Machine Learning, 37, 5–49 (1999)

A Theoretical and Empirical Study of a Noise-Tolerant Algorithm to Learn Geometric Patterns

SALLY A. GOLDMAN sg@cs.wustl.edu
Department of Computer Science, Washington University, St. Louis, MO 63130-4899

STEPHEN D. SCOTT sscott@cse.unl.edu
Department of Computer Science and Engineering, University of Nebraska, Lincoln, NE 68588-0115

Editor: William W. Cohen

Abstract. Developing the ability to recognize a landmark from a visual image of a robot's current location is a fundamental problem in robotics. We describe a way in which the landmark matching problem can be mapped to that of learning a one-dimensional geometric pattern. The first contribution of our work is an efficient noise-tolerant algorithm (designed using the statistical query model) to PAC learn the class of one-dimensional geometric patterns. The second contribution of our work is an empirical study of our algorithm that provides some evidence that statistical query algorithms may be valuable for use in practice for handling noisy data.

Keywords: noise-tolerant PAC learning, statistical query model, landmark matching problem

1. Introduction

We consider the problem of PAC learning the concept class of geometric patterns where the "target" geometric pattern is a configuration of at most k points on the real line. Each instance is a configuration of at most n points on the real line, where it is labeled according to whether or not it visually resembles the target pattern. To capture the notion of visual resemblance we use the *Hausdorff metric* (for example, see Gruber (1983)). Informally, two geometric patterns P and Q resemble each other under the Hausdorff metric if every point in one pattern is "close" to some point in the other pattern.

As a possible application of an algorithm that can learn this concept class, consider the problem of recognizing from a visual image of a robot's current location whether or not it is in the vicinity of a known landmark (where a landmark is a location that is visually different from other locations). Such an algorithm is needed as one piece of a complete navigation system where the robot performs the navigation by planning a path between known landmarks, tracking the landmarks as it goes. Because of inaccuracies in effectors and possible errors in the robot's internal map, when the robot believes it is at landmark L, it should check that it is really in the vicinity of L before heading to the next landmark. Then adjustments can be made if the robot is not at L by either re-homing to L and/or updating its map. We explore applications of our algorithm for learning geometric patterns

to this problem by converting the robot's visual image into a one-dimensional geometric pattern.

One key result of this paper is a polynomial-time algorithm that PAC learns the class of one-dimensional geometric patterns under high noise rates (any rate $<1/2$ of classification noise as well as lesser amounts of other types of noise). We obtain such a noise-tolerant algorithm by using the recently developed statistical query model (Kearns, 1993; Aslam & Decatur, 1998). A second contribution of this work is an empirical study of how an algorithm designed using the statistical query model works on simulated and real data. One goal of this empirical work was to contrast the empirically determined training set sizes with the worst-case lower bounds from our theoretical analysis. Several interesting issues were uncovered when moving from the theoretical to the empirical setting, such as what to do without a value for the algorithm's input ϵ. We found that our algorithm performs very well even with modest amounts of data—much less than the worst-case theoretical results would demand. Surprisingly, when looking at data without any noise, our statistical query algorithm worked as well as a more traditional algorithm that applies a set covering approach to correctly labeled data. While we considered just one learning problem, we are hopeful that algorithms designed using the statistical query model will generally work well in practice (on real, noisy data).

An interesting feature of the concept class studied here is that the target concept is specified by a k-tuple[1] of points on the real line, while the instances are specified by n-tuples of points on the real line where n is potentially much larger than k. Although there are some important distinctions, in some sense our work illustrates a concept class in a continuous domain in which a large fraction of each instance can be viewed as "irrelevant". As in previous work on learning with a large number of irrelevant attributes in the Boolean domain (e.g. Littlestone's work (1988)), our algorithm's sample complexity (the best analog to a mistake bound) depends polynomially on k and $\log n$.

This paper is organized as follows. In the next section we discuss the landmark matching problem in more depth. Then, in Section 3, we review the background material. In Section 4 we formally define the concept class of one-dimensional geometric patterns. Section 5 briefly describes a known covering algorithm to learn patterns. Next, in Section 6, we present our noise-tolerant algorithm. In Section 7 we describe the modifications made to our algorithm for empirical use. Then Section 8 discusses the results of our experiments on real and simulated data. We conclude in Section 9.

2. The landmark matching problem

In this section we explore, in further depth, how this work might apply to the landmark matching problem. It is crucial that the landmark matching algorithm be performed in real-time. To reduce the processing time required by the landmark matching algorithm, some have proposed the use of imaging systems that generate a one-dimensional array of light intensities taken at eye-level (see e.g. Hong et al., 1992; Levitt & Lawton, 1990; Pinette, 1993; Suzuki & Arimoto, 1988). We now briefly describe one such imaging system (Hong et al., 1992; Pinette, 1993). In their robot a spherical mirror is mounted above an upward-pointing camera that enables it to instantaneously obtain a 360° view of the world.

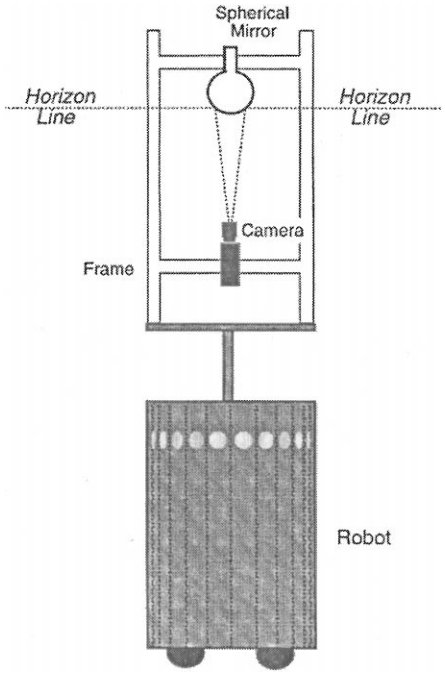

Figure 1. The imaging system on the robot. (This figure comes directly from Pinette's (1993) thesis.)

See figure 1 for a picture of such a robot. The view of the world obtained by this imaging system and the processing performed are shown in figure 2. All points along the eye-level view of the robot (shown by the horizon line in figure 1) project into a circle in the robot's 360° view. Figure 2 shows the panoramic view that results by scanning the 360° view (beginning at due north) in a circle around the robot's horizon line. The panoramic view is sampled along the horizontal line midway between the top and the bottom to produce a one-dimensional array of light intensities (or *signature*) as shown in figure 2.

A common approach to designing landmark matching algorithms uses pattern matching by trying to match the current signature to the signature taken at landmark position L. If one's goal is to determine if the robot is standing exactly at position L, then the pattern matching approach can be implemented to work well. However, in reality, the matching algorithm must determine if the robot is in the vicinity of L (i.e. in a circle centered around L). Because the visual image may change significantly as small movements around L and rotations are made, the pattern matching approach encounters difficulties.

Rather than using a pattern matching approach to match the light intensity array from the current location with the light intensity array of the landmark, we instead propose using a learning algorithm to construct a good hypothesis for performing landmark matching. Intuitively, the learning algorithm is being used to combine a set of positive examples to create a hypothesis that will make good predictions. We obtain the instances by converting the arrays of light intensities into one-dimensional geometric patterns by placing points where there are significant changes in light intensity. Then by applying our algorithm,

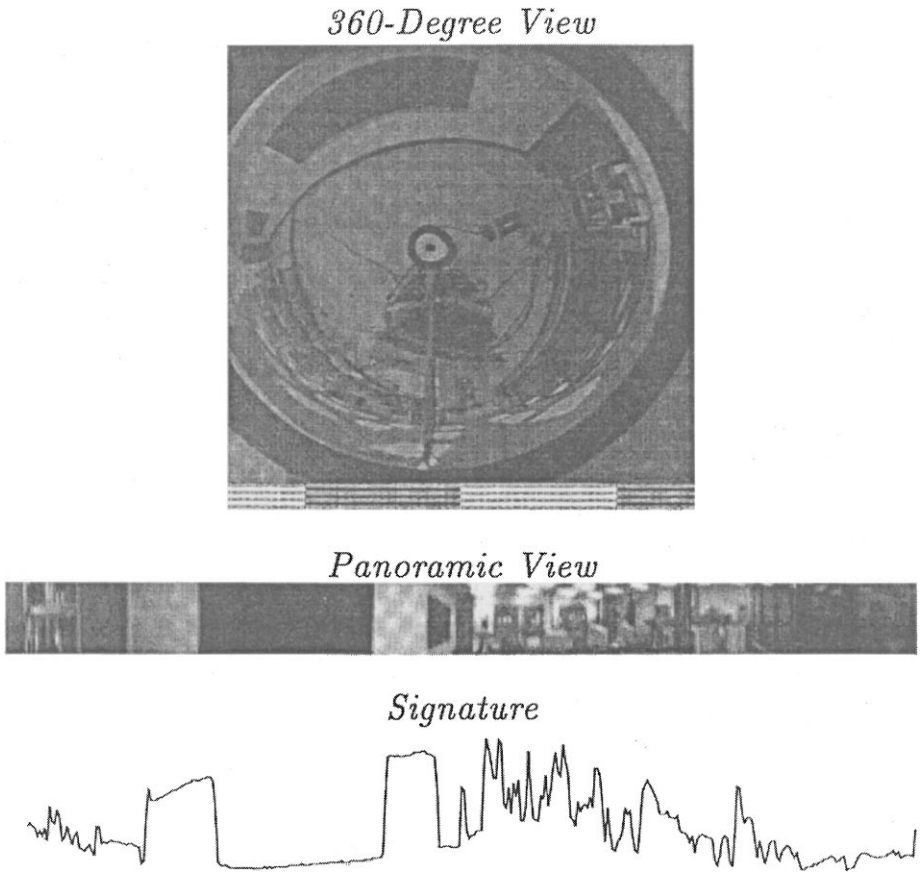

Figure 2. Stages of image processing. (This figure comes directly from Pinette's (1993) thesis.)

giving it a set of positive examples (i.e. patterns[2] obtained from locations in the vicinity of the landmark) and a set of negative examples (i.e. patterns obtained from locations not in the vicinity of the landmark), we construct a hypothesis that can accurately predict whether or not the robot is near the given landmark, assuming that the positive and negative examples are sufficiently distinct.

A natural question raised is why there is a need for learning here. To answer this question, we briefly examine some problems that would occur if a single pattern taken at the landmark location was used as a hypothesis. (The same problems cause difficulties when using a pattern matching approach.) Suppose a landmark location was selected at the position L shown in figure 3 where A and B are the legs of the Gateway Arch in downtown St. Louis. (For simplicity, suppose that nothing besides the arch is in the robot's visual image.) Further, suppose we want the landmark matching system to indicate that the robot is at landmark position L exactly when it is in the dashed circle. Clearly in this example, the

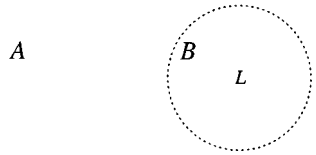

Figure 3. An example to help illustrate a problem that must be overcome. Suppose A and B are the locations of the bases of an arch and L is a landmark location. The visual images obtained within the dashed circle change dramatically.

robot's visual image changes dramatically as the robot moves within this circle. Thus simply using as a hypothesis the single pattern obtained from the visual image obtained from the landmark location would not yield good predictions as to whether or not the robot is "near" the landmark location.

In the learning-based approach we propose here, the navigation system (when selecting L as a landmark) would collect images from locations evenly spaced throughout the dashed circle. Then by using these images as positive examples (and images taken at random locations not in the circle as negative examples), we can apply our learning algorithm to combine all of these images taken near L to obtain a hypothesis to predict if the robot is near L. A valuable feature of the learning algorithm is the generality of its hypothesis class. The concepts (defined precisely in Section 4) are *approximations* to sets of patterns visible within limited regions, and this more general hypothesis class may allow a better fit to the diverse set of positive examples than the simple concept of a single pattern.

Also, when really applying an algorithm for learning one-dimensional geometric patterns to the landmark matching problem, one must handle noisy data. Because of the noise inherent in the data, the problems illustrated above with simply using as a hypothesis the single (noisy) pattern obtained from the visual image at the landmark location would be exacerbated. Our belief is that our algorithm, developed using the statistical query model, will be robust against many types of noise, including those for which the statistical query model has no known theoretical guarantees.

The reader should note that the main results of this paper are the learning algorithm of Section 6, the modifications for empirical use in Section 7, and the work of Section 8.1 that assesses how our noise-tolerant algorithm performs compared to a previous algorithm designed for noise-free data. While we do report empirical results for our algorithm on real data in Section 8.2, these results revealed that more work is required to make our approach a viable solution to the landmark matching problem. Section 9 discusses these issues.

3. Background

In this paper we work within the PAC (probably approximately correct) model as introduced by Valiant (1984, 1985). Details of the model may be found in such textbooks as Kearns and Vazarani (1994), Natarajan (1991), or Anthony and Biggs (1992). We now review the basic definitions and results used here.

3.1. The PAC learning model

In the PAC model, examples of a concept are made available to the learner according to an unknown probability distribution $\mathcal{D}$, and the goal of a learning algorithm is to classify with high accuracy any further (unclassified) instances generated according to the same distribution $\mathcal{D}$.

The *instance domain* $\mathcal{X}$ is the set of all possible objects (instances) that may be made available as data to a learner. A *concept class* $\mathcal{C}$ is a collection of subsets of $\mathcal{X}$, and examples input to the learner are classified according to membership of a *target concept* $C \in \mathcal{C}$. ($\mathcal{C}$ is known to the learner, C is to be learned.) Often $\mathcal{X}$ is decomposed into subsets $\mathcal{X}_n$ according to some natural size measure n for encoding an example. In this paper, we refer to n as the *instance complexity*. For the class of one-dimensional patterns, n is the number of points in an example. Let $\mathcal{X}_n$ denote the set of examples to be classified for each problem of size n, and let $\mathcal{X} = \bigcup_{n\geq 1} \mathcal{X}_n$ denote the *example space*. We say each $X \in \mathcal{X}$ is an *example*.

For each $n \geq 1$, we define each $\mathcal{C}_n \subseteq 2^{\mathcal{X}_n}$ to be a *family of concepts* over $\mathcal{X}_n$, and $\mathcal{C} = \bigcup_{n\geq 1} \mathcal{C}_n$ to be a *concept class* over $\mathcal{X}$. For $C \in \mathcal{C}_n$ and $X \in \mathcal{X}_n$, $C(X)$ denotes the classification of C on example X. We say that an example $X \in C$ is a *positive example* and an example $X \notin C$ is a *negative example*.

Because learning algorithms need a means for representing the functions to be learned, typically associated with each concept class $\mathcal{C}$ is a language $\mathcal{R}_\mathcal{C}$ over a finite alphabet, used for representing concepts in $\mathcal{C}$. Each $r \in \mathcal{R}_\mathcal{C}$ denotes some $C \in \mathcal{C}$, and every $C \in \mathcal{C}$ has at least one representation $r \in \mathcal{R}_\mathcal{C}$. (Here we represent a target concept as a set of points on the real line.) Each concept $C \in \mathcal{C}_n$ has a *size* denoted by $|C|$, which is the representation length of the shortest $r \in \mathcal{R}_\mathcal{C}$ that denotes C. In this paper, we refer to $|C|$ as the *concept complexity*. For ease of exposition, in the remainder of this paper we use $\mathcal{C}$ and $\mathcal{R}_\mathcal{C}$ interchangeably.

Note we are measuring the complexity of a configuration of points by the number of points it contains, and the positions of the points is of no importance. This is based on the assumption of unit cost for representing and operating on a real number, used in the computational geometry and neural network literature, and noted by Valiant (1991) to be typically appropriate for geometrical domains.

To obtain information about an unknown target function $C \in \mathcal{C}_n$, the learner is provided access to labeled (positive and negative) examples of C, drawn randomly according to the unknown target distribution $\mathcal{D}$ over $\mathcal{X}_n$. The learner is also given[3] as input ϵ and δ such that $0 < \epsilon, \delta < 1$, and an upper bound k on $|C|$. The learner's goal is to output, with probability at least $1 - \delta$, a polynomially evaluatable *hypothesis* $\mathcal{H} \subseteq \mathcal{X}_n$ that has probability at most ϵ of disagreeing with C on a randomly drawn example from $\mathcal{D}$ (thus, $\mathcal{H}$ has *error* at most ϵ). If such a learning algorithm exists (that is, an algorithm meeting the above requirements for any $n \geq 1$, any target concept $C \in \mathcal{C}_n$, any target distribution $\mathcal{D}$, any $\epsilon, \delta > 0$, and any $k \geq |C|$), we say that $\mathcal{C}$ is *PAC learnable*. We say that a PAC learning algorithm is a polynomial-time (or efficient) algorithm if the number of examples drawn and computation time are polynomial in $n, k, 1/\epsilon$, and $1/\delta$. For ease of exposition, we drop the subscript "n" and write $\mathcal{C}$ and $\mathcal{X}$ instead of $\mathcal{C}_n$ and $\mathcal{X}_n$.

Note that as originally formulated, PAC learnability also required the hypothesis to be a member of $\mathcal{C}$. Pitt and Valiant (1988) show that under the assumption NP $\neq$ RP, a prerequisite for PAC learnability in this sense is the ability to solve the *consistent hypothesis problem*, which is the algorithmic problem of finding a concept which is consistent with a given sample (that is, containing the positive but not the negative examples of the sample). This implies that if the consistent hypothesis problem is NP-hard for a given concept class (as happens for the concept classes considered here), then the learning problem is hard when $\mathcal{H}$ must come from $\mathcal{C}$.

The more general form of learning that we use here is commonly called *prediction* or *non-proper learning*. The goal is to find *any* polynomial-time evaluatable hypothesis that classifies instances accurately in the PAC sense. Thus the hypothesis need not come from $\mathcal{C}$. This idea of prediction in the PAC model originated in the work of Haussler, Littlestone, and Warmuth (1994), and is discussed in Pitt and Warmuth (1990).

3.2. The VC dimension

The paper of Blumer et al. (1989) identifies an important combinatorial parameter of a class of hypotheses called the *Vapnik-Chervonenkis* (*VC*) *dimension*, which originated in the paper of Vapnik and Chervonenkis (1971). The VC dimension provides bounds on how large a sample size is required in order to have enough information for accurate generalization. (We call this quantity the *sample complexity* of a learning problem; note that given a sufficiently large sample there is still the computational problem of finding a consistent hypothesis.)

