Machine Learning, 25, 51–70 (1996)

PAC Learning of One-Dimensional Patterns

PAUL W. GOLDBERG goldbepw@aston.ac.uk
Dept. of Computer Science and Applied Mathematics, Aston University, Birmingham B4 7ET, U.K.

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

STEPHEN D. SCOTT sds@cs.wustl.edu
Dept. of Computer Science, Washington University, St. Louis, MO 63130

Editor: David Haussler

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 consider the problem of PAC-learning the concept class of geometric patterns where the target geometric pattern is a configuration of k points on the real line. Each instance is a configuration of 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. Informally, two geometric patterns P and Q resemble each other under the Hausdorff metric if every point on one pattern is "close" to some point on the other pattern. We relate the concept class of geometric patterns to the landmark matching problem and then present a polynomial-time algorithm that PAC-learns the class of one-dimensional geometric patterns. We also present some experimental results on how our algorithm performs.

Keywords: PAC learning, landmark matching, robot navigation

1. Introduction

Developing the ability to recognize a landmark from a visual image of a robot's current location is a fundamental problem in robotics. We consider the problem of PAC-learning the concept class of geometric patterns where the "target" geometric pattern is a configuration of k points on the real line. Each instance is a configuration of 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 on one pattern is "close" to some point on the other pattern.

Consider a robot designed to navigate through a large-scaled environment[1]. An important component of a complete autonomous navigation system is an algorithm to recognize a landmark from a visual image of a robot's current location. Suppose that another component of the navigation system has selected a set of key "landmarks" of which the robot has prior knowledge. It is crucial that the robot be able to recognize whether or not it is in the vicinity of a given landmark from a visual image taken from the robot's current location. We shall refer to this problem as the *landmark matching problem*. In his doctoral thesis, Pinette (1993) states that "any general navigation algorithm must be able to match landmarks by their appearance." Namely, when performing navigation a robot plans a path by moving

between known landmarks, tracking landmarks as it goes. Because of inaccuracies in effectors and errors in the robot's internal map, when the robot believes it has reached landmark L, before heading to the next landmark it should check that it is really in the vicinity of L. Then adjustments can be made if the robot is not at L by either traveling towards L and/or updating its map.

We can apply our algorithm to learn geometric patterns to the landmark matching problem by converting the robot's visual image into a one-dimensional geometric pattern. Then this algorithm can be used to predict whether or not the robot is near (or not near) the given landmark. The main result of this paper is a polynomial-time algorithm that PAC-learns the class of one-dimensional geometric patterns. In addition to theoretical results we also provide the learning curves obtained by running our algorithm on simulated data. In addition to providing results on the empirical performance of our algorithm as a function of sample size, these simulations show that the sample complexity requirements are likely to be far less than those of the worst-case theoretical bounds.

An interesting feature of this problem is that the target concept is specified by a k-tuple 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 dual to a mistake-bound) depends polynomially on k and $\log n$.

This paper is organized as follows. In the next section we describe how algorithms for the problem we address could be applied to the landmark matching problem described above. Then in Section 3 we review the PAC learning model and some techniques from learning theory that we apply. In Section 4 we formally define the concept class of one-dimensional geometric patterns. Our main result appears in Section 5, where we describe our algorithm to PAC-learn the class of one-dimensional geometric patterns. Section 6 presents our simulation results. Finally, we conclude in Section 7.

2. The Landmark Matching Problem

In this section we explore, in further depth, how this work relates 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 and Lawton (1990), Pinette (1993), Suzuki and Arimoto (1988)). We now briefly describe one such imaging system (Hong et al. (1992) and Pinnette (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. 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

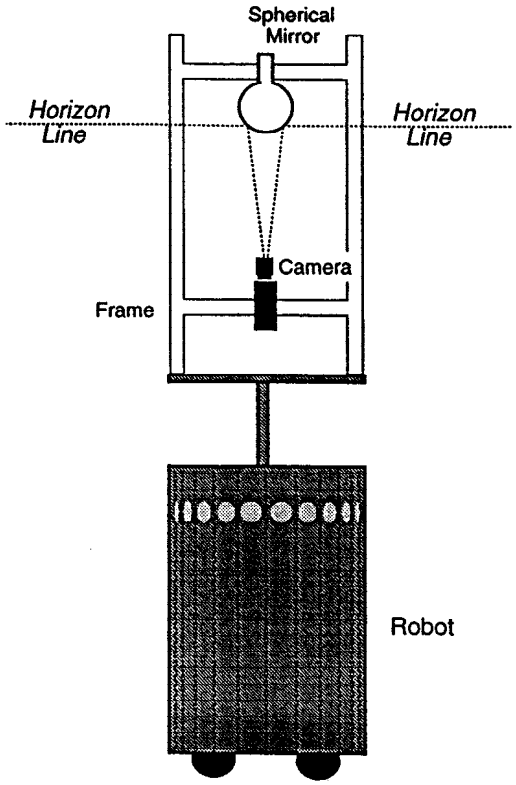

Figure 1. The imaging system on the robot. (This figure comes directly from Pinnette's (1993) thesis.)

(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.

