67577 – Intro. to Machine Learning **Fall semester, 2008/9**

Lecture 10: The Formal (PAC) Learning Model

Lecturer: Amnon Shashua *Scribe: Amnon Shashua*[1]

We have see so far algorithms that explicitly estimate the underlying distribution of the data (Bayesian methods and EM) and algorithms that are in some sense optimal when the underlying distribution is Gaussian (PCA, LDA). We have also encountered an algorithm (SVM) that made no assumptions on the underlying distribution and instead tied the accuracy to the margin of the training data.

In this lecture and in the remainder of the course we will address the issue of "accuracy" and "generalization" in a more formal manner. Because the learner receives only a finite training sample, the learning function can do very well on the training set yet perform badly on new input instances. What we would like to establish are certain guarantees on the accuracy of the learner measured over all the instance space and not only on the training set. We will then use those guarantees to better understand what the large-margin principle of SVM is doing in the context of generalization.

In the remainder of this lecture we will refer to the following notations: the class of learning functions is denoted by C. A learning functions is often referred to as a "concept" or "hypothesis". A target function $c_t \in C$ is a function that has zero error on all input instances (such a function may not always exist).

10.1 The Formal Model

In many learning situations of interest, we would like to assume that the learner receives m examples sampled by some fixed (yet unknown) distribution D and the learner must do its best with the training set in order to achieve the accuracy and confidence objectives. The Probably Approximate Correct (PAC) model, also known as the "formal model", first introduced by Valient in 1984, provides a probabilistic setting which formalizes the notions of accuracy and confidence.

The PAC model makes the following statistical assumption. We assume the learner receives a set S of m instances $\mathbf{x}_1, ..., \mathbf{x}_m \in X$ which are sampled randomly and independently according to a distribution D over X. In other words, a random training set S of length m is distributed according to the product probability distribution D^m. The distribution D is unknown, but we will see that one can obtain useful results by simply assuming that D is fixed — there is no need to attempt to recover D during the learning process. To recap, we make the following three assumptions: (i) D is unkown, (ii) D is fixed throughout the learning process, and (iii) the example instances are sampled independently of each other (are Identically and Independently Distributed — i.i.d.).

We distinguish between the "realizable" case where a target concept $c_t(\mathbf{x})$ is known to exist, and the unrealizable case, where there is no such guarantee. In the realizable case our training examples are $Z = \{(\mathbf{x}_i, c_t(\mathbf{x}_i)\}$, $i = 1, ..., m$ and D is defined over X (since $y_i \in Y$ are given by $c_t(\mathbf{x}_i)$). In the unrealizable case, $Z = \{(\mathbf{x}_i, y_i)\}$ and D is the distribution over $X \times Y$ (each element is a pair, one from X and the other from Y).

[1]Sources: the "learning game example" is based on Ch.1 of Kearns and Vazirani "An introduction to computational learning theory". Also material from pp. 13–25 of Anthony and Bartlett "Neural Network Learning: Theoretical Foundations" was used in the preparation of this lecture.

We next define what is meant by the *error* induced by a concept function $h(\mathbf{x})$. In the realizable case, given a function $h \in C$, the error of h is defined with respect to the distribution D:

$$err(h) = prob_D[\mathbf{x} : c_t(\mathbf{x}) \neq h(\mathbf{x})] = \int_{\mathbf{x} \in X} ind(c_t(\mathbf{x}) \neq h(\mathbf{x})) D(\mathbf{x}) d\mathbf{x}$$

where $ind(F)$ is an indication function which returns '1' if the proposition F is *true* and '0' otherwise. The function $err(h)$ is the probability that an instance $\mathbf{x}$ sampled according to D will be labeled incorrectly by $h(\mathbf{x})$. Let $\epsilon > 0$ be a parameter given to the learner specifying the "accuracy" of the learning process, i.e. we would like to achieve $err(h) \leq \epsilon$. Note that $err(c_t) = 0$.

In addition, we define a "confidence" parameter $\delta > 0$, also given to the learner, which defines the probability that $err(h) > \epsilon$, namely,

$$prob[err(h) > \epsilon] < \delta,$$

or equivalently:

$$prob[err(h) \leq \epsilon] \geq 1 - \delta.$$

In other words, the learner is supposed to meet some accuracy criteria but is allowed to deviate from it by some small probability. Finally, the learning algorithm is supposed to be "efficient" if the running time is polynomial in $1/\epsilon, \ln(1/\delta), n$ and the size of the concept target function $c_t()$ (measured by the number of bits necessary for describing it, for example).

