COMS 6998-4 Fall 2017 Presenter: Che Shen
October 11, 2017 Scribe: Arushi Gupta

Queries in Active Learning

1 Introduction

Recall active learning. In active learning we had a distribution, $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space. We focused on the case where $\mathcal{Y}$ is $\{0, 1\}$. Our set of hypotheses, $\mathcal{H}$ is a mapping from $\mathcal{X}$ to $\mathcal{Y}$. The error of a particular hypothesis is

$$\text{err}(h) = P_{(x,y)\sim\mathcal{D}}[h(x) \neq y]. \tag{1}$$

At each time step we observe $x_t \in \mathcal{X}$ and decide whether or not to query its label with the ultimate goal of returning a hypothesis $\hat{h} \in \mathcal{H}$ such that

$$\text{err}(\hat{h}) \leq \epsilon. \tag{2}$$

Notice that in the active learning framework we looked at before, there was a presumed probability distribution, $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. We assume that the learner receives points sampled from $\mathcal{D}_{\mathcal{X}}$, the marginal of $\mathcal{D}$ over $\mathcal{X}$, and we evaluate the error of the model assuming that points are drawn from $\mathcal{D}$. We also assume that upon receiving a point from $\mathcal{D}_{\mathcal{X}}$, all we may do is choose whether or not to view its label. It would be interesting, however to see what happens when these assumptions are challenged.

2 Extending the active learning framework to the deterministic setting

Our hypotheses will be given similarly as before, as a mapping $h : \mathcal{X} \to \mathcal{Y}$. We will focus on the case where $\mathcal{Y}$ is $\{0, 1\}$. We will assume that our hypothesis class $\mathcal{H}$ is finite, and contains hypotheses $h_1, \ldots, h_n$. We are trying to learn a target hypothesis $h^* \in \mathcal{H}$. We are allowed two query methods, Membership (also called Label) and Equivalence. Membership takes as input some $x \in \mathcal{X}$ and returns its label $h^*(x)$. Equivalent takes as input $\hat{h}$ and returns $\emptyset$ if $\hat{h} = h^*$. If not, it may either be the restricted version of Equivalent, in which case it simply returns False, or it may be the unrestricted version, in which case it returns a counter example, x_0 such that $h(x_0) \neq h^*(x_0)$. Now, for the equivalence queries, we may or may not require $\hat{h} \in \mathcal{H}$, and this will change our querying strategy.

3 Unrestricted Equivalence queries and $\hat{h}$ any function from $\mathcal{X}$ to $\{0, 1\}$

3.1 The naive approach

The naive method would be to use an exhaustive search. That is, for each hypothesis in our hypothesis class, you would run an equivalence query until Equivalent returns “yes”. You would need n-1 queries in the worst case for this.

3.2 Majority Vote

We let

$$\hat{h} = \begin{cases} 1 & \text{if } \sum_{h \in \mathcal{H}} h(x) \geq |\mathcal{H}|/2 \\ 0 & \text{o.w.} \end{cases} \tag{3}$$

(this means that $\hat{h}(x)$ is 1 if half or more of the hypotheses in $\mathcal{H}$ agree that it is 1). We can make an equivalence query with $\hat{h}$. Since we are in the unrestricted equivalence query case, our equivalence query will return a counter example, x_0. Because of the way we designed $\hat{h}$, at least half of the hypotheses in $\mathcal{H}$ will have $h(x_0) = \hat{h}(x_0)$ and we may eliminate these hypotheses. Therefore, we may eliminate at least half of the hypotheses in $\mathcal{H}$ every time, so the algorithm halts after at most $\log n$ rounds.

We wish to consider the case where we cannot query any function $\hat{h}$ that we would like, but first we discuss the relationship between active and online learning.

4 Recall online learning

We have a set $\mathcal{X}$ and a function $h^* : \mathcal{X} \to \{0, 1\}$. The learner has a finite hypothesis space $\mathcal{H} = \{h_1, \ldots, h_n\}$ which contains h^*.

Algorithm 1 Online learning

```
input Training data $((x_1, y_1), (x_2, y_2), \ldots)$.
1: for $t = 1, 2, \ldots$ do
2:     Teacher provides $x_t \in \mathcal{X}$.
3:     Learner predicts $a_t \in \{0, 1\}$.
4:     Teacher reveals $y_t = h^*(x_t)$.
5: end for
```

The goal of the algorithm is to make as few prediction errors as possible. We saw earlier that the halving algorithm is an example of an online learning algorithm.

5 Equivalence of online and active learning

We wish to compare online and active learning to see if they are equivalent problems. The table summarizes some of the goals and operations of online and active learning.

