Active Learning Models and Noise

Sara Stolbach
COMS 6253, Advanced CLT
ss3067@columbia.edu

May 3, 2007

Abstract

I study active learning in general pool-based active learning models as well noisy active learning algorithms and then compare them for the class of linear separators under the uniform distribution.

1 Introduction

There are often cases where data is abundant and easily accessible but labeling the data is costly. For example, in bioinformatics many DNA sequences are available but decoding one sequence can take many hours or days for one person to achieve. This scenario is known as *active learning*; the labels of data are hidden and the learner can pay for the label of any example. This is not captured in the typical PAC supervised learning scenario. An active learning algorithm will want to minimize the number of examples it needs to label due to the expense of labeling. In this paper I will focus on pool-based active learning models, which is when the learner can pay for the label of any example in a pool of unlabeled examples as opposed to another model where points can be created synthetically.

A noisy dataset is difficult in the active learning setting since standard active learning models seek to find the most informative examples which tend to be the most noise-prone. I will first discuss some noiseless active learning models and show why they are noise-prone and then discuss two noisy active learning models and how they compare as well as discuss the next steps that should be taken.

For the most part active learning methods fall under three orthogonal techniques; generalized binary search, opportunistic priors or algorithmic luckiness, and Bayesian assumptions. Opportunistic priors is when a uniform bet over all $\hat{H}$ leads to standard VC generalization bounds. If the algorithm places more weight on a certain hypothesis, it could be excellent if guessed right but worse than usual if guessed wrong. This method is not as practical because of its unpredictability. The other two methods are discussed in the paper. Progress can be measured in a number of ways such as the rate at which the size of the version space decreases and the number of label requests needed. I will be focusing on the number of label requests needed.

2 Preliminaries

Some general definitions used throughout the paper are mentioned here; the variables have these meanings unless specified differently. Let X be the instance space of examples x that are i.i.d over the uniform input distribution D in version space V. Let n be the number of label requests. Let C be the concept class over distribution P. Let η denote the noise rate in noisy models.

A common application examined in active learning is linear separators through the origin of a unit sphere in $\mathbb{R}^d$. X is the set of all data on the surface area of the sphere such that $X = \{x \in \mathbb{R}^d : ||x|| = 1\}$. Each example in X is denoted as $(\vec{x}, t)$ where $\vec{x}$ is the direction of the example and t is the offset. In other words $X = S^d \times [-1, +1]$ where S^d is the unit sphere around the origin of $\mathbb{R}^d$. The distribution D on X is uniform. H is the class of linear separators through the origin, and any $h \in H$ is a homogeneous hyperplane. h is represented by a unit vector $w \in X$ with the classification rule $h(x) = sign(w \cdot x)$. The distance between two hypothesis, u and v in H with respect to D is given by distance$_D(u, v) = \frac{\theta(u,v)}{\pi}$ where $\theta(u, v) = \arccos(u \cdot v)$.

3 General Models

3.1 Bayesian Model

The Query by Committee (QBC) algorithm [1] is one of the most significant pool-based active learning algorithms. It shows that using queries over random unlabeled examples can accelerate the learning concept of some classes over standard learning approaches. Work is done in the Bayesian model which differs from the PAC model in that it is assumed that the target concept is chosen according to a prior distribution P over C and that this distribution is known to the learner. A Bayesian Model has an immediate benefit for active learning since if there is large agreement on unlabeled data you can stop and output the current hypothesis.

The algorithm assumes realizability, meaning a perfect classifier exists. The papers analysis is based on probabilistic assumptions and they show that queries can help accelerate learning of concept classes that are deterministic and noiseless. The paper also goes on to discuss a generalized form of QBC that uses two distributions. It is more computationally intensive but can have an outcome that is not binary or discrete and the inputs can be stochastic.

