TTIC 31250 : Introduction to Theory of Machine Learning

Spring 2020

●

Final

●

Solutions by

Jinhong Du

dujinhong@uchicago.edu

12243476

Exercises

1. **Intervals.** Consider the class $\mathcal{C}$ of intervals on the real line $\mathbb{R}$. That is, a legal target function is specified by an interval $[a, b]$ and classifies an example x as positive if $x \in [a, b]$ and as negative otherwise. Give an algorithm to learn this class in the PAC model, and a proof that a training sample of size $O\left(\frac{1}{\epsilon}\log\frac{1}{\delta}\right)$ is sufficient for your algorithm to achieve error $\leq \epsilon$ with probability $\geq 1 - \delta$.

   The algorithm is given by

   (1) Receive a sample S with size n, sampled from distribution $\mathcal{D}$ and labeled by target function $f \in \mathcal{C}$.

   (2) Return $h(x) = \mathbb{1}_{[a,b]}(x)$ where $a = \min\{x|y = 1, (x, y) \in S\}$ and $b = \max\{x|y = 1, (x, y) \in S\}$.

   The hypothesis produced by this algorithm is consistent with S. Next we show this is a PAC model. Let $f(x) = \mathbb{1}_{[a_0,b_0]}(x)$ be the target function. The probablity of h has error at least ϵ on $\mathcal{D}$ is

$$\begin{aligned}
\mathbb{P}(err_{\mathcal{D}}(h) > \epsilon | err_{\mathcal{S}}(h) = 0) &\leq \mathbb{P}\left(a < a_0 - \epsilon | err_{\mathcal{S}}(h) = 0\right) + \mathbb{P}\left(a > a_0 + \epsilon | err_{\mathcal{S}}(h) = 0\right) \\
&\quad \mathbb{P}\left(b < b_0 - \epsilon | err_{\mathcal{S}}(h) = 0\right) + \mathbb{P}\left(b > b_0 + \epsilon | err_{\mathcal{S}}(h) = 0\right) \\
&= 4\mathbb{P}\left(a < a_0 - \epsilon | err_{\mathcal{S}}(h) = 0\right) \\
&\leq 4\mathbb{P}(x \notin [a_0 - \epsilon, a_0] \text{ for } x \in S) \\
&\leq 4 \prod_{x \in S} \mathbb{P}(x \notin [a_0 - \epsilon, a_0]) \\
&\leq 4(1 - \epsilon)^n \\
&\leq 4e^{-\epsilon n}
\end{aligned}$$

   Let $4e^{-\epsilon n} < \delta$, we have $n \geq \frac{1}{\epsilon}\log\frac{4}{\delta}$. So with sample size $O\left(\frac{1}{\epsilon}\log\frac{1}{\delta}\right)$, the algorithm achieve $err_{\mathcal{D}}(h) \leq \epsilon$ with probability at least $1 - \delta$.

2. **More on margins.** We didn't prove a uniform convergence result for large margin separators in class (e.g., a sample complexity bound for SVM). However, to get some intuition, prove that it is not possible to have a set S of $\frac{1}{\gamma^2} + 1$ points in the unit ball such that every labeling of S is achievable by a linear separator through the origin of margin at least γ. Hint: this has a one-sentence proof.

   An aside: the "ghost sample" proof we gave in class does not immediately give a uniform convergence result from this "VC-dimension-ish" statement, because, when the double-sample S'' is partitoned into S and S', there may be some large-margin labelings of S that do not extend to any large-margin labeling of S''. So a more involved proof is needed.

   Proof. Let S' be any set of points in the unit ball such that every labeling of S' are achievable by a linear separator through the origin of margin at least γ. Let d be the maximum cardinal numbers over all such S'. Then we run the Perceptron Algorithm on any S' with $|S'| = d$. We can change the order of presented points such that it makes d mistakes, while we still have a separator $y = w^{*\top}x$ of margin at least γ at the end. Since the number of mistakes the Perceptron Algorithm makes is at most $\frac{1}{\gamma^2}$, we have that $d \leq \frac{1}{\gamma^2}$. Therefore, linear separators through the origin of margin at least γ can shatter at most $\frac{1}{\gamma^2}$ points. In other word, it is not possible to have a set S of $\frac{1}{\gamma^2} + 1$ points in the unit ball such that every labeling of S is achievable by a linear separator through the origin of margin at least γ. □

3. **Models of learning.** Consider the following four learning models. In each of these we are given a class $\mathcal{C}$, and the goal of the learning algorithm is to *exactly* recover the target concept $c \in \mathcal{C}$.

