, , 1–15 ()

PAC Learning Axis-aligned Rectangles with Respect to Product Distributions from Multiple-instance Examples

PHILIP M. LONG plong@iscs.nus.sg
ISCS Department, National University of Singapore, Singapore 119260, Republic of Singapore

LEI TAN raymondt@microsoft.com
One Microsoft Way, Redmond, WA 98052

Editor: Dana Ron

Abstract. We describe a polynomial-time algorithm for learning axis-aligned rectangles in $\mathbf{Q}^d$ with respect to product distributions from multiple-instance examples in the PAC model. Here, each example consists of n elements of $\mathbf{Q}^d$ together with a label indicating whether any of the n points is in the rectangle to be learned. We assume that there is an unknown product distribution D over $\mathbf{Q}^d$ such that all instances are independently drawn according to D. The accuracy of a hypothesis is measured by the probability that it would incorrectly predict whether one of n more points drawn from D was in the rectangle to be learned. Our algorithm achieves accuracy ϵ with probability $1-\delta$ in

$$O\left(\frac{d^5 n^{12}}{\epsilon^{20}} \log^2 \frac{nd}{\epsilon\delta}\right)$$

time.

Keywords: PAC learning, multiple-instance examples, axis-aligned hyperrectangles

1. Introduction

Dietterich, Lathrop and Lozano-Perez [4] recently introduced the notion of learning from multiple-instance examples, where, rather than learning a function $f : X \to \{0,1\}$ from examples $(x, f(x))$ of its behavior, the learning algorithm instead receives examples of the form $((x_1, ..., x_n), f(x_1) \vee ... \vee f(x_n))$.

They were primarily motivated by a learning problem arising from drug discovery. In particular, they wanted to find algorithms to learn to predict whether a molecule would bind to a particular site. A molecule might have a number of stable shapes. For practical purposes, one can regard a molecule as binding to a site if any of its shapes binds to the site. Further, experiments typically yield information only about whether any of a molecule's shapes binds to a site. For this problem, Dietterich et al argued that high-dimensional axis-aligned rectangles were good models for determining whether a particular shape would bind to a site. They presented experimental studies of different learning algorithms using both natural and synthetic data.

In this paper, we consider the problem of learning axis-aligned rectangles from multiple-instance examples in Valiant's PAC model [12]. We assume that each of

the instances is drawn independently at random according to a fixed, unknown, product distribution D on $\mathbf{Q}^d$. The accuracy of a hypothesis is measured by the probability that it would incorrectly predict whether one of n additional instances drawn according to D would be in the rectangle to be learned.

In this paper, we describe an algorithm which learns to ϵ accuracy with probability at least $1 - \delta$ in time

$$O\left(\frac{d^5 n^{12}}{\epsilon^{20}} \log^2 \frac{nd}{\epsilon\delta}\right).$$

The algorithm of this paper (let's call it A_{mult}) is substantially different from those studied by Dietterich, et al. It works by running d copies of an algorithm A_{pcon} for learning p-concepts [7, 13] to learn the conditional probability that any of the instances of a multiple-instance example will be in the rectangle to be learned, given each of the components of the first instance. Generating p-concepts in this way is reminiscent of the "hidden variable" problems studied by Kearns and Schapire [7].

The p-concepts that arise are piecewise constant, with a constant conditional probability in an interval, and lower conditional probability outside this interval (see Figure 1 for a representative graph). Using the tools of Kearns and Schapire [7], it is fairly easy to see that these p-concepts are of a simple enough form that they can be efficiently learned. The difficulty arises from enforcing the fact that one can reconstruct a good hypothesis for the hidden rectangle from the hypotheses returned by A_{pcon}. What A_{mult} needs is to be able to use each p-concept hypothesis to accurately guess the endpoints of the associated interval. First, loosely speaking, if the two conditional probabilities are too close together, then a constant p-concept can be an accurate enough hypothesis, while yielding no information about the location of the endpoints. Thus, for the sake of A_{mult} we must argue that the two conditional probabilities are sufficiently far apart. This is not necessarily the case. However, if we exclude degenerate cases where the probability of a multiple instance example being labelled 1 is either very large or very small (A_{mult} can easily dispose of these cases before calling the copies of A_{pcon}), then we can ensure that the gap is big enough. Still, if for some p-concept subproblem, the probability that the associated component of the first instance falls in the interval is small enough, then A_{pcon} can still get an accurate enough hypothesis with a constant p-concept by just being inaccurate on the interval. However, we can also argue, if A_{mult} disposes of the degenerate cases, the probability of the interval of each of the components to be learned by copies of A_{pcon} is large enough. The same problem arises concerning the probability of the complement of the interval. Here, this probability can apparently be exponentially small in the parameters of the problem faced by A_{mult}. However, the conditional probability of the p-concepts on this portion of their domains is the same for all of the p-concepts that copies of A_{pcon} are trying to learn. Furthermore, this conditional probability is a simple function of the probability that a multiple-instance example from the original problem is labelled 1 (i.e., that one of the instances is in the rectangle to be learned). Therefore, by estimating this probability, A_{mult} can obtain estimates for use by all the subroutine

p-concept algorithms. With a couple of additional ideas, we are able to enforce the fact that accurate p-concepts yield accurate estimates of the associated endpoints.

The problem studied in this paper can be viewed as that of PAC learning a subclass of the set of unions of n axis-aligned rectangles with respect to product distributions on $\mathbf{Q}^{dn}$. Blumer, Ehrenfeucht, Haussler and Warmuth [3] showed how to learn unions of s rectangles in $\mathbf{Q}^d$ in time polynomial in s for fixed d; Long and Warmuth [9] described an algorithm taking time polynomial in d for fixed s.