Most work on designing landmark matching algorithms uses a pattern matching approach 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 easily 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 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

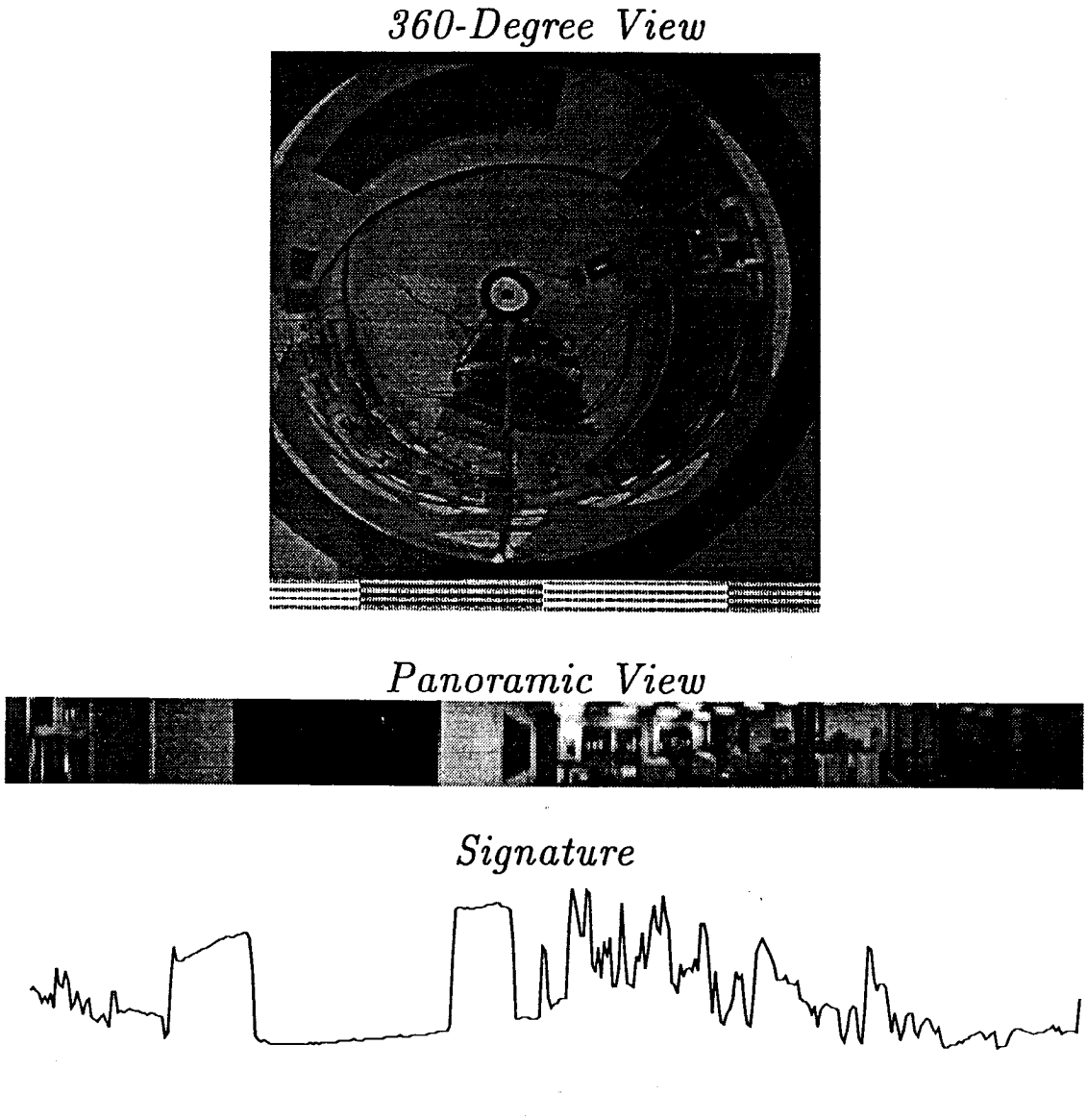

Figure 2. Stages of image processing. (This figure comes directly from Pinnette's (1993) thesis.)

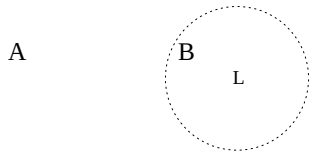

Figure 3. An example to help illustrate a problem that must be overcome. Suppose A and B are the locations of the bases of an arch and L is a landmark location. The visual images, obtained within the dashed circle, change dramatically.

a learning algorithm to construct a good hypothesis for performing landmark matching. Intuitively, the learning algorithm is being used to "average" 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, 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 can construct a hypothesis that can accurately predict whether or not the robot is near the given landmark.

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. (To keep the example simple, suppose that nothing besides the arch was 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 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 a learning algorithm to "combine" all of these images taken near L to obtain a hypothesis that will accurately 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, while the work here assumes noise-free data, 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. We note that by converting the algorithm presented here to one in the statistical query model (Kearns (1993) and Aslam and Decatur (1995)), noise tolerance can be achieved (Goldman and Scott (1996)). Thus by using noise-robust learning algorithms, our learning-based approach can handle noisy data.

