark the best algorithm of each type.

5 Three Learning Algorithms

We now present three learning algorithms and compare their performance on the artificial data set. Each algorithm illustrates a general approach to constructing APRs in the presence of the multiple instance problem. To help the reader keep track of the various algorithms presented in this section, Figure 13 gives a derivation tree that shows how the algorithms are related to one another.

Figure 14 is a schematic diagram of the multiple instance problem in two dimensions. The two coordinate axes represent the measured values of two features (i.e., measured at the point where each ray intersects the molecular surface as in Figure 3). The unfilled shapes represent feature vectors of active molecules. The filled shapes represent feature vectors of inactive molecules. All points with the same shape (e.g., all diamonds) denote feature vectors (variant instances) of the same molecule. The goal of the learning algorithms described in this section is to find a rectangle that includes at least one unfilled point of each shape (i.e., at least one feature vector of each positive molecule) and does not include any filled points (i.e., no feature vectors of any negative molecule).

It should be noted that in addition to the algorithms described here, we have experimented with a large number of variations of these methods. Each method shown here was the best representative chosen from several algorithms having the same fundamental approach.

5.1 "Standard" APR algorithms

An axis-parallel hyper-rectangle can be viewed as a conjunction of tests on the feature values. A simple algorithm can be designed by analogy with the standard algorithm for learning boolean conjunctions (Dietterich & Michalski, 1981). We simply construct the APR that exactly covers all of the positive feature vectors (the "all-positive" APR). This is the maximally-specific conjunctive generalization of the positive instances. We will call this algorithm the "all-positive APR" algo-

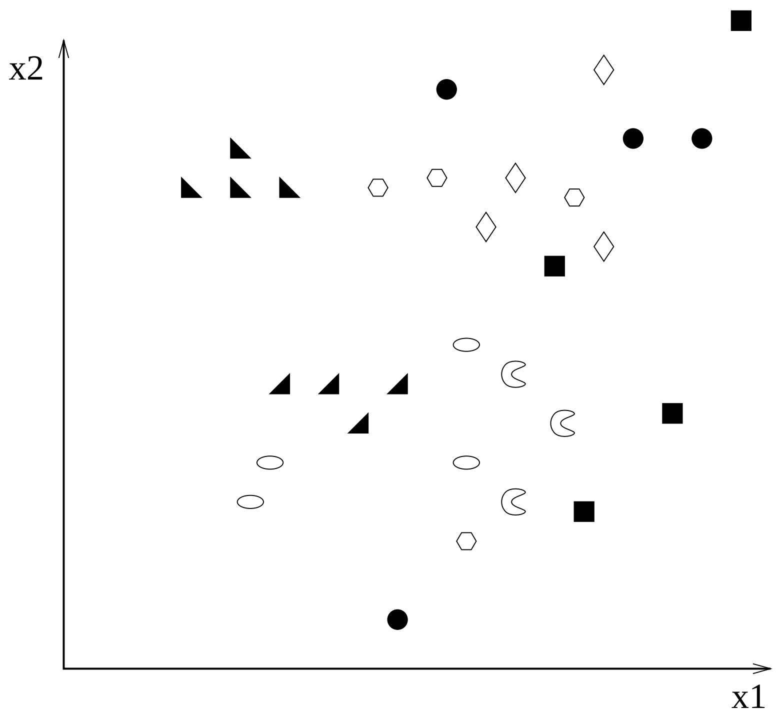

Figure 14: A multiple instance learning problem. Unfilled shapes represent feature vectors of active molecules; filled shapes represent feature vectors of inactive molecules. All points of the same shape denote feature vectors of the same molecule.

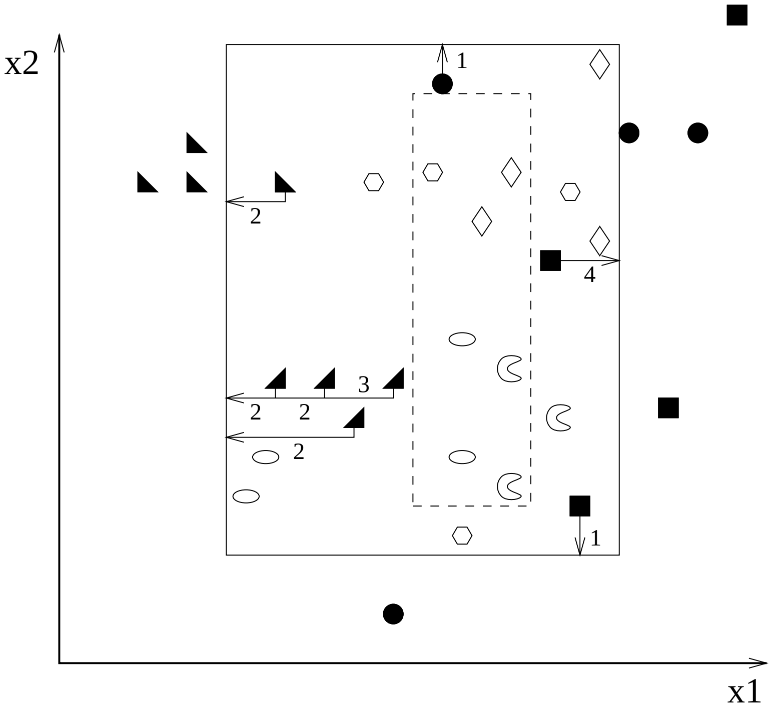

Figure 15: The Elim-Count procedure for identifying the negative instance that is easiest to exclude. The solid rectangle is the smallest rectangle that includes all unfilled points (the all-positive APR). The line next to each included filled point indicates which side of the solid box will be tightened to exclude that point. The number indicates how many unfilled points will also be excluded when the box is tightened. The dashed box indicates the final APR produced by Elim-Count.

rithm. Figure 15 shows the resulting all-positive APR as a solid line bounding-box of the unfilled points.

However, this APR treats every feature as relevant. This is unlikely to be a good hypothesis in this domain, because feature values from nearby rays are highly correlated and because not all parts of the molecular surface are likely to be involved in binding. The obvious next step is to choose a subset of the bounds of this APR that are sufficient to exclude all of the negative instances. This is analogous to the method described by Haussler (1988, 1989). The process of removing bounds from the APR is best organized as a process of adding bounds to an APR that covers the entire feature space. A greedy algorithm considers each bound from the all-positive APR and chooses the bound that eliminates the most negative instances. This bound is then added to the solution, the negative instances are eliminated, and the process repeated until no negatives remain to be eliminated. We will call this the "GFS all-positive APR" algorithm, since it performs Greedy Feature Selection.

One difficulty with this algorithm is that the all-positive APR may contain several negative examples. Figure 15 shows the all-positive APR as a solid line bounding-box of the unfilled points. Note that 8 filled shapes (8 negative instances) are included in this APR. Greedy feature selection to choose a subset of the bounds cannot eliminate any negative examples that were already covered by the all-positive APR. To eliminate these negative instances, we can apply the following greedy algorithm prior to selecting relevant features. For each negative instance, count the minimum number of positive instances that must be excluded from the APR in order to exclude the negative instance. In the figure, these counts are shown next to small lines that indicate which "side" of the APR would be tightened in order to exclude the negative instance. For example, to eliminate the black circle at the top of the APR, we would also need to eliminate the one white diamond in the upper right corner of the APR. The greedy algorithm iteratively chooses to eliminate the negative instance that is easiest to eliminate (i.e., requires eliminating the fewest covered positive instances) until all negatives are eliminated. The resulting APR is the inner, dashed, box in Figure 15. Note that in this example, the resulting APR covers at least one instance of each positive example, but in general, this need not be the case. All instances of a positive example might be excluded by this greedy algorithm. After constructing this APR, we can apply the greedy feature selection algorithm described above to obtain an APR that is guaranteed to have no false positives. We will call this the "GFS elim-count APR" algorithm, because it eliminates negative instances based on counting the number of positive instances that also need to be eliminated (and it performs greedy feature selection).