Algorithm 1 QBC Algorithm

```
Input: $\epsilon > 0$, $\delta > 0$, Gibbs, Sample, Label
Initialize: $n = 0$, $V_0 = C$
repeat
  Call the Sample oracle to get a random instance of $x$.
  Call Gibbs twice to get two predictions $p_1$ and $p_2$ for $x$.
  if $p_1 = p_2$ then
    reject the example
  else
    call the Label($x$) to get $c(x)$, increase $n$ by 1 and set $V_n$ to be all
    concepts $c' \in V_{n-1}$ where $c'(x) = c(x)$
  end if
until more than $t_n$ consecutive examples are rejected.
Output: the Gibbs prediction hypothesis
```

The Query by Committee algorithm (Algorithm 1) uses an oracle denoted **Gibbs**(V, x) which computes the Gibbs prediction rule. It predicts the label of a new example x by randomly choosing an $h \in C$ according to D, restricted to $V \subset C$, and labeling x according to it. Two calls to Gibbs with the same V and x can result in different predictions. The goal is to label x in order to maximize the expected information gain to cause an exponentially fast decrease in the error of the Gibbs prediction rule. However, this is not guaranteed mainly because distribution is ignored. If queries of the same type are always called, the prediction error will stay large. The **Sample** oracle returns an unlabeled example $x \in X$ chosen according to D. A call to the **Label** oracle with input x returns $c(x)$ which is the label of x according to the target concept. The iterations are done in a batch learning scenario until some termination condition is achieved. The termination condition is met when $t_n = \frac{1}{\epsilon} \ln \frac{\pi^2(n+1)^2}{3\delta}$ consecutive examples are rejected.

Theorem 1 *If a concept class C has VC-dimension $0 < d < \infty$ and the expected information gain of queries made by* **QBC** *is uniformly bounded by $g > 0$ bits, then with probability larger than $1 - \delta$ over the random choice of the target concept, the sequence of examples, and the choices made by* **QBC**, *the number of calls to* **Sample** *is smaller than*

$$m_0 = \max\left(\frac{4d}{\epsilon\delta}, \frac{160(d+1)}{g\epsilon}\max\left(6, \ln\frac{80(d+1)}{\epsilon\delta^2 g}\right)^2\right),$$

the number of calls to **Label** *is smaller than*

$$n_0 = \frac{10(d+1)}{g}\ln\frac{4m_0}{\delta}$$

and the probability that the **Gibbs** *prediction that uses the final version space of* **QBC** *makes a mistake is smaller than ϵ.* [1]

It is easy to show that if QBC ever stops then the error of the resulting hypothesis is small with high probability. The real question is will the QBC algorithm stop; The proof of Theorem 1, in [1], shows that it will stop if the number of examples that are rejected between consecutive queries increases with the number of queries; a linear increase. This implies that the probability of accepting a query or making a prediction mistake is exponentially small compared to the number of queries asked (based on information gain $g > 0$).

A common class examined in Active Learning is uniformly distributed half spaces through the origin of the unit sphere in $\mathbb{R}^d$. The information gain from random examples will vanish as d goes to infinity because in a high dimension the volume of the sphere is concentrated near the equator. A typical example will cut the sphere away from the equator which means that query examples are especially important in high dimensions; QBC will likely choose two random points near the equator so the example that separates them will likely be near the equator which implies that QBC can attain a finite information gain in high dimensions. [1] prove the lower bound on the information gain which implies that Theorem 1 holds and that the number of calls to the Sample oracle is $\mathrm{O}(\frac{d}{\epsilon}\log\frac{1}{\delta\epsilon})$ and the number of calls to the label oracle is $O(\frac{d}{\epsilon})$.

The paper also proves that QBC Algorithm has such results for the perceptron class as well by modeling it as a special case of the distributed half-spaces problem. Dasgupta, et al [2] show an algorithm which uses a simple modification to the perceptron update to provide even better results.