3. Background

In this paper we work within the PAC (probably approximately correct) model of computational learning, as introduced by Valiant (1984, 85). 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 $\mathcal{C}$ is decomposed into subclasses $\mathcal{C}_n$ 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.

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 some 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 A exists (that is, an algorithm A meeting the goal 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 any particular instance of a learning problem, the number of variables n is given. For simplicity, we henceforth drop the subscript "n", and write $\mathcal{C}$ and $\mathcal{X}$ instead of $\mathcal{C}_n$ and $\mathcal{X}_n$, noting that all algorithms run in time polynomial in n.

Note, 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 deterministic 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.

The more general form of learning that we use here is commonly called *prediction.* The goal is to find any polynomial-time algorithm that classifies instances accurately in the PAC sense. Thus the algorithm need not define a set that corresponds to some concept in $\mathcal{C}$. This idea of prediction in the PAC model originated in the paper of Haussler, Littlestone and Warmuth (1988), and is discussed in Pitt and Warmuth (1990).

3.2. The VC-dimension and Occam Algorithms

The paper of Blumer et al. (1989) identifies a combinatorial parameter of a class of hypotheses called the *Vapnik-Chervonenkis (VC) dimension*, which originated in the paper of Vapnik and Chervonenkis (1971), that gives 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. Blumer et al. (1989) The VC dimension of concept class $\mathcal{C}$ (which we denote VCD($\mathcal{C}$)) is the size of a largest set $S \subseteq \mathcal{X}$ such that any subset of S is of the form $S \cap C$ for some $C \in \mathcal{C}$, or ∞ if such sets can be arbitrarily large. When it is true that any subset of S is of the form $S \cap C$ for some $C \in \mathcal{C}$, it is said that $\mathcal{C}$ *shatters* S.

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 VCD($\mathcal{C}$) ≥ 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 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. Namely, they showed that any concept $C \in \mathcal{C}$ consistent with a sample of size[4] $\max\left(\frac{4}{\epsilon}\lg\frac{2}{\delta}, \frac{8\,\mathrm{VCD}(\mathcal{C})}{\epsilon}\lg\frac{13}{\epsilon}\right)$ 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 $\Omega\left(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{\mathrm{VCD}(\mathcal{C})}{\epsilon}\right)$ examples in the worst case.

One drawback with the above approach is that the hypothesis must be drawn from $\mathcal{C}$. However, for the problem we study the computational problem of finding such a hypothesis from the class is NP-complete. In fact, the size of the hypothesis output by our algorithm depends on the size of the sample. In particular, the representation complexity of a hypothesis is sublinear in the sample size and polynomial in the parameters n and k. Blumer at al. (1987, 89) show that this achievement of data compression is sufficient to guarantee polynomial learnability. Let $\mathcal{F}^A_{k,n,m}$ be the hypothesis space used by algorithm A for an example complexity of n, a target complexity of k and sample size m. More formally, we say that algorithm A is an *Occam Algorithm* for concept class $\mathcal{C}$ if there exists a polynomial $p(k,n)$ and a constant α, $0 \leq \alpha < 1$, such that for any sample S with $|S| = m \geq 1$ and any k, n, A outputs a hypothesis $\mathcal{H}$ consistent with S such that $size(\mathcal{H}) \leq p(k,n)m^\alpha$.

THEOREM 1 Blumer et al. (1989) *Let A be an Occam algorithm for concept class $\mathcal{C}$ that has hypothesis space $\mathcal{F}^A_{k,n,m}$. If the VC dimension of $\mathcal{F}^A_{k,n,m}$ is at most $p(k,n)(\lg m)^\ell$ for some polynomial $p(k,n) \geq 2$ and $\ell \geq 1$, then A is a PAC-learning algorithm for $\mathcal{C}$ using sample size*

$$m = \max\left(\frac{4}{\epsilon}\lg\frac{2}{\delta}, \frac{2^{\ell+4}p(k,n)}{\epsilon}\left(\lg\frac{8(2\ell+2)^{\ell+1}p(k,n)}{\epsilon}\right)^{\ell+1}\right).$$

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 n points on the real line[5]. A concept is the set of all configurations from $\mathcal{X}_n$ within unit distance[6] under the Hausdorff metric of some "ideal" configuration of k points. The Hausdorff distance between configurations P and Q, denoted $H(P,Q)$, is:

$$\max\left\{\sup_{p\in P}\left\{\inf_{q\in Q}\{dist(p,q)\}\right\}, \sup_{q\in Q}\left\{\inf_{p\in P}\{dist(p,q)\}\right\}\right\}$$

where $dist(p,q)$ is the Euclidean distance between points p and q.