One apparent limitation of our result is that the number n of instances is the same for all examples the learner sees, where, for example, in the drug design application, different molecules assume different numbers of shapes. However, it is easy to see how to replace the assumption that each example contains n instances with the assumption that each example contains at most n instances by running n copies of a fixed-instance-size algorithm, then outputting the hypothesis of the copy that received the most examples. This increases the resources required by roughly a factor of n.

Since they first appeared [8], the results described here have been improved upon. Auer, Long and Srinivasan [1] described an algorithm that does not require that the distribution generating the instances is a product distribution, and that runs in

$$O\left(\frac{d^3n^2}{\epsilon^2}\log\frac{dn\log(1/\delta)}{\epsilon}\log\frac{d}{\delta}\right)$$

time. They still assumed that the instances were independent, but they showed that learning rectangles from multiple-instance examples generated according to an arbitrary distribution on $(\mathbf{Q}^d)^n$ is as hard as learning DNF from single-instance examples, and that a polynomial-time learning algorithm for the dependent-instance case which outputs a rectangle as its hypothesis only exists if $\mathcal{NP} = \mathcal{RP}$.

Blum and Kalai [2] showed that, if the instances are assumed to be independent, learning from multiple-instance examples reduces to single-instance learning with independent misclassification noise, i.e. where the classification in each example is "flipped" with a certain probability. Kearns [6] had introduced the notion of a statistical query, and showed that any class that can be learned from statistical queries can be learned with independent misclassification noise. He then demonstrated that typical learning algorithms can easily be modified to use statistical queries. A wide variety of concept classes have been shown to have polynomial-time algorithms that learn from statistical queries. Blum and Kalai's results imply that each such class is efficiently learnable in the independent-multiple-instance model. Blum and Kalai also established a stronger reduction (to learning with "one-sided" independent misclassification noise), allowing them to show that some classes which are not known to have efficient statistical query learning algorithms are nevertheless learnable from multiple-instance examples. The algorithm of this paper and that of [1] used only the first instance of each multiple-instance example for estimating probabilities; Blum and Kalai found an elegant way to use all n instances, resulting in an improvement in time and sample complexity over the results of [1] by roughly a factor of n.

Instead of measuring the accuracy of the algorithm's hypothesis by the probability that it misclassifies another n instances as to whether any of them is in the target rectangle, one might instead want to measure the accuracy with the probability that a single instance would be misclassified. However, if the probability according to D of the region outside the rectangle to be learned is ϵ, then the probability that a given multiple-instance example is labelled 0 is ϵ^n. Using known techniques [3, 5], this implies that learning from polynomially many examples when the accuracy of the hypothesis is measured with respect to the distribution on individual instances is impossible.[1] Nevertheless, for the drug discovery application, the classification of a collection of n instances corresponds to the binding property of a given molecule, and therefore the probability of correct n-instance classification seems the most relevant measure of the accuracy of a learner's hypothesis.

2. Definitions and main result

Denote the rationals by $\mathbf{Q}$, the positive integers by $\mathbf{N}$, and the real numbers by $\mathbf{R}$.

The following is based on Valiant's PAC model [12].

A *multiple-instance learning algorithm* takes as input $\epsilon, \delta > 0$, and a finite sequence of elements of $(\mathbf{Q}^d)^n \times \{0,1\}$ (n-instance examples), where $d, n \in \mathbf{N}$, and outputs a hypothesis $h : \mathbf{Q}^d \to \{0,1\}$. For a multiple instance learning algorithm A, we will refer to the associated mapping from inputs to outputs also by A.

For each $d, n \in \mathbf{N}, \vec{u}_1, ..., \vec{u}_n \in \mathbf{Q}^d$, and $f : \mathbf{Q}^d \to \{0,1\}$, define

$$\mathrm{OR}_f(\vec{u}_1, ..., \vec{u}_n) = f(\vec{u}_1) \vee ... \vee f(\vec{u}_n).$$

For

$$\sigma = ((\vec{x}_{1,1}, ..., \vec{x}_{1,n}), ..., (\vec{x}_{m,1}, ..., \vec{x}_{m,n})) \in ((\mathbf{Q}^d)^n)^m$$

define

$$\mathrm{sam}_f(\sigma) = (((\vec{x}_{1,1}, ..., \vec{x}_{1,n}), \mathrm{OR}_f(\vec{x}_{1,1}, ..., \vec{x}_{1,n})), \\ ..., ((\vec{x}_{m,1}, ..., \vec{x}_{m,n}), \mathrm{OR}_f(\vec{x}_{m,1}, ..., \vec{x}_{m,n}))).$$

A probability distribution over $\mathbf{Q}^d$ for which the components of a random point are mutually independent is called a *product distribution.*

For $d, n \in \mathbf{N}$, if F is a set of functions from $\mathbf{Q}^d$ to $\{0,1\}$, we say a multiple-instance learning algorithm A (ϵ, δ)-learns F with respect to product distributions from m random n-examples if for all product distributions D over $\mathbf{Q}^d$, for all $f \in F$

$$(D^n)^m\{\sigma : D^n\{\vec{u} : \mathrm{OR}_{A(\epsilon,\delta,\mathrm{sam}_f(\sigma))}(\vec{u}) \neq \mathrm{OR}_f(\vec{u})\} \geq \epsilon\} \leq \delta.$$

Here D^n denotes the probability distribution obtained by sampling n times independently according to D, and $(D^n)^m$ is defined similarly.

We use the unit cost RAM model of computation.