	Online learning	Active learning
Known	set $\mathcal{X}$ and hypothesis space $\mathcal{H}$	set $\mathcal{X}$ and hypothesis space $\mathcal{H}$
Operation	predict y_t for x_t and get feedback	predict $\hat{h}$ and get feedback
Goal	make few mistakes	make few queries

5.1 From active learning to online learning

We want to know if given an algorithm that solves active learning, can we construct an algorithm that performs just as well to solve online learning. More formally, suppose we have an algorithm $\mathcal{A}$ that makes less than M queries in the active learning problem. Can we use it to construct an algorithm $\mathcal{B}$ that makes less than M mistakes in online learning?

1. $\mathcal{A}$ makes an equivalence query with $\hat{h}$
2. $\mathcal{B}$ makes every prediction with $\hat{h}$, i.e. predict the label of x_t to be $\hat{h}(x_t)$ until it makes at least 1 mistake on (x', y')
3. $\mathcal{B}$ presents (x', y') to $\mathcal{A}$ as a counter example.
4. Loop to the first step.

Notice that $\mathcal{A}$ makes $\leq M$ mistakes by assumption, so it will propose at most M incorrect $\hat{h}$'s. $\mathcal{B}$ will run each of these wrong $\hat{h}$s until it finds it's made an error, at which point it gives this incorrect point back to $\mathcal{A}$ as a counter example.

5.2 From online learning to active learning

Suppose we have an algorithm $\mathcal{B}$ that makes less than M mistakes in the online learning problem. Can we use it to construct an algorithm $\mathcal{A}$ that makes less than M queries in active learning.

1. $\mathcal{A}$ makes m copies of the current state of $\mathcal{B}$ and feeds them with $x_1, \ldots, x_m$ and observes the output $\mathcal{B}(x_1), \ldots, \mathcal{B}(x_m)$
2. $\mathcal{A}$ makes an equivalence query with $\hat{h}$ such that $\hat{h}(x_i) = \mathcal{B}(x_i)$ and receives the counter example (x', y')
3. $\mathcal{A}$ chooses the copy of $\mathcal{B}$ that is fed with x' and reveals y' to it.
4. Loop until $\mathcal{A}$ gets the answer "yes".

Here, the active learning algorithm runs m copies of the online learning algorithm. It feeds 1 distinct data point to each of these m copies, and then makes an equivalence query with $\hat{h}$, such that $\hat{h}(x_i) = \mathcal{B}(x_i)$. If there's an example that $\mathcal{B}$ gets wrong, it is allowed to see the label of the point that it made a mistake on. Since $\mathcal{B}$ only makes M mistakes, $\mathcal{A}$ can only make M mistakes.

6 Lower bound in the case where hypotheses given to equivalent queries must be a member of $\mathcal{H}$

In earlier section we saw how majority vote may be used to achieve a $\log n$ mistake rate when we are allowed to run an equivalence query on any function h we like and observe counter examples. We now consider the case where $\hat{h}$ must be in $\mathcal{H}$.

We switch from the language of hypotheses to the language of sets for the following lemma. There are sets $L_1, L_2, \ldots$ which are subsets of some universal space, and we are trying to identify L^*.

Lemma 1. *Suppose the hypothesis space contains a class of distinct sets $L_1, \ldots, L_n$ and there exists a set $L_\cap$ which is not a hypothesis such that for any pair of distinct indices i and j*

$$L_i \cap L_j = L_\cap. \tag{4}$$

Then any algorithm that exactly identifies each of the hypotheses L_i using equivalence and Membership queries must make at least $n - 1$ queries in the worst case.

Proof. This can be done with an adversarial labeling. Start with a set S containing all hypotheses. For a restricted equivalence query the answer is "no", and since we are only allowed to query hypotheses the corresponding hypothesis is removed from S. For a membership query is $x_i \in L_\cap$ the answer is "yes". Otherwise the answer is "no" and the L_i that contains x_i is removed if it is in S. □

We will consider the example of the k-CNF and k-DNF formulas and show they can be efficiently identified using equivalence queries .

6.1 k-CNF and k-DNF

A k-CNF formula is a formula in conjunctive normal form over n variables $x_1, \ldots, x_n$ with at most k literals per clause. For example,

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \tag{5}$$

is a 3-CNF in 4 variables. k-CNF formulas provide a natural hypothesis space over $\{0, 1\}^n$.

6.1.1 1-CNF

Consider the class of 1-CNF formulas of the form $P_1 \wedge P_2 \wedge \cdots \wedge P_n$ where each P_i is either x_i or $\neg x_i$. There are 2^n such formulas, so our hypothesis class contains 2^n elements. Each 1-CNF formula is satisfied by exactly one assignment and no two formulas are satisfied by the same assignment. According to the previous lemma, any algorithm that exactly identifies these 1-CNF formulas using equivalence and Membership queries must make $2^n - 1$ queries in the worst case.