We will say that an algorithm L learns a concept family C in the formal sense (PAC learnable) if for any $c_t \in C$ and for every distribution D on the instance space X, the algorithm L generates efficiently a concept function $h \in C$ such that the probability that $err(h) \leq \epsilon$ is at least $1 - \delta$.

The inclusion of the confidence value δ could seem at first unnatural. What we desire from the learner is to demonstrate a consistent performance regardless of the training sample Z. In other words, it is not enough that the learner produces a hypothesis h whose accuracy is above threshold, i.e., $err(h) \leq \epsilon$, for some training sample Z. We would like the accuracy performance to hold under *all* training samples (sampled from the distribution D^m) — since this requirement could be too difficult to satisfy, the formal model allows for some "failures", i.e, situations where $err(h) > \epsilon$, for some training samples Z, as long as those failures are rare and the frequency of their occurrence is controlled (the parameter δ) and can be as small as we like.

In the unrealizable case, there may be no function $h \in C$ for which $err(h) = 0$, thus we need to define what we mean by the *best* a learning algorithm can achieve:

$$Opt(C) = \min_{h \in C} err(h),$$

which is the best that can be done on the concept class C using functions that map between X and Y. Given the desired accuracy ϵ and confidence δ values the learner seeks a hypothesis $h \in C$ such that:

$$prob[err(h) \leq Opt(C) + \epsilon] \geq 1 - \delta.$$

We are ready now to formalize the discussion above and introduce the definition of the formal learning model (Anthony-Bartlett, pp. 16):

Definition 1 (Formal Model) *Let C be the concept class of functions that map from a set X to Y. A learning algorithm L is a function:*

$$L : \bigcup_{m=1}^{\infty} \{(\mathbf{x}_i, y_i)\}_{i=1}^{m} \rightarrow C$$

from the set of all training examples to C with the following property: given any $\epsilon, \delta \in (0,1)$ there is an integer $m_0(\epsilon, \delta)$ such that if $m \geq m_0$ then, for any probability distribution D on $X \times Y$, if Z is a training set of length m drawn randomly according to the product probability distribution D^m, then with probability of at least $1-\delta$ the hypothesis $h = L(Z) \in C$ output by L is such that $err(h) \leq Opt(C) + \epsilon$. We say that C is learnable (or PAC learnable) if there is a learning algorithm for C.

There are few points to emphasize. The sample size $m_0(\epsilon, \delta)$ is a sufficient sample size for PAC learning C by L and is allowed to vary with ϵ, δ. Decreasing the value of either ϵ or δ makes the learning problem more difficult and in turn a larger sample size is required. Note however that $m_0(\epsilon, \delta)$ does *not depend* on the distribution D! that is, a sufficient sample size can be given that will work for *any* distribution D — provided that D is fixed throughout the learning experience (both training and later for testing). This point is a crucial property of the formal model because if the sufficient sample size is allowed to vary with the distribution D then not only we would need to have some information about the distribution in order to set the sample complexity bounds, but also an adversary (supplying the training set) could control the rate of convergence of L to a solution (even if that solution can be proven to be optimal) and make it arbitrarily slow by suitable choice of D.

What makes the formal model work in a distribution-invariant manner is that it critically depends on the fact that in many interesting learning scenarios the concept class C is not too complex. For example, we will show later in the lecture that any finite concept class $|C| < \infty$ is learnable, and the sample complexity (in the realizable case) is

$$m \geq \frac{1}{\epsilon} \ln \frac{|C|}{\delta}.$$

In lectures 4,5 we will look into concept classes of infinite size and show that despite the fact that the class is infinite it still can be of low complexity!

Before we illustrate the concepts above with an example, there is another useful measure which is the *empirical* error (also known as the sample error) $e\hat{r}r(h)$ which is defined as the proportion of examples from Z on which h made a mistake:

$$e\hat{r}r(h) = \frac{1}{m} |\{i : h(\mathbf{x}_i) \neq c_t(\mathbf{x}_i)\}|$$

(replace $c_t(\mathbf{x}_i)$ with y_i for the unrealizable case). The situation of bounding the true error $err(h)$ by minimizing the sample error $e\hat{r}r(h)$ is very convenient — we will get to that later.

10.2 The Rectangle Learning Problem

As an illustration of learnability we will consider the problem (introduced in Kearns-Vazirani) of learning an axes-aligned rectangle from positive and negative examples. We will show that the problem is PAC-learnable and find out $m_0(\epsilon, \delta)$.