Finally, for each $d \in \mathbf{N}$, for each $\vec{a}, \vec{b} \in \mathbf{Q}^d$, let $r_{\vec{a},\vec{b}} = \prod_{i=1}^d [a_i, b_i]$. Define

$$\mathrm{BOXES}_d = \{r_{\vec{a},\vec{b}} : \vec{a}, \vec{b} \in \mathbf{Q}^d\}.$$

The following is the main result of the paper.

THEOREM 1 *There is a multiple-instance learning algorithm A such that, for all $\epsilon, \delta > 0, d, n \in \mathbf{N}$, A (ϵ, δ)-learns* BOXES_d *with respect to product distributions from*

$$O\left(\frac{d^2 n^6}{\epsilon^{10}} \log \frac{nd}{\epsilon\delta}\right)$$

n-examples in

$$O\left(\frac{d^5 n^{12}}{\epsilon^{20}} \log^2 \frac{nd}{\epsilon\delta}\right)$$

time.

The theorem will be proved by reducing the problem to an associated p-concept problem (Section 3), and then solving this p-concept problem (Section 4). In particular, putting together Lemmas 4 and 8 and simplifying will yield Theorem 1.

3. Reducing to a p-concept problem

In this section, we show how the problem of learning rectangles with respect to product distributions from multiple-instance examples reduces to a p-concept learning problem. We will first need to establish some definitions.

3.1. Definitions

This is the subset of Kearns and Schapire's [7] p-concept model that we need for this paper. (A similar model was independently introduced by Yamanishi [13].)

For each distribution D over $\mathbf{Q}$ and each $f : \mathbf{Q} \to [0,1]$, define the probability distribution $P_{D,f}$ over $\mathbf{Q} \times \{0,1\}$ to be the distribution obtained by first choosing the first component $x \in \mathbf{Q}$ according to D and then choosing the second component so that the probability that it is 1 is $f(x)$.

A *p-concept learning algorithm* A takes as input a finite sequence of elements of $\mathbf{Q} \times \{0,1\}$, and outputs a hypothesis $h : \mathbf{Q} \to [0,1]$. For $\epsilon, \gamma, \delta > 0$, we say that A $(\epsilon, \gamma, \delta)$-learns a set F functions from $\mathbf{Q}$ to $[0,1]$ from m examples if and only if for all $f \in F$, for all distributions D over $\mathbf{Q}$,

$$(P_{D,f})^m\{\sigma : D\{x : |(A(\sigma))(x) - f(x)| > \gamma\} > \epsilon\} \leq \delta.$$

If the three parameters are small, this says that it is highly likely that A's hypothesis is very accurate over almost all of the domain. If, in addition, A only outputs hypotheses in F, following Pitt and Valiant [10], we say that A *properly* $(\epsilon, \gamma, \delta)$-learns F.

Finally, for each $a, b \in \mathbf{Q}, \alpha, \beta \in [0,1]$, define

$$\rho_{a,b,\alpha,\beta}(x) = \begin{cases} \beta & \text{if } x \in [a,b] \\ \alpha & \text{otherwise.} \end{cases}$$

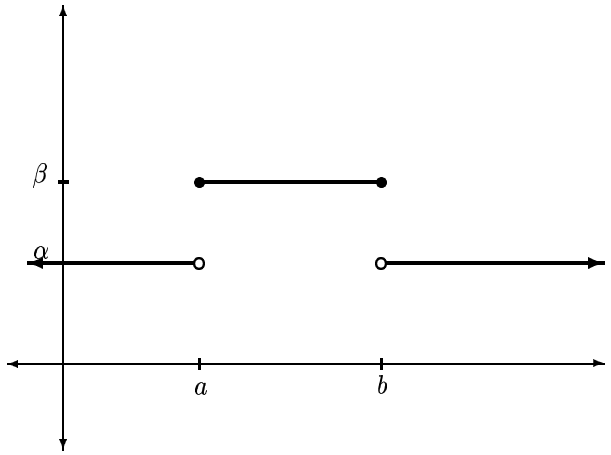

Figure 1. The graph of a typical $\rho_{a,b,\alpha,\beta}$.

A graph of one such $\rho_{a,b,\alpha,\beta}$ is shown in Figure 1. For each $\lambda \geq 0, \mu, \eta \in [0,1]$, let $\mathrm{PINT}_{\lambda,\mu,\eta}$ (PINT stands for "Probabilistic INTervals") consist of all those $\rho_{a,b,\alpha,\beta}$ for which

- $\beta \geq \alpha + \lambda$ (i.e., β is significantly bigger than α), and
- $\alpha \in [\mu - \eta, \mu + \eta]$ (i.e., α is close to μ).

3.2. Reduction

We start with the following straightforward lemma, which describes the p-concepts that the algorithm will learn.

LEMMA 1 *Choose $d, n \in \mathbf{N}$, a product distribution D over $\mathbf{Q}^d$, $\vec{a}, \vec{b} \in \mathbf{Q}^d$, $k \in \{1, ..., d\}$, and $z \in \mathbf{Q}$. If $z \in [a_k, b_k]$,*

$$\begin{aligned} &\Pr_{(\vec{u}_1,...,\vec{u}_n) \in D^n}(\mathrm{OR}_{r_{\vec{a},\vec{b}}}(\vec{u}_1, ..., \vec{u}_n) = 1 \mid u_{1,k} = z) \\ &= 1 - (1 - \Pr_{\vec{u} \in D}(\wedge_{\ell \neq k} u_\ell \in [a_\ell, b_\ell]))(1 - D(r_{\vec{a},\vec{b}}))^{n-1}. \end{aligned} \tag{1}$$