Definition 1 (Blumer et al., 1989). A finite set $\mathcal{S} \subseteq \mathcal{X}$ is *shattered* by the concept class $\mathcal{C}$ if for each of the $2^{|\mathcal{S}|}$ subsets $\mathcal{S}' \subseteq \mathcal{S}$, there is a concept $f \in \mathcal{C}$ that contains all of $\mathcal{S}'$ and none of $\mathcal{S} \backslash \mathcal{S}'$. In other words, for each of the $2^{|\mathcal{S}|}$ possible labelings of $\mathcal{S}$ (where each example $X \in \mathcal{S}$ is either positive or negative), there is some $f \in \mathcal{C}$ that realizes the desired labeling. The *VC dimension* of concept class $\mathcal{C}$ (which we denote $\mathrm{VCD}(\mathcal{C})$) is the size of a largest set $\mathcal{S} \subseteq \mathcal{X}$ that is shattered by $\mathcal{C}$.

As an example, consider the concept class $\mathcal{C}$ of axis-parallel rectangles in $\Re^2$ where points lying on or inside the target rectangle are positive, and points lying outside the target rectangle are negative. First, it is easily seen that there are four points (for example points at $(0, 1)$, $(0, -1)$, $(1, 0)$, $(-1, 0)$) that can be shattered. Thus $\mathrm{VCD}(\mathcal{C}) \geq 4$. We now argue that no set of five points can be shattered. The smallest bounding axis-parallel rectangle defined by the five points is in fact defined by at most four of the points. For p a non-defining point in the set, we see that the set cannot be shattered since it is not possible for p to be classified as negative while also classifying the others as positive. Thus $\mathrm{VCD}(\mathcal{C}) = 4$.

The results of Blumer et al. give a sufficient condition for a prediction algorithm to generalize successfully from example data, in terms of the VC dimension, as given in the following theorem.

Theorem 1 (Blumer et al., 1989). *For sample size*[4] *at least*

$$\max\left(\frac{4}{\epsilon}\lg\frac{2}{\delta}, \frac{8\,\mathrm{VCD}(\mathcal{C})}{\epsilon}\lg\frac{13}{\epsilon}\right)$$

any concept $C \in \mathcal{C}$ consistent with the sample will have error at most ϵ with probability at least $1-\delta$.

Furthermore, Ehrenfeucht et al. (1989) prove that any concept class $\mathcal{C}$ must use $\Omega(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{\mathrm{VCD}(\mathcal{C})}{\epsilon})$ examples in the worst case.

Note that the above results also hold in the case of prediction (i.e. the hypothesis comes from a class $\mathcal{C}'$ that is more expressive than $\mathcal{C}$) by simply substituting $\mathrm{VCD}(\mathcal{C}')$ for $\mathrm{VCD}(\mathcal{C})$. To see this, just assume the algorithm is learning $\mathcal{C}'$ using a hypothesis from $\mathcal{C}'$ but the concepts will only come from $\mathcal{C} \subset \mathcal{C}'$.

3.3. The statistical query model

The noise model we focus on here is that of *random classification noise*. In this model, a random example X is drawn according to an arbitrary, unknown distribution $\mathcal{D}$ and, with probability η (the classification noise rate), X's label is flipped before it is given to the learner. We define $\mathcal{D}_\eta$ to be this noisy distribution.

To obtain a noise-tolerant version of our algorithm we use the *statistical query* model, introduced by Kearns (1993) and enhanced by Aslam and Decatur (1993, 1998), and Decatur (1993). In this model, rather than sampling labeled examples, the learner requests the value of various statistics. An *additive statistical query* (SQ) (Kearns, 1993) oracle returns the probability, within some additive constant, that a provided predicate is true for a random labeled example from $\mathcal{D}$. The particular queries we use are known as *relative statistical queries* (Aslam & Decatur, 1998). These take the form $\mathbf{SQ}(\chi, \mu, \theta)$ where χ is a predicate over labeled examples, μ is a *relative error bound*, and θ is a *threshold*. For target concept C, define $P_\chi \doteq \Pr[\chi(X, C(X)) = 1]$. If $P_\chi < \theta$ then $\mathbf{SQ}(\chi, \mu, \theta)$ may return $\perp$. If $\perp$ is not returned, then $\mathbf{SQ}(\chi, \mu, \theta)$ must return an estimate $\hat{P}_\chi$ such that $P_\chi(1-\mu) \le \hat{P}_\chi \le P_\chi(1+\mu)$. The learner may also request unlabeled examples. As a clarifying example of a relative SQ algorithm, consider the problem of learning a monotone conjunction of Boolean variables as described by Aslam and Decatur (1998). Here the learning algorithm must determine which subset of the variables $\{x_1, \ldots, x_n\}$ is contained in the unknown target function $C \in \mathcal{C}$. The algorithm uses the predicates $\chi_i(X, \ell) \doteq ((x_i = 0) \wedge (\ell = 1))$, so P_{χ_i} is the probability that x_i is false in a positive example. The relative SQ algorithm submits the queries $\mathbf{SQ}(\chi_i, 1/2, \epsilon/n)$ for $1 \le i \le n$. Then the hypothesis returned by the algorithm is the conjunction of the variables x_i for which $\hat{P}_{\chi_i} = 0$ or $\perp$. Since the total error is limited by the sum of the P_{χ_i}s and each $\perp$ (or 0) returned means $P_{\chi_i} \le \epsilon/n$, the total error is at most ϵ.

It is known that all statistical query algorithms are robust against high rates of random classification noise (Kearns, 1993; Aslam & Decatur, 1998; Decatur, 1993). The following theorem indicates the sample complexity required for simulating relative statistical queries

in the presence of classification noise of rate $\eta \leq \eta_b < 1/2$ by drawing noisily labeled examples from the distribution $\mathcal{D}_\eta$. In addition to μ and θ, the theorem uses a parameter ρ. For a particular query χ, ρ is inversely related to that query's dependence on the examples' labels. Formally, define $\chi_\oplus(X, \ell) \doteq (\chi(X, 0) \oplus \chi(X, 1))$, which is true iff χ depends on X's label. Then $\rho = 1$ if $P_{\chi_\oplus} < \theta$ and $\rho = \theta / P_{\chi_\oplus}$ otherwise.

In the following theorem, μ_*, θ_* and ρ_* are lower bounds on μ, θ and ρ over all relative SQs in the algorithm.

Theorem 2 (Aslam and Decatur, 1998). *The total sample complexity required to simulate a relative SQ algorithm in the presence of classification noise* (*of rate* $\eta \leq \eta_b < 1/2$) *is*

$$O\left(\frac{\mathrm{VCD}(\mathcal{Q})}{\mu_*^2\theta_*\rho_*(1-2\eta_b)^2}\left[\log\left(\frac{1}{\mu_*\theta_*\rho_*(1-2\eta_b)}\right) + \log\frac{1}{\delta}\right]\right),$$

where $\mathcal{Q}$ *is the query space.*

In the earlier example, $\mu_* = 1/2$, $\rho_* = \theta_* = \epsilon/n$ and $\mathrm{VCD}(\mathcal{Q}) = \lfloor \lg n \rfloor$. Thus applying Theorem 2 yields a sample complexity of

$$O\left(\frac{n^2 \log n}{\epsilon^2(1-2\eta_b)^2}\left[\log\left(\frac{n}{\epsilon(1-2\eta_b)}\right) + \log\frac{1}{\delta}\right]\right).$$

Statistical query algorithms are also known to be robust against small amounts of malicious errors and distributional errors (Decatur, 1993; Aslam & Decatur, 1998), making them appealing tools for working with noisy data, regardless of the noise process.

While a lot of very nice theoretical results have been proven about the statistical query model, we are unaware of any empirical studies of this model prior to our original work (Goldman & Scott, 1996). One important contribution of this work is an empirical study of a statistical query algorithm. In moving from the theoretical to the empirical setting, our statistical query algorithm required several non-trivial changes. Several of these changes involved removing the dependence on ϵ from our algorithm's decisions, since in practical use, no value for ϵ is given as input to the algorithm. Instead, we searched the parameter space for the most appropriate values given the input data. These changes are similar to changes made by Auer (1997) to an algorithm by Auer, Long, and Srinivasan (1997) to learn the concept class of axis-parallel boxes when *multi-instance examples* are provided (i.e. an example is positive if *any* of its points lies in the target box). Auer searched the parameter space for the value that caused his algorithm to produce the box with the lowest error on the training set, which is similar to how we conduct one of our searches.

4. The class of one-dimensional geometric patterns

For the concept class considered here, the instance space $\mathcal{X}_n$ consists of all configurations of at most n points on the real line.[5] A concept is the set of all configurations from $\mathcal{X}_n$ within

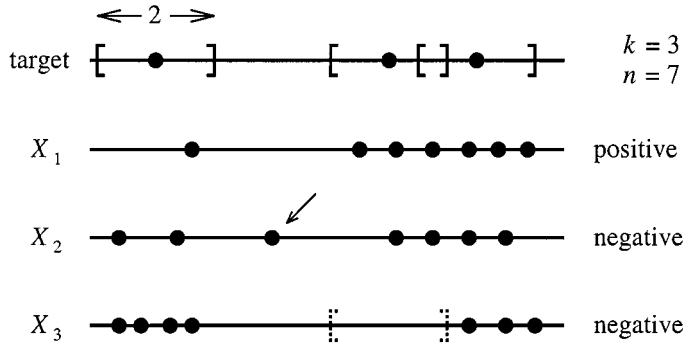

Figure 4. This figure illustrates an example concept from $\mathcal{C}_{3,7}$. The top line shows the target pattern. Around each target point we show an interval that covers all points within unit distance from that point. Every positive example must have every point within one of the above intervals *and* no interval can be empty (e.g. see X_1 above). For an example to be negative, there must be a point in it that is not within unit distance of any target point (e.g. X_2) and/or there are no points in the example within unit distance of some target point (e.g. X_3).

some distance[6] γ under the Hausdorff metric of some "ideal" configuration of at most k points. The Hausdorff distance between configurations P and Q, denoted $\text{HD}(P, Q)$, is:

$$\max\left\{ \max_{p\in P}\left\{ \min_{q\in Q}\{\text{dist}(p,q)\}\right\}, \max_{q\in Q}\left\{ \min_{p\in P}\{\text{dist}(p,q)\}\right\}\right\}$$

where $\text{dist}(p, q)$ is the Euclidean distance between points p and q. In words, if each point in P reports the distance to its nearest neighbor in Q and each point in Q reports the distance to its nearest neighbor in P, then the Hausdorff distance is the maximum of these distances.

Let P be any configuration of points on the real line. Then we define the concept C_P that corresponds to P by $C_P = \{X \in \mathcal{X}_n\colon \text{HD}(P, X) \leq \gamma\}$. For ease of exposition, until Section 8.2 we assume $\gamma = 1$. Figure 4 illustrates an example of such a concept. Thus one can view each concept as a sphere of unit radius in a metric space where P defines the center of the sphere. For any $X \in \mathcal{X}_n$ such that $X \in C_P$, we say that X is a *positive example* of C_P. Likewise, if $X \notin C_P$, we say that X is a *negative example* of C_P. Furthermore, all configurations of points that resemble the given configuration P are contained within this sphere. Finally, the concept class $\mathcal{C}_{k,n}$ that we study is defined as follows.

Definition 2. $\mathcal{C}_{k,n} \doteq \{C_P\colon P$ is a configuration of at most k points on the real line$\}$.

As discussed in Section 1, n may be significantly greater than k. For example, the learner may be asked to predict if a configuration of 100 points is contained within a sphere defined by 3 points. This consideration is, in some sense, analogous to the notion of irrelevant attributes studied in the Boolean domain. Namely, given any positive (respectively, negative) example from $\mathcal{X}_n$, there exists a subset of k of the n points in that example such that the configuration of these k points is also a positive (respectively, negative) example. However,

observe that unlike the Boolean domain, there is no fixed set of points of an instance that are "relevant". Thus if an arbitrary point is removed from an instance it can no longer be determined if that instance was positive or negative before the point was removed.

At first glance, there may appear to be some similarities between $\mathcal{C}_{k,n}$ and the class of the union of at most k intervals over the real line. However, the class of one-dimensional geometric patterns is really quite different (and significantly more complex) than the class of unions of intervals on the real line. One major difference is that for unions of intervals, each instance is a single point on the real line, whereas for $\mathcal{C}_{k,n}$ each instance is a set of n points on the real line. Thus the notion of being able to independently vary the concept complexity and instance complexity does not exist for the class of unions of intervals. Additionally, for the class of unions of intervals, an instance is a positive example simply when the single point provided is contained within one of the k intervals. For $\mathcal{C}_{k,n}$ an instance is positive if and only if it satisfies the following two conditions.

1. Each of the n points in the instance is contained within one of the k width-2 intervals defined by the k target points.
2. There is at least one of the n points in the instance contained within the width-2 interval defined by each of the k target points.

Further we note that Goldberg (1992) has shown that it is NP-complete to find a sphere in the given metric space (i.e. one-dimensional patterns of points on the line under the Hausdorff metric) consistent with a given set of positive and negative examples of an unknown sphere in the given metric space. In other words, given a set $\mathcal{S}$ of examples labeled according to some one-dimensional geometric pattern of k points, it is NP-complete to find some one-dimensional geometric pattern (of *any* number of points) that correctly classifies all examples in $\mathcal{S}$. Thus by applying the result of Pitt and Valiant (1988) (see Section 3.1), assuming NP $\neq$ RP, we cannot PAC learn $\mathcal{C}_{k,n}$ if the hypothesis is constrained to come from $\mathcal{C}_{k',n}$ for any $k' \geq k$. So it is necessary to use a more expressive hypothesis space. Thus to give even further evidence that the class of one-dimensional patterns is significantly more complex than the union of intervals on the real line, observe that the consistency problem for the latter class is trivial to solve.

5. A covering algorithm

We now review a PAC algorithm (Goldberg & Goldman, 1994; Goldberg, Goldman, & Scott, 1996) for learning $\mathcal{C}_{k,n}$. This algorithm (which expects correctly labeled data) is an Occam algorithm (Blumer et al., 1987, 1989). Namely, it draws a sufficiently large sample of size m (polynomial in $k, \lg n, 1/\epsilon$, and $\lg 1/\delta$) and then finds a hypothesis consistent with all examples that has size sublinear in the size of the sample. It builds the hypothesis using a greedy covering algorithm that is based on the observation that it is possible, in polynomial time, to find a concept from $\mathcal{C}_{k+1,n}$ consistent with all the positive examples and a fraction $\frac{1}{2(k+1)}$ of the negative examples. Then the negative examples accounted for are removed and the procedure is repeatedly applied (at most $2(k+1)\lg m$ times) until all negative examples have been eliminated. The intersection of all concepts obtained by

doing this is consistent with the sample and thus forms a hypothesis with error at most ϵ with probability at least $1 - \delta$.

A key drawback of this algorithm is that it is very sensitive to even a small amount of noise in the data. For example a single negative example misclassified as positive could make it impossible to find a pattern that is consistent with all the positively labeled examples. Goldman and Goldberg (1994) were able to modify the algorithm to work when there were false positive examples in the sample. However, it was necessary that the expected number of truly positive examples was higher than the expected number of false positive examples. Also, the techniques they applied would not help to handle even a small rate of false negative examples.

6. Our new noise-tolerant learning algorithm

In this section we describe our new noise-tolerant learning algorithm. It is motivated by the fact that it is possible in polynomial time to find a one-dimensional pattern that is consistent with all positive and at least a fraction $\frac{1}{2(k+1)}$ of the negative examples. Define $\mathcal{P}_k$ as the set of one-dimensional geometric patterns that contain at most k points. Our hypothesis class $\mathcal{C}_{\text{hyp}}$ is the class of functions that consist of the intersection of at most s patterns (where s is given later), each containing at most $3k + 1$ points. Our algorithm will output a hypothesis $\mathcal{H} \in \mathcal{C}_{\text{hyp}}$.

Definition 3.

$$\mathcal{C}_{\text{hyp}} \doteq \left\{ \bigcap_{\leq s} P \colon P \in \mathcal{P}_{3k+1} \right\}.$$

First we draw a sufficiently large unlabeled sample $\mathcal{S}_u$ such that any hypothesis consistent with $\mathcal{S}_u$ would have low prediction error. The standard VC dimension result (Blumer et al., 1989) described in Theorem 1 can be applied to achieve this goal. Next our algorithm constructs a pattern H_+ of at most $3k$ points where for a random positive example X, the distance (under the Hausdorff metric) between X and H_+ is at most one (with high probability). Since we are using an unlabeled sample (since the labels could be corrupted by noise), we decide if point p should be placed in H_+ by using a statistical query that measures the fraction of positive examples that do not have a point within unit distance of p. We show that with high probability, H_+ returned from *Cover-Positives* has a false negative error rate of at most $2\epsilon/5$.

Next, we use H_+ to construct a set of patterns $\mathcal{P}$. These patterns have the property that there exists a subset $\mathcal{P}_*$ for which $\mathcal{H} = \bigcap_{P \in \mathcal{P}_*} P$ correctly classifies most positive examples (so there is a low false negative error rate) and also correctly classifies most negative examples (so there is a low false positive error rate). Each pattern in $\mathcal{P}$ has at most $3k + 1$ points. The key to building $\mathcal{P}$ is to (1) ensure that *each* P placed in $\mathcal{P}$ correctly classifies most positive examples, and (2) ensure that a significant fraction of the negative examples is correctly classified by *some* $P \in \mathcal{P}$. Notice that any negative example correctly classified by a single P placed in $\mathcal{H}$ will be correctly classified as negative by $\mathcal{H}$. We then repeatedly pick the

pattern from $\mathcal{P}$ that reduces the false positive error the most (while keeping the false negative error low). We show that after intersecting at most $s = 4(k+1)\ln\frac{80}{39\epsilon}$ patterns from $\mathcal{P}$ to form $\mathcal{H}$, the overall error rate for $\mathcal{H}$ is at most ϵ. The full algorithm is given in figures 5 and 6 and is described in more detail in Section 6.1. Section 6.2 proves the algorithm's correctness.

Note that along with receiving as inputs ϵ and δ, *Learn-Pattern* also receives an upper bound η_b for the noise rate, and an upper bound k for the number of points in the target pattern. If either of these values is not known a priori then the standard doubling technique can be applied (Haussler et al., 1991). It is important to note that the procedure used to simulate the statistical queries using labeled examples from $\mathcal{D}_\eta$ essentially "inverts" the classification noise process. To achieve this goal it requires a sufficiently accurate estimate of the true noise rate η. Since the learner does not know η but rather only has an upper bound η_b, the entire learning process is repeated for $O(\frac{k}{\epsilon^2}\log\frac{1}{1-2\eta_b})$ values for η—one of which is guaranteed to be sufficiently accurate (Aslam & Decatur, 1998). Then hypothesis testing is used to pick the best of the hypotheses (Haussler et al., 1991).

