COMS 6998-4 Fall 2017 Presenter: Che Shen
October 11, 2017 Scribe: Fei Peng

Query in active learning

1 Introduction

Active learning is the process of learning a hypothesis based on the interaction between a learner and an instructor. An active learner sends a series of queries to an oracle to obtain information about the target hypothesis and eliminate wrong hypotheses.

Recall the setting of active learning. Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the input space and $\mathcal{Y} = \{\pm 1\}$ are the possible labels. (X, Y) is a pair of random variables drawn from $\mathcal{D}$. Let $\mathcal{H}$ be a set of hypotheses mapping from $\mathcal{X}$ to $\mathcal{Y}$. The error of a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ is

$$\text{err}(h) := \Pr(h(X) \neq Y)$$

Let $h^* := \arg\min_{h \in \mathcal{H}} \text{err}(h)$ be a hypothesis of minimum error in $\mathcal{H}$, the goal of active learning is to find a hypothesis $\hat{h} \in \mathcal{H}$ with error $\text{err}(\hat{h})$ close to $\text{err}(h^*)$. In realizable case, we assume that there exists a hypothesis that makes no mistake on all the possible input $x \in \mathcal{X}$ and thus $\text{err}(h^*) = 0$.

An algorithm for active learning proceeds by iteratively drawing a (X, Y) pair from $\mathcal{D}$ where the true label Y is hidden and deciding whether to ask the oracle for Y. A good algorithm should find the target hypothesis and also minimize the number of unlabeled points and queries.

In this lecture, we will see how query is applied in active learning.

2 Deterministic Setting

In deterministic setting, we have an input space $\mathcal{X}$ and a target hypothesis $h^* : \mathcal{X} \to \{0, 1\}$. We assume that the hypothesis space is finite $\mathcal{H} = \{h_1, h_2, ..., h_n\}$ and $h^* \in \mathcal{H}$. The learner can send two types of queries to an oracle:

- Membership: The input is any $x \in \mathcal{X}$ and the output is its label $h^*(x)$
- Equivalence: The input is a hypothesis $\hat{h}$. The output is *yes* if $h^* = \hat{h}$. Otherwise, answer *no* with a counterexample $x \in \mathcal{X}$ s.t. $h^*(x) \neq \hat{h}(x)$

Since we assume $h^* \in \mathcal{H}$, the algorithm will always exactly identify the target hypothesis by seeing enough examples and eliminating all the wrong hypotheses.

Notice that the response of an equivalence query will include a counterexample if the input hypothesis is not the same as h^*. Such an equivalence query is called unrestricted equivalence query. Also, there exists restricted version which only answers yes or no without

any counterexample. In this lecture, we only consider unrestricted equivalence query since the counterexample provides important information about target hypothesis and help reduce complexity.

The choice of counterexample also affects the performance of active learning. A good strategy to properly choose counterexample may result in a more efficient algorithm. Here we assume the selection of a counterexample is arbitrary and consider the worst case.

Next, we will consider two cases. The input $\hat{h}$ of the equivalence query can be any function $f : \mathcal{X} \to \{0, 1\}$ or is limited to be an element of $\mathcal{H}$, i.e. $\hat{h} \in \mathcal{H}$.

2.1 Case 1: Input can be any function

We first consider the case where the input of equivalence query can be any function $f : \mathcal{X} \to \{0, 1\}$. A naive method to find the target hypothesis is to conduct exhaustive searching. Recall we have a finite hypothesis space $\mathcal{H}$, every time the algorithm chooses a candidate hypothesis $h \in \mathcal{H}$ and sends an equivalence query. If the answer is *no*, then h will be removed from hypothesis set. Thus at least one hypothesis can be removed in each iteration until the target hypothesis is found. In the worst case, the algorithm will make $|\mathcal{H}| - 1$ queries.