5.2 An "outside-in" multiple-instance APR algorithm

We can modify the GFS elim-count algorithm to address the multiple instance problem. Instead of merely counting the number of positive instances that must be excluded in order to exclude a negative instance, we can consider excluding positive instances that are "expendable" in the sense that every positive molecule still has at least one positive instance covered by the APR. We implement this by defining a "cost" of excluding each positive instance and choosing to exclude cheap positive instances, as necessary, in order to exclude a covered negative instance.

The cost of excluding a positive instance $m_{i,j}$ of molecule m_i must therefore depend on the other not-yet-excluded positive instances of molecule m_i. If there are many other such surviving positive instances, then the cost of excluding $m_{i,j}$ can be small, because it is less likely that subsequent decisions will end up excluding *all* instances of m_i.

Furthermore, if the other surviving positive instances have feature values that are frequently observed among the positive molecules, then it is likely that those survivors are the most relevant instances and that instance $m_{i,j}$ should be eliminated.

Finally, if variant instance $m_{i,j}$ is very isolated so that few other positive instances have feature values similar to it, then it is probably not an instance that should be included in the APR.

To develop a cost function that incorporates these properties, we employed an estimate of the probability density D_d of all surviving positive instances along feature d. As before, we applied gaussian kernel density estimation to estimate this density. A small gaussian kernel is centered at each value $V_d(m_{i,j})$ of feature d for each surviving instance $m_{i,j}$ of each positive molecule i. These gaussians are then summed to obtain the probability density function D_d. The notation $D_d(m_{i,j})$ indicates the probability (according to D_d) of observing the value $V_d(m_{i,j})$ of feature d.

Let $m_{i,1}, \ldots, m_{i,\nu_i}$ be the surviving variant instances of positive molecule m_i. The cost of eliminating instance $m_{i,j}$ along feature d is defined as

$$\left(- \sum_{l=1, l \neq j}^{\nu_i} D_d(m_{i,l}) \right) + \alpha D_d(m_{i,j}). \tag{1}$$

The first term gives a very low cost if there are many other instances (indexed by l) of the same molecule and they have high probability according to the density estimate D_d. The final term $\alpha D_d(m_{i,j})$ measures the degree to which instance j is isolated from the other positive instances. If $D_d(m_{i,j})$ is small, instance j is very isolated. In our experiments, we used $\alpha = 10.0$. If an instance is the only remaining instance of a molecule, then it receives a cost of 10.0, which is very high. (We experimented with a few other cost functions, but this one worked slightly better than the others.)

We can now apply the same algorithm as "elim-count", except that at each point we choose the negative instance that is cheapest to eliminate (because it eliminates inexpensive positive instances). The algorithm will avoid eliminating the last covered instance of a positive molecule unless the alternative would be even more expensive. We call this algorithm "GFS elim-kde", because it eliminates negative instances based on a kernel density estimate (kde) of the positive instances.

One drawback of GFS elim-kde is that it is quite expensive to compute all of the required kernel density estimates. Let n be the number of features, ν be the number of instances of each positive molecule, N^+ and N^- be the total number of positive and negative instances (respectively), and p be the number of times we must compute the cost with Equation (1). Then the computational cost is bounded by $O(p\nu N^+)$, because we must compute ν kernel density estimates, and each such estimate must process each positive instance. However, p can be quite large itself, since each negative instance may exclude several positive instances along each of the n dimensions. On average in Musk Data Set 1, 0.53 positive instances are excluded along each dimension; in Musk Data Set 2, 0.28 positive instances are excluded along each dimension (these numbers reflect the application of a heuristic to avoid wasting effort on "bad" features). In either case, this means p is still $O(N^-n)$, so the overall computational cost is approximately $O(n\nu N^-N^+)$, which is immense. For Musk Data Set 1, this is approximately 2.0×10^7. For Musk Data Set 2, this is 9.0×10^8.

5.3 An "inside-out" multiple-instance APR algorithm

An alternative to the "outside-in" approach is to construct an APR by starting with a single positive instance and "growing" the APR by expanding it to cover additional positive instances. We have constructed a somewhat complex algorithm based on this approach. We call it the "Iterated Discrimination" algorithm, and it has three basic procedures:

- **Grow:** An algorithm for growing an APR with "tight" bounds along a specified set of features.
- **Discrim:** An algorithm for choosing a set of discriminating features by analyzing an APR.
- **Expand:** An algorithm for expanding the bounds of an APR to improve its generalization ability.

The algorithm works in two phases. In the first phase, the Grow and Discrim procedures are iteratively applied to simultaneously choose a set of discriminating features and construct an APR that has "tight" bounds along those features. In the second phase, the Expand procedure is applied to expand these tight bounds.

We describe each of these procedures in turn.

5.3.1 An algorithm for growing a tight APR

The goal of this algorithm is to find the smallest APR that covers at least one instance of every positive molecule. Let us define the size of an APR as the sum of the widths of all of its bounds:

$$Size(APR) = \sum_d ub_d - lb_d.$$

Many different optimization methods can be applied to this problem, and we have tested simulated annealing, a greedy algorithm, and a backfitting algorithm. We describe the greedy and backfitting procedures here.

We begin the optimization by choosing an initial "seed" positive instance. The greedy procedure then grows the APR by a series of greedy steps. In each greedy step, it identifies the positive instance (of a not-yet-covered positive molecule) that when added to the APR would least increase its size. The APR is then expanded to include that positive instance. This procedure is continued until at least one instance of each positive molecule is covered. Surprisingly, in all of our experiments, the resulting APR was consistent (i.e., it covered no instances of any negative molecules), although this is not required for any of our algorithms.

The backfitting algorithm, which is an extension of the greedy procedure, was first employed in the Projection Pursuit method (Friedman & Stuetzle, 1981). It begins like the greedy algorithm by choosing a seed positive instance and then taking a series of greedy steps. However, after each greedy step, it revisits all previous decisions to consider whether they would be made differently.

Suppose the algorithm has just taken a greedy step to make the l-th decision. Let I_d be the positive instance chosen in step t. The backfitting procedure revisits each of the previous decisions $1, \ldots, l$. When previous decision t is revisited, the algorithm constructs the APR that covers $\{I_1, \ldots, I_{t-1}, I_{t+1}, \ldots, I_l\}$. We will call this APR A^{-t}. It then considers all instances of the same molecule as I_t and identifies the instance that would least increase the size of A^{-t}. This chosen instance replaces I_t in the list of choices made by the algorithm.

The backfitting algorithm then reconsiders choice $t+1$ and so on. It makes repeated passes through the first l choices until progress ceases (i.e., no decisions are changed). It then makes a greedy step to choose the $l+1$st positive instance.

To choose the initial positive instance, we select the positive instance that is closest (in Euclidean distance) to the minimax-APR defined earlier.

Experiments with Musk Data Set 1 and the Artificial Data Set showed that the backfitting procedure always found smaller, tighter APRs than either the greedy procedure or a simulated annealing method. Hence, we have applied the backfitting procedure in all of the experiments reported in this paper.

5.3.2 An algorithm for selecting discriminating features

Once a tight APR has been constructed, it can be applied to select discriminating features. We employed the following greedy algorithm. Let us say that feature d of the tight APR *strongly discriminates against* a negative instance if either (a) that instance lies more than 1Å outside the bounds of the APR along feature d or (b) that instance lies beyond the bounds of the APR, and it lies further outside the bounds along feature d than along any other feature.