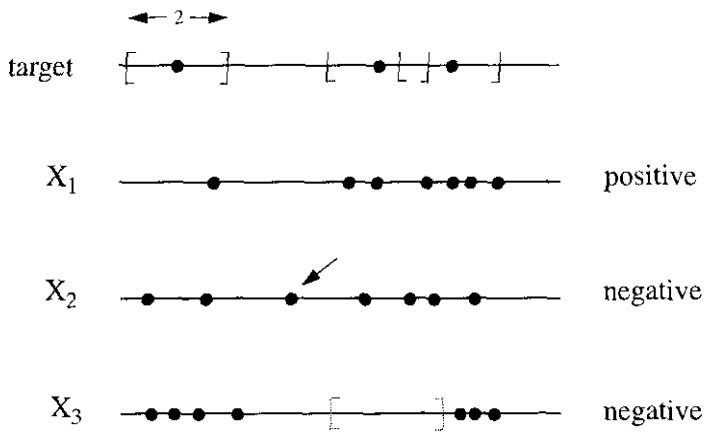

Figure 4. This figure illustrates an example concept from $C_{3,7}$. The top line shows the target pattern. Around each target point we show an interval that covers all points within unit distance from that point. Every positive example must have every point within one of the above intervals *and* no 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 near some target point (e.g. X_3).

of C_P. Furthermore, all configurations of points that resemble the given configuration P are contained within this sphere. Finally, the concept class $C_{k,n}$ that we study is defined as follows: $C_{k,n} = \{C_P \mid P$ 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 $C_{k,n}$ 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 the union of intervals, each instance is a single point on the real line, whereas for $C_{k,n}$ 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. Furthermore, observe that for $C_{k,n}$ each instance (configuration of n points) is an element of a metric space, which has a measure of distance defined between any pair of instances. However, with the class of unions of intervals there is no notion of a distance between

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. Furthermore, observe that for $\mathcal{C}_{k,n}$ each instance (configuration of n points) is an element of a metric space, which has a measure of distance defined between any pair of instances. However, with the class of unions of intervals there is no notion of a distance between instances. Finally, 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 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 S. Thus, assuming NP $\neq$ RP, 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 PAC-Learning Algorithm

Our algorithm is motivated by the fact that while it is NP-complete to find a sphere in this metric space consistent with given sets of positive and negative examples, it is possible in polynomial time to find one that is consistent with all positive and at least a fraction $\frac{1}{2(k+1)}$ of the negative examples, where k is the target concept complexity, the number of points in the configuration defining the target concept. Hence we may build a hypothesis consisting of an intersection of concepts obtained by a greedy set cover algorithm on the negative examples.

We now present our algorithm for learning $\mathcal{C}_{k,n}$. Our algorithm is an Occam algorithm. Define $\mathcal{H}_k$ to be the intersection of at most $2(k+1)\lg m$ concepts from $\mathcal{C}_{k,n}$, where m is the sample size required. Then the algorithm draws a sufficiently large sample of size m (polynomial in $k, \lg n, 1/\epsilon$, and $\lg 1/\delta$) and then outputs a consistent hypothesis from $\mathcal{H}_{k+1}$.

In order to apply Theorem 1 we need to upperbound the VC dimension of $\mathcal{H}_{k+1}$. To achieve this goal we make use of recent results of Goldberg and Jerrum (1995), which identify 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.

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.

The difference between the unit cost model and discretized geometrical problems is significant. Blumer et al. (1989) show that Euclidean n-spheres can be learned in polynomial time under the logarithmic cost model (in which the cost of a real value is the number of bits it occupies). The problem of finding a consistent hypothesis in this class is equivalent to linear programming, and the complexity of this is a major open problem in the unit cost model of real arithmetic. For a discussion of this see Renegar (1992). The NP-completeness results noted above for our learning problems hold also in the discretized case.

We use the following theorem of Goldberg and Jerrum (1995):

THEOREM 2 Goldberg and 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 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 VCD$(\mathcal{C}_{k,n}) \leq 2k \ln(8ekn) \leq 2k \lg(8ekn)$.*

This follows from the fact that the Hausdorff distance between a set of k points on the line and a set of n points on the line depends on a set of kn 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 3 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 $\leq 2\mathrm{VCD}(\mathcal{C}) s \lg(3s)$.*

Combining Corollary 1 with Theorem 3, we get the following result.

COROLLARY 2 *The VC dimension of $\mathcal{H}_{k+1}$ is upperbounded by*

$$\begin{aligned}\mathrm{VCD}(\mathcal{H}_{k+1}) &\leq 8(k+1)^2 \lg(8e(k+1)n) \lg m \lg(6(k+1)\lg m) \\ &\leq 24\sqrt{6}(k+1)^{5/2} \lg(8e(k+1)n)(\lg m)^{3/2}.\end{aligned}$$

Proof: To obtain the first inequality we apply Theorem 3 with $s = 2(k+1)\lg m$ and $\mathrm{VCD}(\mathcal{C}) = 2(k+1)\lg(8e(k+1)n)$. We then get the second inequality by using the inequality $\lg x < 3\sqrt{x}$ for $x > 1$ (which in turn comes from[7] $\ln x < c(x^{1/c} - 1)$ for $c \geq 1$). □

We are now ready to present the main result of this paper:

THEOREM 4 *Let $\mathcal{C} = \cup_{k,n\in N}\mathcal{C}_{k,n}$ be the class of spheres under the Hausdorff metric, whose domain is configurations of up to n points on the real line, and concepts defined by configurations of up to k points on the real line. Then $\mathcal{C}_{k,n}$ is predictable from positive and negative examples with a sample complexity of*

$$m = O\left(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{k^{5/2}\log(kn)}{\epsilon}\log^{5/2}\left(\frac{k\log(kn)}{\epsilon}\right)\right),$$

and time complexity of $O(kmn\log m + mn\log(mn))$.