The perceptron algorithm uses the same concept class of linear separators where datapoints are on the surface area of the unit sphere in $\mathbb{R}^d$. It starts with an initial hypothesis $V_0 \in \mathbb{R}^d$ and in each iteration receives an unlabeled point, makes a prediction $sign(V_t \cdot x_t)$ and during the filtering step it decides whether to ask for its label based on $|V_t \cdot x_t| \leq$ threshold, s_t. If the label is requested the update step is called. The regular perceptron update is "if (x_t, y_t) is misclassified then $V_{t+1} = V_t + y_t x_t$". The error rate cannot be better than $\Omega(1/\sqrt{l_t})$ where l_t is the number of labels queried up to time t. They have changed the update to be "if (x_t, y_t) is misclassified then $V_{t+1} = V_t - 2(V_t \cdot x_t)x_t$". This scaled the update by a factor of $2|V_t \cdot x_t|$ to avoid oscillations cause by points close to the hyperplane. The filtering step is based on s_t; its choice is crucial. [2] set it adaptively by starting it high and keep dividing it in half until a level is reached where enough misclassification points are queried.

The modified perceptron results in a theorem stating that $O(d\log\frac{1}{\epsilon})$ labels are needed when drawing $O(\frac{d}{\epsilon}\log\frac{1}{\epsilon})$ data points at random from the unit sphere in $\mathbb{R}^d$, as opposed to the QBC's need of $O(\frac{d}{\epsilon^2})$ datapoints. It will make $O(d\log\frac{1}{\epsilon})$ errors and have final error $\leq \epsilon$. The bound improvements are based on the change to the update step and the threshold, s_t, used in the filtering step.

The QBC Algorithm would have very poor results in a noisy setting because the wrong examples could be queried for labels producing poor version spaces. In addition, an adversarial noise model could cause the algorithm to never stop.

3.2 Generalized Binary Search

Active learning could also be looked at as an approach to improve the same supervised setting. In a supervised setting of some class with VC dimension d, and error rate ϵ over distribution P some $m = m(\epsilon, d)$ labeled points are needed. Dasgupta, [3] uses a greedy generalized binary search to examine

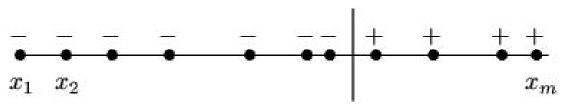

Figure 1: The boundary can be found with just $O(\log m)$ labels using binary search [1]

if fewer then m labels are sufficient to learn the class if the points are not labeled.

In the case of data lying on the real line, and a hypothesis class H of simple threshold functions it is enough to draw $m = O(1/\epsilon)$ random labeled examples from P and return a classifier consistent with them. However, as in Figure 1, if we use unlabeled examples we can use a simple binary search to find the transition from 0 to 1 which requires only $\log m$ labels to infer the rest of them. Hence, there is an exponential improvement. However, what about in the generalized case; Is it possible to pick among $O(m^d)$ possibilities using $o(m)$ labels? If binary search were possible, just $O(d \log m)$ labels would be needed. This is not the case. In $d \geq 2$ there are some cases where the target hypothesis cannot be identified without querying all the labels. However, in the average case the number of labels needed is small.

A variant of a popular greedy scheme is used, where one always asks for the label which most evenly divides the current effective version space weighted by π. π is merely a device for averaging querying counts over some uniform distribution $\hat{H}$. $\hat{H}$ is used instead of H; it reflects the underlying combinatorial structure of the problem, and π can often be chosen to mask its structure. The expected number of labels needed by this strategy is at most $O(\ln |\hat{H}|)$ times that of any other strategy. This is a significant performance guarantee.

A query tree structure is used; there is not always a tree of average depth o(m). The best hope is to come close to minimizing the number of queries and this is done using a greedy approach: Let $S \subseteq \hat{H}$ be the current version space. For each unlabeled x_i, let S_i^+ be the hypothesis which label x_i positive and S_i^- the ones which label it negative. Pick the x_i for which the positive and negative are most nearly equal in π-mass; in other words $\min\{\pi(S_i^+), \pi(S_i^-)\}$ is largest.