If $z \notin [a_k, b_k]$,

$$\Pr_{(\vec{u}_1,...,\vec{u}_n) \in D^n}(\mathrm{OR}_{r_{\vec{a},\vec{b}}}(\vec{u}_1, ..., \vec{u}_n) = 1 \mid u_{1,k} = z) = 1 - (1 - D(r_{\vec{a},\vec{b}}))^{n-1}. \tag{2}$$

Proof: To prove (1), observe that in that case, in order for $\mathrm{OR}_{r_{\vec{a},\vec{b}}}(\vec{u}_1, ..., \vec{u}_n) = 0$, each of the n instances must fail to be in $r_{\vec{a},\vec{b}}$. These events are independent. Since D is a product distribution, if $u_{1,k} \in [a_k, b_k]$, the probability that the other

components imply $\vec{u}_1 \notin r_{\vec{a},\vec{b}}$ is $1 - \Pr_{\vec{u}\in D}(\wedge_{\ell\neq k} u_\ell \in [a_\ell, b_\ell])$, independent of the particular value of $u_{1,k}$. For $\vec{u}_2, ..., \vec{u}_n$, this probability is $1 - D(r_{\vec{a},\vec{b}})$.

The proof of (2) is similar, except that the first instance has already failed by virtue of the fact that $z \notin [a_k, b_k]$. □

Next, we establish a simple lemma that relates approximating each of the intervals of a rectangle $r_{\vec{a},\vec{b}}$ with respect to the projections of a product distribution D over $\mathbf{Q}^d$ to approximating $\mathrm{OR}_{r_{\vec{a},\vec{b}}}$ with respect to D^n.

LEMMA 2 *Choose $d, n \in \mathbf{N}$, $\epsilon > 0$. Choose a distribution D over $\mathbf{Q}^d$. Choose $\vec{a}, \vec{b}, \hat{\vec{a}}, \hat{\vec{b}} \in \mathbf{Q}^d$.*

If, for each $k \in \{1, ..., d\}$,

$$\Pr_{\vec{u}\in D}(u_k \in [a_k, b_k] \Delta [\hat{a}_k, \hat{b}_k]) \leq \frac{\epsilon}{dn},$$

where Δ denotes the symmetric difference, then

$$D^n(\mathrm{OR}_{r_{\vec{a},\vec{b}}} \Delta \mathrm{OR}_{r_{\hat{\vec{a}},\hat{\vec{b}}}}) \leq \epsilon.$$

Proof: Obviously,

$$\begin{aligned} & D^n(\mathrm{OR}_{r_{\vec{a},\vec{b}}} \Delta \mathrm{OR}_{r_{\hat{\vec{a}},\hat{\vec{b}}}}) \\ & \leq \Pr_{(\vec{u}_1,...,\vec{u}_n)\in D^n}(\vee_{j=1}^n \vee_{k=1}^d u_{j,k} \in [a_k, b_k] \Delta [\hat{a}_k, \hat{b}_k]) \\ & \leq \sum_j \sum_k \Pr_{(\vec{u}_1,...,\vec{u}_n)\in D^n}(u_{j,k} \in [a_k, b_k] \Delta [\hat{a}_k, \hat{b}_k]). \end{aligned}$$

This completes the proof. □

We need the following technical lemma. The idea is that if x is a good estimate of y, then $x^{1-1/n}$ must be a good estimate of $y^{1-1/n}$.

LEMMA 3 *Choose $\epsilon > 0$, $0 < \nu \leq \epsilon/2$, x and y. Choose $n \in \mathbf{N}$, $n \geq 2$. If*

$$x \geq \epsilon,$$

and

$$|x - y| \leq \nu,$$

then

$$|x^{1-1/n} - y^{1-1/n}| \leq \nu\sqrt{\frac{2}{\epsilon}}.$$

Proof: Define $f : \mathbf{R} \to \mathbf{R}$ by $f(u) = u^{1-1/n}$. Then $f'(u) = (1 - 1/n)u^{-1/n}$. So if $u \geq \epsilon - \nu$,

$$f'(u) \leq (1 - 1/n)(\epsilon - \nu)^{-1/n}.$$

Since $n \geq 2$ and $\nu \leq \epsilon/2$, this implies that for $u \geq \epsilon - \nu$, $f'(u) \leq \sqrt{2/\epsilon}$. Applying the mean value theorem together with the fact that x and y are both at least $\epsilon - \nu$ completes the proof. □

Now we are ready for the lemma giving the reduction.

LEMMA 4 *Assume that there are functions* $\varphi_1, \varphi_2 : \mathbf{R} \times \mathbf{R} \times \mathbf{R} \to \mathbf{N}$ *such that for each* $\lambda, \mu, \eta \in [0,1]$, *there is an algorithm* A *such that for each* $\epsilon, \gamma, \delta > 0$, A *properly* $(\epsilon, \gamma, \delta)$*-learns* $\mathrm{PINT}_{\lambda,\mu,\eta}$ *with respect to product distributions from* $\varphi_1(\epsilon, \gamma, \delta)$ *examples in* $\varphi_2(\epsilon, \gamma, \delta)$ *time.*

Then there is a multiple-instance learning algorithm B *such that, for all* $1/2 > \epsilon, \delta > 0, d, n \in \mathbf{N}$, $n \geq 2$, B (ϵ, δ)*-learns* BOXES_d *from*

$$\varphi_1\left(\frac{\epsilon}{nd}, \frac{\epsilon^2}{32n}, \frac{\delta}{2d}\right) + \left\lceil 32768 \frac{n^2}{\epsilon^5} \log \frac{4}{\delta} \right\rceil$$

n-examples in

$$O\left(d\varphi_2\left(\frac{\epsilon}{nd}, \frac{\epsilon^2}{32n}, \frac{\delta}{2d}\right) + \frac{n^2}{\epsilon^5} \ln \frac{1}{\delta}\right)$$

time.