With this definition of "strongly discriminate", we then choose features iteratively by selecting in each step the feature that strongly discriminates against the largest number of negative instances. Those negative instances are then removed from further consideration, and the process is repeated until enough bounds have been selected so that all negative instances are excluded.

The rationale for the 1Å "margin" beyond the APR is that in our experiments with the Artificial Data Set, irrelevant features were found to discriminate many training-set negative instances, but by only small margins. Hence, small values for the margin do not robustly identify the relevant features. Various other margins were considered, and the results are relatively insensitive to this parameter, as long as it is larger than 0.5Å.

5.3.3 Iterative selection of positive instances and discriminating features

The Iterated Discrimination algorithm alternates the application of these first two algorithms as follows. First, the backfitting algorithm is applied to construct a tight APR with bounds on *all* features. Then a subset of those features is selected as discriminating features. The backfitting algorithm is applied again to construct a tight APR, but this time, it only measures the size of the APR along the *discriminating* features. This can cause it to choose different positive instances (and hence, different bounds). The feature selection procedure is again invoked to further narrow the set of discriminating features. This back-and-forth loop continues until it converges (which typically requires 3–4 iterations).

It is interesting to compare the behavior of the backfitting algorithm on the first iteration with its behavior in subsequent iterations. During the first iteration, more than half of the features are irrelevant, and yet the backfitting algorithm is trying to minimize the total size of the APR. The APR bounds tend to be wider along the irrelevant features than along the relevant ones (because the surface of the positive molecules is more variable along those rays). Hence, when the irrelevant features are discarded, the backfitting algorithm can alter its choices quite substantially. During subsequent iterations, the choice of relevant features hardly changes at all. As a result, the choice of positive instances does not change much either, so that the algorithm converges rapidly. We expect this to be generally true for this kind of algorithm (more details are given below in Section 6.2).

5.3.4 Expanding the APR to improve generalization

Experiments on the Artificial Data Set showed that using the tight APR resulting from the previous two methods does not generalize well. It is typically so tight that it excludes most positive instances in the test set (as well as all negative instances). The problem is that the APR was constructed to be exactly big enough to cover at least one positive instance of each molecule in the training set, but no bigger.

To overcome this problem, we once more turned to kernel density estimation. For each relevant feature of the tight APR, we apply kernel density estimation to estimate the probability that a positive instance will satisfy the bounds on that feature. Our goal is to expand the bounds so that with high probability, new positive instances will fall inside the APR.

This algorithm is controlled by two user-specified parameters: ϵ and τ. The ϵ parameter specifies the amount of probability that should lie outside the expanded bounds of the APR along each feature dimension. The τ parameter determines the width of the gaussian kernel. The width of the kernel is set so that if all of the positive instances were centered between the APR bounds, the kernel density estimator would conclude that τ of the probability lay within the bounds. Hence, if the positive instances were normally distributed, $\epsilon + \tau = 1$.

For each relevant dimension d of the APR, the algorithm first computes the width of the kernel using τ. It then expands the lower and upper bounds of the APR along d so that $\epsilon/2$ probability lies below the lower bound (according to the kernel density estimate) and $\epsilon/2$ probability lies above the upper bound. If the tail of the kernel density estimate contains less than $\epsilon/2$ probability, then that bound is not changed.

This is illustrated in Figure 16. Here, the tall vertical lines show the initial, tight bounds of the APR. The medium-height vertical lines show the expanded bounds—they have moved outward so that the probability under each tail of the density estimate is exactly $\epsilon/2 = 0.01$. Note that the lower bound has expanded considerably, because there are many positive instances very close to the tight lower bound (and consequently a larger tail). The upper bound has hardly expanded at all, because the positive instances were already quite sparse near the tight bound.

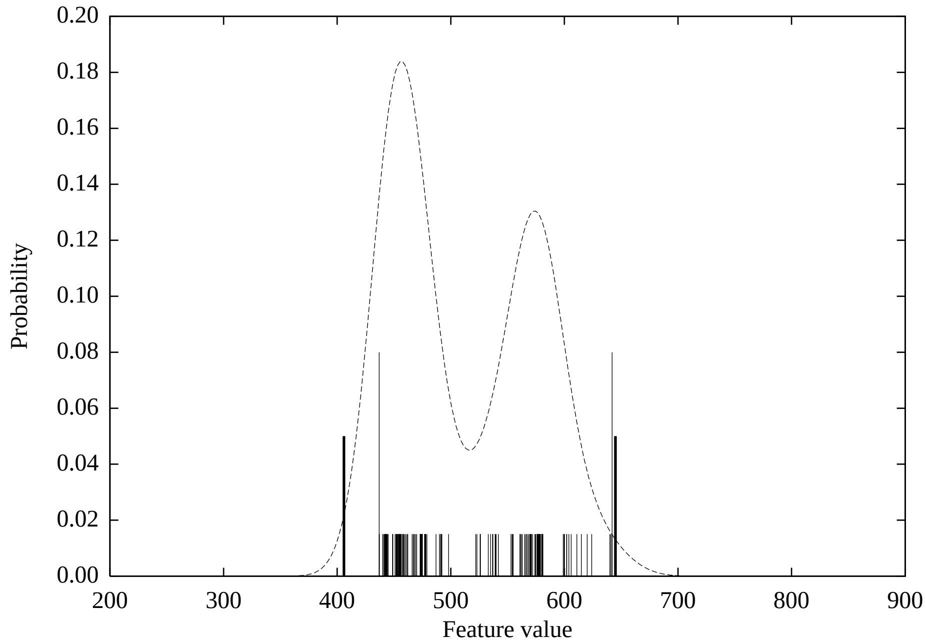

Figure 16: Expanding APR bounds via kernel density estimation (Musk Data Set 1). Small vertical lines mark the value of one feature for each positive feature vector. Curve shows an estimate of the probability density of these values. Tall vertical lines show bounds of initial, tight APR. Thicker medium vertical lines show the expanded bounds.

Table 2: Artificial Data Set performance. 92 training set molecules, 500 test set molecules.

Algorithm	True Positives	False Negatives	False Positives	True Negatives	Errors	% Correct
iterated-discrim APR[a]	237	22	45	196	67	86.6
backpropagation[b]	248	11	120	121	131	73.8
GFS elim-count APR	244	15	126	115	141	71.8
C4.5 (pruned)	249	10	132	109	142	71.6
GFS elim-kde APR	239	20	124	117	144	71.2
all-positive APR	187	72	75	166	147	70.6
GFS all-positive APR	258	1	157	84	158	68.4

[a] $\tau = 0.999$, $\epsilon = 0.002$

[b] 127 hidden units, learning rate 0.1, momentum 0.6, 600 epochs

6 Experimental Results

6.1 Results on the Artificial Data Set

Table 2 shows the results of running each of these APR algorithms on the Artificial Data Set. In addition, we show the results of the C4.5 decision tree algorithm and the backpropagation neural network algorithm, both of which ignored the multiple-instance problem and treated all of the positive instances as positive examples.

To evaluate the generalization performance, we constructed 500 additional molecules. Each new molecule was generated by randomly choosing (with replacement) a molecule from Musk Data Set 1 and replacing its feature vectors with artificial feature vectors. Hence, the proportion of positive and negative molecules, and the number of feature vectors per molecule are the same in the artificial test set as in the artificial training set.