6.1. *Details of the algorithm*

As described above, *Cover-Positives* takes as input a set of points S from the examples in the unlabeled sample $\mathcal{S}_u$. It returns H_+, a pattern with at most $3k$ points that, with high probability, has a false negative error rate of at most $2\epsilon/5$. It does so via the following procedure. Let S' be the set of points of S that are not within unit distance of a point in the current H_+ and have not been deleted in Line 12 of *Cover-Positives* (figure 5). At each step we take the leftmost point p of S' and consider adding the point $q = p+1$ to H_+ (q is the rightmost point that would cover p). The statistical query of Line 6 of *Cover-Positives* ($\chi_{\text{PosNotNear}}(q)$) estimates the probability that no point in a random positive example is within unit distance of q. If this estimate is too large, we do not add q to H_+ since it would cause H_+ to misclassify too many positive examples. However, there might be significant positive weight more than unit distance to the left of q. Thus we shift q left to $p'+1$ where p' is the rightmost point in S left of p (see figure 7). If we find a q to the right of $p-1$ that is near a sufficiently large amount of positive weight, then we add it to H_+ and all points in S' within unit distance of q are removed since q covers them. Otherwise we continue sliding q to the left, setting it to $p'+1$ where p' is the next point left of p'. If ever $q < p-1$, then we can show that p must be from a negative example and thus can be removed from S'. The proof of correctness of *Cover-Positives* is in Lemma 2.

Then we use H_+ to construct the set of patterns $\mathcal{P}$. Recall that an example is negative because either (1) it has no point within unit distance of some target point t or (2) it has a point that is not within unit distance of any target point. Let Δ be a value smaller than the minimum distance between two points in S. For ease of exposition, suppose for a moment that we knew the labels of the points in S (where for each point $p \in X \in \mathcal{S}_u$ we define the label of p to be that of X). We now consider the two possible cases for why $X \in \mathcal{S}_u$ could be negative.

Case 1. No point $p \in X$ is within unit distance of some target point t. Thus there is some width-2 interval I consistent with all the positive examples from $\mathcal{S}_u$ that does not contain

Learn-Pattern$(\epsilon, \delta, k, \eta_b)$
▷ $\mathcal{S}_u$ is an unlabeled sample from $\mathcal{D}_\eta$ of size m_u
▷ $S = \{p \in X : X \in \mathcal{S}_u\}$ (*i.e.* the points from examples in $\mathcal{S}_u$)
▷ $\mathcal{S}_\ell$ is a labeled sample from $\mathcal{D}_\eta$ of size m_ℓ (used by the SQ procedures to compute statistics)
▷ $\chi_{ExtraFalseNeg}(\mathcal{H}_1, \mathcal{H}_2) \doteq ((\mathcal{H}_1(X) = 1) \wedge (\mathcal{H}_2(X) = 0) \wedge (\ell = 1))$
▷ $\chi_{FalsePos}(\mathcal{H}) \doteq ((\mathcal{H}(X) = 1) \wedge (\ell = 0))$
▷ $\chi_{FixedFalsePos}(P, \mathcal{H}) \doteq (((\mathcal{H} \cap P)(X) = 0) \wedge (\mathcal{H}(X) = 1) \wedge (\ell = 0))$
▷ $\chi_{PosNotNear}(q) \doteq (\forall_{r \in X}(dist(r, q) > 1) \wedge (\ell = 1))$
▷ All χs implicitly take a labeled example (X, ℓ)

1 $H_+ \leftarrow$ **Cover-Positives**(S)
2 $\mathcal{P} \leftarrow \{H_+\}$
3 For each $r \in S$
4 **Add-Good-Pattern**$(\mathcal{P}, H_+ \cup \{r + 1\}, H_+)$
5 **Add-Good-Pattern**$(\mathcal{P}, H_+ \cup \{r - 1 - \Delta\}, H_+)$ ▷ Δ is smaller than the minimum
6 For each $r_\ell, r_r \in S$ (for $r_\ell < r_r$) distance between two points in S
7 $R \leftarrow \{x : (x \in H_+) \wedge (x \in [r_\ell - 1 + \Delta, r_r + 1 - \Delta])\}$
8 If $R \neq \emptyset$
9 **Add-Good-Pattern**$(\mathcal{P}, (H_+ \setminus R) \cup \{r_\ell - 1\}, H_+)$
10 **Add-Good-Pattern**$(\mathcal{P}, (H_+ \setminus R) \cup \{r_r + 1\}, H_+)$
11 **Add-Good-Pattern**$(\mathcal{P}, (H_+ \setminus R) \cup \{r_\ell - 1\} \cup \{r_r + 1\}, H_+)$
12 $\mathcal{H} \leftarrow$ the always true hypothesis
13 While $\mathbf{SQ}(\chi_{FalsePos}(\mathcal{H}), 1/9, \epsilon/2) > 4\epsilon/9$ (and not $\perp$) and the loop has executed at most s times
14 $P_{best} \leftarrow \text{argmax}_{P \in \mathcal{P}} \left\{\mathbf{SQ}\left(\chi_{FixedFalsePos}(P, \mathcal{H}), 1/3, \frac{\epsilon}{6(k+1)}\right)\right\}$
15 $\mathcal{H} \leftarrow \mathcal{H} \cap P_{best}$
16 $\mathcal{P} \leftarrow \mathcal{P} \setminus \{P_{best}\}$
17 Return $\mathcal{H}$

Cover-Positives(S)
1 $S' \leftarrow S$
2 $H_+ \leftarrow \emptyset$
3 Repeat ▷ Greedily cover points in S' using points within distance 1 of most pos exs
4 $p' \leftarrow p \leftarrow$ leftmost point in S'
5 $q \leftarrow p + 1$
6 While $(\mathbf{SQ}(\chi_{PosNotNear}(q), 15/17, \epsilon/(5k)) > 2\epsilon/(85k))$ and $(q \geq p - 1)$
7 Slide p' to the rightmost point to its left in S and set $q \leftarrow p' + 1$
8 If $q \geq p - 1$ then
9 $H_+ \leftarrow H_+ \cup \{q\}$
10 $S' \leftarrow S' \setminus \{x \in S' : dist(q, x) \leq 1\}$
11 Else
12 $S' \leftarrow S' \setminus \{p\}$
13 Until $S' = \emptyset$
14 Return H_+

Add-Good-Pattern$(\mathcal{P}, P, H_+)$
1 $\mathcal{H}_{H_+} \leftarrow \{H_+\}$
2 $\mathcal{H}_P \leftarrow \{P\}$
3 If $\mathbf{SQ}(\chi_{ExtraFalseNeg}(\mathcal{H}_{H_+}, \mathcal{H}_P), 1/3, \epsilon/(20k)) \leq \epsilon/(30k)$ (or $\perp$)
4 $\mathcal{P} \leftarrow \mathcal{P} \cup \{P\}$

Figure 5. *Learn-Pattern* is our noise-tolerant algorithm to learn one-dimensional patterns. It takes as input the error bound ϵ, confidence parameter δ, an upper bound η_b for the noise rate, and an upper bound k for the number of points in the target concept. The standard doubling technique can be applied if either η_b or k is not provided. *Cover-Positives* creates a pattern H_+ that guarantees a false negative error rate of at most $2\epsilon/5$. *Add-Good-Pattern* checks if P correctly classifies most positive examples, and if so it adds P to the set of patterns $\mathcal{P}$.

$\mathbf{SQ}(\chi_{FalsePos}(\mathcal{H}), \mu, \theta)$
  $count1 \leftarrow$ no. of exs $(X, \ell) \in S_\ell$ where $(\mathcal{H}(X) = 1) \wedge (\ell = 0)$
  $count2 \leftarrow$ no. of exs $(X, \ell) \in S_\ell$ where $\mathcal{H}(X) = 1$
  Return $(count1 - \hat{\eta} \cdot count2)/(|S_\ell|(1 - 2\hat{\eta}))$

Figure 6. The procedure used to estimate P_χ for $\chi = \chi_{\text{FalsePos}}(\mathcal{H}) \doteq ((\mathcal{H}(X) = 1) \wedge (\ell = 0))$ with a noise rate estimate of $\hat{\eta}$ using $\mathcal{S}_\ell$ drawn from the noisy distribution $\mathcal{D}_\eta$. The procedures to simulate the other statistical queries are similar.

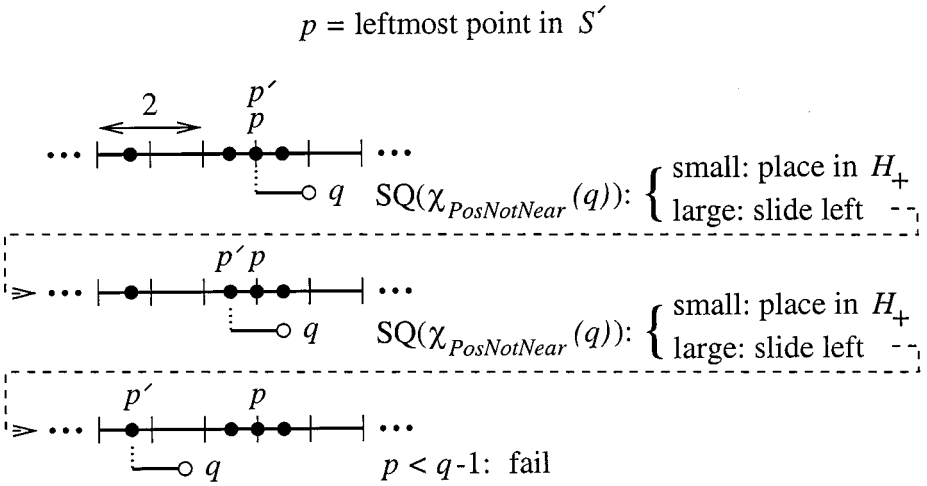

Figure 7. An example of adding a point in H_+ to cover p, the leftmost point of S' (also the leftmost uncovered point of S). The distance between adjacent tick marks is unit distance. If q is such that it would not misclassify too many positive examples (i.e. a sufficiently small value is returned by the SQ on $\chi_{\text{PosNotNear}}(q)$ of Line 6 of *Cover-Positives*), then it is placed in H_+ and all points of S' within unit distance of q are removed. Otherwise we slide q left to $p' + 1$ where p' is the rightmost point left of p in S.

any point in X. By adding the midpoint of I to H_+, X would be correctly classified. For any width-2 interval, there is some width-2 interval of the form $[r, r + 2]$ (a center of $r + 1$) or $[r - 2 - \Delta, r - \Delta]$ (a center of $r - 1 - \Delta$) for $r \in S$ that is equivalent with respect to the points in S. When $I = [r, r + 2]$, we are assuming that r is from a positive example, so r is included in I. Similarly, when $I = [r - 2 - \Delta, r - \Delta]$, we are assuming that r is from a negative example, so r is excluded from I. (Only these two cases are used because of the way that H_+ is built from left to right; see the proof of Lemma 3 for more detail.) Since we do not have labels for $\mathcal{S}_u$ we treat each point $p \in S$ as a positive point (in Line 4 of *Learn-Pattern*) and a negative point (in Line 5 of *Learn-Pattern*).

Case 2. Example X has a point p that is not within unit distance of a target point. Thus there is some interval I that contains p and does not contain any point from a positive example of $\mathcal{S}_u$. So in this case it suffices to just consider all intervals $[r_\ell + \Delta, r_r - \Delta]$ for $r_\ell, r_r \in S$. By removing any points in H_+ that are within unit distance of I and adding

a point just outside that range to the left (at $r_\ell - 1$) and/or to the right (at $r_r + 1$), any point $p \in I$ will be classified as negative. These steps are taken in Lines 9, 10, and 11 of *Learn-Pattern*.

The query on Line 3 of *Add-Good-Pattern* ensures that no pattern we add has false negative error that is significantly larger than that of H_+. This stage of the algorithm is proven correct in Lemma 3.

The while loop of Line 13 of *Learn-Pattern* is the final stage of the algorithm. It repeatedly selects the pattern P_{best} of $\mathcal{P}$ that reduces $\mathcal{H}$'s false positive error the most (as measured by the query in Line 14). The loop repeats until $\mathcal{H}$'s false positive error is sufficiently small (as measured by the query in Line 13) or until the loop has executed s times. This greedy covering of the negative weight is shown to be correct in the proof of Theorem 3.

6.2. *Proof of correctness*

The main result of this section is the following.

Theorem 3. *For*

$$D = O\left(k^2 \log n \cdot \log\left(\frac{1}{\epsilon}\right) \cdot \left(\log k + \log\log\frac{1}{\epsilon}\right)\right),$$

$$|\mathcal{S}_u| = O\left(\frac{k}{\epsilon}\log\frac{1}{\delta} + \frac{Dk}{\epsilon}\log\frac{k}{\epsilon}\right),$$

and

$$|\mathcal{S}_\ell| = O\left(\frac{Dk^2}{\epsilon^2(1-2\eta_b)^2}\left(\log\frac{k}{\epsilon(1-2\eta_b)} + \log\frac{1}{\delta}\right)\right),$$

with probability at least $1-\delta$, Learn-Pattern outputs a hypothesis $\mathcal{H}$ with error at most ϵ even under random classification noise of rate $\eta \le \eta_b < 1/2$.

We now state and prove a sequence of lemmas that are used to prove this main result. First we give results that upperbound the VC dimensions of the hypothesis space and of the query space. We will use these results with Theorem 1 to show in Lemma 1 that any hypothesis consistent with $\mathcal{S}_u$ will have error at most $\epsilon/(80k)$ with probability at least $1-\frac{\delta}{2}$. Then Lemma 2 argues that H_+, the output of *Cover-Positives*, has sufficiently small error on the positive examples, and Lemma 3 shows that when some of the patterns derived from H_+ are intersected, the intersection is consistent with negative examples from $\mathcal{S}_u$. Finally, Lemma 4 proves that the SQ estimating the false positive error rate is sufficiently accurate. We then combine these lemmas to prove Theorem 3.

To upperbound the VC dimensions of $\mathcal{C}_{\text{hyp}}$, our hypothesis class and $\mathcal{Q}$, our query space, we apply a result of Goldberg and Jerrum (1995), which identifies general situations where the VC dimension of a hierarchical concept class is guaranteed to be only polynomial in n and k, as required for PAC learning.

Theorem 4 (Goldberg & Jerrum, 1995). *Let $\{\mathcal{C}_{k,n}: k, n \in \mathbf{N}\}$ be a family of concept classes where concepts in $\mathcal{C}_{k,n}$ and instances are represented by k and n real values, respectively. Suppose that the membership test for any instance and any concept C of $\mathcal{C}_{k,n}$ can be expressed as a boolean formula $\Phi_{k,n}$ containing $\sigma = \sigma(k, n)$ distinct atomic predicates, each predicate being a polynomial inequality over $k + n$ variables (representing C and x) of degree at most $d = d(k, n)$. Then* $\mathrm{VCD}(\mathcal{C}_{k,n}) \leq 2k \ln(8ed\sigma)$.

Corollary 1. *Let $\mathcal{C}_{k,n}$ be sets of points on the line under the Hausdorff metric. Then* $\mathrm{VCD}(\mathcal{C}_{k,n}) \leq 2k \ln(8ekn) \leq 2k \lg(8ekn)$.

This follows from the fact that unit distance (with respect to the Hausdorff metric) between a set of k points on the line and a set of n points on the line can be determined by a set of kn distinct degree 1 inequalities in their coordinates.

Combined with a result from Blumer et al. (1989) we can upperbound the VC dimension of our hypothesis class.

Theorem 5 (Blumer et al., 1989). *For concept class $\mathcal{C}$, the class of concepts defined by the intersection of at most s concepts from $\mathcal{C}$ has VC dimension at most* $2\mathrm{VCD}(\mathcal{C})s \lg(3s)$.

We will later show that $\mathcal{H}$ is the intersection of at most $s = 4(k+1)\ln(\frac{80}{39\epsilon})$ patterns from $\mathcal{P}$. Combining this with Corollary 1 and Theorem 5, we get the following result.

Corollary 2. *The VC dimension of $\mathcal{C}_{\mathrm{hyp}} = \{\bigcap_{\leq s} P: P \in \mathcal{P}_{3k+1}\}$ is upperbounded by*

$$
\begin{aligned}
&2[2(3k+1)\lg(8en(3k+1))] \cdot 4(k+1)\ln\left(\frac{80}{39\epsilon}\right) \cdot \lg\left(3 \cdot 4(k+1)\ln\left(\frac{80}{39\epsilon}\right)\right) \\
&\quad = 16(k+1)(3k+1)\lg(8en(3k+1)) \cdot \ln\left(\frac{80}{39\epsilon}\right) \cdot \left[\lg(12(k+1)) + \lg\ln\left(\frac{80}{39\epsilon}\right)\right] \\
&\quad = O\left(k^2(\log n + \log k)\log\left(\frac{1}{\epsilon}\right) \cdot \left(\log k + \log\log\frac{1}{\epsilon}\right)\right) \\
&\quad = O\left(k^2 \log n \cdot \log\left(\frac{1}{\epsilon}\right) \cdot \left(\log k + \log\log\frac{1}{\epsilon}\right)\right).
\end{aligned}
$$

Proof: To obtain the first equation we apply Theorem 5 with $s = 4(k+1)\ln(\frac{80}{39\epsilon})$ and $\mathrm{VCD}(\mathcal{C}) = 2(3k+1)\lg(8en(3k+1))$. The rest follows from algebra. □

Next, we bound the VC dimension of the SQ query space. This bound is a consequence of Corollary 2.

Corollary 3. *The VC dimension of the SQ query space $\mathcal{Q}$ is upperbounded by*

$$
O\left(k^2 \log n \cdot \log\left(\frac{1}{\epsilon}\right) \cdot \left(\log k + \log\log\frac{1}{\epsilon}\right)\right).
$$

Proof: In the Appendix.

We now combine Theorem 1 with Corollary 2 to derive the sample size required for $\mathcal{S}_u$.

Lemma 1. *Any hypothesis consistent with $\mathcal{S}_u$ has error at most $\epsilon/(80k)$ with probability at least $1-\frac{\delta}{2}$. This also means that with probability at least $1-\frac{\delta}{2}$, S has a point in any interval of weight at least $\epsilon/(80k)$.*

Proof: Using an error bound of $\epsilon/(80k)$ and confidence of $1-\frac{\delta}{2}$, Theorem 1 requires a sample complexity of

$$m_u = \max\left(\frac{320k}{\epsilon}\lg\frac{4}{\delta}, \frac{640k\,\mathrm{VCD}(\mathcal{C}_{\mathrm{hyp}})}{\epsilon}\lg\frac{1040k}{\epsilon}\right),$$

which is consistent with the value of $|\mathcal{S}_u|$ given in Theorem 3. □

Throughout the remainder of this section we assume that $\mathcal{S}_u$ is a *good sample*, i.e. any hypothesis consistent with the examples of $\mathcal{S}_u$ (if the labels were known) will have error at most $\epsilon/(80k)$. We also assume that *all* of the statistical queries are correctly simulated (the size of $\mathcal{S}_\ell$ will be chosen so that with probability at least $1-\delta/2$ all of the statistical queries are correctly simulated).