6.1.2 Identification of k-CNF

Proposition 1. *There is an algorithm that achieves exact identification of k-CNF formulas using unrestricted equivalence queries and the number of equivalence queries is polynomial in* n^k.

We show that this is true by constructing an adversarial labeling. Initially, let ϕ be the conjunction of all clauses C over n variables with at most k literals per clause. There are no more than $(2n+1)^k$ of these. Test ϕ for equivalence with ϕ^*. If not equivalent, change ϕ according to the counter example. Since a clause is removed each time a counterexample is seen, the number of equivalence queries is bounded by $(2n+1)^k$.

We will now consider an example Consider a 2-CNF in 3 variables

$$\phi^* = (x_1 \vee x_2) \wedge (x_1 \vee x_3) \tag{6}$$

Query with	Return
$(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee x_3)$	(1,0,0)
$(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\neg x_2 \vee x_3)$	(0,1,1)
$(x_1 \vee x_2) \wedge (x_1 \vee x_3)$	yes

7 Extending equivalence query to the stochastic setting

Looking back at active learning, we see that we are using Membership queries in the stochastic setting. However, in the case where some class examples are rare, we could need lots of Membership queries to even find 1 example of a data point which belongs to the rare class. Equivalence queries in the stochastic setting may return counter examples that are not interesting and do not help us narrow down our hypotheses. We define a new oracle called search which returns counter examples to version spaces.

Assume again that there is some distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. We are learning with a sequence of nested hypothesis classes $H_0 \subset H_1 \subset \cdots \subset H_k \cdots$. Let $H = \cup_{i=0}^{\infty} H_k$.

```
SEARCH

input: Set of hypotheses  $V \subset H$
```

```
output: labeled examples $(x,\ h*(x))$ s.t. $h(x) \neq h*(x)\ \forall\ h \in V$
or $\emptyset$ if there is no such example
```

Thus SEARCH returns a counter example such that all hypotheses in V disagree, that is, it returns a counterexample that is a systematic mistake for the hypotheses in V.

Proposition 2. *For any call $x \in X$ such that $LABEL(x) = h^*(x)$, we can construct a call to SEARCH that achieves a no lesser reduction in the region of disagreement.*

Proof. Let $V_x = H_{+1}(x) = \{h \in \mathcal{H} \text{ s.t. } h(x) = +1\}$. and let $H_{SEARCH}(V)$ be the hypothesis in H consistent with the output of $SEARCH_H(V)$. If $h^*(x) = +1$. then $SEARCH_H(V_x)$ returns $\emptyset$. In this case, the reduction in the region of disagreement is the same regardless of whether we use SEARCH or LABEL. If $h^*(x) = -1$ then $SEARCH(V_x)$ returns a valid counter example and eliminates all of $H_{+1}(x)$. Thus $H_{SEARCH}(V_x) \subset H_{LABEL}(x)$ □

7.1 LARCH algorithm

Let $H_k(S) = \{h \in H_k : \forall(x, y) \in S, h(x) = y\}$. The pseudocode for the LARCH algorithm is included. LARCH combines LABEL and SEARCH. LARCH first calls SEARCH in Step 3. If SEARCH returns $\emptyset$ SEARCH checks to see if the last call to CAL resulted in a small enough error, halting if so in Step 6, and decreasing the allowed error rate if not in Step 8. If SEARCH returns a counter example the complexity of the hypothesis class is increased. One interesting property of LARCH is that $H_k(S) \subset H_k$ may have a version space that is more easily active learned than H_k itself, since it may have a markedly smaller disagreement coefficient. We give a quick definition of the disagreement coefficient here:

The region of disagreement for V is defined to be

$$DIS(V) = \{x \in \mathcal{X} : \exists h_1, h_2 \in V s.t. h_1(x) \neq h_2(x)\}, \tag{7}$$

and let

$$B(h, r) = \{h' \in \mathbb{C} : P[h(X) \neq h'(X) \leq r]\}. \tag{8}$$

Then the disagreement coefficient of h with respect to r is

$$\theta_h = \sup_{r>0} \frac{P[DIS(B(h, r))]}{r}. \tag{9}$$

Algorithm 2 LARCH algorithm

```
input Nested hypothesis classes $H_0 \subset H_1 \subset \cdots$, oracles LABEL and SEARCH ; learning
    parameters $\epsilon, \delta \in (0,1)$
1: initialize $S \leftarrow \emptyset$ $k = 0$, $l = 0$
2: for $i = 1, 2, \ldots$ do
3:    $e \leftarrow SEARCH_{H_k}(H_k(S))$
4:    if $e = \perp$ then
5:       if $2^{-l} \leq \epsilon$ then
6:          return any $h \in H_k(S)$
7:       else
8:          $l \leftarrow l + 1$
9:       end if
10:   else
11:      $S = S \cup e$
         $k \leftarrow \min\{k' : H_{k'}(S) \neq \emptyset\}$
12:   end if
13:   $S \leftarrow S \cup CAL(H_k(S), LABEL, 2^{-l}, \delta/(i^2 + i))$
14: end for
```