Proof: Consider the algorithm B that performs the following steps in order:

- Algorithm B sets $\hat{p}$ to be the fraction of the first $\left\lceil 32768 \frac{n^2}{\epsilon^5} \log \frac{4}{\delta} \right\rceil$ examples whose label is 1. If $\hat{p} \leq 3\epsilon/4$, it outputs $\emptyset$ and halts, and if $\hat{p} \geq 1 - 3\epsilon/4$, it outputs $\mathbf{Q}^d$ and halts. (This step is to test for degenerate cases and eliminate them with high probability. The estimate $\hat{p}$ of the probability that a multiple-instance example is labelled 1 needs to be especially accurate, though, due to way it is used by the subroutine p-concept learning algorithms.)
- If it didn't halt in the previous step, Algorithm B sets $\mu = 1 - (1 - \hat{p})^{1-1/n}$. (This is our estimate for the conditional probability that a multiple-instance example is labelled 1, given that a particular instance is not in the rectangle to be learned. This will be passed to the subroutine p-concept learning algorithms, which will then constrain their guesses at α (see Figure 1) to be near μ.) Next,
  - for each $k \in \{1, ..., d\}$, Algorithm B constructs a sample for an Algorithm A for properly learning

    $$\mathrm{PINT}_{\epsilon^2/(8n),\mu,\epsilon^2/(64n)}$$

    by pairing the kth component of the first instance of each of the remaining examples with the example's label. That is, if

    $$((\vec{x}_{1,1}, ..., \vec{x}_{1,n}), y_1), ..., ((\vec{x}_{m,1}, ..., \vec{x}_{m,n}), y_m)$$

    are the remaining examples, it feeds

    $$(x_{1,1,k}, y_1), ..., (x_{m,1,k}, y_m)$$

to Algorithm A. (Recall that $\mathrm{PINT}_{\epsilon^2/(8n),\mu,\epsilon^2/(64n)}$ is the set of "probabilistic intervals" like that in Figure 1 where the two conditional probabilities are at least $\epsilon^2/(8n)$ apart and the lesser of the two is $\epsilon^2/(64n)$-close to μ.)

- Algorithm B now outputs $r_{(\hat{a}_1,\ldots,\hat{a}_d),(\hat{b}_1,\ldots,\hat{b}_d)}$ (i.e. the rectangle formed through the intervals returned by the p-concept learning algorithm).

Choose $d, n \in \mathbf{N}$, a product distribution D over $\mathbf{Q}^d$ and $\vec{a}, \vec{b} \in \mathbf{Q}^d$.

By the standard Hoeffding bound (see [11, Appendix B]), the probability that $|\hat{p} - D^n(\mathrm{OR}_{r_{\vec{a},\vec{b}}})| > \frac{\epsilon^{2.5}}{128n}$ is at most $\delta/2$. (Here and elsewhere in this proof, we will refer to $\mathrm{OR}_{r_{\vec{a},\vec{b}}}$ and the set on which it evaluates to 1 interchangeably.) Therefore, it is sufficient to prove that, if

$$|\hat{p} - D^n(\mathrm{OR}_{r_{\vec{a},\vec{b}}})| \leq \frac{\epsilon^{2.5}}{128n}, \tag{3}$$

then

$$\Pr(D^n(\mathrm{OR}_{r_{\vec{a},\vec{b}}} \Delta \mathrm{OR}_{r_{\vec{a},\vec{b}}}) \geq \epsilon) \leq \delta/2.$$

Assume (3) holds. If Algorithm B halts during the first bullet, obviously its error is suitably small. If B does not halt during the first bullet, (3) implies

$$\epsilon/2 \leq D^n(\mathrm{OR}_{r_{\vec{a},\vec{b}}}) \leq 1 - \epsilon/2. \tag{4}$$

Define

$$\alpha = 1 - (1 - D(r_{\vec{a},\vec{b}}))^{n-1}.$$

Applying Lemma 1, for each k, for each $z \notin [a_k, b_k]$,

$$\alpha = \Pr_{(\vec{u}_1,\ldots,\vec{u}_n) \in D^n} (\mathrm{OR}_{r_{\vec{a},\vec{b}}}(\vec{u}_1, \ldots, \vec{u}_n) = 1 \mid u_{1,k} = z). \tag{5}$$

In other words, the p-concept describing the relationship between the kth component of the first instance of a multiple-instance example and the label evaluates to α outside of $[a_k, b_k]$. We claim that Lemma 3 implies that μ is a good estimate of α, i.e. that

$$|\mu - \alpha| \leq \frac{\epsilon^2}{64n}. \tag{6}$$

To see this, note that (3) implies that

$$|(1 - \hat{p}) - D^n(\neg \mathrm{OR}_{r_{\vec{a},\vec{b}}})| \leq \frac{\epsilon^{2.5}}{128n}.$$

Plugging in the fact that $D^n(\neg \mathrm{OR}_{r_{\vec{a},\vec{b}}}) = (1 - D(r_{\vec{a},\vec{b}}))^n$, we get

$$|(1-\hat{p}) - (1 - D(r_{\vec{a},\vec{b}}))^n| \leq \frac{\epsilon^{2.5}}{128n}.$$

and by (4) $(1 - D(r_{\vec{a},\vec{b}}))^n \geq \epsilon/2$. Applying Lemma 3, this implies

$$|(1-\hat{p})^{1-1/n} - (1 - D(r_{\vec{a},\vec{b}}))^{n-1}| \leq \frac{\epsilon^2}{64n},$$

which in turn implies

$$|(1-(1-\hat{p})^{1-1/n}) - (1 - (1 - D(r_{\vec{a},\vec{b}}))^{n-1})| \leq \frac{\epsilon^2}{64n}.$$

Substituting the definitions of μ and α yields (6).