Generalized binary search would clearly have poor results in a noisy set-

[1]This figure was taken from Dasgupta's paper [3]

ting because as previously mentioned, the datapoints that are chosen to be labeled tend to be the most noise-prone. A small amount of adversarial noise can cause the datapoints that would be chosen to divide the version space to give virtually no help in learning the concept.

4 Noisy Models

There are two active learning models that work with arbitrary classification noise. The only restriction is that samples are drawn *i.i.d* from some underlying distribution. The results hold for any sort of mechanism used to generate the noise. The algorithms have different restrictions on η. However it is important to note that Kaariainen [10] shows a lower bound of $\Omega(\frac{\eta^2}{\epsilon^2})$ on the sample complexity of any active learner and therefore it can not be hoped to achieve speedups when η is large.

4.1 Agnostic Active Learning

Balcan et al [4] describe an algorithm known as the A^2 algorithm , (Algorithm 2), which is noise tolerant. It was the first noise tolerant algorithm and shows some positive results. They produce bounds for Linear Threshold Functions and linear separators under the uniform distribution where the algorithm is successful for any amount of noise and shows exponential improvements if $\eta < \frac{\epsilon}{16}$. However, the algorithm is not very sample efficient.

A^2 relies on a subroutine such as the VC bound or Occam's Razor bound to compute the lower bound, $LB(S, h, \delta)$, and the upper bound, $UB(S, h, \delta)$ on the true error rate of h, $err_P(h)$, by using a sample S of examples drawn i.i.d from P such that $LB(S, h, \delta) \leq err_P(h) \leq UB(S, h, \delta)$ holds for all h with probability $1 - \delta$.

A^2 algorithm can be viewed as a robust selective sampling algorithm [9]. Selective sampling keeps track of two spaces; the current version space, H_i, consistent with all labels queried so far and the region of uncertainty, R_i. The region of uncertainty includes all datapoints $x \in X$, which have two hypothesis that do not agree on it. In each round i of the selective sampling algorithm, a random unlabeled example is picked from R_i and queried, eliminating all hypothesis in H_i inconsistent with the received label. In the agnostic case we cannot eliminate an hypothesis based on a single example.

The A^2 algorithm samples a set of examples S_i and uses UB and LB to calculate the disagreement of a region, $\text{DISAG}_D(H_i)$. The disagreement of a region is $\text{DISAG}_D(H_i) = \Pr_{x\ D}[\exists h_1, h_2 \in G : h_1(x) \neq h_2(x)]$. If all $h \in H_i$ agree on some region it can be safely eliminated thereby reducing the region of uncertainty. This eliminates all hypotheses whose lower bound is greater than the minimum upper bound. Each round completes when S_i is large enough to reduce half of its region of uncertainty. Therefore, the number of rounds is bounded by $\log \frac{1}{\epsilon}$. The algorithm stops when

$$\text{DISAG}_D(H_i)(\min_{h\in H_i'} UB(S_i, h, \delta_k) - \min_{h\in H_i'} LB(S_i, h, \delta_k)) \leq \epsilon.$$

A^2 returns $h = \text{argmin}(\min_{h\in H_i'} UB(S_i, h, \delta_k))$ where i and k is the iteration that the algorithm was in when it satisfied the condition.

The bounds for the class of Linear Separators under the Uniform Distribution over the unit sphere for A^2 is described in a later section.