For a hypothesis $\mathcal{H}$ let the false negative error of $\mathcal{H}$ be the probability with respect to $\mathcal{D}$ that $(\mathcal{H}(X)=0)\wedge(\ell=1)$. We now argue that the false negative error rate of H_+ is at most $\frac{2\epsilon}{5}$. Suppose that p is a point in H_+. For a positive example X, $H_+(X)$ could only be positive if there is a point of X within unit distance of p. Thus any p that violates this condition should not be put in H_+. If a point p is within unit distance of point q we say that *p covers q*.

Lemma 2. *With probability at least $1-\delta$, the procedure* Cover-Positives *returns a pattern H_+ of at most $3k$ points for which the false negative error rate of H_+ is at most $\frac{2\epsilon}{5}$.*

Proof: With probability at least $1-\delta$, we have that $\mathcal{S}_u$ is a good sample and all statistical queries are correctly simulated.

Let P_χ be the probability that an example X has all of its points greater than distance one from point q (i.e. $\Pr[\chi_{\mathrm{PosNotNear}}(q)=1]$). Let $\hat{P}_\chi$ be the estimate returned for P_χ by the SQ oracle for $\mu=15/17$ and $\theta=\epsilon/(5k)$. At each step we take the leftmost point p of S' and consider adding the point $q=p+1$ to H_+ (q is the rightmost point that would cover p). If $\hat{P}_\chi > 2\epsilon/(85k)$, we do not add q to H_+ since it would cause H_+ to misclassify too many positive examples. However, there might be significant positive weight more than unit distance to the left of q. Thus we shift q left to $p'+1$ where p' is the rightmost point in S left of p (see figure 7). If we find a q with $\hat{P}_\chi \le 2\epsilon/(85k)$ to the right of $p-1$ then it is added to H_+ and all points in S' within unit distance of q are removed since q covers them. Otherwise we continue sliding q to the left, setting it to $p'+1$ where p' is the next point left of p'. If ever $q < p-1$, then, as we show in Lemma 5 (in the Appendix), p must be from a negative example and thus can be removed from S'.

If $\hat{P}_\chi \leq 2\epsilon/(85k)$ then $P_\chi \leq \frac{\hat{P}_\chi}{1-\mu} \leq \frac{2\epsilon}{85k} \cdot \frac{17}{2} = \epsilon/(5k)$. Further, since $\theta = \epsilon/(5k)$ it follows that for any point q added to H_+, $P_\chi \leq \epsilon/(5k)$. Finally, since S has a point in any interval of weight at least $\epsilon/(80k)$ (by Lemma 1), for each target interval some point q_* considered has $P_\chi \leq \epsilon/(80k)$. For this q_*, $\hat{P}_\chi \leq P_\chi(1+\mu) \leq \frac{\epsilon}{(80k)} \cdot \frac{32}{17} = \frac{2\epsilon}{85k}$. Thus for each target interval, a point is placed in H_+. We show in Lemma 6 (in the Appendix) that only the leftmost and rightmost points placed in H_+ for each target interval add error. So H_+'s total false negative error rate is at most $2k \cdot \frac{\epsilon}{5k} = \frac{2\epsilon}{5}$. □

Next we must prove that the patterns placed in $\mathcal{P}$ are sufficient to properly classify the negative examples in $\mathcal{S}_u$ (and hence can be used to create a hypothesis with a low false positive error rate).

Lemma 3. *With probability at least $1-\delta$, there exists a set of patterns from $\mathcal{P}$ that when intersected are consistent with the negative examples of $\mathcal{S}_u$.*

Proof: With probability at least $1-\delta$, we have that $\mathcal{S}_u$ is a good sample, and all statistical queries are correctly simulated.

Recall that an example is negative because either (1) it has no point within unit distance of some target point t or (2) it has a point that is not within unit distance of any target point. Let H_+ be the pattern returned by *Cover-Positives*. Recall from figure 5 that $S = \{p \in X : X \in \mathcal{S}_u\}$ and Δ is smaller than the minimum distance between two points in S. For ease of exposition, suppose for a moment that we knew the labels of the points in S (where for each point $p \in X \in \mathcal{S}_u$ we define the label of p to be that of X). We now consider the two possible cases for why $X \in \mathcal{S}_u$ could be negative.

Case 1. No point $p \in X$ is within unit distance of some target point t. Thus there is some width-2 interval I consistent with all the positive examples from $\mathcal{S}_u$ that does not contain any point in X. By adding the midpoint of I to H_+, X would be correctly classified. For any width-2 interval, there is some width-2 interval of the form $[r, r+2]$ (a center of $r+1$) or $[r-2-\Delta, r-\Delta]$ (a center of $r-1-\Delta$) for $r \in S$ that is equivalent with respect to the points in S. When $I = [r, r+2]$, we are assuming that r is from a positive example, so r is included in I. Similarly, when $I = [r-2-\Delta, r-\Delta]$, we are assuming that r is from a negative example, so r is excluded from I. (Only these two cases are used because of the way that H_+ is built from left to right; see below.) Since we do not have labels for $\mathcal{S}_u$ we treat each point $p \in S$ as a positive point (in Line 4 of *Learn-Pattern*) and a negative point (in Line 5 of *Learn-Pattern*).

Case 2. Example X has a point p that is not within unit distance of a target point. Thus there is some interval I that contains p and does not contain any point from a positive example of $\mathcal{S}_u$. So in this case it suffices to just consider all intervals $[r_\ell + \Delta, r_r - \Delta]$ for $r_\ell, r_r \in S$. By removing any points in H_+ that are within unit distance of I and adding a point just outside that range to the left (at $r_\ell - 1$) and/or to the right (at $r_r + 1$), any point $p \in I$ will be classified as negative. These steps are taken in Lines 9, 10, and 11 of *Learn-Pattern*.

We now argue that some patterns correctly classifying all negative weight regions described in Cases 1 and 2 above are inserted into $\mathcal{P}$. First observe that all regions of negative

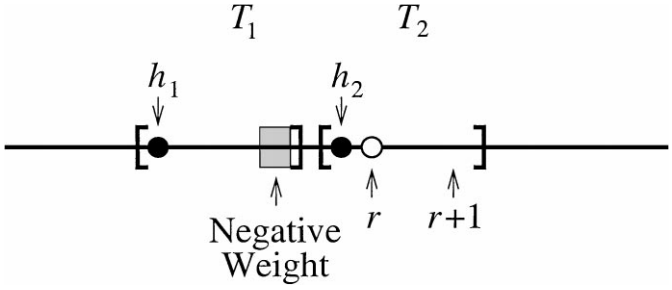

Figure 8. An example of a pattern created by Line 4 of *Learn-Pattern*. The points h_1 and h_2 are points from H_+. This line places a point at $r+1$, which is more than unit distance from negative weight when the negative weight is covered by an H_+ point to its right (h_2).

weight that cause their corresponding examples to be misclassified by H_+ must lie within unit distance of some point in H_+. Otherwise H_+ would correctly classify those examples as negative. This means that all such negative weight regions lie within distance 3 of some target point. We now consider Cases 1 and 2.

Patterns addressing Case 1 are built in Lines 4 and 5 of *Learn-Pattern*. An example of Line 4's purpose is shown in figure 8. Here the negative weight is from examples that are negative because they have no point in target interval T_2. However, H_+ point h_2 that covers positive weight in T_2 also covers the negative weight region. So the desired interval of Case 1 is $I = [r, r+2]$. When I's midpoint is added to pattern P, P correctly classifies all negative examples associated with the negative weight. One such r that works is the leftmost positive point in T_2. By Lemma 1, the weight of the positive points in T_2 to the left of r is at most $\epsilon/(80k)$. This is the most positive weight correctly classified by H_+ that is not correctly classified by P. So for pattern P, the true probability of the SQ of Line 3 of *Add-Good-Pattern* is $P_\chi \leq \epsilon/(80k)$, implying $\hat{P}_\chi \leq \frac{4}{3} \cdot \frac{\epsilon}{(80k)} = \frac{\epsilon}{60k}$. Thus P will be added to $\mathcal{P}$.

An example of Line 5's purpose is in figure 9. Here the negative weight is from examples that are negative because they have no point in target interval T_1. However, H_+ point h_1 that covers positive weight in T_1 also covers the negative weight region. So the desired interval

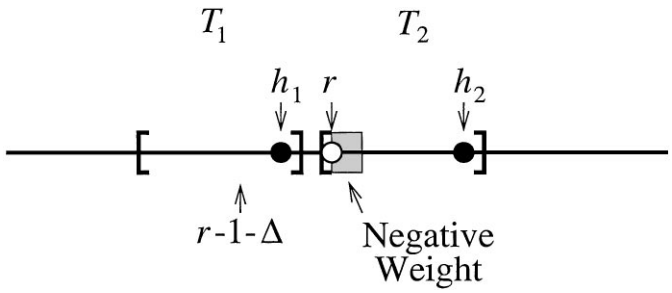

Figure 9. An example of a pattern created by Line 5 of *Learn-Pattern*. The points h_1 and h_2 are points from H_+. This line places a point at $r-1-\Delta$, which is more than unit distance from negative weight when the negative weight is covered by an H_+ point to its left (h_1).

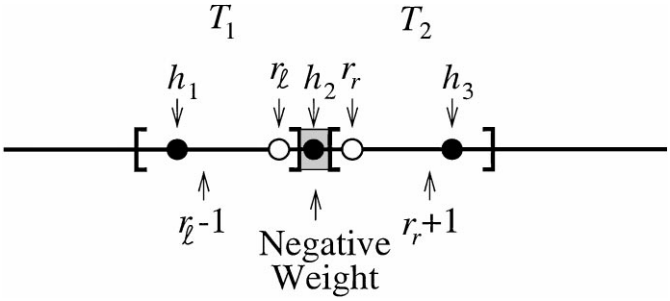

Figure 10. An example of a pattern created by Line 11 of *Learn-Pattern*. It is meant to remove all H_+ points (h_2) from unit distance of the negative weight region and place points at $r_\ell - 1$ and $r_r + 1$, which are more than unit distance from the negative weight. These points cover the positive weight uncovered by the deletion of h_2.

of Case 1 is $I = [r - 2 - \Delta, r - \Delta]$. When I's midpoint is added to pattern P, P correctly classifies all negative examples associated with the negative weight. One such r that works is the leftmost negative point in the negative weight region. Any other H_+ point in T_1 must also cover the negative weight region because otherwise the negative examples with points in the region would be correctly classified by H_+. So P with its extra point at $r - 1 - \Delta$ covers all the positive weight that H_+ covers. So for pattern P, the true probability of the SQ of Line 3 of *Add-Good-Pattern* is $P_\chi = 0$, implying $\hat{P}_\chi = 0$. Thus P will be added to $\mathcal{P}$. Note that the results for Lines 4 and 5 also hold when the negative weight does not lie in another target interval (though these fall under Case 2).

Patterns addressing Case 2 are built in Lines 9, 10, and 11 of *Learn-Pattern*. We only discuss Line 11 because it is similar to Lines 9 and 10. An example of Line 11's purpose is in figure 10. Here the negative weight is from examples that are negative because they have at least one point that is not in any target interval. However, H_+ point h_2 that covers positive weight in T_1 and T_2 also covers the negative weight region. So the desired interval of Case 2 is $I = [r_\ell + \Delta, r_r - \Delta]$. When the H_+ points within unit distance of I are removed and points are placed in P just beyond unit distance of I to the left and right, pattern P correctly classifies all negative examples associated with the negative weight. Two points r_ℓ and r_r that work are the rightmost positive point in T_1 (for r_ℓ) and the leftmost positive point in T_2 (for r_r). By Lemma 1, the weight of the positive points in T_1 to the right of r_ℓ is at most $\epsilon/(80k)$. The same holds for the weight of the positive points in T_2 to the left of r_r. So the most positive weight correctly classified by H_+ that is not correctly classified by P is at most $2\epsilon/(80k) = \epsilon/(40k)$. So for pattern P, the true probability of the SQ of Line 3 of *Add-Good-Pattern* is $P_\chi \leq \epsilon/(40k)$, implying $\hat{P}_\chi \leq \frac{4}{3} \cdot \frac{\epsilon}{(40k)} = \frac{\epsilon}{30k}$. Thus P will be added to $\mathcal{P}$. The patterns created in Lines 9 and 10 are similar, except the most positive weight they cannot cover is half as much as for Line 11. □

We now prove that the statistical query χ_{FalsePos} used to estimate the false positive error rate provides sufficient accuracy. For (X, ℓ), a random labeled example drawn from $\mathcal{D}_\eta$, define $P_\chi \doteq \Pr[(\mathcal{H}(X) = 1) \wedge (\ell = 0)]$. That is, P_χ is the false positive error rate. Let $\hat{P}_\chi$ be the estimate returned by the statistical query oracle for P_χ for $\mu = 1/9$ and $\theta = \epsilon/2$.

Lemma 4. *Let T be the test (or predicate) "$(\hat{P}_\chi \leq 4\epsilon/9) \vee (\hat{P}_\chi = \perp)$." Once $P_\chi \leq 2\epsilon/5$ then the hypothesis $\mathcal{H}$ will pass test T. Furthermore, for any hypothesis $\mathcal{H}$ that passes T, $P_\chi \leq \epsilon/2$.*

Proof: If $P_\chi \leq 2\epsilon/5$ then $\hat{P}_\chi \leq P_\chi(1+\mu) \leq \frac{2\epsilon}{5} \cdot \frac{10}{9} = 4\epsilon/9$, which satisfies T. Also, if $\hat{P}_\chi \leq 4\epsilon/9$ then $P_\chi \leq \hat{P}_\chi \cdot \frac{1}{1-\mu} = \frac{4\epsilon}{9} \cdot \frac{9}{8} = \epsilon/2$. Similarly, if $\hat{P}_\chi = \perp$, then by definition $P_\chi \leq \epsilon/2$. □

We now prove the main result of this section.

Proof [Theorem 3]: First we argue that $\mathcal{S}_\ell$ is sufficiently large so that with probability at least $1-\delta/2$, all of the statistical queries are properly simulated. Examination of figure 5 reveals that the minimum values of μ and θ over all queries are $1/9$ and $\epsilon/20k$, respectively, so these values are entered as μ_* and θ_* in Theorem 2. In the worst case, every query in the algorithm depends on the examples' labels, so $\rho_* = \theta_*$ in Theorem 2. Thus the sample size required for the (noisy) labeled examples so that the statistical queries of *Learn-Pattern*, *Add-Good-Pattern* and *Cover-Positives* are simulated with confidence at least $1 - \frac{\delta}{2}$ is

$$O\left(\frac{\text{VCD}(\mathcal{Q})k^2}{\epsilon^2(1-2\eta_b)^2}\left(\log\frac{k}{\epsilon(1-2\eta_b)} + \log\frac{1}{\delta}\right)\right)$$

where $\text{VCD}(\mathcal{Q})$ is given in Corollary 3. These queries can tolerate classification noise of rate $\eta \leq \eta_b < 1/2$. Furthermore, with probability at least $1-\delta/2$, $\mathcal{S}_u$ is a good sample (Lemma 1). For the remainder of this proof we assume that $\mathcal{S}_u$ is a good sample and *all* statistical queries are correctly simulated. From the above, with probability at least $1-\delta$, this will be the case.

We now prove that the false positive error rate of the final hypothesis is at most $\epsilon/2$. Let e_i be the false positive error of $\mathcal{H}$ after i iterations of the while loop of *Learn-Pattern*. Let $P_P = \Pr[((\mathcal{H} \cap P)(X) = 0) \wedge (\ell = 0) \wedge ((\mathcal{H}(X) = 1))]$, and let $\hat{P}_P$ be the estimate for P_P from an SQ oracle with $\mu = 1/3$ and $\theta = \frac{\epsilon}{6(k+1)}$. Since (by Lemma 1) correctly classifying all negative points from $\mathcal{S}_u$ would guarantee a false positive error rate of at most $\epsilon/(80k) \leq \epsilon/80k$, from Lemma 3 we have that there exists a set of patterns from $\mathcal{P}$ that when intersected yield a false positive error rate of at most $\epsilon/(80k) \leq \epsilon/80k$. In fact, we have something even stronger. Let $\mathcal{S}^-$ be the set of negative examples in $\mathcal{S}_u$. Notice that at least $|\mathcal{S}^-|/2$ of the negative examples fall into Case 1 or at least $|\mathcal{S}^-|/2$ of the negative examples fall into Case 2 (where Case 1 and Case 2 are as in the proof of Lemma 3). Since there are only k target points, if $|\mathcal{S}^-|/2$ of the negative examples fall into Case 1 then by an averaging argument, there is some interval (and thus some pattern in $\mathcal{P}$) that would properly classify at least $|\mathcal{S}^-|/(2k)$ of the examples of $\mathcal{S}^-$. Similarly, since the portions of the real line that are not within unit distance of any target point form at most $k+1$ contiguous intervals, when $|\mathcal{S}^-|/2$ of the negative examples fall into Case 2, by an averaging argument, there is some pattern in $\mathcal{P}$ that would properly classify at least $|\mathcal{S}^-/(2k+1)|$ of the negative

examples. Thus,

$$\exists P \in \mathcal{P}\colon P_P \geq \left(e_i - \frac{\epsilon}{80}\right)\frac{1}{2(k+1)}.$$

Let P be a pattern meeting the above condition. For this pattern,

$$\hat{P}_P \geq \left(e_i - \frac{\epsilon}{80}\right)\frac{1}{2(k+1)} \cdot \frac{2}{3} = \left(e_i - \frac{\epsilon}{80}\right)\frac{1}{3(k+1)}.$$

When $e_i \leq 2\epsilon/5$ we exit the while loop (by Lemma 4). Thus

$$P_P \geq \left(\frac{2\epsilon}{5} - \frac{\epsilon}{80}\right)\frac{1}{2(k+1)} = \frac{31\epsilon}{160(k+1)} \geq \frac{\epsilon}{6(k+1)}.$$

We now compute the maximum number of rounds s of the while loop needed until $e_i \leq \frac{2\epsilon}{5}$, completing the learning task (by Lemma 4). Since we know there is a P for which $\hat{P}_P \geq (e_i - \frac{\epsilon}{80})\frac{1}{3(k+1)}$, it immediately follows that $P_{P_{\text{best}}} \geq (e_i - \frac{\epsilon}{80})\frac{1}{3(k+1)}\frac{3}{4} = (e_i - \frac{\epsilon}{80})\frac{1}{4(k+1)}$ (where P_{best} is as defined in figure 5). Let $c = \frac{1}{4(k+1)}$. Thus we obtain the recurrence: $e_i \leq e_{i-1} - (e_{i-1} - \frac{\epsilon}{80})c = e_{i-1}(1-c) + c\epsilon/80$ with $e_0 \leq 1$. Solving the recurrence yields

$$\begin{aligned} e_i &\leq e_0(1-c)^i + \frac{\epsilon}{80} - \frac{\epsilon}{80}(1-c)^i \\ &\leq (1-c)^i - \frac{\epsilon}{80}(1-c)^i + \frac{\epsilon}{80} \\ &= \frac{\epsilon}{80} + (1-c)^i\left(1 - \frac{\epsilon}{80}\right) \leq \frac{\epsilon}{80} + (1-c)^i. \end{aligned}$$

By selecting $i \geq s = \frac{1}{c}\ln(\frac{80}{39\epsilon}) = 4(k+1)\ln(\frac{80}{39\epsilon})$, we get that the false positive error rate $e_i \leq \epsilon/2$.