For algorithms that require the user to specify various parameters, we chose the parameters that gave the best test set performance. For Iterated Discrimination, the best parameter values were $\tau = 0.999$ and $\epsilon = 0.002$. For backpropagation, we conducted a systematic search for good parameters using an Adaptive Solutions CNAPS computer with 128 processors (in 32-bit mode). The best parameters employed a single hidden layer of 127 sigmoid units, learning rate 0.1, momentum 0.6 (with weight updates applied once every 16 patterns), and trained for 600 epochs.

The results clearly show that the Iterated-Discrimination algorithm is far superior to all of the other algorithms tested. The performance of Iterated Discrimination is statistically significantly different from all of the other algorithms ($p < 0.001$, binomial test for the difference of two proportions). None of the other algorithms can be distinguished statistically from each other (at $p < 0.05$).

It is not surprising that the "all-positive APR" and the "GFS elim-count APR" algorithms did poorly, since neither of these addresses the multiple instance problem. However, it is somewhat surprising that the "GFS elim-kde APR" algorithm did so badly, since it *does* explicitly consider the multiple instance problem in deciding which positive instances to remove from its APR. We experimented with many variations of the "GFS elim-kde APR" algorithm. For example, one variation required that each negative instance be discriminated from the positive instances by a "margin of safety." This hurt performance (false negatives increased without decreasing false positives). Another variation we explored was to employ various methods of expanding or shrinking the APR both before and after feature selection. None of these worked as well as the GFS elim-kde algorithm described above.

Table 3: Artificial Data Set. Success in identifying the correct relevant features.

Algorithm	True Relevants	False Irrelevants	False Relevants	True Irrelevants	Errors	% Correct
iterated-discrim APR	38	2	18	108	20	88.0
GFS all-positive APR	18	22	12	114	34	79.5
GFS elim-kde APR	23	17	43	83	60	63.9
GFS elim-count APR	24	16	37	89	53	68.1
all-positive APR	40	0	126	0	126	24.1

An advantage of artificial data is that we can measure more than generalization performance. Table 3 shows how well each algorithm did at choosing the 40 correct features. We can see that the Iterated Discrimination algorithm was the most successful at identifying relevant features. All of the other algorithms (except the all-positive APR), could only correctly identify about half of the relevant features, whereas Iterated Discrimination found 38 of the 40 features.

From this data we conclude that the superior performance of "Iterated Discrimination APR" is primarily explained by its ability to find the relevant features. This in turn appears to be the result of its approach of first finding a very tight APR that covers at least one positive instance of every positive molecule. Along the relevant features, this tight APR is much tighter than along the irrelevant dimensions. In contrast, the bounding box of all of the positive examples—which is the starting point for all of the other algorithms—has wide bounds along both the relevant and irrelevant dimensions. Figure 17 shows the widths of the bounds for the all-positive APR (as a percentage of the widths of the bounds of all the feature vectors). The first 40 features are the true relevant features. Some of them clearly have tighter bounds than the irrelevant features, but most of them have wide bounds indistinguishable from the irrelevant features.

In contrast, Figure 18 shows the widths of the bounds for the tight APR constructed by the backfitting algorithm during the first iteration of Iterated Discrimination. The relevant features are clearly identified. It should be noted, however, that on the real musk data sets, the analogous plot does not separate the features nearly so clearly.

A drawback of the Iterated Discrimination algorithm is the need to choose values for τ and ϵ. Figure 19 shows how performance on the Artificial Data Set varies as a function of these two parameters. In general, as τ increases, the best choice for ϵ decreases. This is what we would expect: large values of τ mean that the gaussian kernel is smaller, and hence, the probability under the tails of the distribution decreases. To sufficiently widen the APR bounds, the desired probability under the tails (ϵ) must decrease too. The peak performance of Iterated Discrimination on the artificial data was obtained with $\tau = 0.999$ and $\epsilon = 0.002$.

Unfortunately, these parameter values cannot be confidently extrapolated to Musk Data Set 1, because the artificial data were generated using gaussian distributions, and these match the gaussian kernel density estimator very well. Hence, we need to look at Musk Data Set 1 to choose the best parameters.

We can summarize our analysis of the Artificial Data Set as follows:

- The artificial data pose a difficult learning problem. Neither backpropagation nor C4.5 can perform well on this data set.

- Algorithms that ignore the multiple instance problem do not perform well.

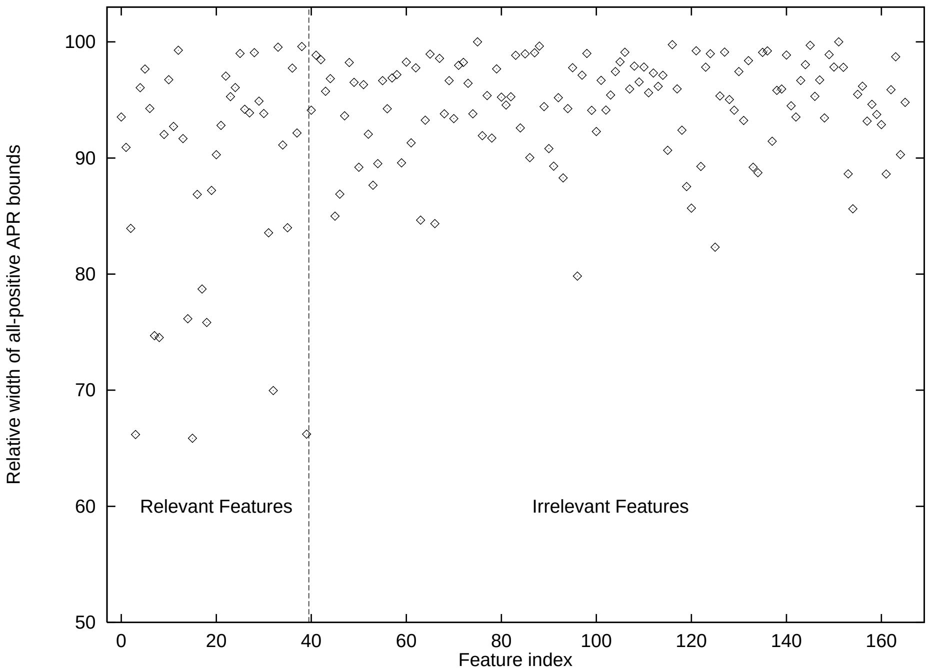

Figure 17: Bound widths for the all-positives APR on the Artificial Data Set. Vertical bar separates the relevant features (0–39) from the irrelevant features (40–165).

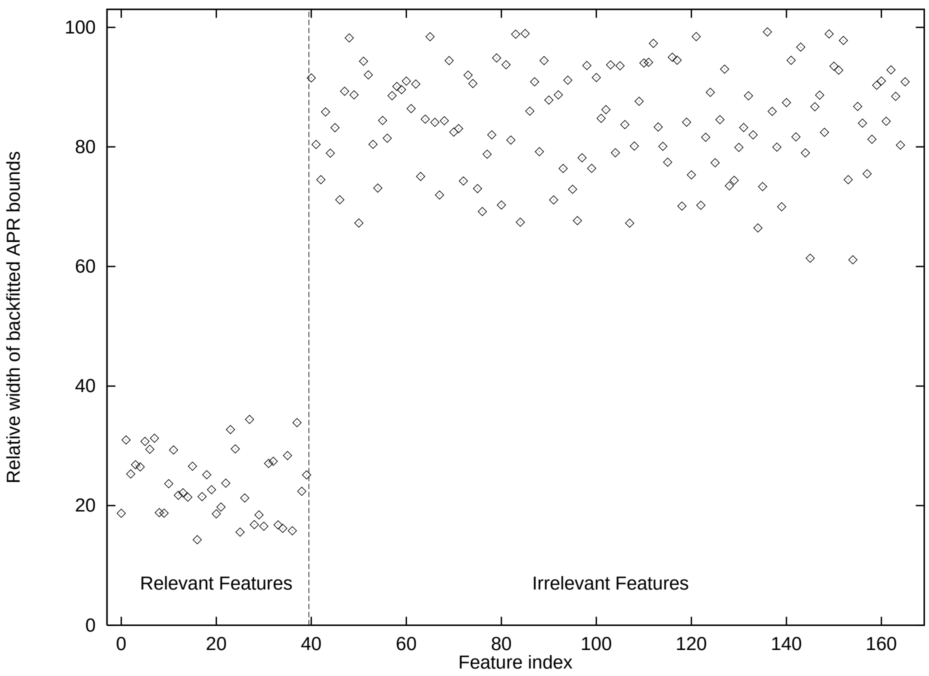

Figure 18: Bound widths for the backfitted APR on the Artificial Data Set. Vertical bar separates the relevant features (0–39) from the irrelevant features (40–165).