In the rectangle learning game we are given a training set consisting of points in the 2D plane with a positive '+' or negative '-' label. The positive examples are sampled inside the target rectangle (parallel to the main axes) R and the negative examples are sampled outside of R. Given m examples sampled i.i.d according to some distribution D the learner is supposed to generate an approximate rectangle R' which is consistent with the training set (we are assuming that R exists) and which satisfies the accuracy and confidence constraints.

We first need to decide on a learning strategy. Since the solution R' is not uniquely defined given any training set Z, we need to add further constraints to guarantee a unique solution. We will choose R' as the axes-aligned concept which gives the *tightest fit* to the positive examples, i.e., the smallest area axes-aligned rectangle which contains the positive examples. If no positive examples are given then $R' = \emptyset$. We can also assume that Z contains at least three non-collinear positive examples in order to avoid complications associated with infinitesimal area rectangles. Note that we could have chosen other strategies, such as the middle ground between the tightest fit to the positive examples and the tightest fit (from below) to the negative examples, and so forth. Defining a strategy is necessary for the analysis below — the type of strategy is not critical though.

We next define the error $err(R')$ on the concept R' generated by our learning strategy. We first note that with the strategy defined above we always have $R' \subset R$ since R' is the tightest fit solution which is consistent with the sample data (there could be a positive example outside of R' which is not in the training set). We will define the "weight" $w(E)$ of a region E in the plane as

$$w(E) = \int_{\mathbf{x} \in E} D(\mathbf{x}) d\mathbf{x},$$

i.e., the probability that a random point sampled according to the distribution D will fall into the region. Therefore, the error associated with the concept R' is

$$err(R') = w(R - R')$$

and we wish to bound the error $w(R - R') \leq \epsilon$ with probability of at least $1 - \delta$ after seeing m examples.

We will divide the region $R - R'$ into four strips $T_1', ..., T_4'$ (see Fig.10.1) which overlap at the corners. We will estimate $prob(w(T_i') \geq \frac{\epsilon}{4})$ noting that the overlaps between the regions makes our estimates more pessimistic than they truly are (since we are counting the overlapping regions twice) thus making us lean towards the conservative side in our estimations.

Consider the upper strip T_1'. If $w(T_1' \leq \frac{\epsilon}{4})$ then we are done. We are however interested in quantifying the probability that this is not the case. Assume $w(T_1') > \frac{\epsilon}{4}$ and define a strip T_1 which starts from the upper axis of R and stretches to the extent such that $w(T_1) = \frac{\epsilon}{4}$. Clearly $T_1 \subset T_1'$. We have that $w(T_1') > \frac{\epsilon}{4}$ iff $T_1 \subset T_1'$. Furthermore:

Claim 1 $T_1 \subset T_1'$ *iff* $\mathbf{x}_1, ..., \mathbf{x}_m \not\in T_1$.

Proof: If $\mathbf{x}_i \in T_1$ the the label must be positive since $T_1 \subset R$. But if the label is positive then given our learning strategy of fitting the tightest rectangle over the positive examples, then $\mathbf{x}_i \in R'$. Since $T_1 \not\subset R'$ it follows that $\mathbf{x}_i \not\in T_1$. $\square$

We have therefore that $w(T_1' > \frac{\epsilon}{4})$ iff no point in T_1 appears in the sample $S = \{\mathbf{x}_1, ..., \mathbf{x}_m\}$ (otherwise T_1 intersects with R' and thus $T_1' \subset T_1$). The probability that a point sampled according to the distribution D will fall outside of T_1 is $1 - \frac{\epsilon}{4}$. Given the independence assumption (examples are drawn i.i.d.), we have:

$$prob(\mathbf{x}_1, ..., \mathbf{x}_m \not\in T_1) = prob(w(T_1' > \frac{\epsilon}{4})) = (1 - \frac{\epsilon}{4})^m.$$