Finally, we prove that the false negative error rate of the final hypothesis is at most $\epsilon/2$. From Lemma 2 we know that H_+ has a false negative error rate of at most $2\epsilon/5$. Given the parameters of the SQ on Line 3 of *Add-Good-Pattern*, if the test passes we know $\hat{P}_\chi \leq \epsilon/(30k)$, implying $P_\chi \leq \frac{\hat{P}_\chi}{1-\mu} \leq \frac{\epsilon}{30k} \cdot \frac{3}{2} = \frac{\epsilon}{20k}$. If $\perp$ is returned, then $P_\chi \leq \theta = \epsilon/(20k)$. Thus each pattern $P \in \mathcal{P}$ adds at most $\epsilon/(20k)$ of false negative error to H_+'s false negative error. Call a region of positive weight *exposed* by pattern P if the region was covered by H_+ but is not covered by P. So each pattern in $\mathcal{P}$ exposes at most $\epsilon/(20k)$ of positive weight. Note that each point p added to H_+ to form a pattern P that passes the test acts as the midpoint of an interval that overlaps one or both edges of some target interval. Otherwise, P's false negative error is 1. Thus the positive weight exposed by P lies at one end of some target interval. Consider all the positive weight at one side of some target interval T that is exposed by the patterns in some set $\mathcal{P}' \subseteq \mathcal{P}$. Now consider a pattern P' that exposes the most positive weight of all patterns in $\mathcal{P}'$. The positive weight

exposed by P' is a superset of the union of the positive weight exposed by all patterns in $\mathcal{P}'$. So we can charge the extra false negative error of all patterns in $\mathcal{P}'$ to the single pattern P'. Since the number of target intervals is at most k, there exists $\mathcal{P}'' \subseteq \mathcal{P}$ with $|\mathcal{P}''| \leq 2k$ such that we can charge all the extra false negative error of the patterns in $\mathcal{P}$ to the patterns in $\mathcal{P}''$. So the total additional false negative error introduced by the intersection of the patterns in $\mathcal{P}$ is at most $2k \cdot \frac{\epsilon}{20k} = \epsilon/10$. Since the hypothesis $\mathcal{H}$ is the intersection of patterns of $\mathcal{P}$, $\mathcal{H}$'s total false positive error is at most $\frac{2\epsilon}{5} + \frac{\epsilon}{10} = \epsilon/2$. Thus we have now shown that the overall error of the final hypothesis is at most $\epsilon/2 + \epsilon/2 = \epsilon$.

Finally, it is easily seen that *Learn-Pattern* runs in time polynomial in $m_u + m_\ell$. □

Note that along with receiving as inputs ϵ and δ, *Learn-Pattern* also receives an upper bound η_b for the noise rate, and an upper bound k for the number of points in the target pattern. If either of these values is not known a priori then the standard doubling technique (Haussler et al., 1991) can be applied. It is important to note that the procedure used to simulate the statistical queries using labeled examples from $\mathcal{D}_\eta$ essentially "inverts" the noise process. To achieve this goal it requires a sufficiently accurate estimate of the true noise rate η. From Aslam and Decatur (1998), since the learner does not know η but rather only has an upper bound η_b, the entire learning process is repeated for $O(\frac{k}{\epsilon^2}\log\frac{1}{1-2\eta_b})$ values for $\hat{\eta}$—one of which is guaranteed to be sufficiently accurate. Then hypothesis testing (Haussler et al., 1991) is used to select the best of the hypotheses.

7. Modifications for empirical use

When designing a PAC learning algorithm, we assume the learner is given the desired error bound ϵ and confidence parameter δ. Then the learner draws the number of training examples that it needs to *guarantee* that the final hypothesis has error at most ϵ with probability at least $1 - \delta$. In contrast to this theoretical setting, when performing empirical studies typically one studies how the accuracy of the hypothesis changes as a function of the number of training examples. Thus the amount of data is fixed and the learner wants to construct the most accurate hypothesis that it can.

At first this difference in viewpoint does not seem significant—one can easily fix δ, obtain a function of m that gives the accuracy that can be guaranteed, and then use this as the value of ϵ in the algorithm. Consider Line 6 of *Cover-Positives*. The algorithm uses a statistical query to compute an estimate $\hat{P}_\chi$ for $\chi_{\text{PosNotNear}}(q)$. Then depending on whether or not $\hat{P}_\chi > 2\epsilon/(85k)$, different steps are taken. If the learner used the theoretical bounds to compute the worst-case value of ϵ for the given training data size and then used this value of ϵ in this cutoff for Line 6 of *Cover-Positives*, we might doom our algorithm's empirical performance to match the worst-case bounds. Similar problems can be seen in Line 13 of *Learn-Pattern* and Line 3 of *Add-Good-Pattern*.

To apply *Learn-Pattern* in the empirical setting so that we could obtain the most accurate hypothesis possible, we made the following changes. First, the cutoff used in Line 6 of *Cover-Positives* (to replace the $2\epsilon/(85k)$) had to be chosen very carefully. In particular, when building H_+ we found the performance (as measured by the false negative error rate of the H_+ obtained) to be very sensitive to this cutoff value. If the cutoff is too high then the

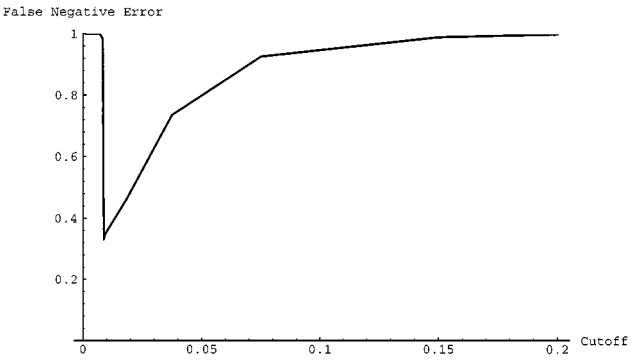

Figure 11. A typical example of H_+'s error versus the cutoff value used to build H_+. This motivates the binary search technique in determining the best cutoff.

false negative error rate is high. If the cutoff is too low, by even a small amount, then at least one target interval has no corresponding point in H_+, causing all truly positive examples to be misclassified. This behavior is exhibited in figure 11, a typical plot of the false negative error rate of H_+ versus the cutoff used to build it based on the simulated data of Section 8.1. Since an optimal cutoff depends on the sample size, noise rate, *and* the specific distribution for the data, in practice the only way to obtain the best results is to have the algorithm *learn* the best cutoff while also learning the target concept. Thus our simulations selected the cutoff empirically using a binary search procedure to find one that produced an H_+ with the lowest measured false negative error rate on the training data. Our algorithm performed a binary search on the interval [0, 0.3] for the appropriate cutoff. As long as H_+'s false negative error rate decreased, the lower half of the interval was searched. When the error rate increased, the upper half was searched. We found that this simple heuristic works well in general: in the experiments on simulated data (Section 8.1), this procedure failed to find a good cutoff less than 1% of the time, and we easily detected this event since the final hypotheses of those runs misclassified all positive examples. In the experiments on real data (Section 8.2), this procedure always found a good cutoff. However, we feel there is likely still room for improvement.

We now describe how the values for the other cutoffs were empirically selected (the pseudocode is given in figures 12–14). First note that in Line 14 of *Learn-Pattern* we simply pick the pattern with the largest measured value for $\hat{P}_{\chi_{\text{FixedFalsePos}}}$. Also, we exit the while loop of Line 13 of *Learn-Pattern* when the false positive error does not show any significant improvement (as measured by $FFP_{P_{\text{best}}}$ in *Learn-Pattern-Empirical*). We now discuss how we selected the cutoff used in Line 3 of *Add-Good-Pattern* (which has been merged into *Learn-Pattern-Empirical*). For the sake of efficiency we wanted to avoid using another binary search since running the rest of the algorithm to completion is significantly more time consuming than just running *Cover-Positives-Empirical* to completion. Since we used a binary search to obtain a pattern H_+ with the smallest possible false negative error rate, we first computed the measured false negative error rate of H_+ (which we denote by FN_{H_+}). The idea is to then only add patterns to our hypothesis $\mathcal{H}$ if they do not cause any

Learn-Pattern-Empirical() ▷ No arguments this time
▷ $\mathcal{S}_u$ is an unlabeled sample from $\mathcal{D}_\eta$ of size m_u
▷ $S = \{p \in X : X \in \mathcal{S}_u\}$ (*i.e.* the points from examples in $\mathcal{S}_u$)
▷ $\mathcal{S}_\ell$ is a labeled sample from $\mathcal{D}_\eta$ of size m_ℓ (used by the SQ procedures to compute statistics)
▷ (Replaced $\chi_{ExtraFalseNeg}(\mathcal{H}_1, \mathcal{H}_2)$ with $\chi_{FalseNeg}(\mathcal{H})$)
▷ $\chi_{FalseNeg}(\mathcal{H}) \doteq ((\mathcal{H}(X) = 0) \wedge (\ell = 1))$
▷ $\chi_{FalsePos}(\mathcal{H}) \doteq ((\mathcal{H}(X) = 1) \wedge (\ell = 0))$
▷ $\chi_{FixedFalsePos}(P, \mathcal{H}) \doteq (((\mathcal{H} \cap P)(X) = 0) \wedge (\mathcal{H}(X) = 1) \wedge (\ell = 0))$
▷ $\chi_{PosNotNear}(q) \doteq (\forall_{r \in X}(dist(r, q) > 1) \wedge (\ell = 1))$
▷ All χs implicitly take a labeled example (X, ℓ)

1 $H_+ \leftarrow$ **Build-H_+-Empirical**(S)
2 $FN_{H_+} \leftarrow$ **SQ-Empirical**$(\chi_{FalseNeg}(\{H_+\}))$
3 $\mathcal{P} \leftarrow \{H_+\}$
4 For each $r \in S$
5 $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{H_+ \cup \{r + 1\}\}$
6 $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{H_+ \cup \{r - 1 - \Delta\}\}$ ▷ Δ is smaller than the minimum
7 For each $r_\ell, r_r \in S$ (for $r_\ell < r_r$) distance between two points in S
8 $\quad R \leftarrow \{x : (x \in H_+) \wedge (x \in [r_\ell - 1 + \Delta, r_r + 1 - \Delta])\}$
9 $\quad$ If $R \neq \emptyset$
10 $\quad\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{(H_+ \setminus R) \cup \{r_\ell - 1\}\}$
11 $\quad\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{(H_+ \setminus R) \cup \{r_r + 1\}\}$
12 $\quad\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{(H_+ \setminus R) \cup \{r_\ell - 1\} \cup \{r_r + 1\}\}$
13 $\mathcal{H} \leftarrow$ the always true hypothesis
▷ Greedily cover negative weight until improvement to false pos error is small
14 $FFP_{P_{best}} \leftarrow 1.0$ ▷ This will track how much FP error P_{best} fixes
15 While $(\mathcal{P} \neq \emptyset)$ and $\left(FFP_{P_{best}} > 0.0005\right)$
16 $\quad P_{best} \leftarrow \text{argmax}_{P \in \mathcal{P}}\{$**SQ-Empirical**$(\chi_{FixedFalsePos}(P, \mathcal{H}))\}$
17 $\quad FFP_{P_{best}} \leftarrow$ **SQ-Empirical**$(\chi_{FixedFalsePos}(P, \mathcal{H}))$
18 $\quad \mathcal{P} \leftarrow \mathcal{P} \setminus \{P_{best}\}$
19 $\quad$ If $\left($**SQ-Empirical**$(\chi_{FalseNeg}(\mathcal{H} \cap P_{best})) \leq FN_{H_+} + 0.01\right)$ and
$\quad\quad \left(FFP_{P_{best}} > 0.0005\right)$ then
20 $\quad\quad \mathcal{H} \leftarrow \mathcal{H} \cap P_{best}$
21 Return $\mathcal{H}$

Figure 12. *Learn-Pattern-Empirical* is *Learn-Pattern* (figure 5) with modifications for empirical use. The arguments that *Learn-Pattern* used are omitted since they are not required by *Learn-Pattern-Empirical*. Arguments μ and θ are omitted from the SQs since these arguments are only used to determine the size of the labeled sample $\mathcal{S}_\ell$ in Theorem 3 and empirically we have fixed amounts of data.

significant increase in the false negative error rate. Finally, for the sake of efficiency we only applied this test when the pattern was selected as P_{best} in Line 14 of *Learn-Pattern*. Thus, for our empirical test, we replaced the test in Line 3 of *Add-Good-Pattern* by a test used just after Line 14 of *Learn-Pattern* in which we used $\chi_{\text{FalseNeg}} \doteq ((\mathcal{H}(X) = 0) \wedge (\ell = 1))$ to measure the false negative error rate of the hypothesis $\mathcal{H} \cap P_{\text{best}}$ (done on Line 19 of *Learn-Pattern-Empirical*). We then execute Line 15 of *Learn-Pattern* (on Line 20 of *Learn-Pattern-Empirical*) only when the estimate returned for the SQ was at most $FN_{H_+} + 0.01$. Otherwise, we move to the next best pattern until one passes this test.

By empirically selecting our cutoffs we have another very important benefit. First, since the amount of training data is fixed and the cutoffs used for our statistical queries are empirically selected, there is no place in which the value for k is used. Thus, the modification of our

Build-H_+-Empirical(S)
1 $cutoff_{low} \leftarrow 0.0$
2 $cutoff_{high} \leftarrow 0.3$
3 $H_{best} \leftarrow$ **Cover-Positives-Empirical**$(S, cutoff_{high})$
4 $FN_{high} \leftarrow FN_{best} \leftarrow$ **SQ-Empirical**$(\chi_{FalseNeg}(\{H_{best}\}))$
5 $cutoff_{tmp} \leftarrow (cutoff_{low}+cutoff_{high})/2$
▷ Loop until $cutoff_{high}$ and $cutoff_{low}$ are identical w.r.t. $\mathcal{S}_\ell$
6 While $cutoff_{high}-cutoff_{low} \geq 1/|\mathcal{S}_\ell|$
7 $H_{tmp} \leftarrow$ **Cover-Positives-Empirical**$(S,cutoff_{tmp})$
8 $FN_{tmp} \leftarrow$ **SQ-Empirical**$(\chi_{FalseNeg}(\{H_{tmp}\}))$
9 If $FN_{tmp} < FN_{best}$ then ▷ $cutoff_{tmp}$ produced a better hypothesis
10 $H_{best} \leftarrow H_{tmp}$
11 $FN_{best} \leftarrow FN_{tmp}$
12 If $FN_{tmp} > FN_{high}$ ▷ $cutoff_{tmp}$ is too low
13 $cutoff_{low} \leftarrow cutoff_{tmp}$
14 Else ▷ $cutoff_{tmp}$ might be too high
15 $cutoff_{high} \leftarrow cutoff_{tmp}$
16 $FN_{high} \leftarrow FN_{tmp}$
17 $cutoff_{tmp} \leftarrow (cutoff_{low}+cutoff_{high})/2$
18 Return H_{best}

Cover-Positives-Empirical$(S,cutoff)$
1 $S' \leftarrow S$
2 $H_+ \leftarrow \emptyset$
3 Repeat ▷ Greedily cover points in S' using points within distance 1 of most pos exs
4 $p' \leftarrow p \leftarrow$ leftmost point in S'
5 $q \leftarrow p+1$
6 While (**SQ-Empirical**$(\chi_{PosNotNear}(q)) > cutoff$) and $(q \geq p-1)$
7 Slide p' to the rightmost point to its left in S and set $q \leftarrow p'+1$
8 If $q \geq p-1$ then
9 $H_+ \leftarrow H_+ \cup \{q\}$
10 $S' \leftarrow S' \setminus \{x \in S' : dist(q,x) \leq 1\}$
11 Else
12 $S' \leftarrow S' \setminus \{p\}$
13 Until $S' = \emptyset$
14 Return H_+

Figure 13. *Build-H_+-Empirical* uses a binary search over $[0, 0.3]$ to find the best cutoff for building H_+ in *Cover-Positives-Empirical*. When $cutoff_{tmp}$ is chosen, a candidate H_+ (called H_{tmp}) is built and its false negative error is computed. As long as H_{tmp}'s false negative error rate decreases, the lower half of the interval $[cutoff_{low}, cutoff_{high}]$ is searched. When the rate increases, the upper half is searched. H_{best} tracks the best H_+ found so far. *Cover-Positives-Empirical* is identical to *Cover-Positives* (figure 5) except that the cutoff in Line 6 is given as an argument.

SQ-Empirical$(\chi_{FalsePos}(\mathcal{H}))$
$count1 \leftarrow$ no. of exs $(X,\ell) \in S_\ell$ where $(\mathcal{H}(X) = 1) \wedge (\ell = 0)$
Return $count1/|S_\ell|$

Figure 14. The procedure used in the empirical version of our algorithm to estimate P_χ for $\chi = \chi_{\text{FalsePos}}(\mathcal{H}) \doteq ((\mathcal{H}(X) = 1) \wedge (\ell = 0))$. All this procedure does is compute the fraction of labeled examples in $\mathcal{S}_\ell$ that satisfy the predicate. The procedures to simulate the other statistical queries are similar. Note that the arguments μ and θ are omitted since these arguments are only used to determine the size of the labeled sample $\mathcal{S}_\ell$ in Theorem 3 and empirically we have fixed amounts of data.

algorithm developed for empirical use does *not* need to know k. Much more significantly, we obtained very good results without having to run the algorithm with $O(\frac{k}{\epsilon^2}\log\frac{1}{1-2\eta_b})$ different estimates for η (see Section 6). In our empirical studies, we estimated the statistical queries by just using a labeled (noisy) sample and computing the fraction of the examples for which the predicate χ was true (see figure 14). Below, we briefly provide an analytical basis for why this performs well. Furthermore, notice that since we are no longer trying to invert the noise process, we can reduce the harm caused when the noise model is not simply that of random classification noise.