Algorithm 2 A^2*Algorithm*

```
Input: $\epsilon$, Sample Oracle for D, Label Oracle O, H
Initialize: $i = 1$, $D_i = D$ $H_i = H$, $S_i = \emptyset$, and $k = 1$
while $\text{DISAG}_D(H_i)(\min_{h\in H_i} UB(S_i, h, \delta_k) - \min_{h\in H_i} LB(S_i, h, \delta_k)) > \epsilon$ do
  Set $S_i = \emptyset$, $H_i' = H_i$, $k = k + 1$
  while $\text{DISAG}_D(H_i') \geq \frac{1}{2}\text{DISAG}_D(H_i)$ do
    if $\text{DISAG}_D(H_i')(\min_{h\in H_i'} UB(S_i, h, \delta_k) - \min_{h\in H_i'} LB(S_i, h, \delta_k)) \leq \epsilon$ then
      Output: $h = \text{argmin}(\min_{h\in H_i'} UB(S_i, h, \delta_k))$
    else
      $S_i' = 2|S_i| + 1$ sample from $D$ satisfying $\exists h_1, h_2 \in H_i : h_1(x) \neq h_2(x)$
      $S_i = S_i \cup \{(x, O(x)) : x \in S_i'\}$;
      $H_i' = \{h \in H_i : LB(S_i, h, \delta_k) \leq \min_{h\in H_i'} UB(S_i, h', \delta_k)\}$; $k = k + 1$;
    end if
  end while
  $H_{i+1} = H_i'$, $D_{i+1} = D_i$ conditioned on $\exists h_1, h_2 \in H_i : h_1(x) \neq h_2(x)$, $i = i + 1$
end while
Output: $h = \text{argmin}(\min_{h\in H_i'} UB(S_i, h, \delta_k))$
```

4.2 Teaching Dimension and Active Learning

Hanneke [5] describes a general active learning noise-tolerant algorithm which is based on exact learning with membership queries. He shows the first nontrivial general upper bound on label complexity in an active learning noise model. In exact learning, the algorithm is required to identify the oracle's actual target function rather then approximating it with high probability. There is no classification noise and the algorithm can ask for the label of any example. In a sense it is a limiting case of PAC active learning. An additional restriction to the algorithm in [5], is that it only works for arbitrary persistent classification noise; meaning the label of a datapoint cannot change from one query to the next.

The goal of exact learning is to ask for labels $f(x)$ until the only concept in C consistent with the observed labels is the target $f \in C$. $C \subseteq C_F$ where F is the corresponding σ-algebra of set X and C_F is the set of all F-measurable $f : X \to \{-1, 1\}$. MembHalving (Hegedüs [6]) is an example of an exact learning algorithm. It uses majority vote to continously minimize the version space until only one hypothesis is left. Querying a specifying set for h_{maj} guarantees that we at least halve the version space each round because it is guaranteed that either h makes a mistake or we identify f.

Definition 1 $\forall f \in C_F, XTD(f, V, U) = inf(\{t | \exists R \subseteq U : |\{h \in V : h(R) = f(R)\}| \leq 1 \wedge |R| \leq t\}$

The teaching dimension is the minimum number of instances a teacher must reveal to uniquely identify *any* target concept chosen from the class. The exact teaching dimension is a more restrictive form; The function, $f(R)$, of a minimal subset, $R \subseteq U$, can be satisfied by only one hypothesis, $h(R)$, and $|R|$ is at most the value of $t = XTD$.

The Teaching Dimension and Active Learning algorithm (TDA), (Algorithm 3), works by continuously reducing the size of the version space until it is between specified sizes. The method **Reduce** achieves this by getting the minimal specifying set, R_i, of the subsequence $S_i \in U$ based upon the majority, h_{maj}, of V. $\bar{V_i}$ is the set of $h \in V$ where $h(R_i) \neq Oracle(R_i)$. Reduce gets the minimal set r times where $\bar{V}$ is all $h \in V$ that did appeared in $> \theta \cdot r$ of the sets $h(R_i)$. It returns $V' = V \setminus \bar{V}$; it is unlikely that these datapoints are noisy. It then get the labels from the version space that should be used for the final hypothesis via the method **Label** and returns the hypothesis which has the smallest error. Label gets the minimal specifying