Actually we can do this better using Majority Vote. Construct a fake hypothesis $\hat{h}$ that may not exist in $\mathcal{H}$

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

In each iteration, the algorithm constructs $\hat{h}$ and sends an equivalence query. Every query answered *no* must reduce the cardinality of $\mathcal{H}$ by at least one half. Thus, $\log(|\mathcal{H}|)$ queries suffice for the algorithm to find h^*.

Notice that exhaustive search and Majority vote algorithm are exactly the same as the consistent algorithm and halving algorithm introduced in online learning. Online learning and active learning are actually equivalent. Since this is not relevant to what we will discuss in the next section, the proof will be described in Appendix A.

2.2 Case 2: Input must be a member of $\mathcal{H}$

Now consider the setting that the input of equivalence query must be a member of the hypothesis space $\mathcal{H}$. Exhaustive search still works, but Majority vote algorithm is not applicable because the fake hypothesis $\hat{h}$ may not exist in the hypothesis space. Angluin, 1988 [1] gives the lower bound of queries.

Lemma 1. *Suppose the hypothesis space contains a class of distinct sets L_1, L_2, ..., L_n and there exists a set $L_\cap$, such that for any pair of distinct indices i and j,*

$$L_i \cap L_j = L_\cap$$

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

Proof. Suppose the target hypothesis is L^*. In the worst case, the algorithm cannot find the target hypothesis by luckily choosing L^* or $x \in L^*$. For an equivalence query with the hypothesis $L \neq L^*$, the answer is *no* and L will be removed. For a membership query with an element $x \in L_\cap$, the reply is *yes*. Otherwise, the reply is *no*. If $x \in L_\cap$, then the hypothesis set has no change because $x \in L_i$ for any i. If $x \in L_i \backslash L_\cap$, then $x \notin L_j$ for any $j \neq i$. So only L_i will be removed.

Each query removes at most one hypothesis, so $n-1$ queries are required in the worst case, which proves Lemma 1. □

Lemma 1 can be strengthened if $L_\cap$ is not a hypothesis.

Lemma 2. *Suppose the hypothesis space contains a class of distinct sets L_1, L_2, ..., 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,*

$$L_i \cap L_j = L_\cap$$

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.

We now show that k-CNF formulas can be identified using queries. Let k-CNF denote the set of formulas in conjunctive normal form over n variables x_1,x_2,...,x_n with at most k literals per clause. A literal refers to x or $\neg x$ for any variable x. k-CNF has a natural hypothesis space over $\{0,1\}^n$.

First consider a simple case that $k = 1$ and each variable appears only once. The hypothesis of $\mathcal{H}$ has the form of

$$P_1 \wedge P_2 \wedge ... \wedge P_n$$

where P_i is either x_i or $\neg x_i$.

Obviously, $|\mathcal{H}| = 2^n$. Each 1-CNF formula is satisfied by exactly one assignment and any two formulas cannot be satisfied by the same assignment. According to Lemma 1, any algorithm that exactly identifies these 1-CNF formulas using restricted equivalence and membership queries must make at least $2^n - 1$ queries in the worst case.

For the general cases where $k \geq 1$ and unrestricted equivalence query is allowed, we have the following proposition.

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

Proof. Initially let ϕ be the conjunction of all clauses C over n variables with at most k literals per clause. There are at most $(2n+1)^k$ such clauses. All the clauses in the target formula ϕ^* must exist in ϕ. The goal of the algorithm is to remove all the incorrect clauses from ϕ.

Send an equivalence query with ϕ. If not equivalent, a counterexample x will be returned. Since $\phi^*(x) = \text{true}$, we can remove all the clauses in ϕ that are not satisfied by assignment x. Each query removes at least one clause, so the number of equivalence queries is bounded by $(2n+1)^k$ which is polynomial in n^k. □

3 Stochastic Setting

Equivalence query sometimes may be inefficient because it has to ensure the input hypothesis predicts the same result as h^* for any $x \in \mathcal{X}$. Suppose x is randomly drawn from a hidden distribution $\mathcal{D}$, then it is impossible to cover all the possible input values. Thus, instead of exactly identifying the target hypothesis, we wish to find an approximate hypothesis that has a large probability of predicting the true label $h^*(x)$ for any $x \in \mathcal{X}$. Another problem of equivalence query is that the counterexample from the oracle may provide no useful information. So we want to find a natural way to restrict the searchable space to a set of good counterexamples instead of returning an arbitrary one.