Choose $k \in \{1, ..., d\}$. Define

$$\nu_k = \Pr_{\vec{u} \in D}(\wedge_{\ell \neq k} u_\ell \in [a_\ell, b_\ell])$$

and

$$\beta_k = 1 - (1 - \nu_k)(1 - D(r_{\vec{a},\vec{b}}))^{n-1}.$$

By Lemma 1, together with (5), in its kth call, if D_k is the distribution on the kth component of a vector drawn according to D, algorithm A receives examples drawn according to $P_{D_k, \rho_{a_k}, b_k, \alpha, \beta_k}$.

In order to show that

$$\rho_{a_k, b_k, \alpha, \beta_k} \in \mathrm{PINT}_{\epsilon^2/(8n), \mu, \epsilon^2/(64n)},$$

we'd like to lower bound

$$\beta_k - \alpha = \nu_k (1 - D(r_{\vec{a},\vec{b}}))^{n-1}. \tag{7}$$

We begin by working on ν_k. By (4),

$$\begin{aligned}
\epsilon/2 &\leq D^n(\mathrm{OR}_{r_{\vec{a},\vec{b}}}) \\
&= D^n\{(\vec{u}_1, ..., \vec{u}_n) : \vee_{j=1}^n \vec{u}_j \in r_{\vec{a},\vec{b}}\} \\
&\leq \sum_{i=1}^n D^n\{(\vec{u}_1, ..., \vec{u}_n) : \vec{u}_j \in r_{\vec{a},\vec{b}}\} \\
&= nD(r_{\vec{a},\vec{b}}).
\end{aligned}$$

But directly from the definition, $\nu_k \geq D(r_{\vec{a},\vec{b}})$. Thus,

$$\nu_k \geq \frac{\epsilon}{2n}. \tag{8}$$

Now we want to lower bound $(1 - D(r_{\vec{a},\vec{b}}))^{n-1}$. Recall that the second inequality from (4) is $D^n(\neg\mathrm{OR}_{r_{\vec{a},\vec{b}}}) \geq \epsilon/2$. But $D^n(\neg\mathrm{OR}_{r_{\vec{a},\vec{b}}}) = (1 - D(r_{\vec{a},\vec{b}}))^n$. Thus

$$(1 - D(r_{\vec{a},\vec{b}}))^{n-1} \geq (1 - D(r_{\vec{a},\vec{b}}))^n \geq \epsilon/2.$$

Putting this together with (8) and (7), we have

$$\beta_k - \alpha \geq \frac{\epsilon^2}{4n}. \tag{9}$$

Putting together (9) and (6), the p-concept $\rho_{a_k,b_k,\alpha,\beta_k}$ which A is trying to learn during its kth call is in

$$\mathrm{PINT}_{\epsilon^2/(8n),\mu,\epsilon^2/(64n)}.$$

Thus, since A was given $\varphi_1(\frac{\epsilon}{nd}, \frac{\epsilon^2}{32n}, \frac{\delta}{2d})$ examples on the kth call, with probability at least $1 - \delta/(2d)$, the hypothesis $\rho_{\hat{a}_k,\hat{b}_k,\hat{\alpha}_k,\hat{\beta}_k}$ output by A satisfies

$$\Pr_{\vec{u}\in D}(|\rho_{\hat{a}_k,\hat{b}_k,\hat{\alpha}_k,\hat{\beta}_k}(u_k) - \rho_{a_k,b_k,\alpha,\beta_k}(u_k)| > \epsilon^2/(32n)) \leq \epsilon/(nd). \tag{10}$$

We claim that

$$u_k \in [\hat{a}_k, \hat{b}_k]\Delta[a_k, b_k] \;\Rightarrow\; |\rho_{\hat{a}_k,\hat{b}_k,\hat{\alpha}_k,\hat{\beta}_k}(u_k) - \rho_{a_k,b_k,\alpha,\beta_k}(u_k)| > \epsilon^2/(32n). \tag{11}$$

In other words, informally, where $[\hat{a}_k, \hat{b}_k]$ is wrong about $[a_k, b_k]$, $\rho_{\hat{a}_k,\hat{b}_k,\hat{\alpha}_k,\hat{\beta}_k}$ is a bad estimate of $\rho_{a_k,b_k,\alpha,\beta_k}$. First, since A is a proper learning algorithm, its hypothesis $\rho_{\hat{a}_k,\hat{b}_k,\hat{\alpha}_k,\hat{\beta}_k}$ is in

$$\mathrm{PINT}_{\epsilon^2/(8n),\mu,\epsilon^2/(64n)},$$

and therefore $\hat{\beta}_k \geq \hat{\alpha}_k + \epsilon^2/(8n)$ and $|\hat{\alpha}_k - \mu| \leq \epsilon^2/(64n)$. Since $|\mu - \alpha| \leq \epsilon^2/(64n)$, $|\hat{\alpha}_k - \alpha| \leq \epsilon^2/(32n)$. (In other words, Algorithm A couldn't help but to get α approximately right, simply by virtue of the fact that it is a proper learning algorithm.) From here, we break up our proof of (11) into cases:

- If $u_k \in [a_k, b_k] - [\hat{a}_k, \hat{b}_k]$, then since $|\hat{\alpha}_k - \alpha| \leq \epsilon^2/(32n)$,

  $$\rho_{\hat{a}_k,\hat{b}_k,\hat{\alpha}_k,\hat{\beta}_k}(u_k) = \hat{\alpha}_k \leq \alpha + \epsilon^2/(32n), \tag{12}$$

  but in this case

  $$\rho_{a_k,b_k,\alpha,\beta_k}(u_k) = \beta_k. \tag{13}$$

  Putting together (13), (12), and (9) implies (11) in this case.