Algorithm 3 ReduceAndLabel (TDA)

```
**Input:** Finite $V \in C_F, U = \{x_1, x_2, \ldots, x_m\} \in X^m$, values $\epsilon, \delta, \hat{\eta} \in (0, 1]$.
**Initialize:** $u = \lfloor |U|/(5 \ln |V|) \rfloor$, $V_0 = V, i = 0$
**repeat**
  $i = i + 1$
  Let $U_i = \{x_{1+u(i-1)}, x_{2+u(i-1)}, \ldots, x_{ui}\}$
  $V_i = \text{Reduce}(V_{i-1}, U_i, \frac{\delta}{48 \ln |V|}, \hat{\eta} + \frac{\epsilon}{2})$
**until** $|V_i| > \frac{3}{4}|V_{i-1}$ or $|V_i| \leq 1$
Let $\bar{U} = \{x_{ui+1)}, x_{ui+2)}, \ldots, x_{ui+l}\}$, where $l = \lceil 12 \frac{\hat{\eta}}{\epsilon^2} \ln \frac{12|V|}{\delta} \rceil$
$L = \text{Label}(V_{i-1}, \bar{U}, \frac{\delta}{12}, \hat{\eta} + \frac{\epsilon}{2})$
**Output:** Concept $h \in V_i$ having smallest $er_L(h)$, (or any $h \in V$ if $V_i = \emptyset$).
```

set for V based upon h_{maj} as in Reduce, and labels those points. It then looks at all examples in $\bar{U}$ that were not in the minimal set and labels those based upon its majority value over its value all its values from the subsets if $h(R_i) = h_{maj}(R_i) = Oracle(R_i)$.

It is important to use subsamples of size $< \frac{1}{16\eta}$ in TDA because the probability of such a subsample containing a noisy sample is small.

Theorem 2 *Let $n = \lfloor \frac{1}{16(\eta + 3\epsilon/4)} \rfloor$, and let N be the size of a minimal $\frac{\epsilon}{2}$-cover of C. Let $l = \lceil 48 \frac{\eta + \epsilon/2}{\epsilon^2} \ln \frac{12N}{\delta} \rceil$. Let $s = \lceil (397 \ln \frac{96 \ln N}{\delta}) \rceil (4 \ln N) + \lceil 167 \frac{l}{n} \ln \frac{36l}{\delta} \rceil$, and $t = XTD(C, D, n, \frac{\delta}{2s})$. Then the number of labels queried in $C, D, \epsilon, \delta, \eta$ is $\leq ts = O(t(\frac{\eta^2}{\epsilon^2} + 1)(d \log \frac{1}{\epsilon} + log \frac{1}{\delta})(\log \frac{d}{\epsilon\delta}))$.* [5]

The theorem states that the bound is st; s is based on n, N, and l, and t is the extended teaching dimension. This implies that the upper bound for any concept class is based upon its extended teaching dimension.

The number of datapoints TDA requires is based upon the size of V where it is known that $|V| \leq N < 2(\frac{4e}{\epsilon} \ln \frac{4e}{\epsilon}^2)$ [11]. So, the number of datapoints is

$$m \leq \left\lceil 224 \frac{\eta + \epsilon/2}{\epsilon^2} \ln \frac{48 \ln |2(\frac{4e}{\epsilon} \ln \frac{4e}{\epsilon}^2)|}{\delta} \right\rceil (5 \ln |2(\frac{4e}{\epsilon} \ln \frac{4e}{\epsilon}^2)|).$$

The concept class of axis aligned rectangles is shown as an application whose XTD$(C, D, n, \delta) \leq O\left(\frac{n^2}{\lambda} \log \frac{nm}{\delta}\right)$. Results were not shown for any other concept classes; in particular the most common application in the active learning model as described in QBC and A^2 is not mentioned. Why wasn't this concept class examined? How does this algorithm compare to the other noisy model, A^2?