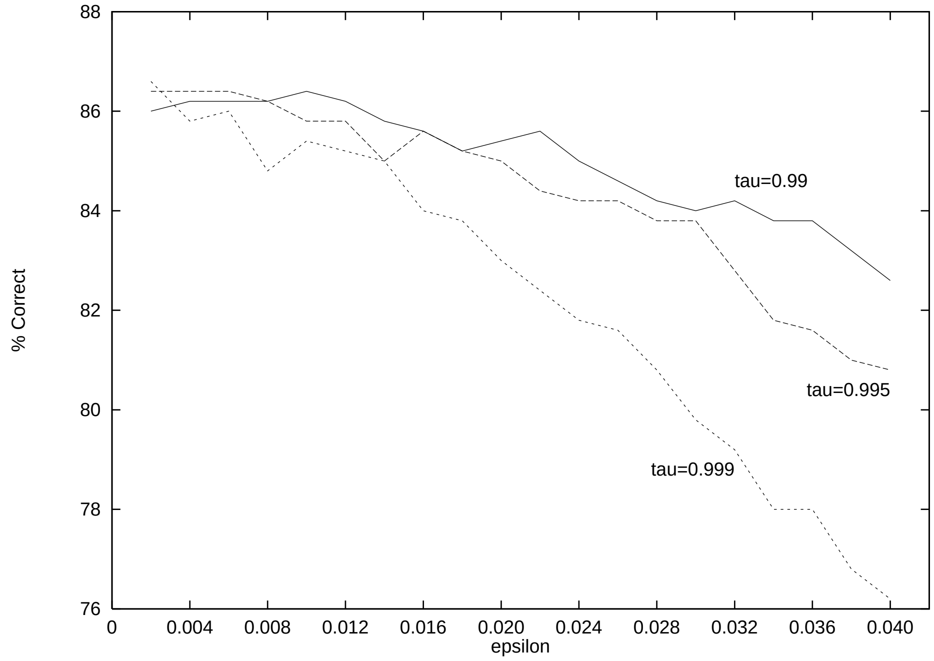

Figure 19: Performance of Iterated Discrimination on the Artificial Data Set as a function of its parameters ϵ and τ.

- The "inside-out" approach to constructing APR's performs best. This is a result of its superior ability to identify discriminating features.

6.2 Results on Musk Data Set 1

We now turn to the first of the two musk data sets. Table 4 summarizes the performance of the algorithms on this data as measured by 10-fold cross-validation. A 95% confidence interval is shown in the last column. From this, we can see that the top three methods are statistically indistinguishable. This probably reflects the difference between the Artificial Data Set, where features are either relevant or irrelevant, and the musk data sets, where more features are (probably) measuring relevant parts of the molecular surface, but where these measurements are redundant.

As with the Artificial Data Set, we have chosen parameter values for Iterated Discrimination and Backpropagation to optimize cross-validated accuracy. For backpropagation, we found that it was exceedingly difficult to train a network. Parameters that worked for the Artificial Data Set did not work for Musk Data Set 1. We believe this primarily reflects the fact that the bias of standard multi-layer sigmoid units is not appropriate for learning axis-parallel rectangles. Note that all of the APR methods performed better than the non-APR methods.

The real musk data in Table 4 demonstrate even more clearly than the artificial data in Table 2 the importance of not ignoring the multiple instance problem. The top two algorithms in the table are the two methods that explicitly attempt to solve the multiple-instance problem. As with the Artificial Data Set, the best algorithm is Iterated Discrimination, which makes only 7 errors on test molecules. Note that all of the non-multiple-instance algorithms (except the all-positive APR) have high false positive rates. This is to be expected, since if they mistakenly classify *any* feature

Table 4: 10-fold cross-validation performance on Musk Data set 1 (92 molecules)

Algorithm	True Positives	False Negatives	True Positives	False Negatives	Errors	% Correct
iterated-discrim APR[a]	42	5	2	43	7	92.4 [87.0–97.8]
GFS elim-kde APR	46	1	7	38	8	91.3 [85.5–97.1]
GFS elim-count APR	46	1	8	37	9	90.2 [84.2–96.3]
GFS all-positive APR	47	0	15	30	15	83.7 [76.2–91.2]
all-positive APR	36	11	7	38	18	80.4 [72.3–88.5]
backpropagation[b]	45	2	21	24	23	75.0 [66.2–83.8]
C4.5 (pruned)	42	5	24	21	29	68.5 [40.9–61.3]

[a] $\epsilon = 0.01, \tau = 0.999$ or $\epsilon = 0.02$ and $\tau = 0.995$.

[b] 127 hidden units, learning rate 0.001, no momentum, 950 epochs.

vector of a molecule as positive, then the molecule is classified as positive.

This partly explains the particularly poor performance of backpropagation and C4.5. Additionally, those algorithms do not have the advantage of knowing that good hypotheses should take the form of axis-parallel rectangles. Hence, even though both C4.5 and backpropagation can represent APR's, they choose other, less appropriate, hypotheses in this domain.

Table 5 gives some insight into the behavior of the Iterated Discrimination algorithm. For each fold of the 10-fold cross-validation, this table shows how the number of relevant dimensions and the set of selected positive instances changes. First note that only two or three iterations are performed by the algorithm in each fold. The choice of relevant dimensions is essentially unchanged after the first iteration. This shows the critical importance of the heuristic for selecting relevant dimensions. The choice of relevant dimensions has a big influence on which positive instances are chosen by the backfitting algorithm to be the "active" variants. In the second iteration, the algorithm changes its choice for 15–25% of the positive molecules.

Figure 20 shows the sensitivity of Iterated Discrimination to the choice of parameter values. There are several things to note. First, larger values of τ give better performance. Second, the values of ϵ giving peak performance are quite wide, especially for $\tau = 0.999$. When we compare this figure to Figure 19, we see that unfortunately, the Artificial Data Set does not accurately predict the point of peak generalization—it suggests much smaller values for ϵ (which correspond to much wider APR bounds). Similarly, we will see below that Musk Data Set 1 does not accurately predict the point of peak generalization for Musk Data Set 2.

What values of τ and ϵ shall we choose for Musk Data Set 2? Based on the wide plateau for $\tau = 0.999$, it is an obvious choice. However, with the wide plateau, it isn't clear what value of ϵ to choose. One thing to consider is that Musk Data Set 2 has only 39 positive molecules instead of the 47 positive molecules in Musk Data Set 1. This will mean that the tight APR produced by backfitting will have narrower bonds on Musk Data Set 2 than on Musk Data Set 1. This suggests that we choose smaller values of ϵ, because those will produce wider APR bounds. Hence, we will choose $\epsilon = 0.01$, since it is the smallest value giving peak performance on Musk Data Set 1.

