CS269: Machine Learning Theory
Lecture 17: Learning from Labeled and Unlabeled Data
November 22, 2010

Lecturer: Jennifer Wortman Vaughan
Scribe: Georgios Georgiadis, Ryan R. Rosario, Alan Roytman

1 Introduction

This lecture focused on methods of combining labeled and unlabeled data to learn a classifier.

As a motivating example, suppose we would like to classify web pages as either fraudulent or not fraudulent. In this case, obtaining unlabeled data (i.e., webpages) is easy. However, labeling such data can be very costly since it requires humans to manually look at each webpage and determine whether or not it is a scam. We might hope that by making use of the *unlabeled* data in a clever way, we could learn a classifier without requiring as much labeled data as we would normally need.

In this lecture, we consider two learning models: semi-supervised learning, and active learning.

2 Semi-Supervised Learning

In this lecture, we will only very briefly discuss semi-supervised learning to get a sense of the types of assumptions that people make and the types of algorithms that can be used. We will spend most of the lecture on active learning.

Suppose we are given data of the form $S_L = (\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m)$ with additional unlabeled examples $S_U = \vec{x}_{m+1}, \vec{x}_{m+2}, \vec{x}_{m+k}$ where k is some finite index. The total data set is then $S = S_L \cup S_U$.

The idea behind semi-supervised learning is that we want the function we learn to be "compatible" with S. For example, if we think that our data can be classified using a large margin, this immediately rules out separators that cut through dense parts of the input space. (This is the assumption behind the "Transductive SVM" algorithm.) It must be noted that if we do not make certain assumptions about the data, then the unlabeled points S_U do not help much.

2.1 Co-Training for Semi-Supervised Learning

We briefly discuss the Co-Training algorithm, which is based on a specific assumption about the compatibility between the distribution D and the target. We will not go into the analysis of this algorithm in detail, but more information can be found in the original paper[1] from COLT, which has close to two-thousand citations and was the winner of the ICML 10-Year Best Paper Award in 2008.

Co-Training assumes two "views" of the data, where each input $\vec{x}$ is a pair,

$$\vec{x} = (\vec{x}_1, \vec{x}_2)$$

[1] Blum, A., Mitchell, T. Combining labeled and unlabeled data with co-training. COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann, 1998, p. 92-100.

In the context of the web page classification motivating example, $\vec{x}_1$ may be the meta data associated with the web page such as the title, text, etc. and $\vec{x}_2$ the words in links pointing to the web page.

Assume there exists functions c_1, c_2 such that

$$c_1(\vec{x}_1) = c_2(\vec{x}_2) = c(\vec{x})$$

for any $\vec{x}$ with positive weight from D. This is where the compatibility between the distribution D and the target is involved. That is, we only care about agreement between the views on points $(\vec{x}_1, \vec{x}_2)$ that have non-zero weight with respect to D.

To extract a classifier, one can use iterative co-training.

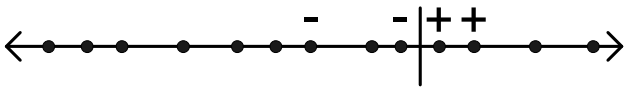
1. Use the labeled data to learn the initial h_1, h_2.
2. First use h_1 to label examples that it's confident about and then feed these to our learner to update h_2.
3. Then use h_2 to label examples that it's confident about and then feed these to our learner to update h_1.
4. Keep repeating this process.

One limitation to iterative co-training and semi-supervised learning in general is that there are many assumptions that must be satisfied for it to yield decent results. Although co-training has many assumptions that must be satisfied, it can work well in practice, and has become a very popular algorithm.

3 Active Learning

In active learning, the learning algorithm is allowed to interactively query for labels on specifically requested points. In contrast to being given every label for each point, in some settings it is possible to achieve a lower label complexity (i.e. the number of labels needed to guarantee an error of at most ϵ with probability at least $1 - \delta$).

For one-dimensional thresholds, we can show that there is an opportunity to improve on the label complexity in the active learning setting. In the traditional ("passive" learning) setting, we argued that $O(\frac{1}{\epsilon})$ examples are sufficient to obtain error at most ϵ. However, in the active learning setting, we can perform a binary search and get a much better bound on the label complexity. In particular, we begin by querying the label of the point in the middle. If this point is negative, then we know that the target threshold must lie on the right side (similarly, if the point is positive, we know the target must lie on the left side). We can repeat this process and obtain a label complexity of $O(\log \frac{1}{\epsilon})$. The following is an example of running the binary search procedure:



However, there are some classes, such as one-dimensional intervals, which cannot achieve a lower label complexity in the active learning setting. In particular, consider the task of distinguishing between the following two cases:

1. There exists some positive interval of weight ϵ.
2. The target function is the all-negative function (i.e. every point is labeled with a "$-$").