4.3 Linear Separators under the Uniform Distribution

Model	# of Datapoints	# of Labels Queried
QBC	$\mathrm{O}(\frac{d}{\epsilon}\log\frac{1}{\delta\epsilon})$	$O(\frac{d}{\epsilon})$
Modified Perceptron	$O(\frac{d}{\epsilon}\log\frac{1}{\epsilon})$	$O(d\log\frac{1}{\epsilon})$
A^2	$\frac{64}{\epsilon^2}\left(2d\ln(\frac{12}{\epsilon})+\ln(\frac{4}{\delta})\right)$	$O\left(d\left(d\ln d+\ln\frac{1}{\delta'}\right)\ln\frac{1}{\epsilon}\right)$
TDA	$\left\lceil 224\frac{\eta+\epsilon/2}{\epsilon^2}\ln\frac{48\ln\lvert 2(\frac{4e}{\epsilon}\ln\frac{4e}{\epsilon}^2)\rvert}{\delta}\right\rceil \times (5\ln\lvert 2(\frac{4e}{\epsilon}\ln\frac{4e}{\epsilon}^2)\rvert)$	$O>((\frac{2^d}{\sqrt{d}})(\frac{\eta^2}{\epsilon^2}+1)\times (d\log\frac{1}{\epsilon}+log\frac{1}{\delta})(\log\frac{d}{\epsilon\delta}))$

Table 1: Bounds for models with the class of linear separators under D

A common application analyzed is data drawn from the unit sphere in $\mathbb{R}^d$ where the labels are divided by a linear separator that goes through the origin of the sphere. The teaching dimension and active learning model did not examine this case which would be useful in analyzing the difference between the two noisy models. I will show the upper bounds for each algorithm with this application and provide an analysis of the two relative to eachother. Table 1 displays the bounds on datapoints and queries for each of the models mentioned that analyzed this classifier. QBC and Modified Perceptron use the Perceptron algorithm on the classifier to produce these bounds.

4.3.1 A^2

A^2 algorithm shows exponential improvements for the linear separator over the unit sphere. It is well-designed for this application due to the minimization that it does to the area of uncertainty.

Theorem 3 *Let $X, H, andD$ be as defined above, and let LB and UB be the VC bound. Then for any $0<\epsilon<\frac{1}{2}, 0<\eta<\frac{\epsilon}{16\sqrt{d}}$, and $\delta>0$, with probability*

$1 - \delta$, *the algorithm* A^2 *requires*

$$O\left(d\left(d\ln d + \ln\frac{1}{\delta'}\right)\ln\frac{1}{\epsilon}\right)$$

calls to the labeling oracle for linear separators under the uniform distribution, where $\delta' = \frac{\delta}{N^2(\epsilon,\delta,H)}$ *and* $N(\epsilon,\delta,H) = O\left(\ln\frac{1}{\epsilon}\left(d^2\ln d + \ln\frac{d\ln 1/\epsilon}{\delta}\right)\right)$. [4]

δ is based on $N(\epsilon,\delta,H)$ which is an upper bound on the number of bound (LB and UB) evaluations needed in the algorithm.

If H is a set of functions from X to $\{-1,1\}$ with finite VC Dimension, $d \geq 1$, then for any $\epsilon, \delta > 0$, the sample size required from D, an arbitrary but fixed probability distribution, is $\frac{64}{\epsilon^2}\left(2d\ln(\frac{12}{\epsilon}) + \ln(\frac{4}{\delta})\right)$. This is based upon standard sample complexity bounds from Anthony and Bartlett [12] and it implies with probability at least $1-\delta$ that $|err(h) - e\hat{r}r(h)| \leq \epsilon$ for all $h \in H$.

4.3.2 *TDA*

The number of queries required in the TDA model is based upon the exact teaching dimension of the concept class. This proposes some problems because regardless of the number of datapoints given by the teacher the separator cannot be learned. This is because there are infinitely many linear separators in $\mathbb{R}^d$. The concept space must be discretized to $X = \{0,1\}^d$ to produce some results.