- If $u_k \in [\hat{a}_k, \hat{b}_k] - [a_k, b_k]$, the fact that $|\hat{\alpha}_k - \alpha| \leq \epsilon^2/(32n)$ and $|\hat{\alpha}_k - \hat{\beta}_k| \geq \epsilon^2/(8n)$ implies, using the triangle inequality, that $|\hat{\beta}_k - \alpha| \geq \epsilon^2/(8n) - \epsilon^2/(32n)$, establishing (11) in this case.

Now, (11) and (10) together imply that with probability at least $1 - \delta/(2d)$,

$$\Pr_{\vec{u}\in D}(u_k \in [\hat{a}_k, \hat{b}_k]\Delta[a_k, b_k]) \leq \epsilon/(nd). \tag{14}$$

Since k was chosen arbitrarily, with probability at least $1-\delta/2$, for *all* $k \in \{1, ..., d\}$ the above holds, and applying Lemma 2 completes the proof. □

4. Solving the p-concept problem

The results of Kearns and Schapire [7] imply that each $\mathrm{PINT}_{\lambda,\mu,\eta}$ can be learned in polynomial time, but for our application we need a proper learning algorithm.

4.1. Definitions

The following definition is due to Pollard [11]. Choose a set F of functions from $\mathbf{Q}$ to $[0,1]$. We say that F *shatters* $(u_1,s_1),...,(u_d,s_d)$ if for each $q_1,...,q_d \in \{0,1\}^d$, there is an $f \in F$ such that for all $i \in \{1,...,d\}$, $f(u_i) \geq s_i \Leftrightarrow q_i = 1$. The *pseudo-dimension* of F is the size of the longest sequence shattered by F.

A p-concept learning algorithm A *minimizes quadratic loss with respect to* F iff for all $(x_1,y_1),...,(x_m,y_m) \in \mathbf{Q} \times \{0,1\}$, if $h = A((x_1,y_1),...,(x_m,y_m))$, then h minimizes $\sum_{i=1}^{m}(h(x_i)-y_i)^2$ from among functions in F.

4.2. Learning $\mathbf{PINT}_{\lambda,\mu,\eta}$

We begin by recording a lemma that follows directly from the work of Kearns and Schapire [7, Lemma 2.2 and Theorem 5.1].

LEMMA 5 ([7]) *Choose a set F of functions from $\mathbf{Q}$ to $[0,1]$. If d is the pseudo-dimension of F, then any algorithm which minimizes quadratic loss with respect to F properly (ϵ,γ,δ)-learns F from*

$$O\left(\frac{1}{\epsilon^2\gamma^4}\left(d\log\frac{1}{\epsilon\gamma}+\log\frac{1}{\delta}\right)\right)$$

examples.

Next, we bound the pseudo-dimension of $\mathrm{PINT}_{\lambda,\mu,\eta}$. The proof uses ideas from the bound on the VC-dimension of BOXES_2 of Blumer, Ehrenfeucht, Haussler and Warmuth [3].

LEMMA 6 *For any $\lambda \geq 0, \mu,\eta \in [0,1]$, the pseudo-dimension of $\mathrm{PINT}_{\lambda,\mu,\eta}$ is at most 4.*

Proof: Choose $\lambda \geq 0, \mu,\eta \in [0,1]$. Assume for contradiction that

$$(u_1,s_1),...,(u_5,s_5)$$

is shattered by $\mathrm{PINT}_{\lambda,\mu,\eta}$.

Assume without loss of generality that

$$s_1 = \min\{s_1,...,s_5\}.$$

Choose $\mathrm{left},\mathrm{right},\mathrm{up} \in \{2,...,5\}$ such that

$$\begin{aligned} u_{\mathrm{left}} &= \min\{u_2,...,u_5\} \\ u_{\mathrm{right}} &= \max\{u_2,...,u_5\} \\ s_{\mathrm{up}} &= \max\{s_2,...,s_5\}. \end{aligned}$$

Choose middle from $\{2, ..., 5\} - \{\text{left}, \text{right}, \text{up}\}$.

The definition of shattering implies that there exist $a, b \in Q, \alpha, \beta \in [0, 1], \beta \geq \alpha$ such that

- $\rho_{a,b,\alpha,\beta}(u_1) < s_1$,
- for each $i \in \{\text{left}, \text{right}, \text{up}\}$, $\rho_{a,b,\alpha,\beta}(u_i) \geq s_i$, and
- $\rho_{a,b,\alpha,\beta}(u_{\text{middle}}) < s_{\text{middle}}$.

Fix such a, b, α, β.

Since $\rho_{a,b,\alpha,\beta}(u_1) < s_1$, it must be the case that $\alpha < s_1$, and since

$$s_1 = \min\{s_1, ..., s_5\},$$

this means that α is less than all the s_i's. Thus, for each $i \in \{\text{left}, \text{right}, \text{up}\}$, $\rho_{a,b,\alpha,\beta}(u_i) \geq s_i$ by virtue of the fact that $u_i \in [a, b]$ and $s_i \leq \beta$. But, by the definitions, this implies that $u_{\text{middle}} \in [a, b]$ and $s_{\text{middle}} \leq \beta$, which in turn implies $\rho_{a,b,\alpha,\beta}(u_{\text{middle}}) \geq s_{\text{middle}}$, a contradiction. □

Next, we describe a simple lemma analyzing a subroutine in the minimizing quadratic loss algorithm.