   Equivalence Query: In this model the algorithm can propose a hypothesis h and is told either that h is correct, or else is given a counterexample x such that $h(x) \neq c(x)$. This is really the same as the Mistake-Bound model.

   Restricted Equivalence Query: Same as above except h must be from class $\mathcal{C}$.

   Membership Query Only: In this model the algorithm can propose examples x and is told their labels. But the algorithm is *not* allowed any mistakes / equivalence queries: it must exactly recover the target given only the membership queries.

   Teacher-directed: Like the membership query model, but now a teacher who knows the target function proposes the examples to be queried. The examples must *uniquely identify* the target concept, in the sense that no other $c' \in \mathcal{C}$ is consistent with this set of data. I.e., think of yourself as the teacher, trying to teach an adversarial (but consistent and proper) learning algorithm.

   For each class of functions below, state in which of the above models it can or cannot be learned/taught using a number of queries polynomial in n, along with a brief explanation.

   (a) The class of all functions over $\{0,1\}^n$ having *exactly one* positive example.

   The class of all functions over $\{0,1\}^n$ having exactly one positive example can be represented by $\mathcal{C}_1 = \{c(x) = \mathbb{1}_{\{x=y\}} | y \in \{0,1\}^n\}$. Suppose the target function is $c(x) = \mathbb{1}_{\{x=x_0\}}$ for $x_0 \in \{0,1\}^n$, then to recover c is equivalent to identify x_0.

   (1) For equivalence query, we initialize $h(x) = 0$, and since $h(x) \neq c(x)$ only when $x = x_0$, x_0 is revealed and c is recovered.

   (2) For the restricted equivalence query, we cannot learn c as h must be from $\mathcal{C}_1$, i.e. there exists $x_1 \in \{0,1\}^n$ such that $x_1 \neq x_0$ and $h(x_1) = 1$. So the oracle can always return x_1 and we need to exhaustive search $\{0,1\}^n$ until we reach x_0, which takes exponential time in n in general.

   (3) For the membership query, we cannot learn c for similar reasons as we need to exhaustive search $\{0,1\}^n$ until we reach x_0.

   (4) For the teacher-directed model, we can learn c as $\{(x_0, 1)\}$ is uniquely identify c and will be revealed.

   (b) The class of all functions over $\{0,1\}^n$ having *at most one* positive example.

   The class of all functions over $\{0,1\}^n$ having at most one positive example can be represented by $\mathcal{C}_2 = \mathcal{C}_1 \cup \{c(x) \equiv 0\}$.

   (1) For equivalence query, we initialize $h(x) = 0$ and make a query, if h is correct then we are done. Otherwise, $c \in \mathcal{C}_1$ and same argument in (a) applies. So we can learn $\mathcal{C}_2$.

   (2) For the restricted equivalence query, since now the zero function is in $\mathcal{C}_2$, this case is equivalent to (1), so we can learn $\mathcal{C}_2$.

   (3) For the membership query, we cannot learn c due to the same reason in (a) (3) if $c \neq 0$.

Solution (cont.)

(4) For the teacher-directed model, we cannot learn c since if $c = 0$, the set of examples that uniquely identify c is $\{(x, 0) | x \in \{0, 1\}^n\}$, which has more than poly(n) points.

(c) The class of monotone conjunctions over $\{0, 1\}^n$ (including the conjunction of nothing, which is always positive).

Let $\mathcal{C}_3$ be the class of monotone conjunctions over $\{0, 1\}^n$ (including the conjunction of nothing, which is always positive).

(1) For equivalence query, we initialize $h(x)$ to be the monotone conjunction of all variable and make a query, if h is correct then we are done. Otherwise, if we receive a counterexample $(x, 1)$, then we drop variables which are 0 in x. As the algorithm will make at most n mistakes, if all variables are dropped, then c is the conjunction of nothing. So we can learn $\mathcal{C}_3$.

(2) For the restricted equivalence query, we can learn $\mathcal{C}_3$ by the same algorithm in (1).

(3) For the membership query, we can learn $\mathcal{C}_3$ by querying examples e_i for $i = 1, \ldots, n$ where e_i is the unit vector with its ith entry being one, to see if the ith variable in c or not.