Theorem 4 *Let XTD be as defined above and* $X = \{0,1\}^d$ *and no datapoints lie on the separator. The bound on the number of labels queried in* $C, D, \epsilon, \delta, \eta$ *for linear separators under the uniform distribution in* X *is* $> (\frac{2^d}{\sqrt{d}})(\frac{\eta^2}{\epsilon^2} + 1)(d\log\frac{1}{\epsilon} + log\frac{1}{\delta})(\log\frac{d}{\epsilon\delta})$

Proof: The teaching dimension of linearly separable functions is 2^d [8]. We are only concerned with linear separators through the origin which implies that we only need to be concerned about the datapoints that lie near the origin. It is possible to shift any one of those datapoints slightly without changing the others and thereby require a different linear separator. The teaching dimension is on the order of $\frac{2^d}{\sqrt{d}}$.[2] The XTD is even worse since it is a more restrictive case. However, the TD is poor in itself and it is therefore

[2]This value was received from Rocco Servedio

not necessary to find the XTD. The proof is therefore an approximate bounds based on substituting $t = XTD$ with TD and s as defined in Theorem 2. □

Based upon Theorem 4, it would not be a good idea to use TDA for linear separators under the uniform distribution. The poor bounds in this class were caused by the dependence of the TDA algorithm on the XTD. It can be assumed that if the XTD for a class is small that the algorithm would perform well and therefore be a good algorithm to use. It would appear that this was the reason that linear separators over the unit sphere was not examined in [5]. Hanneke shows the classifier of axis-aligned rectangles, which have a low XTD. (However, they required discreteness as well). In the class of linear separators under the uniform distribution, A^2 algorithm has un upper bound of significantly less queries and would therefore be a better choice.

5 Conclusion and Open Questions

I have described a number of active learning algorithms and analyzed and compared the A^2 and TDA algorithms. I have produced results that show that A^2 is better for linear separators over the unit sphere. Some open questions:

1. In order to fully analyze and compare the two algorithms, what bounds does A^2 have for axis-aligned rectangles? (the concept class shown using TDA)

2. TDA is useful since it is a general active learning and noisy model, however it does not do well in the setting analyzed by many other papers in this topic. Can TDA be altered so that it does not depend on the exact teaching dimension?

3. Can a general algorithm be written which would produce reasonable bounds for all the applications?

4. Can general bounds be made for A^2?

6 References

[1] Y. Freund, S. Seung, E. Shamir, and N. Tishby. (1997) Selective Sampling using the query by committee algorithm. *Machine Learning*, 28:133-168.
[2] S. Dasgupta, A. Kalai, and C. Monteleoni. (2005) Analysis of perceptron-based active learning. *COLT*.
[3] S. Dasgupta (2004) Analysis of a greedy active learning strategy. *NIPS*.
[4] M.-F. Balcan, A. Beygelzimer, J. Langford. (2006) Agnostic Active Learning. *Proc. of the 23rd International Conference on Machine Learning*.
[5] S. Hanneke. (2007) Teaching Dimension and the Complexity of Active Learning. *COLT*.
[6] T. Hegedüs. (1995) Generalised teaching dimensionand the query complexity of learning. *Proc. of the 8th Annual Conference on Computational Learning Theory*.
[7] S. A. Goldman and M. J. Kearns. (1995) On the complexity of teaching. *Journal of Computer and System Sciences*, 50:20-31.
[8] M. Anthony, G. Brightwell, J. Shawe-Taylor (1995) On specifying Boolean functions by labelled examples. *Discrete Applied Mathematics*, 61:1-25.
[9] D. Cohen, L. Atlas, & R. Ladner (1994). Improving generalization with active learning. Machine Learning, 15(2), 201-221.
[10] M. Kaarianen (2005). On active learning in the nonrealizable case. *NIPS Workshop on foundations of Active Learning*.
[11] D. Haussler (1992). Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100:78-150.
[12] M. Anthony & P. Bartlett (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press.