Note that any method of choosing parameters for Musk Data Set 2 based on experiments with Musk Data Set 1 risks some overfitting, because the two data sets share many molecules. However, because Musk Data Set 2 has many more conformations and, as we shall see below, because the best parameters turn out not to be those that worked best for Musk Data Set 1, we believe the degree of "contamination" is small.

Table 5: Musk Data Set 1. 10-fold cross-validation. For each fold, the left half of the table indicates the change in the number of relevant dimensions (starting with 166) with each iteration of the Iterated Discrimination algorithm. The right half of the table indicates how many of the instances selected by backfitting changed in each iteration. The values for iteration 1 show the number of positive molecules in the training set (and hence, the total number of selected instances).

	Change in Relevant Dimensions			Change in Selected Instances		
	Iteration:			Iteration:		
Fold	1	2	3	1	2	3
0	−142	−1	0	42	6	1
1	−145	0		42	10	
2	−141	0		42	2	
3	−140	0		42	7	
4	−143	0		42	8	
5	−143	0		42	8	
6	−142	0		42	6	
7	−142	−1	0	43	7	2
8	−144	0		43	6	
9	−141	0		43	8	

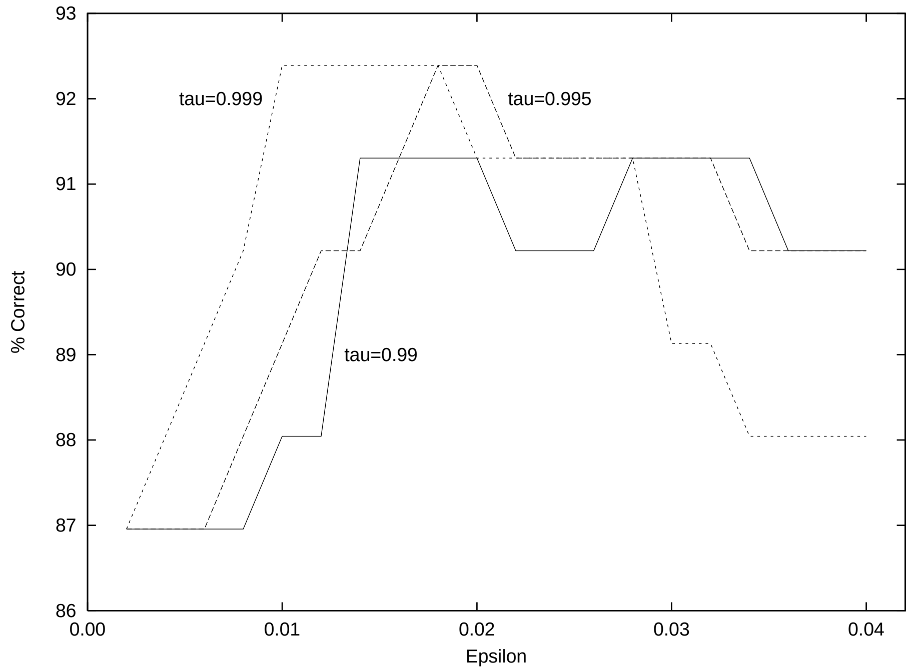

Figure 20: Performance of Iterated Discrimination on Musk Dat Set 1 as a function of τ and ϵ.

Table 6: 10-fold cross-validation performance on Musk Data Set 2 (102 molecules)

Algorithm	True Positives	False Negatives	True Positives	False Negatives	Errors	% Correct
iterated-discrim APR[a]	30	9	2	61	11	89.2 [83.2–95.2]
GFS elim-kde APR	32	7	13	50	20	80.4 [72.7–88.1]
GFS elim-count APR	31	8	17	46	25	75.5 [67.1–83.8]
all-positive APR	34	5	23	40	28	72.6 [63.9–81.2]
backpropagation[b]	16	23	10	53	33	67.7 [58.6–76.7]
GFS all-positive APR	37	2	32	31	34	66.7 [57.5–75.8]
most frequent class	0	39	0	63	39	61.8 [52.3–71.2]
C4.5 (pruned)	32	7	35	28	42	58.8 [49.3–68.4]

[a] $\tau = 0.999, \epsilon = 0.01$

[b] 127 hidden units, learning rate 0.20, no momentum, 60 epochs

6.3 Results on Musk Data Set 2

Table 6 shows the final results for running all algorithms on the very large Musk Data Set 2. Again we have shown a 95% confidence interval for the percentage of correct classification by each algorithm. Once again we encountered extreme difficulty in training a multi-layer sigmoid net on this data set. None of the parameter values that worked for either Musk Data Set 1 or the Artificial Data Set resulted in any effective training for this data set (e.g., squared error did not decrease or else behaved chaotically). The parameters employed here (127 hidden units, 60 epochs, learning rate 0.2, no momentum) where chosen by performing a cross-validation within the training set of the first fold of the 10-fold cross-validation. This is not strictly fair, since the parameter choice should have been repeated within the training set of *each* fold of the cross-validation. In any case, however, the performance is very poor.

The relative ranking of the algorithms is almost the same as with Musk Data Set 1 (the "GFS all-positive APR" has dropped below "backpropagation"). However, the gap between Iterated Discrimination and the other algorithms has increased so that it is now statistically significant ($p < 0.10$). All of the APR algorithms except for "GFS all-positive APR" outperform the non-APR algorithms (C4.5 and backpropagation). C4.5 even performed worse (on percent-correct basis) than the trivial strategy of guessing the most frequent class, although the difference is not statistically significant.

Figure 21 shows the cross-validated performance of Iterated Discrimination for various values of the ϵ and τ parameters. Note that the values of $\epsilon = 0.01$ and $\tau = 0.999$ chosen on the basis of Musk Data Set 1 do not give the peak performance. Peak performance of 91.2% is attained for any of the following parameter values: ($\tau = 0.99$, $\epsilon = 0.012$), ($\tau = 0.99$, $\epsilon = 0.014$), and ($\tau = 0.995$, $\epsilon = 0.008$). This matches the best performance reported for an APR-like neural network algorithm on this same data set (Jain, Dietterich, Lathrop, et al., 1994), where parameter values were also chosen after cross-validation. Performance of at least 89.2% is robust over a wide range of parameter values.

Figure 22 shows a visualization of the binding hypothesis learned from the entire Musk Data Set 2 applied to classify conformation 18 of a molecule named "256" (a true musk; the molecule names are drawn from Bersuker, et al., 1991). Each of the two stereo pairs in the figure can provide a three-dimensional picture if viewed through a stereo viewer or by converging your eyes at a point beyond the page so that the two images fuse and come into focus. All of the bounds in Figure 22