We now analytically compute the difference between using the method provided by Aslam and Decatur (1998) for obtaining the estimates for the statistical queries and our method of just directly computing the fraction of examples for which the predicate is true. For ease of exposition we just focus on the procedure given in figure 15 for estimating the false negative error of $\mathcal{H}$ (where we use it to evaluate the hypothesis $\mathcal{H} \cap P_{\text{best}}$). Using the notation of figure 15, the estimate specified by Aslam and Decatur (1998), denoted by $\hat{P}_T$, should be

$$\frac{count1 - \hat{\eta} \cdot count2}{m_\ell(1-2\hat{\eta})}.$$

The estimate we used for our empirical studies (denoted by $\hat{P}_E$) was $count1/m_\ell$. Let $count1_{H_+}$ (respectively $count2_{H_+}$) be the value for $count1$ (respectively, $count2$) when $\mathcal{H} = \{H_+\}$.

We now compute the value we used for the cutoff when using $\hat{P}_T$ throughout. We intersect P_{best} with $\mathcal{H}$ when

$$\frac{count1 - \hat{\eta} \cdot count2}{m_\ell(1-2\hat{\eta})} \leq \frac{count1_{H_+} - \hat{\eta} \cdot count2_{H_+}}{m_\ell(1-2\hat{\eta})} + 0.01,$$

so we intersect P_{best} with $\mathcal{H}$ when

$$count1 \leq count1_{H_+} + 0.01 \cdot m_\ell + \hat{\eta}(count2 - count2_{H_+} - 0.02 \cdot m_\ell).$$

Similarly, when using $\hat{P}_E$ throughout, we intersect P_{best} with $\mathcal{H}$ when

$$\frac{count1}{m_\ell} \leq \frac{count1_{H_+}}{m_\ell} + 0.01.$$

$\mathbf{SQ}(\chi_{FalseNeg}(\mathcal{H}), \mu, \theta)$
  $count1 \leftarrow$ no. of exs $(X,\ell) \in S_\ell$ where $(\mathcal{H}(X) = 0) \wedge (\ell = 1)$
  $count2 \leftarrow$ no. of exs $(X,\ell) \in S_\ell$ where $\mathcal{H}(X) = 0$
  Return $(count1 - \hat{\eta} \cdot count2)/(|\mathcal{S}_\ell|(1-2\hat{\eta}))$

Figure 15. The procedure used to simulate statistical queries to be provably tolerant of classification noise (Aslam and Decatur (1998)). This procedure estimates the false negative error rate P_χ for $\chi = \chi_{\text{FalseNeg}}(\mathcal{H}) \doteq ((\mathcal{H}(X) = 0) \wedge (\ell = 1))$ with a noise rate estimate of $\hat{\eta}$ using $\mathcal{S}_\ell$ drawn from the noisy example oracle $\mathcal{D}_\eta$.

Solving for *count*1 here we obtain that using a direct estimate we intersect P_{best} with $\mathcal{H}$ when

$$count1 \leq count1_{H_+} + 0.01 \cdot m_\ell.$$

The difference between the two methods is an additive factor

$$\hat{\eta}(count2 - count2_{H_+} - 0.02 \cdot m_\ell)$$

for the cutoff based on the value for *count*1. This factor is partly balanced by the savings gained by not having to run the algorithm for many different noise rate estimates. Finally, note that for the estimates of Line 16 of *Learn-Pattern-Empirical*, since we are essentially just comparing two estimates to see which is best, there is even less of a cost for using the more straightforward estimates.

8. Experimental results

Our algorithm was run through two different sets of tests. The first set involved running the algorithm on randomly generated training and test patterns. These runs allowed us to estimate the algorithm's empirical data requirements. The second set of tests used real data gathered for Pinette's (1993) thesis. These runs revealed what kinds of data might be encountered by a robot in an office environment and suggested more work that is required to make our approach a viable solution to the landmark matching problem (see Section 9).

8.1. Simulated data

Three different experiments using simulated data were performed for our algorithm. We first compared our new noise-tolerant SQ algorithm (when $\eta = 0$) to the old noise-intolerant covering algorithm reviewed in Section 5. We generated three random targets of $k = 15$ real points, each on the real interval $[1, 100]$. We then built a test set of 1000 examples (each with $n = 30$ points) for each target. Each example had a 1/2 probability of being negative. Each negative example was made to be very similar to the positive examples by the following construction. With probability 1/2 the example was negative because some point was not within unit distance of any target point, and only one point was placed not near any target point. Also, with probability 1/2 the example was negative because some target point was not within unit distance of any example points, and only one target point was not near any points in that example. The training sets were made in the same fashion, each with m examples, where m varied from 20 to 1000 in increments of 20. For the old algorithm of Section 5, all m examples were properly labeled. For the new (SQ) algorithm, 1/3 of the examples were unlabeled (and used for $\mathcal{S}_u$) and 2/3 were properly labeled (and used for $\mathcal{S}_\ell$). For each training set, each algorithm generated a hypothesis that was evaluated with the corresponding test set to measure its accuracy. Eight runs were made per target and all these evaluations were averaged and then the results from the 3 targets were averaged. The results appear in figure 16. Surprisingly, the new noise-tolerant algorithm performed

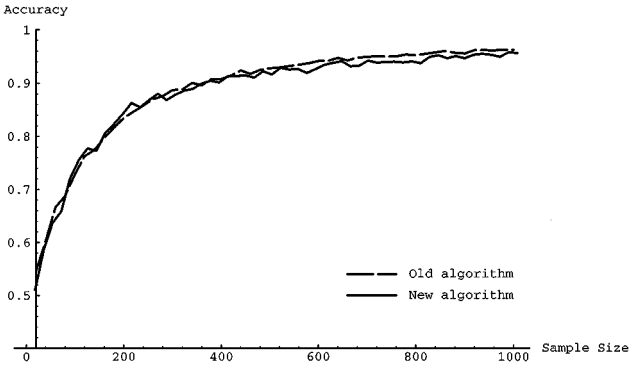

Figure 16. Our new algorithm's performance versus the old noise-intolerant one when $\eta = 0$.

nearly as well as the old algorithm which was designed specifically for noise-free data. This is despite the fact that the old algorithm's worst-case data requirements are significantly less. A possible explanation for this is that for this concept class, an unlabeled sample and statistics based on a properly labeled sample are empirically almost as useful for learning as a properly labeled sample with individual labels available.

For the runs of our new algorithm on training sets corrupted with classification noise, the test and training sets were made the same way as above, but each training example's label was flipped with probability η. The test sets were uncorrupted. The training sample size m varied from 150 to 2100 in increments of 150. Four runs were made for each of the three targets. The results appear in figure 17 and seem quite good, especially when compared to the amount of data specified by the worst-case lower bounds to achieve similar results. Specifically, with $\eta = 0.05$, over 90% accuracy is obtained with $m = 1000$ training examples. By contrast, for $\epsilon = 0.1$, $\delta = 0.5$, $\eta = 0.05$, $n = 30$ and $k = 15$, the worst-case sample size is $m > 4 \times 10^9$.

Next we studied the performance of our algorithm in a noise model much closer to the type of noise we expect to see in real data for the landmark matching problem. Intuitively,

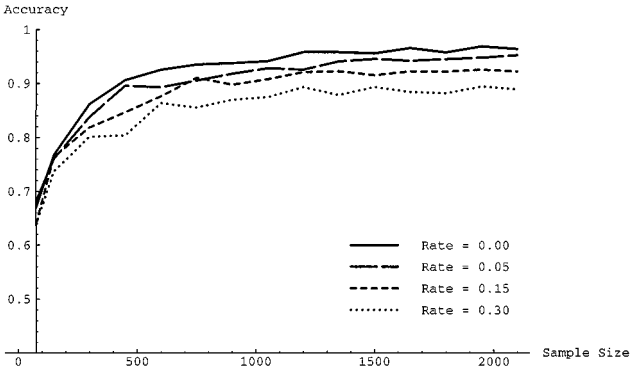

Figure 17. Performance of our algorithm for various classification noise rates.

in practice we expect to see examples that are corrupted in the sense that some points are shifted left or right, some spurious points are present, and some true points are absent. We attempted to model these phenomena with the following generalized noise process. We first created a set of examples, but did not add any classification noise to the set. Then for each example X, we created a set S_{add} of points to add to it, where $|S_{\text{add}}| + 1$ was geometrically distributed with parameter p_{add}, i.e. we flipped a biased coin (with probability of heads $= p_{\text{add}}$) until heads came up. Then we placed $T - 1$ points in S_{add}, where T is the total number of flips. Each point in S_{add} was created by uniformly at random selecting a point in X and "wiggling" it. The wiggling process consisted of finding d_1 (geometrically distributed with parameter p_{wig}), randomly choosing a direction to wiggle based on a fair coin flip, and then moving the point a distance $0.1d_1$ in the chosen direction. Then we created a set S_{del} of points to delete from X, where $|S_{\text{del}}| + 1$ was geometrically distributed with parameter p_{del}. Each point in S_{del} was selected uniformly at random from X. Once S_{add} and S_{del} were created, the new noisy example was $X' = \text{wiggle}(X \setminus S_{\text{del}}) \cup S_{\text{add}}$, where $\text{wiggle}(X)$ applies the aforementioned wiggling process to each point in X, except the distance moved is $0.1d_2$ where $d_2 + 1$ is geometrically distributed with parameter p_{wig}, allowing for no wiggle if the total number of coin flips to get heads is 1.

In the plots of figure 18, we selected $p_{\text{add}} = p_{\text{wig}} = p_{\text{del}} = 7/8$ (so $\text{E}[|S_{\text{add}}|] = \text{E}[|S_{\text{del}}|] = \text{E}[d_2] = 1/7$ and $\text{E}[d_1] = 8/7$) and $p_{\text{add}} = p_{\text{wig}} = p_{\text{del}} = 15/16$ (so $\text{E}[|S_{\text{add}}|] = \text{E}[|S_{\text{del}}|] = \text{E}[d_2] = 1/15$ and $\text{E}[d_1] = 16/15$). On average, in this noise process, probabilities of $7/8$ in some way corrupted over 98% of all examples and caused about 18% of the positive examples to become negative and about 1% of the negative examples to become positive. Probabilities of $15/16$ in some way corrupted over 86% of all examples and caused about 8.5% of the positive examples to become negative and about 0.6% of the negative examples to become positive. In the plots, m varied from 150 to 2100 in increments of 150 and from 2400 to 3000 in increments of 300. There were two runs made on each of three targets. As with the previous experiment, the test data were not corrupted. For comparison with performance on classification noise, we included the $\eta = 0.30$ curve from figure 17. In the over 200 runs of the algorithm on this generalized noise process, the H_+ cutoff tuning procedure described in Section 7 twice failed to find a good cutoff. Subsequently the final

Figure 18. Performance of our algorithm for the generalized noise process.

hypotheses of those runs misclassified all positive examples. These two data points were detected and omitted from figure 18.

The poorer performance of the algorithm on generalized noise is not surprising when one considers that the process is indirectly introducing a significant amount of classification noise and is adding distributional errors. These distributional errors affect both the labeled and unlabeled examples, so building of H_+ is much more difficult because the algorithm's view of the distribution of points is skewed.

8.2. Real data

Twelve experiments were performed in which we ran our algorithm on real data sets. These sets came from the "room" data set from Pinette's (1993) thesis. The data sets were one-dimensional images (signatures) taken in a room at the coordinates shown in figure 19. In this figure, adjacent points are separated by 1 foot. The signatures taken at these points were light intensities at 1° intervals, so each array was 360 pixels wide. We grouped some signatures together, as indicated by the dashed boxes. The circled point in each group is the center of that group. Each group's center location could be considered the target location when that group of patterns was used as the set of positive examples.

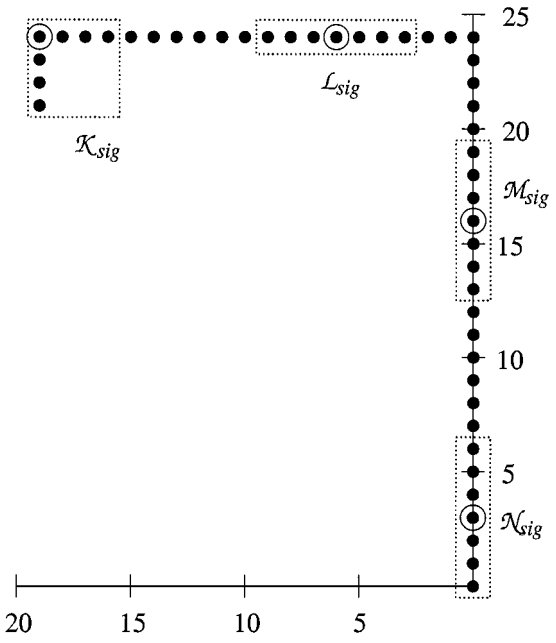

Figure 19. Coordinates where the one-dimensional images (signatures) of the "room" data set were shot. Units on the axes are in feet. Dotted boxes indicate groups of signatures that were given to the algorithm as training and testing data. The circled points highlight the center point of each data set; this point could be considered the target location when that set's patterns were used as positive examples.

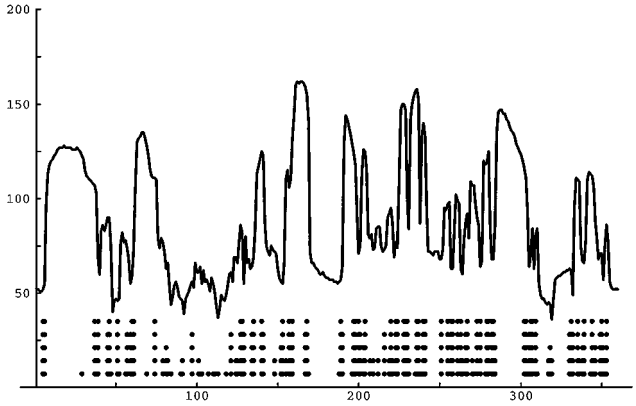

Figure 20. The signature taken at location (0, 0) and the results of spike detection. From the bottom pattern to the top one, the ν values are 0.05, 0.07, 0.09, 0.11 and 0.13.

The signatures from the points of figure 19 were mapped to one-dimensional patterns via a simple spike detector. This spike detector first scanned the entire array to determine its maximum and minimum values. Then the spike detector placed a point wherever the absolute value of the change in intensity exceeded some factor ν times the difference between the maximum and minimum. For every $\mathcal{Y} \in \{\mathcal{K}, \mathcal{L}, \mathcal{M}, \mathcal{N}\}$, we ran the spike detector on each signature in $\mathcal{Y}_{\text{sig}}$ using $\nu \in \{0.05, 0.07, 0.09, 0.11, 0.13\}$ and placed the resultant pattern into set $\mathcal{Y}_\nu$. Figure 20 is the signature taken at (0, 0) in figure 19 along with the patterns from the spike detector run on all values of ν. In the figure, $\nu = 0.13$ is the top pattern and $\nu = 0.05$ is the bottom pattern.

After mapping all signatures to patterns, we used the sets of patterns in twelve experiments. For each experiment we, for each value of ν, used one of $\mathcal{K}_\nu$, $\mathcal{L}_\nu$, $\mathcal{M}_\nu$ and $\mathcal{N}_\nu$ for the positive examples and two of the remaining sets for the negative examples. Due to a shortage of examples, the set of unlabeled examples was just the union of the positives and the negatives. Thus for each combination of patterns there were 7 positive examples, 14 negative examples and 21 unlabeled examples. For each run, we also used $\gamma \in \{15, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40\}$ as the maximum Hausdorff distance for which the learning algorithm considers two patterns to be visually similar (see Section 4).

The algorithm was evaluated using leave-one-out cross-validation, where one of the examples was withheld from the training set (withheld from both the positive or negative set and the unlabeled set) and was used to test the hypothesis created by *Learn-Pattern-Empirical*. The minimum error for each run is reported in Table 1 as well as which values of ν and γ yielded these minima. Since it's not realistic to report only the minima, Table 2 reports the errors for $\gamma = 30$ and $\nu = 0.09$. This point was chosen because it had one of the lowest overall error rates, and many runs had a minimum there. General plots of the results of the runs are in figures 21–28.

Overall, results were mixed. When $\mathcal{M}_\nu$ or $\mathcal{N}_\nu$ provided the positives, the error rates were low. However, some runs had high error rates, especially when $\mathcal{L}_\nu$ provided the positive examples and $\mathcal{K}_\nu$ provided some of the negative examples (and vice-versa). Additionally,

Table 1. Minimum values of the error for each run of the algorithm. The values of γ and ν that yielded these minima are given in the third column. The notation ... in some sets indicates that more such points exist. The "Min Count" column indicates how many points achieved the minimum.

Positive set	Negative sets	Min error	Min at (γ, ν)	Min count
$\mathcal{K}_\nu$	$\mathcal{L}_\nu, \mathcal{M}_\nu$	0.0952	{(24, 0.11), (24, 0.13), (26, 0.11), (26, 0.13), (34, 0.13), ...}	8
	$\mathcal{L}_\nu, \mathcal{N}_\nu$	0.0476	{(26, 0.11), (26, 0.13), (34, 0.09), (40, 0.13)}	4
	$\mathcal{M}_\nu, \mathcal{N}_\nu$	0.0476	{(32, 0.09), (34, 0.09)}	2
$\mathcal{L}_\nu$	$\mathcal{K}_\nu, \mathcal{M}_\nu$	0.0476	{(36, 0.05)}	1
	$\mathcal{K}_\nu, \mathcal{N}_\nu$	0.0476	{(40, 0.05)}	1
	$\mathcal{M}_\nu, \mathcal{N}_\nu$	0.0952	{(26, 0.05), (26, 0.07), (26, 0.09), (30, 0.05), (32, 0.05)}	5
$\mathcal{M}_\nu$	$\mathcal{K}_\nu, \mathcal{L}_\nu$	0.0000	{(28, 0.05)}	1
	$\mathcal{K}_\nu, \mathcal{N}_\nu$	0.0000	{(24, 0.09), (28, 0.05), (28, 0.09), (30, 0.09), (32, 0.09), ...}	9
	$\mathcal{L}_\nu, \mathcal{N}_\nu$	0.0000	{(22, 0.05), (22, 0.07), (24, 0.05), (24, 0.07), (24, 0.09), ...}	10
$\mathcal{N}_\nu$	$\mathcal{K}_\nu, \mathcal{L}_\nu$	0.0000	{(15, 0.07), (15, 0.09), (15, 0.13), (18, 0.07), (18, 0.09), ...}	14
	$\mathcal{K}_\nu, \mathcal{M}_\nu$	0.0000	{(15, 0.07), (15, 0.09), (15, 0.11), (15, 0.13), (18, 0.07), ...}	19
	$\mathcal{L}_\nu, \mathcal{M}_\nu$	0.0000	{(15, 0.07), (15, 0.09), (15, 0.11), (18, 0.07), (18, 0.09), ...}	27
Average		0.0317		

Table 2. Values of the error for each run of the algorithm for $\gamma = 30$ and $\nu = 0.09$.