One way to do stochastic equivalence checking is to randomly draw a set of examples from $\mathcal{D}$, send membership queries to the oracle and check whether the candidate hypothesis $\hat{h}$ is consistent with all these samples. If *yes*, then we may conclude that $\hat{h}$ is close to h^*. However, this method suffers from class imbalance problem, since finding an example from the rare class will be really difficult and also increase label complexity.

In this section, we will introduce a new oracle SEARCH proposed by Beygelzimer, Hsu, Langford, Zhang, 2016 [2] and then reveal its advantage through a union-intervals example. We consider learning with a nested sequence of hypotheses classes $H_0 \subset H_1 \subset ... \subset H_k \subset ...$, where $H_k \subseteq \mathcal{Y}^{\mathcal{X}}$ has VC dimension d_k. For a set of labeled samples $S \subset \mathcal{X} \times \mathcal{Y}$, let $H_k(S) := \{h \in H_k : \forall (x,y) \in S, h(x) = y\}$ be the set of hypotheses in H_k consistent with S. Let $\text{err}(h) := \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$ denote the error of hypothesis h with respect to distribution $\mathcal{D}$, and $\text{err}(h, S)$ be the error rate of h on the labeled example of S. Let $h_k^* = \arg\min_{h \in H_k} \text{err}(h)$ and $k^* = \arg\min_{k \geq 0} \text{err}(h_k^*)$. For simplicity, we assume the minimum is attained at some finite k^*. Finally, we define $h^* := h_{k^*}^*$, the optimal hypothesis in the sequence of classes. The goal of the learner is to learn a hypothesis which has a error rate not much more than that of h^*.

3.1 Oracle

We introduce two oracles LABEL and SEARCH. LABEL is the same as the membership query. Given an input $x \in \mathcal{X}$, LABEL returns its true label $h^*(x)$. SEARCH is similar to equivalence query but has some difference.

The input of SEARCH is a version space V. SEARCH checks the predictions of all the hypotheses in V. If there exists a specific input x for which all the hypotheses in V predict the wrong label, then SEARCH will return x as a counterexample. Here x is a systematic mistake

Oracle 1: $\textsc{Search}_H(V)$ (where $H \in \{H_k\}_{k=0}^{\infty}$)

input : Set of hypothesis $V \subset H$

output : Labeled example $(x, h^*(x))$ s.t. $h(x) \neq h^*(x)$ for all $h \in V$, or $\perp$ if there is no such example.

which results from the complexity of current model. It implies that current hypotheses are too simple and the learning algorithm should consider more complicated model. i.e. hypothesis in H_{k+1}

Now we consider the connection between these two oracles. Take learning threshold classifiers as an example, we'll show that Search is as effective as Label in reducing the region of disagreement.

The hypothesis set is $H := \{h_w : w \in [0,1]\}$ on the input space $\mathcal{X} = [0,1]$. In the realizable case,

$$h_w(x) = \begin{cases} +1 & \text{if } w \leq x \leq 1 \\ -1 & \text{if } 0 \leq x \leq w \end{cases}$$

A learner with only the Label oracle can achieve logarithmic query complexity by binary search.

Assume that binary search starts with querying the label of x. Let $V_x = H_{+1}(x) = \{h \in H : h(x) = 1\}$. We can query $\textsc{Search}(V_x)$. If Search returns $\perp$, there is no counterexample $(x', -1)$ such that $x' \geq x$ and $h^*(x') = -1$, which means the target threshold $w^* \leq x$. So we can reduce the region of disagreement to $[0, x)$. Else if Search returns a counterexample $(x', -1)$, it means $w^* > x'$, so we can reduce the region of disagreement to $(x', 1]$. As a result, Search can reduce the region of disagreement no less than Label and achieve logarithmic query complexity without getting the label of a particular point.