Proof: To build the hypothesis we use a greedy set cover 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 $\varphi = \frac{1}{2(k+1)}$ of the negative examples. Then the negative examples accounted for are removed and the procedure is repeatedly applied until all negative examples have been eliminated. The intersection of all concepts obtained by doing this is consistent with the sample, and assuming that enough negative examples are removed at each stage, it is an Occam algorithm.

Let r denote the number of rounds until all negative examples have been covered. Then since $\frac{1}{1-\varphi} = 1 + \frac{\varphi}{1-\varphi} \geq 1+\varphi$, it is easily seen that

$$r \leq \log_{\frac{1}{1-\varphi}} m \leq \log_{1+\varphi} m = (\log_{1+\varphi} 2)(\lg m).$$

We next apply the inequality $(1+\varphi)^x \geq 1+\varphi x$ with $x = 1/\varphi$. This gives that $(1+\varphi)^{1/\varphi} \geq 2$, and thus $\log_{1+\varphi} 2 \leq 1/\varphi$. Applying this to the above upperbound for r shows that

$$r \leq (\log_{1+\varphi} 2)(\lg m) \leq \frac{1}{\varphi}\lg m = 2(k+1)\lg m.$$

Finally, the hypothesis output is the intersection of the r concepts obtained in this manner.

Thus by applying Theorem 1 with $p(k,n) = 24\sqrt{6}(k+1)^{5/2}\lg(8e(k+1)n)$ and $\ell = 3/2$ we get that any hypothesis from $\mathcal{H}_{k+1,n,m}$ that is consistent with a sample of size

$$\begin{aligned} m = &\max\left(\frac{4}{\epsilon}\lg\frac{2}{\delta}, \frac{1536\sqrt{3}(k+1)^{5/2}\lg(8e(k+1)n)}{\epsilon}\lg^{5/2}\left(\frac{4800\sqrt{30}(k+1)^{5/2}\lg(8e(k+1)n)}{\epsilon}\right)\right) \\ = &\ O\left(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{k^{5/2}\log(kn)}{\epsilon}\log^{5/2}\left(\frac{k\log n}{\epsilon}\right)\right) \end{aligned} \quad (1)$$

will have error at most ϵ with probability at least $1-\delta$.

What remains is to prove that in polynomial time we can find a concept $\mathcal{H}$ from $\mathcal{C}_{k+1,n}$ that is consistent with all positive examples and at least a fraction $1/(2k+1)$ of the uncovered negative examples. Recall that there are two ways for an example to be negative: either there is a point in the example that is not near[8] any target point (e.g. X_2 in Figure 1), or no points in the example are near some target point (e.g. X_3 in Figure 1). Let $\mathcal{N}$ be the set of negative examples that remain at the start of a round. By a simple averaging argument it follows that one of the following holds.

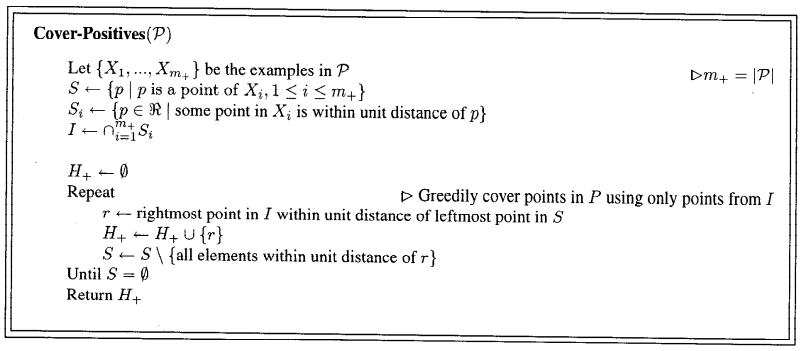

Figure 5. Algorithm to preprocess the examples to find a pattern that is consistent with all positive examples. Note that any pattern consistent with the points in P must be a subset of I.

Case 1: At least $|\mathcal{N}|/2$ of the negative examples have no points near some target point. Thus, by an averaging argument, there is some width 2 interval I_1 containing at least one point from each of the positive examples that does not contain points in at least $\frac{|\mathcal{N}|}{2k}$ of the negative examples.

Case 2: At least $|\mathcal{N}|/2$ of the negative examples have a point that is not near a target point. Since the portions of the real line that are not near any target point form at most $k+1$ contiguous intervals, by an averaging argument, there is some interval I_2 containing points from at least $\frac{|\mathcal{N}|}{2(k+1)}$ distinct negative examples and no points from the positive examples.

Our complete algorithm is given in Figures 5 and 6.

The algorithm **Cover-Positives** greedily covers the points from the positive examples. It first computes the set S_i of points that are unit distance from some point in positive example X_i. Then the points used for the greedy covering are selected from $I = S_1 \cap \cdots \cap S_{m_+}$. This ensures that every positive example has some point within unit distance of a point in H_+. Then a greedy covering is used to be sure that H_+ has a point within unit distance of a point from every positive example. Thus the final H_+ returned will be consistent with all positive examples.