Positive set	Negative sets	Error
$\mathcal{K}_{0.09}$	$\mathcal{L}_{0.09}, \mathcal{M}_{0.09}$	0.1428
	$\mathcal{L}_{0.09}, \mathcal{N}_{0.09}$	0.1429
	$\mathcal{M}_{0.09}, \mathcal{N}_{0.09}$	0.1428
$\mathcal{L}_{0.09}$	$\mathcal{K}_{0.09}, \mathcal{M}_{0.09}$	0.1905
	$\mathcal{K}_{0.09}, \mathcal{N}_{0.09}$	0.2381
	$\mathcal{M}_{0.09}, \mathcal{N}_{0.09}$	0.1905
$\mathcal{M}_{0.09}$	$\mathcal{K}_{0.09}, \mathcal{L}_{0.09}$	0.1429
	$\mathcal{K}_{0.09}, \mathcal{N}_{0.09}$	0.0000
	$\mathcal{L}_{0.09}, \mathcal{N}_{0.09}$	0.0000
$\mathcal{N}_{0.09}$	$\mathcal{K}_{0.09}, \mathcal{L}_{0.09}$	0.0476
	$\mathcal{K}_{0.09}, \mathcal{M}_{0.09}$	0.1429
	$\mathcal{L}_{0.09}, \mathcal{M}_{0.09}$	0.0000
Average		0.1151

most of these errors came from false negatives. We believe that this phenomenon was primarily caused by too few data points that were improperly grouped. Since we had no information concerning what was in the signatures in our data set, we grouped them so that the groups would be spread evenly throughout the room, not so that each group was near a good landmark.

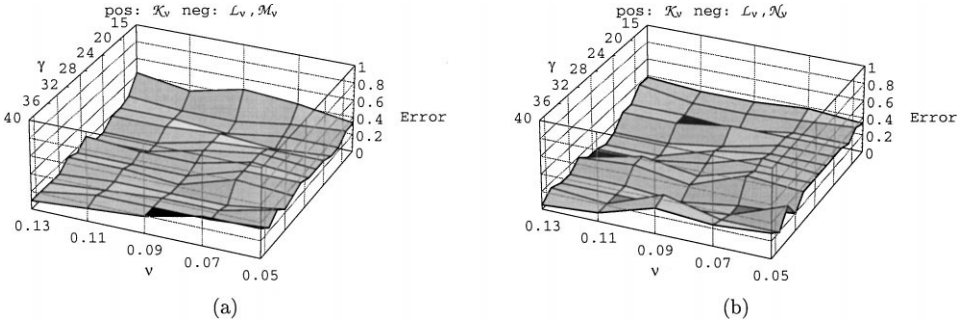

Figure 21. Performance of our algorithm when (a) the positive examples came from $\mathcal{K}_\nu$ and the negative examples came from $\mathcal{L}_\nu$ and $\mathcal{M}_\nu$; and (b) the positive examples came from $\mathcal{K}_\nu$ and the negative examples came from $\mathcal{L}_\nu$ and $\mathcal{N}_\nu$.

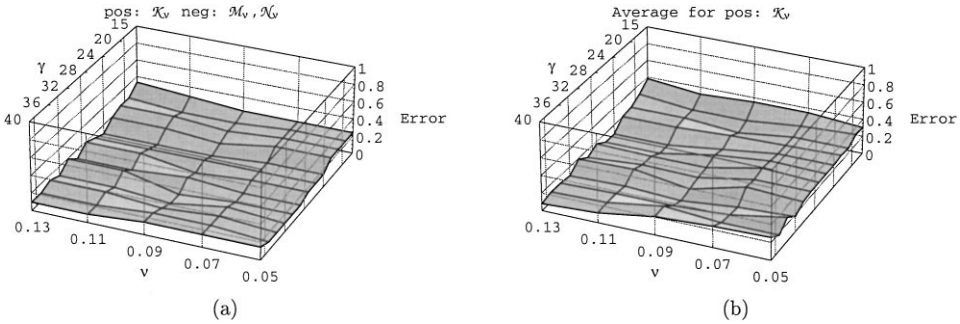

Figure 22. Performance of our algorithm when (a) the positive examples came from $\mathcal{K}_\nu$ and the negative examples came from $\mathcal{M}_\nu$ and $\mathcal{N}_\nu$; and (b) average error when the positive examples came from $\mathcal{K}_\nu$.

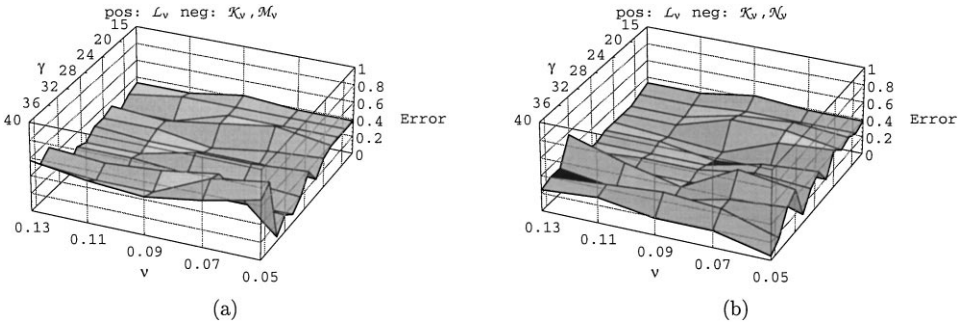

Figure 23. Performance of our algorithm when (a) the positive examples came from $\mathcal{L}_\nu$ and the negative examples came from $\mathcal{K}_\nu$ and $\mathcal{M}_\nu$; and (b) the positive examples came from $\mathcal{L}_\nu$ and the negative examples came from $\mathcal{K}_\nu$ and $\mathcal{N}_\nu$.

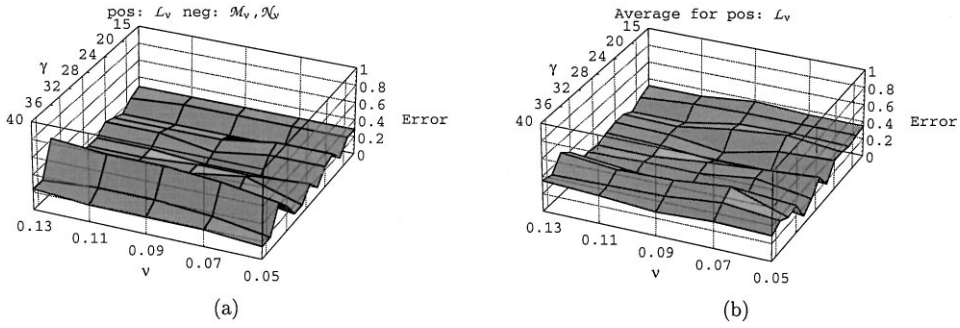

Figure 24. Performance of our algorithm when (a) the positive examples came from $\mathcal{L}_\nu$ and the negative examples came from $\mathcal{M}_\nu$ and $\mathcal{N}_\nu$; and (b) average error when the positive examples came from $\mathcal{L}_\nu$.

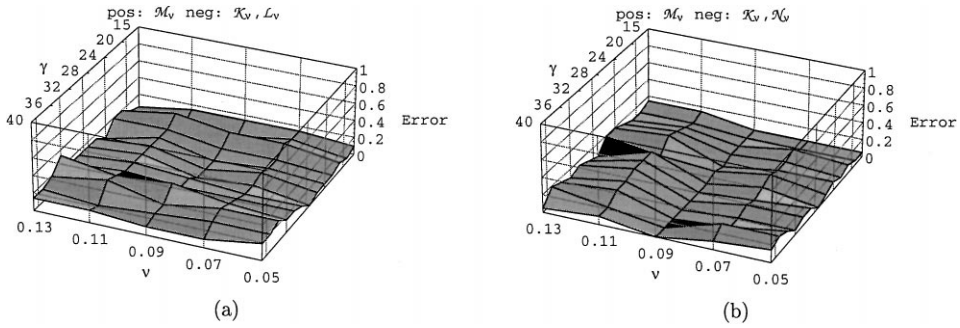

Figure 25. Performance of our algorithm when (a) the positive examples came from $\mathcal{M}_\nu$ and the negative examples came from $\mathcal{K}_\nu$ and $\mathcal{L}_\nu$; and (b) the positive examples came from $\mathcal{M}_\nu$ and the negative examples came from $\mathcal{K}_\nu$ and $\mathcal{N}_\nu$.

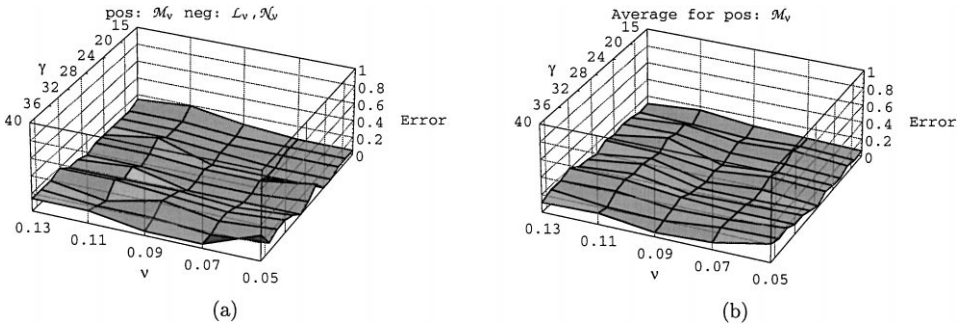

Figure 26. Performance of our algorithm when (a) the positive examples came from $\mathcal{M}_\nu$ and the negative examples came from $\mathcal{L}_\nu$ and $\mathcal{N}_\nu$; and (b) average error when the positive examples came from $\mathcal{M}_\nu$.

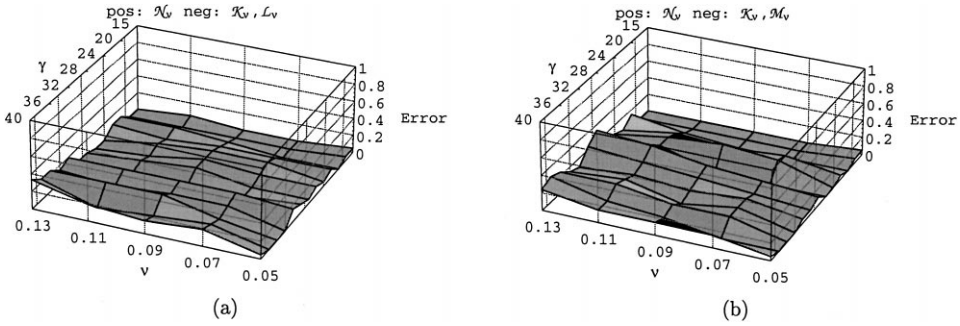

Figure 27. Performance of our algorithm when (a) the positive examples came from $\mathcal{N}_\nu$ and the negative examples came from $\mathcal{K}_\nu$ and $\mathcal{L}_\nu$; and (b) the positive examples came from $\mathcal{N}_\nu$ and the negative examples came from $\mathcal{K}_\nu$ and $\mathcal{M}_\nu$.

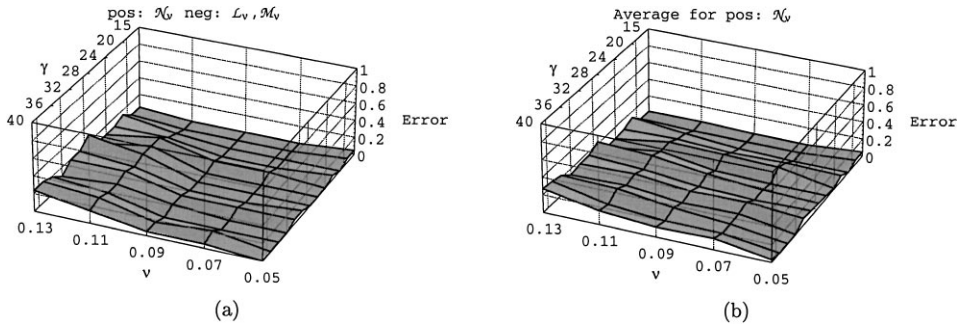

Figure 28. Performance of our algorithm when (a) the positive examples came from $\mathcal{N}_\nu$ and the negative examples came from $\mathcal{L}_\nu$ and $\mathcal{M}_\nu$; and (b) average error when the positive examples came from $\mathcal{N}_\nu$.

In fact, as Table 3 indicates, the Hausdorff distances between patterns in $\mathcal{L}_{0.09}$ and $\mathcal{L}_{0.09}$ are very similar to those between patterns in $\mathcal{K}_{0.09}$ and $\mathcal{L}_{0.09}$. On average, a pattern randomly selected from $\mathcal{K}_{0.09}$ is more similar to a random pattern from $\mathcal{L}_{0.09}$ than another random pattern from $\mathcal{L}_{0.09}$ is. (We also found that patterns in sets $\mathcal{M}_{0.09}$ and $\mathcal{N}_{0.09}$ are much more similar to patterns in the same set than to patterns in different sets, accounting for the improved performance when $\mathcal{M}_\nu$ and $\mathcal{N}_\nu$ supplied the positive examples.) Thus not only do some patterns in separate sets appear as similar as some in the same set, some patterns in the same set are as dissimilar as those in separate sets, even though the signatures in the same set are generally similar with each other and dissimilar to signatures in different sets. While we expect our algorithm to handle these discrepancies, we believe that the amount of available data (seven positives and fourteen negatives) is insufficient. The empirical work using real data also revealed a potential problem with the approach in this paper: information is lost when mapping from signatures to patterns, and there might not be enough data to compensate for this loss. We tried splitting the learning process into two phases: one using patterns from low to high transitions in the signatures and one using patterns from high to

Table 3. Minimum, maximum and average Hausdorff distances between patterns of Set 1 and patterns of Set 2 for $\nu = 0.9$. When Set 1 = Set 2, no comparisons are made between identical patterns since those Hausdorff distances are always zero.

Set 1	Set 2	Min	Max	Avg
$\mathcal{K}_{0.09}$	$\mathcal{K}_{0.09}$	10.0000	46.0000	23.9048
$\mathcal{K}_{0.09}$	$\mathcal{L}_{0.09}$	7.0000	60.0000	30.1224
$\mathcal{K}_{0.09}$	$\mathcal{M}_{0.09}$	10.0000	54.0000	33.5714
$\mathcal{K}_{0.09}$	$\mathcal{N}_{0.09}$	32.0000	58.0000	48.4286
$\mathcal{L}_{0.09}$	$\mathcal{L}_{0.09}$	7.0000	57.0000	32.0952
$\mathcal{L}_{0.09}$	$\mathcal{M}_{0.09}$	30.0000	53.0000	42.2449
$\mathcal{L}_{0.09}$	$\mathcal{N}_{0.09}$	34.0000	55.0000	48.5918
$\mathcal{M}_{0.09}$	$\mathcal{M}_{0.09}$	6.0000	22.0000	12.5714
$\mathcal{M}_{0.09}$	$\mathcal{N}_{0.09}$	10.0000	27.0000	19.0816
$\mathcal{N}_{0.09}$	$\mathcal{N}_{0.09}$	7.0000	16.0000	10.8095

low transitions in the signatures. However this process fared no better than our original procedure. But we do believe that additional data, especially positive examples, would help immensely, particularly if the points are evenly spaced throughout the circle rather than on a line through the circle's center.

9. Concluding remarks

As expected, the simulation performance was dramatically better (with respect to sample size and noise rate) than worst-case theoretical results specify. Indeed, we believe it to be extremely difficult, if not impossible, to specify a distribution that pushes this algorithm to worst-case performance in terms of sample size and time complexity.

Our preliminary results indicate that SQ algorithms may be a very good method for designing noise-tolerant algorithms that work well in practice. We had expected that for the noise-free case our statistical query algorithm would not perform as well as the covering algorithm designed to work only with noise-free data. However, we found that it worked nearly as well as the traditional covering algorithm.

As mentioned in Section 2, more experimental work is required to make our mapping and learning algorithm a viable solution to the landmark matching problem. One major avenue of future work is to test the algorithm's performance on larger real data sets, preferably ones that have points evenly spaced throughout circles centered at the target locations rather than only having points on a line through the circles' centers. Also, the positive examples should come from target locations that are good landmarks. Additionally, while we focused on the landmark matching problem, our algorithm should be applicable to any data representable as a one-dimensional array of values, e.g. sonar data, temporal difference information, or amplitudes of a waveform. An interesting problem is to investigate applications of this algorithm to other types of data.

After obtaining larger and different data sets, we can explore other methods to map the signatures to one-dimensional data as well as other procedures to set γ and selecting the cutoffs applied to the SQs when building H_+ and in the rest of the algorithm. After mapping the signatures to one-dimensional data, we should run a general procedure on the data to determine if the negative examples are sufficiently dissimilar from the positives for our algorithm to be effective. The procedure that produced the results of Table 3 is a step in that direction.

The next step would be to run more rigorous simulations, including measuring the variance of the error rates and determining confidence intervals. This would allow us to contrast our approach to others, including e.g. a simple instance-based algorithm to act as a baseline. This would also allow us to more thoroughly empirically evaluate the applicability of an SQ algorithm to this application domain, i.e. determine how well an SQ algorithm can handle noise models besides classification noise, malicious errors and distributional errors.

There is also the open work of exploring the use of metrics other than the Hausdorff metric, perhaps to incorporate more information into the patterns, e.g. direction of change and/or magnitude of change. One possibility is to learn the concept of superpositions of at most k probability density functions (pdfs), where each pdf gives the probability of finding a point at a particular location. Each example would be a superposition of at most n pdfs, and the metric could be the difference between the two superpositions integrated from $-\infty$ to ∞. This metric could be thresholded to yield a binary concept, or normalized to yield a p-concept (Kearns & Schapire, 1994), where the learner learns the *probability* that an example is positive. The p-concept model is useful in cases where an example's classification is influenced by a randomized process or information not visible to the learner. A probabilistic approach might be more amenable to a practical robot system since it is less sensitive to small perturbations in the training and testing data than our current system that uses thresholding.

Another possible approach to landmark matching is outlined in Goldman, Kwek, and Scott (1997). Their algorithm takes a discretized and bounded version of the class studied here, maps the examples' points to boolean attributes, and applies Winnow (Littlestone, 1988) to learn an arbitrary boolean function over the attributes. Their algorithm can learn geometric patterns of any constant dimension, so the one-dimensional signatures can be mapped to two-dimensional patterns, if desired. Additionally, their use of Winnow makes their algorithm *agnostic*, meaning that the target concept is *any* classifier and the algorithm attempts to find the best hypothesis from some fixed hypothesis space, making no assumptions whatsoever about the target. Winnow also affords their algorithm tolerance of classification and attribute noise. Finally, minor modifications of their algorithm grant it tolerance of *concept shift*, where the target concept changes over time. This can occur in landmark matching if the signature of a particular location changes due to objects moving, appearing and disappearing. While their algorithm has a sound theoretical foundation, empirical tests are required to contrast its performance with that of the algorithm presented here.