Repeating the same analysis to regions T_2', T_3', T_4' and using the union bound $P(A \cup B) \leq P(A) + P(B)$ we come to the conclusion that the probability that *any* of the four strips of $R - R'$ has weight greater that $\epsilon/4$ is at most $4(1 - \frac{\epsilon}{4})^m$. In other words,

$$prob(err(L') \geq \epsilon) \leq 4((1 - \frac{\epsilon}{4})^m \leq \delta.$$

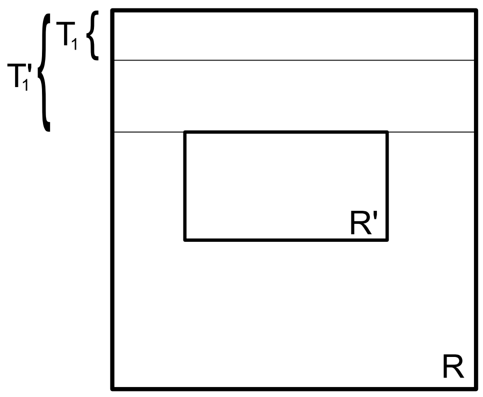

Figure 10.1: Given the tightest-fit to positive examples strategy we have that $R' \subset R$. The strip T_1 has weight $\epsilon/4$ and the strip T_1' is defined as the upper strip covering the area between R and R'.

We can make the expression more convenient for manipulation by using the inequality $e^{-x} \geq 1 - x$ (recall that $1 + (1/n))^n < e$ from which it follows that $(1 + z)^{1/z} < e$ and by taking the power of rz where $r \geq 0$ we obtain $(1 + z)^r < e^{rz}$ then set $r = 1, z = -x$):

$$4(1 - \frac{\epsilon}{4})^m \leq 4e^{-\frac{\epsilon m}{4}} \leq \delta,$$

from which we obtain the bound:

$$m \geq \frac{4}{\epsilon} \ln \frac{4}{\delta}.$$

To conclude, assuming that the learner adopts the tightest-fit to positive examples strategy and is given at least $m_0 = \frac{4}{\epsilon} \ln \frac{4}{\delta}$ training examples in order to find the axes-aligned rectangle R', we can assert that with probability $1 - \delta$ the error associated with R' (i.e., the probability that an $(m + 1)$'th point will be classified incorrectly) is at most ϵ.

We can see form the analysis above that indeed it applies to any distribution D where the only assumption we had to make is the independence of the draw. Also, the sample size m behaves well in the sense that if one desires a higher level of accuracy (smaller ϵ) or a higher level of confidence (smaller δ) then the sample size grows accordingly. The growth of m is linear in $1/\epsilon$ and linear in $\ln(1/\delta)$.

10.3 Learnability of Finite Concept Classes

In the previous section we illustrated the concept of learnability with a particular simple example. We will now focus on applying the learnability model to a more general family of learning examples. We will consider the family of all learning problems over finite concept classes $|C| < \infty$. For example, the conjunction learning problem (over boolean formulas) which we looked at in Lecture 1 with n literals contains only 3^n hypotheses because each variable can appear in the conjunction or not and if appears it could be negated or not. We have shown that n is the lower bound on the number of mistakes on the worst case analysis any on-line algorithm can achieve. With the definitions we have above on the formal model of learnability we can perform accuracy and sample complexity analysis that will apply to any learning problem over finite concept classes. This was first introduced by Valiant in 1984.

In the realizable case over $|C| < \infty$, we will show that any algorithm L which returns a hypothesis $h \in C$ which is consistent with the training set Z is a *learning algorithm* for C. In other words, any finite concept class is learnable and the learning algorithms simply need to generate consistent hypotheses. The sample complexity m_0 associated with the choice of ϵ and δ can be shown as equal to: $\frac{1}{\epsilon} \ln \frac{|C|}{\delta}$.

In the unrealizable case, any algorithm L that generates a hypothesis $h \in C$ that *minimizes* the empirical error (the error obtained on Z) is a learning algorithm for C. The sample complexity can be shown as equal to: $\frac{2}{\epsilon^2} \ln \frac{2|C|}{\delta}$. We will derive these two cases below.

10.3.1 The Realizable Case

Let $h \in C$ be some consistent hypothesis with the training set Z (we know that such a hypothesis exists, in particular $h = c_t$ the target concept used for generating Z) and suppose that

$$err(h) = prob[\mathbf{x} \sim D : h(\mathbf{x}) \neq c_t(\mathbf{x})] > \epsilon.$$

Then, the probability (with respect to the product distribution D^m) that h agrees with c_t on a random sample of length m is at most $(1-\epsilon)^m$. Using the inequality we saw before $e^{-x} \geq 1 - x$ we have:

$$prob[err(h) > \epsilon \;\&\&\; h(\mathbf{x}_i) = c_t(\mathbf{x}_i), \quad i = 1, ..., m] \leq (1-\epsilon)^m < e^{-\epsilon m}.$$