We now argue that H_+ is a pattern of at most k points that is consistent with all examples in $\mathcal{P}$. First note that k points defining the target concept must be in I since each example in $\mathcal{P}$ must have a point near each target point. Then by a simple inductive argument, it can be shown that for all i, the ith leftmost points in H_+ cover all points from the examples of $\mathcal{P}$ that are within unit distance of the first i points of the target concept.

Next our algorithm **Learn-1d-Pattern** finds either an interval I_1 (correpsonding to Case 1) or an interval I_2 (corresponding to Case 2) that when intersected with $\mathcal{H}$ will cover at least $|\mathcal{N}|/(2(k+1))$ of the misclassified negative examples. The procedure **Find-I1** takes as input the set of positive examples $\mathcal{P}$ and uncovered negative examples $\mathcal{N}$ and searches

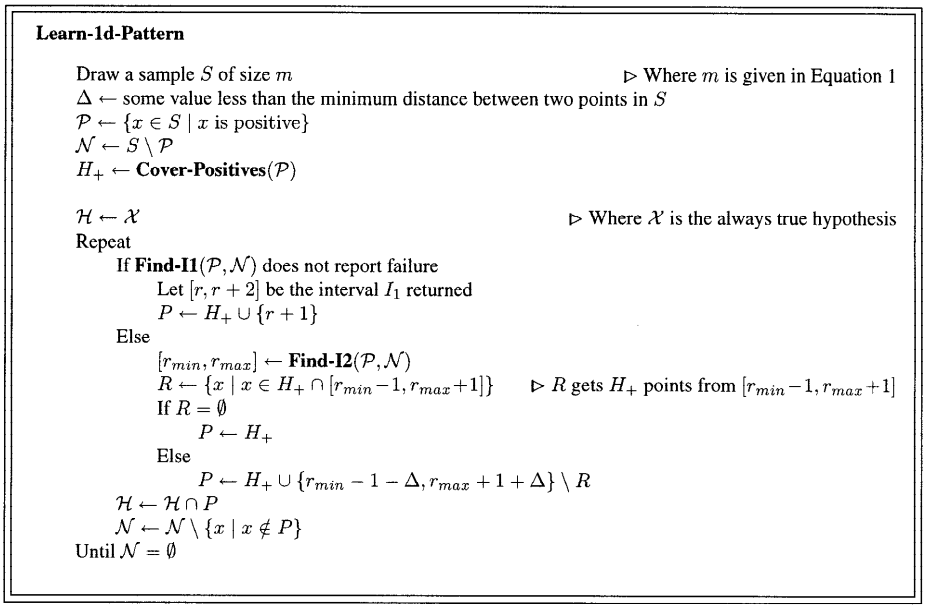

Figure 6. Algorithm to PAC learn a one-dimensional pattern. First a pattern H_+ of k points is greedily formed that is guaranteed to correctly classify all examples in $\mathcal{P}$. Next a greedy set covering technique is used to find a set of patterns (each consistent with all examples in $\mathcal{P}$) that when intersected with H_+ will correctly classify all negative examples.

for an interval I_1 of width 2 that contains at least one point from each positive example and does not contain any point in at least $|\mathcal{N}|/(2k)$ of the negative examples. This interval can be found by placing all mn points on one line and sliding a width 2 window over them while updating records of which examples are represented in the current window. It is easily seen that this can be done in $O(mn)$ time. The procedure returns the first interval that satisfies the condition (if one exists), or otherwise returns failure.

Our algorithm first calls **Find-I1**. By the above argument we know that if **Find-I1** returns failure, then Case 2 must apply. In this situation, our algorithm then uses procedure **Find-I2** which takes as input $\mathcal{P}$ and $\mathcal{N}$ and returns an interval I_2 that contains points from at least $\frac{|\mathcal{N}|}{2(k+1)}$ distinct negative examples and no points from the positive examples. As with **Find-I1**, **Find-I2** runs in $O(mn)$ time.

We have already argued that either **Find-I1** or **Find-I2** will succeed. If **Find-I1** succeeds then since the pattern P added to $\mathcal{H}$ has at most one point added to H_+, $P \in \mathcal{C}_{k+1,n}$ as desired. Also it is easily seen that P is consistent with the $\frac{|\mathcal{N}|}{2k}$ negative examples that have no point in the returned interval I_1.

In **Find-I2**, $R = \{x \mid x \in H_+ \cap [r_{min} - 1, r_{max} + 1]\}$ is the set of points of H_+ that cover points in I_2, i.e. the points that must be removed from P to make it consistent with the negative examples with points in I_2. If **Find-I2** succeeds and $R = \emptyset$ then certainly $P \in \mathcal{C}_{k+1,n}$ and the required number of negative examples are covered. Finally, note that by the definition of I, all points in R must be within unit distance of either r_{min} or r_{max}. Thus the selected $P \in \mathcal{C}_{k+1,n}$ will be consistent with the $\frac{|\mathcal{N}|}{2(k+1)}$ negative examples that have a point in $[r_{min}, r_{max}]$.