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

Proof. Let $V_x := H_{+1}(x) := \{h \in H : h(x) = +1\}$. We will show that any call to Label can be transformed to a call to Search with V_x. For simplicity, assume that there is no systematic mistake. Let $H_{\textsc{Label}}(x)$ denote the candidate hypotheses after calling $\textsc{Label}(x)$ and $H_{\textsc{Search}}(S)$ the candidate hypotheses after calling $\textsc{Search}(S)$.

There are two cases: If $h^*(x) = +1$, then a call to Label will return $(x, 1)$ which will remove all the hypotheses that predict -1. $H_{\textsc{Label}}(x) = V_x$. We can do the same thing by calling $\textsc{Search}(V_x)$. Because all the hypotheses in V_x predict the correct label for x, $\textsc{Search}(V_x)$ returns $\perp$ and $H \backslash V_x$ is removed. So $H_{\textsc{Search}}(V_x) = V_x = H_{\textsc{Label}}(x)$. If $h^*(x) = -1$, $H_{\textsc{Label}}(x) = H \backslash V_x$. $\textsc{Search}(V_x)$ returns a counterexample $(x, -1)$ eliminating V_x. So $H_{\textsc{Search}}(V_x) = H \backslash V_x = H_{\textsc{Label}}(x)$.

In both cases, we have $H_{\textsc{Search}}(V_x) \subseteq H_{\textsc{Label}}(x)$ and hence $\text{Dis}(H_{\textsc{Search}}(V_x)) \subseteq \text{Dis}(H_{\textsc{Label}}(x))$. □

3.2 LARCH

We introduce an algorithm LARCH that combines LABEL and SEARCH. It is different from LABEL-only algorithm in that it first calls SEARCH to check if there exists a systematic mistake. If so it will move to H_{k+1} and consider more complicated hypothesis classes.

Theorem 1. *Assume that* $\mathrm{err}(h^*) = 0$. *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* S *in* LARCH *at the first time that* $k \geq k'$. *Fix any* $\epsilon, \delta \in (0,1)$. *If* LARCH *is run with inputs hypothesis classes* $\{H_k\}_{k=0}^{\infty}$, *oracles* LABEL *and* SEARCH, *and learning parameters* ϵ, δ, *then with probability at least* $1-\delta$: LARCH *halts after at most* $k^* + \log(1/\epsilon)$ *for-loop iterations and returns a classifier with error rate at most* ϵ; *furthermore, it draws at most* $\tilde{O}(k^* d_{k^*}/\epsilon)$ *unlabeled examples from* $\mathcal{D}_{\mathcal{X}}$, *makes at most* $k^* + \log(1/\epsilon)$ *queries to* SEARCH, *and at most* $\tilde{O}((k^* + \log(1/\epsilon) \cdot (max_{k' \leq k^*} \theta_{k'}(\epsilon)) \cdot d_k \cdot \log^2(1/\epsilon))$ *queries to* LABEL.

Algorithm 1: LARCH

input : a nested sequence of hypotheses classes $H_0 \subset H_1 \subset ...$; oracles LABEL and SEARCH; learning parameters $\epsilon, \delta \in (0,1)$

initialize : $S \leftarrow \emptyset, (index)k \leftarrow 0, l \leftarrow 0$

for $i = 1, 2, ...$ **do**

  $e \leftarrow \mathrm{SEARCH}_{H_k}(H_k(S))$

  if $e = \perp$ **then** `// no counterexample found`

    if $2^{-l} \leq \epsilon$ **then**

      return *any* $h \in H_k(S)$

    else

      $l \leftarrow l + 1$

    end

  else `// counterexample found`

    $S \leftarrow S \cup \{e\}$

    $k \leftarrow \min\{k' : H_{k'}(S) \neq \emptyset\}$

  end

  $S \leftarrow S \cup \mathrm{CAL}(H_k(S), \mathrm{LABEL}, 2^{-l}, \delta/(i^2+i))$

end

Union-of-intervals example

Consider a hypothesis class where each hypothesis is a union of non-trivial intervals in $\mathcal{X} := [0,1]$, assuming that $D_{\mathcal{X}}$ is uniform.