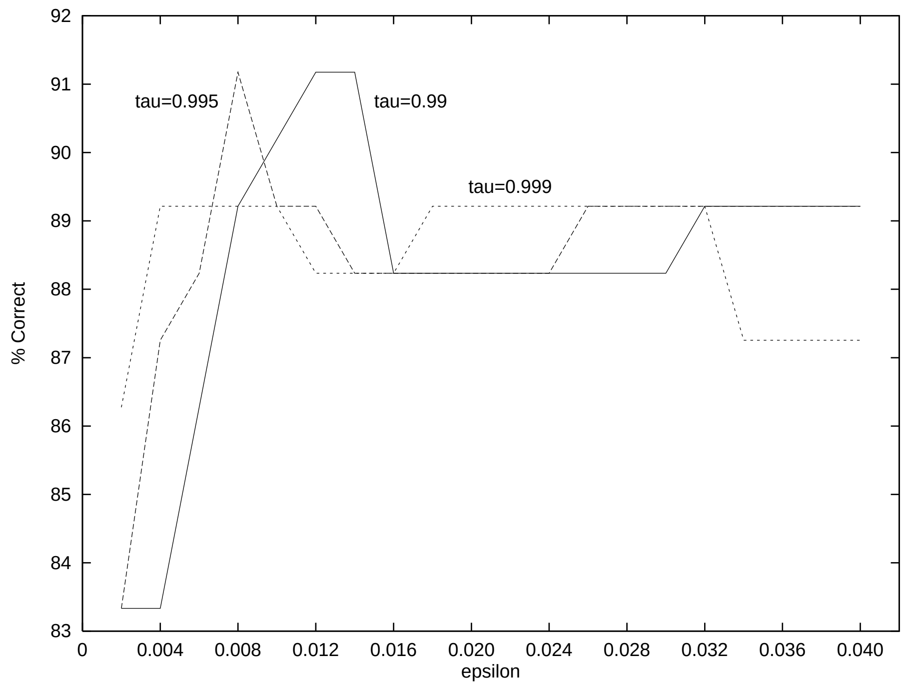

Figure 21: Performance of Iterated Discrimination on Musk Dat Set 2 as a function of τ and ϵ.

are satisfied, so "256" is correctly classified as a musk molecule.

Figure 23 shows the same APR applied to classify conformation 4 of a molecule named "253", which is a non-musk. Here, all but two bounds (59 and 29) are satisfied. The two violated bounds both identify a region of the molecule that does not "stick out" far enough. All other conformations of molecule "253" are also classified as non-musk, so "253" is correctly classified as a non-musk.

Figure 24 shows the APR surrounding a bond-graph (structure diagram) of conformation 18 of molecule "256." This helps us visualize which regions of the molecule were more important for musk activity. We can see that the preponderance of the bounds are testing the shape of the end of the molecule opposite the oxygen atom. This is a region of hydrophobic "bulk", and previous chemical studies had also emphasized the importance of this region for musk activity (Fehr, Galindo, Haubrichs & Perret, 1989). There is also a notable absence of any APR bounds on the middle left side of the molecule, which suggests that the molecules have some shape freedom in this area. Finally, note that only a few, tight bounds are needed to test the position of the rings along the axis orthogonal to the plane of the paper. This probably reflects the fact that all conformations are aligned so that their aromatic rings are superimposed, so there is very little variation between musk and non-musk molecules along this axis.

Displays such as these could be employed by chemists to suggest changes in the molecules to improve binding. For example, molecule "253" could be made to satisfy the APR by adding an ethyl (C_2H_5) group to the molecule in the region of bounds 29 and 59. This would change it from a non-musk into a musk. In fact, it becomes molecule "256"!

Figure 22: The Musk Data Set 2 APR binding hypothesis applied to classify conformation 18 of molecule "256" (a true musk). Dashed lines depict the Connolly surface of the molecule. The heavily shaded area at the bottom is the surface of the oxygen atom. Line segments depict the APR bounds along selected rays, and the index number of each ray is given. The upper two frames give a stereo pair showing a "front view"; the lower two frames show the "back view" of the same molecule. All APR bounds are satisfied.

Figure 23: The Musk Data Set 2 APR binding hypothesis applied to classify conformation 4 of molecule "253" (a non-musk). The upper two frames give a "front view"; the lower two frames give a "back view" of the same molecule. Note that bounds 59 (in the "front view") and 29 (in the "back view") are violated.

Figure 24: The Musk Data Set 2 APR and a structure diagram (bond graph) for conformation 18 of molecule "256." All APR bounds are shown in this single stereo pair. The oxygen atom is at the bottom on the image.

7 Discussion

In a standard problem of learning axis-parallel rectangles, the learning algorithm must solve two problems: select the relevant features and set bounds along those features. The multiple instance problem adds a third difficulty: the algorithm must choose which positive instances to treat as genuine positives.

From the experiments presented above, we can see that setting the bounds along the features is the easiest of these three problems. In Iterated Discrimination, we postponed setting the exact bound values until we had already determined the relevant features and the genuine positive instances.

On the other hand, we found it essential to coordinate the choice of relevant features and positive instances. For example, in data not shown, we stopped Iterated Discrimination after one iteration, and the resulting performance was substantially worse.

The Artificial Data Set was critical to helping us debug and understand our algorithms. In particular, note that while the performance of the "GFS elim-kde APR" algorithm on Musk Data Set 1 was indistinguishable from Iterated Discrimination, the artificial data revealed it to be much worse, because it was not selecting relevant features very well. This was borne out in the Musk Data Set 2 experiments, where GFS elim-kde performed much worse. Similar behavior was observed for the "GFS elim-count APR" algorithm. We strongly recommend the artificial data set approach to algorithm development and evaluation.

The significance of these results for drug design is limited by three factors. First, the algorithms in this paper address only two-class qualitative data. While the effect of some drugs can only be measured by qualitative response, there are usually quantitative measures of drug efficacy in human subjects and in laboratory assays. Hence, medicinal chemists are primarily interested in algorithms for predicting real-valued activites. As we mentioned above, Jain, Dietterich, Lathrop, et al. (1994), and Jain, Koile, Bauer, et al. (1994) describe an APR-like neural-network based method, called COMPASS, that can make quantitative activity predictions.

Second, the algorithms in this paper assume that a conjunction of conditions must be satisfied for binding. This is not always the case. For example, many drugs of medical importance are "antagonist" drugs—their job is to prevent the natural compound from binding (e.g., by blocking access to the binding site). Different antagonist drugs may operate by fitting in different binding sites or by binding in different modes to the same general binding site. It is easy to conceive of extensions to the algorithms reported here that could handle multiple binding modes, and hence, have broader applicability in drug design. This is an important direction for further research.

The third limitation of the algorithms discussed here is that they are based on placing each molecule in a standard position and orientation with respect to the 162 rays. For many classes of molecules, it is not difficult to choose a standard position and orientation. However, for highly flexible molecules or very diverse sets of molecules, it can be much more difficult. Dietterich, Jain, Lathrop & Lozano-Perez (1994) and Jain, Dietterich, Lathrop et al. (1994) describe a method called *dynamic reposing* that permits the relative orientations of the molecules to change slightly during learning. In comparisons with other state-of-the-art methods, Jain, Koile, Bauer & Chapman (1994) show that dynamic reposing permits more accurate and robust activity predictions. We have conducted initial experiments with dynamic reposing using APRs, but because the APR gives only a yes/no response, it does not provide the quantitative signal needed to control reposing. Attempts to define such a signal for APRs have not yet succeeded.

The need for dynamic reposing raises another interesting direction for research. In this paper, we have considered what might be called the discrete multiple-instance problem: each input object can be represented as a finite set of possible instances. While there are many applications that exhibit this problem, there are other applications where the space of possible instances is continuous and infinite. The alternative positions and orientations of molecules provide an example of this continuous multiple-instance problem. Related problems arise in optical character recognition (Simard, Le Cun & Denker, 1993).

8 Conclusions

The multiple instance problem is an important problem that arises in real-world tasks where the training examples are ambiguous: a single example object may have many alternative feature vectors that describe it, and yet only one of those feature vectors may be responsible for the observed classification of the object. In particular, the problem arises in drug activity prediction, where each training example is a molecule (and its observed binding strength), but where each feature vector describes a possible shape (conformation) of the molecule. Because binding strength is most likely the result of a single shape fitting into a binding site, usually only one of the feature vectors properly represents the active molecular shape.