Thus our final hypothesis $\mathcal{H}$ consists of an intersection of at most $2(k+1)\lg m$ concepts from $\mathcal{C}_{k+1,n}$. The correctness thus follows from that fact that m was selected to satisfy Theorem 1. For the stated time complexity we first assume that the mn points from the sample are sorted. The rest of the preprocessing takes $O(mn)$ time. The main loop in **Learn-1d-Pattern** is executed $O(k \log m)$ times, with each execution taking $O(mn)$ time. Thus the stated time complexity follows. □

6. Simulation Results

To empirically evaluate the algorithm, we simulated it on uniformly distributed random test data. For each pair of values of k and n, we generated 3 random targets of k real points each on the real interval $[1, 100]$. Then a test set of 500 examples was randomly generated for each target. For each target we created 10 training sets, each with m examples, where m varied from 20 to 1000 in increments of 20. Each training example had a 0.5 probability of being negative, and if negative, it had a 0.5 probability of being Case 1. All negative examples were made as deceptive as possible by only allowing one point not near a target point (if Case 1) or only allowing one target point not near any example points (if Case 2). For each training set, the algorithm generated a hypothesis which was evaluated with the corresponding test set to measure its accuracy. All 10 evaluations per target were averaged and then the results from the 3 targets were averaged. Figure 7 describes our simulation procedure.

The curves in Figures 8 and 9 are connected points separated by increments of 20, where each point is the average of $10 \cdot 3 = 30$ hypothesis generations and evaluations. No attempt was made to fit a smooth curve to the points, so sudden changes in slope merely indicate minor variations in accuracy. Figure 8 indicates how the accuracy of the hypothesis relates to the number of examples in the training set for different values of k and n. Even for values as high as $k = 20$ and $n = 40$, the hypothesis generated by the algorithm was almost 95% correct when trained on 1000 examples. By contrast, substituting $\epsilon = 0.95$, $\delta = 0.5$, $k = 20$ and $n = 40$ into Equation 1 yields a worst-case sample complexity of $m = 3.806 \times 10^{11}$, more than 8 orders of magnitude beyond what our empirical results suggest.

Figure 9 exhibits the relationship between the hypothesis accuracy and the number of examples in the training set as k varies but n remains constant at 50. In this case, varying k did not greatly impact accuracy. For $k = 10$, the hypothesis generated by the algorithm was almost 96% correct when trained on 1000 examples. For $k = 30$, the hypothesis generated by the algorithm was almost 93% correct when trained on 1000 examples. By contrast,

Simulation-Procedure

For each (k, n) pair:
Generate 3 random targets C_i, $i = 1, 2, 3$ of k points each on $[1, 100]$.
Make a test set T_i, $i = 1, 2, 3$ of 500 examples for each target.
For $20 \leq m \leq 1000$ in increments of 20:
For each target C_i:
For $1 \leq j \leq 10$ in increments of 1:
Generate training set S_j with m random examples consistent with C_i:
Flip a fair coin.
If heads:
Generate a positive example by first randomly placing a point in each target interval. The other $n - k$ points are placed by randomly selecting a target interval and then randomly placing the point in the selected interval.
If tails:
Flip another fair coin.
If heads:
Generate an example that is negative because it does not have a point near some target interval. Do this by first randomly picking the target interval. Then randomly place the n points near the remaining $k - 1$ target points (like above where first one of the $k - 1$ intervals is picked and then the point is randomly placed in the selected interval).
If tails:
Generate an example that is negative because it has a point that is not near any target point. In this, one point is randomly placed among all parts of the line not near any target interval and the remaining $n - 1$ points are randomly placed within the k target intervals.
Run **Learn-1d-Pattern** to create a hypothesis $\mathcal{H}_j$ consistent with S_j.
Evaluate T_i with $\mathcal{H}_j$ and calculate the accuracy = the proportion of the examples consistent with C_i's evaluation of T_i.
Average the accuracies for target C_i and this value of m.
Average the averages of the 3 targets and insert this (k, n) curve into the final plot.

Figure 7. Summary of the simulation procedure.

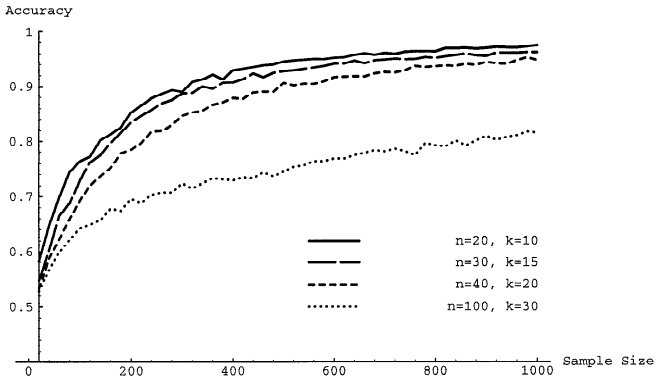

Figure 8. Hypothesis accuracy versus sample size for different values of n and k.