(4) For the teacher-directed model, we can learn c. The set of examples that uniquely identify c is $\{(e_i, c(e_i)) | i = 1, \ldots, n\}$, from which we know which variables are in c.

(d) The class of decision lists over $\{0, 1\}^n$.

Let $\mathcal{C}_4$ be the class of decision lists over $\{0, 1\}^n$.

(1) For equivalence query, we can learn $\mathcal{C}_4$ as we have showed in Homework 1 Problem 4 by other class of functions.

(2) For the restricted equivalence query, we cannot learn $\mathcal{C}_4$ by $\mathcal{C}_4$ since the presence of counterexamples can be adversarial to the algorithm. It is hard for any algorithm to learn how to select and sort the if-else terms in polynomial time.

(3) For the membership query, we cannot learn $\mathcal{C}_4$. The reason is the same as in (2).

(4) For the teacher-directed model, we cannot learn $\mathcal{C}_4$. The reason is the same as in (2).

4. **XOR of Conjunctions.** Consider the class $\mathcal{C}$ of "XORs of two conjunctions". This is the class of functions f that can be described as $f(x) = T_1(x) \oplus T_2(x)$ where T_1 and T_2 are conjunctions, and "$\oplus$" is XOR; so, $f(x)$ is positive if x satisfies *exactly one* of T_1 or T_2. For instance, the following is an XOR of two conjunctions:

$$x_1 x_3 \overline{x}_5 \oplus x_2 x_4.$$

This function is positive on examples 11100 and 11011, and is negative on examples 00000 and 11110.

Give an algorithm that learns this class $\mathcal{C}$ over $\{0, 1\}^n$ in the mistake-bound model. Your algorithm should have a mistake bound polynomial in n and should be efficient (running in polynomial time per example) too. Your algorithm need *not* use $\mathcal{C}$ as its hypothesis representation. If you get stuck, then for partial credit give an *inefficient* algorithm.

(Inefficient Algorithm) If we don't care computation time, we can use the Halving Algorithm:

(1) Let $\mathcal{H} = \mathcal{C}$ where at least one consistent hypothesis h exists. $|\mathcal{C}| = O(2^{2n+1})$.

(2) Predict with the majority of the $|\mathcal{H}|$ consistent experts in the class $\mathcal{H}$ until we make a mistake.

(3) At least half of the experts in $\mathcal{H}$ will also make a mistake; remove them from the class and renew the class $\mathcal{H}$.

(4) Repeat the above procedure (2)-(3).

The algorithm makes at most $\log_2(|\mathcal{C}|) = O(2n+1)$ mistakes. This gives a inefficient algorithm.
(Efficient Algorithm)
First we consider three cases:

(1) If a variable x_i is in both T_1 and T_2, suppose that $T_1(x) = x_i T_1'(x), T_2(x) = x_i T_2'(x)$, then $T_1(x) \oplus T_2(x) = x_i \wedge (T_1'(x) \oplus T_2(x))$ which is a decision list.

(2) If a variable x_i is in T_1 and $\overline{x}_i$ is in T_2, suppose that $T_1(x) = x_i T_1'(x), T_2(x) = \overline{x}_i T_2'(x)$, then $T_1(x) \oplus T_2(x)$ is equivalent to $\texttt{if } x_i = 1 \texttt{ then } T_1(x) \texttt{ else } T_2(x)$, which is also a decision list.

(3) If the above two cases are not satisfied, then the variables in T_1 and T_2 are different (ignoring negation), this is equivalent to a 2-layer decision tree with root T_1 and same conditions T_2 in the second layer.

So an XOR-AND can be expressed as a decision tree with rank at most n. My guess is that we can first deal with the first two cases, for example, the algorithm should

(a) Decide whether case (2) is possible, if yes, our task becomes learning two conjunctions.

(b) If no, look at case (1), if yes, we can reduce number of variables in T_1 and T_2.

(c) If no, then consider case (3), now T_1 and T_2 are uncorrelated, we are able to use list-and-cross type algorithm to identify the target function.

But sorry I have no enough time to finish it.

5. **Course evaluation.** Please fill out the online course evaluation and write down "Done". (These are totally anonymous, so you get full credit for writing "Done" without actually doing the evaluation, but please do the evaluation it's quite useful for us.) You may give yourself an extra 15 minutes in addition to the allotted 24 hours to complete this question.

Done.