LEMMA 7 *There is an algorithm which, given $c_1, c_2 \in \mathbf{Q}, \xi_1 > 0, \xi_2, \xi_3 \in \mathbf{Q}$, minimizes*

$$f(x, y) = x^2 + c_1 x + y^2 + c_2 y$$

subject to

$$x \geq y + \xi_1$$
$$\xi_2 \leq y \leq \xi_3$$

in constant time.

Proof: First, straightforward calculus implies that, absent any constraints, $x^2 + c_1 x$ and $y^2 + c_2 y$ are minimized at $-c_1/2$ and $-c_2/2$ respectively, and further, that as one of x or y is moved away from its optimal value while the other is held constant, f increases monotonically.

If $-c_1/2$ and $-c_2/2$ satisfy the constraints, the algorithm is done. Suppose $-c_1/2$ and $-c_2/2$ do not satisfy the constraints. Suppose that in this case (x, y) is a feasible solution that does not make any of the constraints tight. Then one can find an improved feasible solution by fixing one of x or y, and moving the other toward its optimal unconstrained value. Therefore, any optimal feasible solution makes one of the three constraints tight. Making any of the constraints tight allows one to express one variable as a linear function of the other, reducing the problem to that of minimizing a convex quadratic in one variable subject to bounds on that variable, trivially solvable in constant time. One can therefore solve the three such problems arising from making the three constraints tight, and output the minimal solution. □

Now we are ready to give the algorithm for learning $\text{PINT}_{\lambda,\mu,\eta}$.

LEMMA 8 *Choose* $\lambda, \mu, \eta \in [0,1]$. *There is an algorithm* A *such that, for any* $\epsilon, \gamma, \delta > 0$, A *properly* $(\epsilon, \gamma, \delta)$*-learns* $\mathrm{PINT}_{\lambda,\mu,\eta}$ *from*

$$O\left(\frac{1}{\epsilon^2\gamma^4}\left(\log\frac{1}{\epsilon\gamma} + \log\frac{1}{\delta}\right)\right)$$

examples in

$$O\left(\frac{1}{\epsilon^4\gamma^8}\left(\log\frac{1}{\epsilon\gamma} + \log\frac{1}{\delta}\right)^2\right)$$

time.

Proof: From Lemmas 5 and 6, the sample complexity bound holds. What remains is to show that one can minimize quadratic loss on $(x_1, y_1), ..., (x_m, y_m) \in \mathbf{Q} \times \{0,1\}$ for $\mathrm{PINT}_{\lambda,\mu,\eta}$ in $O(m^2)$ time.

Assume without loss of generality that $x_1 \leq ... \leq x_m$. Then, for any $i \in \{1, ..., m-1\}$, if β and α are fixed, the quadratic loss of $\rho_{a,b,\alpha,\beta}$ does not change for $b \in [x_i, x_{i+1})$ and for $a \in (x_i, x_{i+1}]$. Thus, it suffices to consider the values of a and b that coincide with sample points, together with one value on either end of the whole sample.[2] This implies that one only need try $O(m^2)$ pairs of values for a and b. Finally, Lemma 7 implies that, for fixed a, b, one can minimize the quadratic loss of $\rho_{a,b,\alpha,\beta}$ as a function of β and α, subject to the constraints imposed on them by virtue of membership in $\mathrm{PINT}_{\lambda,\mu,\eta}$, in constant time, as long as we have access to the number of positive and negative points to the left and right of each of the elements of the sample. By sweeping from left to right maintaining running counts, this can be taken care of in an $O(m)$ time preprocessing step. □

Acknowledgments

We are very grateful to Avrim Blum for pointing out a mistake in an earlier version of this paper. We also thank Pankaj Agarwal and Dana Ron for valuable conversations, and anonymous referees for their comments.

This work was done while Phil Long was at Duke University supported by Office of Naval Research grant N00014–94–1–0938. Lei Tan was supported by a Duke University Department of Computer Science Fellowship.

Notes

1. We thank Dana Ron for helpful communication on this point.
2. Note that one actually only needs to try positive points since it's apparent that the optimal interval should always end at positive points. Otherwise, one can have lower quadratic loss by screening out the outermost negative points.

References

1. P. Auer, P.M. Long, and A. Srinivasan. Approximating hyper-rectangles: learning and pseudo-random sets. *Proceedings of the 29th ACM Symposium on the Theory of Computing*, 1997.
2. A. Blum and A. Kalai. A note on learning from multiple-instance examples. *Machine Learning*, this issue.
3. A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *JACM*, 36(4):929–965, 1989.
4. T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1–2):31–71, 1997.
5. A. Ehrenfeucht, D. Haussler, M. Kearns, and L.G. Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82(3):247–251, 1989.
6. M.J. Kearns. Efficient noise-tolerant learning from statistical queries. *Proceedings of the 25th ACM Symposium on the Theory of Computing*, 1993.
7. M.J. Kearns and R.E. Schapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48(3):464–497, 1994.
8. P.M. Long and L. Tan. PAC learning axis-aligned rectangles with respect to product distributions from multiple-instance examples. *The 1995 Conference on Computational Learning Theory*, pages 228–234, 1996.
9. P.M. Long and M. K. Warmuth. Composite geometric concepts and polynomial predictability. *Information and Computation*, 113(2):203–252, 1994.
10. L. Pitt and L.G. Valiant. Computational limitations on learning from examples. *Journal of the Association for Computing Machinery*, 35(4):965–984, 1988.
11. D. Pollard. *Convergence of Stochastic Processes*. Springer Verlag, 1984.
12. L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
13. Kenji Yamanishi. A learning criterion for stochastic rules. *Machine Learning*, 1992. Special Issue on the Proceedings of the 3nd Workshop on Computational Learning Theory, to appear.