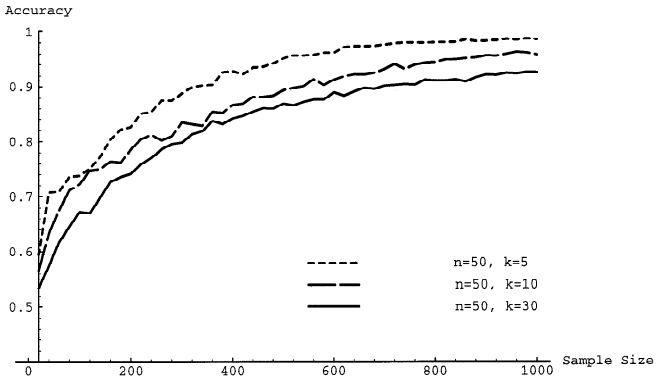

Figure 9. Hypothesis accuracy versus sample size for varying values of k.

substituting $\epsilon = 0.9$, $\delta = 0.5$, $k = 30$ and $n = 50$ into Equation 1 yields a worst-case sample complexity of $m = 1.286 \times 10^{12}$, more than 9 orders of magnitude beyond what our empirical results suggest. This disparity is due to the worst-case assumptions used throughout the analysis.

7. Concluding Remarks

In this paper we have presented an algorithm to efficiently PAC learn the class of one-dimensional geometric patterns. As discussed in Section 2, we feel that this algorithm can provide a novel way in which to perform landmark matching.

One interesting direction of further research is to consider the problem of learning two-dimensional geometric patterns. Goldberg (1992) has shown that while there is a considerable loss of efficiency in extending this technique from the one-dimensional to the two-dimensional case, the resulting algorithm still runs in polynomial time. This loss is caused by the increased search space of regions which define appropriate local features which perform the function of eliminating a significant fraction of the negative examples. It can be shown that a suitable set of regions to search over is the set of regions bounded by up to two vertical lines passing through points occurring in the examples, and two unit circles centered at points occurring in the examples. Can improved algorithms for higher dimensions be obtained?

Another very important research direction, that must be addressed to apply this work to real-world problems, is extending our algorithm so that it still performs well when there is noise in the data. Goldberg and Goldman (1994) have presented an algorithm that works when there is one-sided random classification noise (where only the labels are wrong). However, it would be nice to extend this work to handle general random classification noise and even small amounts of other types of noise. (See Goldman and Scott (1996) for recent work in achieving this goal.)

Finally, after we have extended our algorithm to tolerate the types of noise that we expect to see in real-world examples, we intend to obtain performance curves using real-world data rather than simulated data.

Acknowledgments

We thank Brian Pinette for allowing us to include his figures in our paper. We also thank Stephen Judd and Tom Hancock for several very useful discussion about the material in Section 3. Finally, we thank the COLT committee members for their comments.

Paul Goldberg carried out this research while visiting Washington University, and working at Sandia National Laboratories, supported by the U.S. Department of Energy under contract DE-AC04-76AL85000. Sally Goldman and Stephen Scott are supported in part by NSF National Young Investigator Grant CCR-9357707 with matching funds provided by Xerox Corporation, Palo Alto Research Center and WUTA.

Notes

1. By a large-scaled environment we mean that not all landmarks are visible from all locations in the environment.
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. 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. All results presented here apply if unit distance is replaced by some fixed distance since we can just rescale.
7. The exponent of $\lg m$ can be reduced arbitrarily close to 1 by just increasing the value of c.
8. For ease of exposition, we say that an example point within unit distance from a given target point is *near* that target point.

References

Anthony, M., & Biggs, N. (1992). *Computational Learning Theory: an Introduction*. Cambridge University Press.

Aslam, A., & Decatur, S. (1995). Specification and simulation of statistical query algorithms for efficiency and noise tolerance. *Proceedings of the Eighth Annual ACM Conference on Computational Learning Theory* (pp. 437–446). 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 Association for Computing Machinery*, 36(4), 929–965.

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., & 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. & 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*.

Goldberg, P. (1992). *PAC-Learning Geometrical Figures.* PhD thesis, Department of Computer Science, University of Edinburgh.

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 and P. M. Willis, editors, *Convexity and its applications*. Brikhauser 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. (1988). Predicting $\{0, 1\}$ functions on randomly drawn points. *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science* (pp. 100–109).

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 Annuual ACM Symposium on Theory of Computing* (pp. 392–401). New York, NY: ACM Press.

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 Kaufman Publishers.

Pinette, B. (1993). *Image-Based Navigation Through Large-Scaled Environments*. PhD thesis, University of Massachusetts, Amherst.

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

Pitt, L. & Warmuth, M. K. (1990). Prediction preserving reducibility. *J. of Comput. Syst. Sci.*, 41(3), 430–467. Special issue of the for the *Third Annual Conference of Structure in Complexity Theory* (Washington, DC., June 1988).

Renegar, J. (1992). On the Computational Complexity and Geometry of the First-Order Theory of the Reals. Part 1 (of 3). *Journal of Symbolic Computation*, 13, 255–299.

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. *Procs of the 9th International Joint Conference on AI* (pp. 560–566).

Valiant, L. G. (1991). A View of Computational Learning Theory. *NEC Research Symposium: Computation and Cognition* (ed. C.W. Gear), SIAM, Philadelphia.

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 20, 1995
Accepted April 20, 1995
Final Manuscript June 11, 1996