For $k \geq 0$, let H_k be the hypothesis class of the union of up to k intervals in $[0,1]$ and H_0 contains only negative hypothesis. The target hypothesis h^* is the union of k^* non-empty intervals.

Because the disagreement coefficient of H_1 is $\Omega(1/\epsilon)$, the probability that a LABEL-only active learner draws an input that an help distinguish hypotheses in H_1 can be really small

and the learning will not be very effective.

However, the first SEARCH query by LARCH provides a counterexample to H_0, which must be a positive example $(x_1, +1)$. Hence, $H_1(S_{[1]})$ is the class of intervals that contain x_1 with disagreement coefficient $\theta_1 \leq 4$.

Before LARCH advances its index to a value k (for any $k \leq k^*$), S must already contain at least k positive examples and $k-1$ negative examples separating them. The disagreement coefficient of the set of unions of k intervals consistent with $S_{[k]}$ is at most $4k$, independent of ϵ.

SEARCH actually narrows the searchable space and increases the probability of finding useful inputs to eliminate wrong hypotheses. As a result, the sample complexity and label complexity is polynomial in $\log(1/\epsilon)$ which is a great improvement compared with $\tilde{O}(1/\epsilon)$.

4 Appendix

A Online learning and Active learning

We will show that online learning and active learning are equivalent problems. Given an online learning algorithm, we can construct another algorithm to solve active learning problem, and vice versa.

Proposition 3. *Given an algorithm $\mathcal{A}$ that makes less then M equivalence queries in the active learning problem, we can construct an algorithm $\mathcal{B}$ that makes less than M mistakes in online learning.*

Proof. Every time $\mathcal{A}$ makes an equivalence query with hypothesis $\hat{h}$, $\mathcal{B}$ will predict $\hat{h}(x_t)$ for each x_t until it makes one mistake on (x', y'). Then $\mathcal{B}$ presents (x', y') to $\mathcal{A}$ as a counterexample. Since $\mathcal{A}$ makes at most M equivalence queries, the loop will happen at most M times. It means that $\mathcal{B}$ will make at most M mistakes. □

Proposition 4. *Suppose we have an algorithm $\mathcal{B}$ that makes less than M mistakes in the online learning problem, then we can construct an algorithm $\mathcal{A}$ that makes less than M queries in active learning.*

Proof. Algorithm $\mathcal{A}$ makes N copies of the current state of $\mathcal{B}$ and feeds them with $x_1, x_2, ..., x_N$ to see the output $\mathcal{B}(x_1), ..., \mathcal{B}(x_N)$ where $N = |\mathcal{X}|$. $\mathcal{A}$ constructs a hypothesis $\hat{h}$ s.t. $\hat{h}(x_i) = \mathcal{B}(x_i)$ and get a counterexample (x',y'). Now $\mathcal{A}$ knows the true label of x' is y'. It uses the copy of $\mathcal{B}$ fed with x' and reveals y' to it. Repeat until $\mathcal{A}$ gets answer *yes*.

Notice that every time $\mathcal{A}$ makes an equivalence query and gets answer *no*, it indicates that $\mathcal{A}$ finds a mistake of $\mathcal{B}$. Since $\mathcal{B}$ makes at most M mistakes, $\mathcal{A}$ makes at most M equivalence queries before it gets answer *yes*. □

References

[1] Dana Angluin. "Queries and concept learning". In: *Machine learning* 2.4 (1988), pp. 319–342.

[2] Alina Beygelzimer et al. "Search Improves Label for Active Learning". In: *CoRR* abs/1602.07265 (2016). URL: http://arxiv.org/abs/1602.07265.