Intuitively, if we query for a point and discover that its label is negative, it is not clear that this reduces our search space in any substantial way. There may be some points labeled positively to the left, or there may be some to the right, or there may be none at all. It turns out that, in order to distinguish between these two cases, we need $\frac{1}{\epsilon}$ labels. Hence, we get no improvement over the traditional, passive learning setting.

3.1 The Cohn-Atlas-Ladner Algorithm

The Cohn-Atlas-Ladner Algorithm[2] (CAL for short) is a well-known algorithm used in the active learning setting. It works by considering the current version space (initially starting out with the entire hypothesis class H) and reducing it if necessary based on some criterion. In particular, if there exist two hypotheses h, h' in the version space that disagree on the current point, the algorithm requests the label of the point. In the following algorithm, V_t represents the version space at time t, $\vec{x}_t$ is the data received at time t, and T is the number of examples.

```
Initialize V_1 = H
foreach t = 1, 2, 3, ..., T do
    Receive some x_t
    if ∃h, h' ∈ V_t s.t. h(x_t) ≠ h'(x_t) then
        Query label y_t
        V_{t+1} = {h ∈ V_t : h(x_t) = y_t}
    else
        V_{t+1} = V_t
return an arbitrary h ∈ V_{T+1}
```

Algorithm 1: CAL Algorithm.

As we stated it in Algorithm 1, the CAL algorithm can be extremely inefficient since it potentially requires searching through all of the functions in the version space (which could be infinite) every time we see a new point. However, if we have an efficient algorithm that returns a consistent hypothesis or null if none exists, we can use this algorithm as a black box to get an efficient implementation of the CAL algorithm. Using the black box algorithm, which we call Learn, the CAL algorithm can be implemented by maintaining a set of pairs $(\vec{x}_t, y_t)$ such that there exists a hypothesis in H consistent with the labeling in the set. Algorithm 2, which makes use of the Learn subroutine, is a more formal description of the procedure.

3.2 CAL Claim

Claim 1. *The number of labels that are required by CAL to produce a function $h \in \mathcal{H}$ such that with probability at least $1 - \delta$, the error $\mathrm{err}(h) \leq \epsilon$ is:*

$$\tilde{O}\left(\theta\left(d + log\left(\frac{1}{\delta}\right)\right) log\left(\frac{1}{\epsilon}\right)\right).$$

In the above expression $\tilde{O}$ is the same as big-O notation, slightly modified to hide extra log terms that we are not interested in. d is the VC dimension of $\mathcal{H}$ and θ is the so called "Disagreement Coefficient". As we will see in the next section, θ depends on the hypothesis class $\mathcal{H}$, the data distribution D, and the

[2]Cohn, D., Atlas, L., and Ladner, R. Improving generalization with active learning. Machine Learning, 15(2):201-221, 1994.

```
Initialize S = {}
foreach t = 1, 2, 3, ... do
    Receive some x_t
    if Learn(S ∪ {(x_t, 1)}) and Learn(S ∪ {(x_t, 0)}) return valid functions then
        Query label y_t
        S = S ∪ {(x_t, y_t)}
    else if Learn(S ∪ {(x_t, 1)}) returns a valid function then
        S = S ∪ {(x_t, 1)}
    else
        S = S ∪ {(x_t, 0)}
return Learn(S)
```

Algorithm 2: Equivalent CAL Algorithm Using the Learn Subroutine.

(unknown) target $c \in C$. Note that in the above expression, there is a logarithmic dependence on $1/\epsilon$, a result which is somewhat satisfying. As long as the disagreement coefficient is small, active learning can be done using fewer labels than passive learning.

3.3 Disagreement Coefficient[3]

An important part of the expression in Claim 1 is the θ parameter. To determine what θ is let's define the following quantities:

Definition 1.

$$d(h, h') = Pr_{\vec{x} \sim D}[h(\vec{x}) \neq h'(\vec{x})].$$

$d(h, h')$ can be interpreted as a distance measure between the two functions. If the two functions completely agree on all data points, then $d(h, h') = 0$. The quantity gets larger and larger as the two functions disagree more and more.