Appendix

This appendix provides the proof of Corollary 3 as well as two lemmas required to prove Lemma 2. We start with the proof of Corollary 3.

Corollary 3. *The VC dimension of the SQ query space $\mathcal{Q}$ is upperbounded by*

$$O\left(k^2 \log n \cdot \log\left(\frac{1}{\epsilon}\right) \cdot \left(\log k + \log \log \frac{1}{\epsilon}\right)\right).$$

Proof: Define $\mathcal{Q}_\chi$ to be the query space for query χ. Then $\text{VCD}(\mathcal{Q}_{\chi_{\text{FalsePos}}})$ and $\text{VCD}(\mathcal{Q}_{\chi_{\text{FixedFalsePos}}})$ are asymptotically equivalent to $\text{VCD}(\mathcal{C}_{\text{hyp}})$ because each of these query spaces can only shatter a set that can be shattered by $\mathcal{C}_{\text{hyp}}$, our hypothesis space.[7] Also, $\text{VCD}(\mathcal{Q}_{\chi_{\text{ExtraFalseNeg}}}) = O(k \log n)$ follows from Corollary 1 and Theorem 5 because $\mathcal{Q}_{\chi_{\text{ExtraFalseNeg}}}$ is no more expressive than the space of intersections of two patterns of at most $3k+1$ points each.

Finally, $\text{VCD}(\mathcal{Q}_{\chi_{\text{PosNotNear}}}) = O(\log n)$. We will show this by arguing that for $\mathcal{Q}_{\chi_{\text{PosNotNear}}}$ to shatter a set $\mathcal{V}$ of examples, the number of points per example is exponential in $|\mathcal{V}|$. Since the number of points per example is at most n, the logarithmic bound on $\text{VCD}(\mathcal{Q}_{\chi_{\text{PosNotNear}}})$ follows.

Recall that query $\chi_{\text{PosNotNear}}(q)$ is true iff the given example X is positive and q is not within unit distance of any point in X. Thus each point q uniquely defines an instance of $\mathcal{Q}_{\chi_{\text{PosNotNear}}}$. Therefore to shatter a set $\mathcal{V}$ of examples, the points in the examples of $\mathcal{V}$ must be placed such that for all i between 1 and $|\mathcal{V}|$, there exists some point q within unit distance of some point in exactly i of $\mathcal{V}$'s examples and not within unit distance of any point in $\mathcal{V}$'s remaining examples. This way the i examples will make $\chi_{\text{PosNotNear}}(q)$ false and the rest will make $\chi_{\text{PosNotNear}}(q)$ true. Such a q must exist for each of the $\binom{|\mathcal{V}|}{i}$ combinations of examples. We now lowerbound the number of points per example for this to be possible.

Consider the case when $\lfloor |\mathcal{V}|/2 \rfloor$ of the examples falsify the predicate and the remainder satisfy it. There are $\binom{|\mathcal{V}|}{|\mathcal{V}|/2}$ such combinations. The most efficient use of the points is to stagger them in a diagonal configuration as in figure 29. In this configuration, every contiguous group of $\lfloor |\mathcal{V}|/2 \rfloor$ points lies within some unit interval. If several "diagonals" (with the points placed in different examples) are abutted, each such diagonal can pick up at most $|\mathcal{V}|$ of the $\binom{|\mathcal{V}|}{|\mathcal{V}|/2}$ possible combinations. Since each point can lie in at most two diagonals, each diagonal costs each example at least 1/2 of a point. So the minimum number

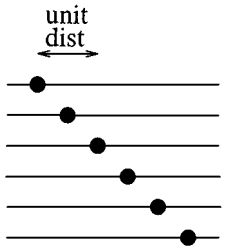

Figure 29. An example of the most efficient configuration of points in $\mathcal{V}$ for the case when $\lfloor |\mathcal{V}|/2 \rfloor$ of the examples falsify $\chi_{\text{PosNotNear}}(q)$ and the rest satisfy it. In this example, $|\mathcal{V}| = 6$, so each contiguous group of 3 points lies within some unit interval.

of points per example to shatter $\mathcal{V}$ is at least

$$\frac{\binom{|\mathcal{V}|}{|\mathcal{V}|/2}}{2|\mathcal{V}|}.$$

Applying Stirling's approximation gives a lower bound on the minimum number of points per example as

$$\Omega\left(\frac{|\mathcal{V}|^{|\mathcal{V}|}}{(|\mathcal{V}|/2)^{|\mathcal{V}|}}\sqrt{\frac{|\mathcal{V}|}{(|\mathcal{V}|/2)^2}}\frac{1}{|\mathcal{V}|}\right) = \Omega\left(\frac{|\mathcal{V}|^{|\mathcal{V}|}\cdot 2^{|\mathcal{V}|+1}}{|\mathcal{V}|^{|\mathcal{V}|+1}\sqrt{|\mathcal{V}|}}\right) = \Omega\left(\frac{2^{|\mathcal{V}|}}{|\mathcal{V}|^{3/2}}\right).$$

So the minimum number of points per example is at least $c_1 \cdot 2^{|\mathcal{V}|}/|\mathcal{V}|^{3/2}$ for some constant c_1 and sufficiently large $|\mathcal{V}|$. Since the number of points per example is at most n, it follows that $c_2 + |\mathcal{V}| - (3/2)\lg|\mathcal{V}| \leq \lg n$ for another constant c_2. Also, since $|\mathcal{V}| - (3/2)\lg|\mathcal{V}| \geq c_3(3/2)\lg|\mathcal{V}|$ for another constant c_3, we get $(3/2)\lg|\mathcal{V}| \leq (1/c_3)\lg n$. Combined, these inequalities imply

$$|\mathcal{V}| \leq \lg n + \frac{3}{2}\lg|\mathcal{V}| - c_2 \leq \left(1 + \frac{1}{c_3}\right)\lg n - c_2 = O(\log\ n).$$

Finally, we apply a standard result that says for classes $\mathcal{C}_1$ and $\mathcal{C}_2$, $\text{VCD}(\mathcal{C}_1 \cup \mathcal{C}_2) \leq \text{VCD}(\mathcal{C}_1) + \text{VCD}(\mathcal{C}_2) + 1$. Since $\mathcal{Q} = \mathcal{Q}_{\chi_{\text{ExtraFalseNeg}}} \cup \mathcal{Q}_{\chi_{\text{FalsePos}}} \cup \mathcal{Q}_{\chi_{\text{FixedFalsePos}}} \cup \mathcal{Q}_{\chi_{\text{PosNotNear}}}$, the result follows. □

Now we prove two lemmas needed by the proof of Lemma 2. We first show that if, during *Cover-Positives*, $q < p - 1$ then p must be from a negative example and can be removed from S'.

Lemma 5. *Let q, p and S' be as in the proof of Lemma 2. If, while sliding q to the left as in the proof of Lemma 2, we discover that $q < p - 1$, then p must be from a negative example and thus can be removed from S'.*

Proof: If in fact p is from a positive example, then it lies within unit distance of a point t in the target concept along with at least one point from every other positive example, implying that every other positive example has at least one point within distance 2 of p. If the distribution of examples is such that at least $1 - \epsilon/(80k)$ of the positive examples had their point(s) within unit distance of t lying to the right of p, then an H_+ point would have been placed at $p + 1$, covering p. The only other possibility is that at least $\epsilon/(80k)$ of the positive examples had their point(s) within unit distance of t lying to the left of p. In this case (figure 30), by Lemma 1 there must have been at least one positive example point in S to the left of p but within distance 2 of p (since this point was also within unit distance of t). Call the leftmost such point p'. A little thought shows that $p' + 1$ covers at least as much of t's interval as $p + 1$ does. Since S has a point in any interval of weight at least

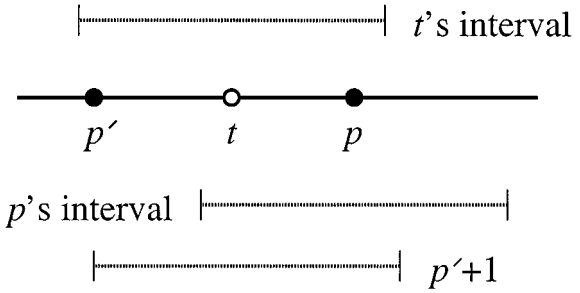

Figure 30. An example of how if p were from a positive example, *Cover-Positives* would have found an H_+ point covering it. The intervals indicated in the figure are of width 2 and centered at their respective points. In this figure at least $\epsilon/(80k)$ of the positive examples had their point(s) within unit distance of t lying to the left of p. Thus at least one positive example point from S lies to the left of p but within distance two of p. The leftmost such point is p'. Since $p'+1$ covers enough positive weight to be included in H_+ and also covers p, $p'+1$ is an H_+ point covering p if p were from a positive example.

$\epsilon/(80k)$ (by Lemma 1), the positive weight to p''s left is at most $\epsilon/(80k)$. Thus at least $1-\epsilon/(80k)$ of the positive weight within unit distance of t is to the right of p'. All this weight and p are within unit distance of $p'+1$, so $p'+1$ is an H_+ point covering p. So if no such $p'+1$ was found, then p must have come from a negative example. □

We next show that when *Cover-Positives* is finished, $|H_+| \leq 3k$ and that only two H_+ points per target interval add error, completing the proof of Lemma 2.

Lemma 6. *$|H_+| \leq 3k$ and only two H_+ points per target interval add error.*

Proof: Based on the construction of H_+, we know that for two H_+ points p_i^+ and p_j^+, $p_i^+ < p_j^+$ for all $i < j$, i.e. H_+ is built strictly from left to right. For target point t (figure 31), the first point p_1^+ placed in H_+ for t must be placed at some location at or to the right of $t-2$ (otherwise p_1^+ would not cover any points near t). Since p_1^+ covers at least one point within unit distance from t, the first place we try to place p_2^+ is at some point strictly to the right of t (recall that p_2^+ is unit distance to the right of the leftmost uncovered point). Since t covers all the weight of t's interval and S has a point in any interval with weight at least $\epsilon/(80k)$, p_2^+ will not slide left of t in *Cover-Positives*. Thus $p_2^+ \geq t$. Therefore p_1^+ and p_2^+

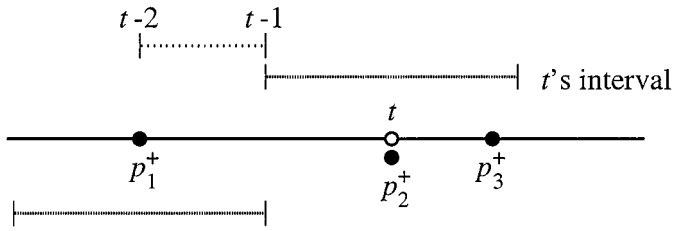

Figure 31. An example of how no more than three H_+ points can be added for any target point. The intervals indicated in the figure are of width 2 and centered at their respective points.

are all that are needed to cover all positive weight of t's interval. So the third point p_3^+ will only cover already covered positive weight and uncovered negative weight (lying outside t's interval). So it must be placed at some location that is strictly to the right of $p_2^+ \geq t$ and will cover some points in S outside of t's interval. The point p_3^+ will be just close enough to t to cover a sufficiently large amount of positive weight in t's interval, and no closer. So no point strictly to the right of p_3^+ can be far enough left to cover sufficient positive weight due to t. Thus at most 3 points are placed in H_+ for each target point.

Since all H_+ points must be within unit distance of positive weight from t, it follows that if three H_+ points p_1^+, p_2^+ and p_3^+ are placed for target point t, then $t - 2 \leq p_1^+ < p_2^+ < p_3^+ \leq t + 2$. So p_2^+ cannot cover any negative weight not already covered by p_1^+ and p_3^+. We can therefore charge the error introduced by these points to p_1^+ and p_3^+, supporting the claim that error is added by at most two H_+ points per target point. □

Acknowledgments

We thank David Mathias and Daniel Dooly for their comments on drafts of this paper. We also thank Brian Pinette for permission to use his thesis figures and for test data for our algorithm. Finally, we thank the ICML committee members and the *Machine Learning* referees for their comments.

Sally Goldman and Stephen Scott were supported in part by NSF National Young Investigator Grant CCR-9357707 with matching funds provided by Xerox PARC and WUTA. Stephen Scott performed this work at Washington University.

Notes

1. In this paper, k and n are upper bounds on the number of points per concept and example, i.e. they need not have exactly k and n points.
2. We assume that the portion of the complete navigation system that selects the landmarks will gather a set of images near the landmark to be used as the positive examples for training.
3. If the demand of polynomial-time computation below is replaced with expected polynomial-time computation, then the learning algorithm need not be given the parameter k, but could "guess" it instead (Haussler et al., 1991).
4. Note that throughout this paper, lg will be used for the base-2 logarithm and ln will be used for the natural logarithm. When the base of the logarithm is not significant (such as when using asymptotic notation), we use log.
5. Note that throughout this paper, the word "point" will refer to a single point on the real line, and we shall use the term "a configuration of points" when speaking of an instance.
6. When applying our learning algorithm to the landmark matching problem, γ can be different for each landmark.
7. Since $\mathcal{Q}_{\chi_{\text{FixedFalsePos}}}$ contains hypotheses with one extra pattern than allotted for in Corollary 2, its VC dimension is slightly higher, but still asymptotically equivalent.

References

Anthony, M., & Biggs, N. (1992). *Computational learning theory*. Cambridge University Press.

Aslam, A., & Decatur, S. (1993). General bounds on statistical query learning and PAC learning with noise via hypothesis boosting. *Proceedings of the 34th Annual Symposium on Foundations of Computer Science* (pp. 282–291). Los Alamitos, CA: IEEE Computer Society Press.

Aslam, A., & Decatur, S. (1998). Specification and simulation of statistical query algorithms for efficiency and noise tolerance. *Journal of Computer and System Sciences*, Earlier version in COLT '95, to appear.

Auer, P. (1997). On learning from multi-instance examples: Empirical evaluation of a theoretical approach. *Machine Learning: Proceedings of the Fourteenth International Conference* (pp. 21–29). San Francisco, CA: Morgan Kaufmann Publishers.

Auer, P., Long, P. M., & Srinivasan, A. (1997). Approximating hyper-rectangles: Learning and pseudo-random sets. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing* (pp. 314–323). New York, NY: ACM Press.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1987). Occam's razor. *Information Processing Letters*, *24*, 377–380.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, *36*(4), 929–965.

Decatur, S. (1993). Statistical queries and faulty PAC oracles. *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory* (pp. 262–268). New York, NY: ACM Press.

Ehrenfeucht, A., Haussler, D., Kearns, M., & Valiant, L. G. (1989). A general lower bound on the number of examples needed for learning. *Information and Computation*, *82*, 247–261.

Goldberg, P. (1992). *PAC-learning geometrical figures.* Ph.D. thesis, Department of Computer Science, University of Edinburgh.

Goldberg, P., & Goldman, S. (1994). Learning one-dimensional geometric patterns under one-sided random misclassification noise. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory* (pp. 246–255). New York, NY: ACM Press.

Goldberg, P., Goldman, S., & Scott, S. (1996). PAC-learning one-dimensional geometric patterns. *Machine Learning*, *25*(1), 51–70.

Goldberg, P., & Jerrum, M. (1995). Bounding the Vapnik-Chervonenkis dimension of concept classes parameterized by real numbers. *Machine Learning*, *18*, 131–148. Special issue for the *Sixth Annual ACM Conference on Computational Learning Theory*.

Goldman, S., Kwek, S., & Scott, S. (1997). Agnostic learning of geometric patterns. *Proceedings of the Tenth Annual Conference on Computational Learning Theory* (pp. 325–333). New York, NY: ACM Press.

Goldman, S., & Scott, S. (1996). A theoretical and empirical study of a noise-tolerant algorithm to learn geometric patterns. *Machine Learning: Proceedings of the Thirteenth International Conference* (pp. 191–199). San Francisco, CA: Morgan Kaufmann.

Gruber, P. M. (1983). Approximation of convex bodies. In P. M. Gruber & J. M. Willis (Eds.), *Convexity and its applications*. Brikhäuser Verlag.

Haussler, D., Kearns, M., Littlestone, N., & Warmuth, M. K. (1991). Equivalence of models for polynomial learnability. *Information and Computation*, *95*(2), 129–161.

Haussler, D., Littlestone, N., & Warmuth, M. K. (1994). Predicting $\{0, 1\}$ functions on randomly drawn points. *Information and Computation*, *115*(2), 284–293.

Hong, J., Tan, X., Pinette, B., Weiss, R., & Riseman, E. M. (1992). Image-based homing. *IEEE Control Systems Magazine*, *12*(1), 38–45.

Kearns, M. (1993). Efficient noise-tolerant learning from statistical queries. *Proceedings of the 25th Annual ACM Symposium on Theory of Computing* (pp. 392–401). New York, NY: ACM Press.

Kearns, M., & Schapire, R. (1994). Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, *48*(3), 464–497.

Kearns, M., & Vazirani, U. (1994). *An introduction to computational learning theory*. Cambridge, MA: MIT Press.

Levitt, T. S., & Lawton, D. T. (1990). Qualitative navigation for mobile robots. *Artificial Intelligence*, *44*(3), 305–360.

Littlestone, N. (1988). Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*, 285–318.

Natarajan, B. K. (1991). *Machine learning: A theoretical approach*. San Francisco, CA: Morgan Kaufmann Publishers.

Pinette, B. (1993). *Image-based navigation through large-scaled environments.* Ph.D. thesis, University of Massachusetts, Amherst.

Pitt, L., & Valiant, L. (1988). Computational limitations on learning from examples. *Journal of the ACM*, *35*, 965–984.

Pitt, L., & Warmuth, M. K. (1990). Prediction preserving reducibility. *Journal of Computer and System Sciences*, *41*(3), 430–467. Special issue of the *Third Annual Conference of Structure in Complexity Theory* (Washington, DC., June 1988).

Suzuki, H., & Arimoto, S. (1988). Visual control of autonomous mobile robot based on self-organizing model for pattern learning. *Journal of Robotic Systems*, *5*(5), 453–470.

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

Valiant, L. G. (1985). Learning disjunctions of conjunctions. *Proceedings of the 9th International Joint Conference on AI*, 560–566.

Valiant, L. (1991). A view of computational learning theory. *Computation & Cognition: Proceedings of the First NEC Research Symposium*, 32–51.

Vapnik, V. N., & Chervonenkis, A. Ya. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, *16*(2), 264–280.

Received April 2, 1997
Accepted October 21, 1998
Final manuscript October 20, 1998