7.2 Union intervals example

Let $S_{[k']}$ be the set of labeled examples S in LARCH at the first time that $k \geq k'$. let $\theta_{k'}(\cdot)$ be the disagreement coefficient of $H_{k'}(S_{[k']})$. Let $X = [0,1]$. The hypothesis h^* is the union of intervals in X assuming that D_X is uniform. For $k \geq 0$ let H_k be the hypothesis class of the union of up to k intervals in $[0,1]$ (H_0 containing only the always negative hypothesis). $\mathcal{H} = \cup_{i=0}^{\infty} H_i$. Thus h^* is the union of k^* intervals.

Now, we know that the disagreement coefficient of H_1 is $\omega(1/\epsilon)$. This is true because we can choose the target function as the empty interval and thepick $1/\epsilon'$ disjoint closed intervals on the real line with probability mass $\geq \epsilon$ and let the h_i be the hypothesis which take value 1 on these intervals. Hence LABEL-only active learning algorithms like CAL are not very effective. The first query by LARCH provides a counterexample to H_0 denoted by $(x_1, +1)$ (note it must have a positive label since H_0 was the null hypothesis). Hence $H_1(S_{[1]})$ is the class of intervals that contain x_1 with disagreement coefficient $\theta_1 \leq 4$. The 4 comes from the fact that for any $h_[a,b]$ the hypotheses in $B(h,r)$ must have their starts and ends within r of a and b respectively. What do we know about θ_k?. Before LARCH advances its index to value k, S must already contain at least k positive examples and $k-1$ negative examples separating them. The disagreement coefficient of H_k is O(k) so Theorem 1 implies that with high probability LARCH makes at most $k^* + \log(1/\epsilon)$ queries to SEARCH and $\tilde{O}((k^*)^3 \log(1/\epsilon) + (k^*)^2 \log^3(1/\epsilon))$ queries to LABEL.

Theorem 1. *Assume that* $\text{err}(h^*) = 0$. *For each* $k' \geq 0$ *let* $\theta_{k'}(\cdot)$ *be the disagreement* *coefficient of $H_{k'}(S_{[k']})$ where $S_{[k']}$ is the set of labeled examples in S in LARCH at the first time that $k \geq k'$. Fix any $\epsilon, \delta \in (0,1)$. If LARCH runs with inputs hypothesis classes $\{H_k\}_{k=0}^{\infty}$, oracles LABEL and SEARCH then w.p. at least $1-\delta$ LARCH halts after at most $k^* + \log_2(1/\epsilon)$ for-loop iterations and returns a classifier with error rate at most ϵ. Furthermore, it draw at most $\tilde{O}(k^* d_{k^*}/\epsilon)$ unlabeled examples from D_X, makes at most $k^* + \log_2(1/\epsilon)$ queries to SEARCH and at most $\tilde{O}(k^* + \log(1/\epsilon)) \cdot (\max_{k' \leq k^*} \theta_{k'}(\epsilon)) d_{k^*} \cdot log^2(1/\epsilon)$ queries to LABEL*

Proof. By union bound, there is an event with probability at least $1-\delta$ such that each call to CAL made by LARCH satisfies the high probability guarantee. When the algorithm halts, it returns a hypothesis $h \in H_k(S)$. Then

$$P_{(x,y)\sim D}[h(x) \neq y \wedge x \in Dis(H_k(S))] \leq \epsilon. \tag{10}$$

For the number of for loop iterations, note that in each loop either k_i or l is incremented. $k_i \leq k^*$, $I \leq \log_2(1/\epsilon)$. For the number of queries to search and label note that $SEARCH \leq k^* + log_2(1/\epsilon)$. In each loop, CAL makes label queries $\tilde{O}(\theta_{k_i}(\epsilon) \cdot d_{k_i} \cdot I_i^2 \cdot polylog(i))$. Taking the sum over all iterations we get the result. □

References

[1] Dana Angluin. Concept learning, queries, supervised learning *Machine Learning*, 2(4):319–342, 1988.

[2] Alina Beygelzimer , Daniel J. Hsu , John Langford and Chicheng Zhang Search Improves Label for Active Learning *NIPS* 29:3342-3350, 2016.