Definition 2.

$$B(h, r) = \{h' \in \mathcal{H} : d(h, h') \leq r\}.$$

The above quantity, $B(h, r)$, can be interpreted geometrically as a ball, with center h and radius r. The functions, h', that fall inside this radius, r, are at distance at most r away from h.

Definition 3.

$$DIS(V) = \{\vec{x} : \exists h, h' \in V, h(\vec{x}) \neq h'(\vec{x})\}.$$

If V is the current version space, then $DIS(V)$ is the set of data points $\vec{x}$ for which there are two functions in the version space that disagree. Suppose that c is the true target function. If $V_t \subseteq B(c, r)$ for a small value of r (say, $r < \epsilon$), then we know that any function output by CAL will have a small error. Intuitively, if active learning can help reduce the number of labels that we need, then it should be the case that the probability of querying additional labels at this point should be very small. Since we query a label

[3]Hanneke, S. (2007). A Bound on the Label Complexity of Agnostic Active Learning. In proceedings of the 24th Annual International Conference on Machine Learning (ICML).

whenever two functions in the version space disagree, this means we would like $Pr_{X\sim D}[X \in DIS(V_t)]$ to be small. If $V_t \subseteq B(c,r)$, then

$$Pr_{X\sim D}[X \in DIS(V_t)] \leq Pr_{X\sim D}[X \in DIS(B(c,r))].$$

Putting this all together, we would like $Pr_{X\sim D}[X \in DIS(B(c,r))]$ to be small when r is small. The disagreement coefficient θ captures this notion. In particular, we define θ as follows:

Definition 4.

$$\theta = \sup_{r>0} \frac{Pr_{X\sim D}[X \in DIS(B(c,r))]}{r}.$$

We would like the number of labels that are required by CAL to be small when the error is small. From Definition 4 though if θ is big, even if the error is small, the probability of querying for a label will be high. Only when θ is small will the probability be small.

3.4 Example 1: 1-Dimensional threshold function class

We can easily calculate θ for the one dimensional threshold function class. Following Definition 1, $d(h,h')$ is the probability of the interval between the two threshold values as shown in Figure 1. In other words $d(h,h') =$ weight that D puts on $[h,h']$.

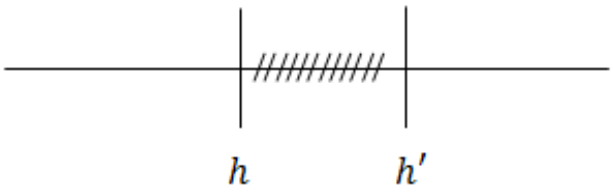


Figure 1: Illustrative example of the Disagreement Region for the 1-D threshold function class

Following Definition 2, $B(c,r)$ is the set of functions that fall in an r-weighted interval to the left or right of c, weighted by the probability distribution D. According to Definition 3 then, $DIS(B(c,r)) = B(c,r)$, since the set of values of x for which these threshold functions can disagree is the same. Figure 2 illustrates this argument. Note here, that we are abusing notation and we are using intervals to represent both sets of points and sets of possible thresholds.

We can then evaluate θ as follows:

$$\theta = \sup_{r>0} \frac{Pr_{X\sim D}[X \in DIS(B(c,r))]}{r} = \frac{2r}{r} = 2.$$

And using Claim 1, we get an expression for the number of labels required which is on the order of $\tilde{O}\left(2log\left(\frac{1}{\epsilon}\right)\right)$.

3.5 Example 2: 1-Dimensional interval function class

In the case of 1-Dimensional interval functions, it is easy to show that the disagreement coefficient $\theta = \frac{1}{w}$, where w is the weight that the distribution D puts on the positive interval. This explains why active learning does not help in general in this case, and also captures our intuition that the hard cases for active learning are the ones where we would like to distinguish very small positive regions from the all-negative hypothesis.

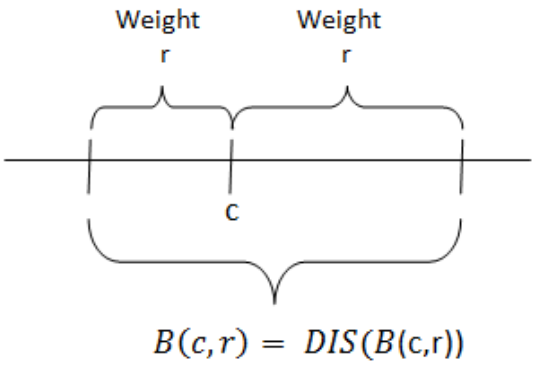


Figure 2: Illustrative example of the Ball and Disagreement set for the 1-D threshold function class