We presented a representation for molecular shape and a representation for binding hypotheses. In feature space, each hypothesis corresponds to an axis-parallel rectangle (APR). We presented three general approaches to designing APR algorithms: (a) ignore the multiple instance problem, (b) start with the bounding APR of all positive examples and shrink it while attending to multiple instances (the "outside in" approach), and (c) start with a single-point APR and grow it while considering multiple instances (the "inside out" approach). Experiments clearly show that the "inside out" approach is the best. Ignoring the multiple instance problem—either with APR algorithms, neural networks, or decision trees—gives quite poor performance. The "outside-in" approach has great difficulty identifying relevant features of the APR.

Even when the multiple-instance problem is ignored, APR algorithms generally out-perform neural networks and decision trees on this task even though in principle networks and trees can both represent APRs. This is a good illustration of the importance of choosing an appropriate bias for inductive learning algorithms.

Drug activity prediction and the multiple instance problem are both important subjects for future research. A particularly interesting issue is how to design multiple-instance modifications for decision trees, neural networks, and other popular machine learning algorithms.

9 Acknowledgements

The authors wish to thank George Whitesides for providing the initial impetus to consider shape representation and the musk domain. Without his early involvement, none of this work would have been conducted.

This research was initiated at Arris Pharmaceutical Corporation. Many people participated in the development of this project including Barr Bauer, David Chapman, Roger Critchlow, Ajay Jain, Brad Katz, Kimberle Koile, John Park, and Teresa Webster. The authors thank all of these teammates and Mike Ross, Arris Chief Technical Officer, for their help in bringing this project to a successful conclusion.

The data sets described in this paper have been deposited at the Irvine Machine Learning database repository (Murphy & Aha, 1994).

10 References

Aha, D. W. (1992). Generalizing from case studies: A case study. In *Proceedings of the Ninth International Conference on Machine Learning* (pp. 1–10). San Francisco: Morgan Kaufmann.

Beets, M. G. J. (1987). *Structure-activity relationships in human chemoreception.* London: Applied Science Publishers.

Bersuker, I. B., Dimoglo, A. S., Yu. Gorbachov, M., Vlad, P. F., Pesaro, M. (1991). *New Journal of Chemistry, 15*, (5) 307–320.

Blankley, C. J. (1983). Introduction: A review of QSAR methodology. In J. G. Topliss (Ed.) *Quantitative structure-activity relationships of drugs.* New York: Academic Press.

Buchanan, B. G. & Mitchell, T. M. (1978). Model-directed learning of production rules. In D. A. Waterman & F. Hayes-Roth (Eds.), *Pattern-Directed Inference Systems.* New York: Academic Press. 297–312.

Chang, W. C., Guida, W. C., Still, J. (1989). An internal coordinate Monte Carlo method for searching conformational space. *Journal of the American Chemical Society, 111,* 4379–4386.

Connolly, M. L. (1983). Solvent-accessible surfaces of proteins and nucleic acids. *Science, 221*, 709–713.

Cramer, R. D. III, Patterson, D. E., Bunce, J. D. (1988). Comparative molecular field analysis (CoMFA). 1. Effecto f shape on binding of steroids to carrier proteins. *Journal of the American Chemical Society, 110,* 5959–5967.

Crippen, G. M., & Havel, T. F. (1988). *Distance geometry and molecular conformation.* New York: John Wiley & Sons.

Dietterich, T. G., & Flann, N. S., (1988). An inductive approach to solving the imperfect theory problem. Technical Report No. 88-30-2. Corvallis, Oregon: Oregon State University, Department of Computer Science. Presented at the 1988 AAAI Spring Symposium on Explanation-Based Learning.

Dietterich, T. G., Jain, A., Lathrop, R., Lozano-Perez, T. (1994). A comparison of dynamic reposing and tangent distance for drug activity prediction. *Advances in Neural Information Processing Systems, 6.* San Mateo, CA: Morgan Kaufmann.

Dietterich, T. G., & Michalski, R. S. (1981). Inductive learning of structural descriptions: Evaluation criteria and comparative review of selected methods. *Artificial Intelligence, 16*, 257–294.

Engle, M. L., Burks, C. (1993). Artificially generated data sets for testing DNA sequence assembly algorithms. *Genomics, 16,* 286–288.

Fehr, C., Galindo, J., Haubrichs, R., & Perret, R. (1989). New aromatic musk odorants: Design and synthesis. *Helvetica Chimica Acta, 72*, 1537–53.

Friedman, J. H. & Stuetzle, W. (1981). Projection pursuit regression. *Journal of the American Statistical Association, 76,* 817–723.

Ham, C. L., Jurs, P.C. (1985). Structure-activity studies of musk odorants using pattern recognition: monocyclic nitrobenzenes. *Chemical Senses, 10,* 491–505.

Hansch, C. & Fujita, T. (1964). ρ-σ-π Analysis. A method for the correlation of biological activity and chemical structure. *Journal of the American Chemical Society, 86,* 1616–1626.

Hansch, C. & Leo, A. J. (1979). *Substituent constants for correlation analysis in chemistry and biology.* New York: Wiley Interscience.

Haussler, D., (1988). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence, 36* (2), 177–222.

Haussler, D. (1989). Learning conjunctive concepts in structural domains. *Machine Learning, 4* (1), 7–40.

Jain, A. N., Dietterich, T. G., Lathrop, R. H., Chapman, D., Critchlow, R. E., Bauer, B. E., Webster, T. A., Lozano-Perez, T. (1994). Compass: A shape-based machine learning tool for drug design. *Journal of Computer Aided Molecular Design, 8* (6), 635–652.

Jain, A., Koile, K., Bauer, B., Chapman, D. (1994) Compass: Predicting biological activities from molecular surface properties. Performance comparisons on a steroid benchmark. *J. Medicinal Chemistry, 37*, 2315–2327.

Kuntz, I. D. (1992). Structure-based strategies for drug design and discovery. *Science, 257,* 1078–1082.

Lindsay, R., Buchanan, B., Feigenbaum, E., & Lederberg, J. (1980). *Applications of artificial intelligence to organic chemistry: The Dendral project.* New York: McGraw-Hill.

Murphy, P. M., & Aha, D. W. (1994). *UCI Repository of machine learning databases* [Machine-readable data repository]. Irvine, CA: University of California, Department of Information and Computer Science.

Narvaez, J. N., Lavine, B. K., Jurs, P. C. (1986). Structure-activity studies of musk odorants using pattern recognition: bicyclo- and tricyclo-benzenoids. *Chemical Senses, 11* (1), 145–156.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis.* London: Chapman and Hall.

Simard, P., Le Cun, Y., Denker, J. (1993). Efficient pattern recognition using a new transformation distance. In Hanson, S. J., Cowan, J. D., Giles, C. L. (Eds.) *Advances in Neural Information Processing Systems 5*, San Mateo, CA: Morgan Kaufmann. 50–58.

Soderlund, C., Burks, C. (1994). GRAM and *genfragII:* Simulating and solving the single-digest partial restriction map problem. *Comp. Applic. Biosci. (CABIOS), 10* 349–358.

Still, W.C. (undated). Macromodel program. Distributed by Columbia University, New York.

Theimer, E.T., Davies, J.T. (1967). Olfaction, musk odor, and molecular properties. *J. Agr. Food Chem., 15,* 6–14.

Weiner, S. J., Kollman, P. A., Case, D. A., Singh, U. C., Ghio, C., Alagona, G., Profeta, S. Jr., & Weiner, P. (1984). A new force field for molecular mechanical simulation of nucleic acids and proteins. *Journal of the American Chemical Society, 106*, 765–784.

Weiner, S. J., Kollman, P. A., Nguyen, D. J., & Case, D. A. (1986). An all atom force field for simulations of proteins and nucleic acids. *Journal of Computational Chemistry, 7* (2), 230–252.