We wish to bound the error *uniformly*, i.e., that $err(h) \leq \epsilon$ for all concepts $h \in C$. This requires the evaluation of:

$$prob[\max_{h \in C}\{err(h) > \epsilon\} \;\&\&\; h(\mathbf{x}_i) = c_t(\mathbf{x}_i), \quad i = 1, ..., m].$$

There at most $|C|$ such functions h, therefore using the Union-Bound the probability that *some* function in C has error larger than ϵ and is consistent with c_t on a random sample of length m is at most $|C|e^{-\epsilon m}$:

$$\begin{aligned}
& prob[\exists h : err(h) > \epsilon \;\&\&\; h(\mathbf{x}_i) = c_t(\mathbf{x}_i), \quad i = 1, ..., m] \\
& \leq \sum_{h: err(h) > \epsilon} prob[h(\mathbf{x}_i) = c_t(\mathbf{x}_i), \quad i = 1, ..., m] \\
& \leq |h : err(h) > \epsilon| e^{-\epsilon m} \\
& \leq |C| e^{-\epsilon m}
\end{aligned}$$

For any positive δ, this probability is less than δ provided:

$$m \geq \frac{1}{\epsilon} \ln \frac{|C|}{\delta}.$$

This derivation can be summarized in the following theorem (Anthony-Bartlett, pp. 25):

Theorem 1 *Let C be a finite set of functions from X to Y. Let L be an algorithm such that for any m and for any $c_t \in C$, if Z is a training sample $\{(\mathbf{x}_i, c_t(\mathbf{x}_i))\}$, $i = 1, ..., m$, then the hypothesis $h = L(Z)$ satisfies $h(\mathbf{x}_i) = c_t(\mathbf{x}_i)$. Then L is a learning algorithm for C in the realizable case with sample complexity*

$$m_0 = \frac{1}{\epsilon} \ln \frac{|C|}{\delta}.$$

10.3.2 The Unrealizable Case

In the realizable case an algorithm simply needs to generate a consistent hypothesize to be considered a learning algorithm in the formal sense. In the unrealizable situation (a target function c_t might not exist) an algorithm which minimizes the empirical error, i.e., an algorithm L generates $h = L(Z)$ having minimal sample error:

$$\hat{err}(L(Z)) = \min_{h \in C} \hat{err}(h)$$

is a learning algorithm for C (assuming finite $|C|$). This is a particularly useful property given that the true errors of the functions in C are unknown. It seems natural to use the sample errors $\hat{err}(h)$ as estimates to the performance of L.

The fact that given a large enough sample (training set Z) then the sample error $\hat{err}(h)$ becomes close to the true error $err(h)$ is somewhat of a restatement of the "law of large numbers" of probability theory. For example, if we toss a coin many times then the relative frequency of 'heads' approaches the true probability of 'head' at a rate determined by the law of large numbers. We can bound the probability that the difference between the empirical error and the true error of some h exceeds ϵ using Hoeffding's inequality:

Claim 2 *Let h be some function from X to $Y = \{0, 1\}$. Then*

$$prob[|\hat{err}(h) - err(h)| \geq \epsilon] \leq 2e^{(-2\epsilon^2 m)},$$

for any probability distribution D, any $\epsilon > 0$ and any positive integer m.

Proof: This is a straightforward application of Hoeffding's inequality to Bernoulli variables. Hoeffding's inequality says: Let X be a set, D a probability distribution on X, and $f_1, ..., f_m$ real-valued functions $f_i : X \to [a_i, b_i]$ from X to an interval on the real line ($a_i < b_i$). Then,

$$prob\left[|\frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{x}_i) - E_{\mathbf{X}\sim D}[f(\mathbf{x})]| \geq \epsilon\right] \leq 2e^{-\frac{2\epsilon^2 m^2}{\sum_i (b_i - a_i)^2}} \tag{10.1}$$

where

$$E_{\mathbf{X}\sim D}[f(\mathbf{x})] = \frac{1}{m}\sum_{i=1}^{m} \int f_i(\mathbf{x}) D(\mathbf{x}) d\mathbf{x}.$$

In our case $f_i(\mathbf{x}_i) = 1$ iff $h(\mathbf{x}_i) \neq y_i$ and $a_i = 0, b_i = 1$. Therefore $(1/m)\sum_i f_i(\mathbf{x}_i) = \hat{err}(h)$ and $err(h) = E_{\mathbf{X}\sim D}[f(\mathbf{x})]$. $\square$

The Hoeffding bound almost does what we need, but not quite so. What we have is that for any *given* hypothesis $h \in C$, the empirical error is close to the true error with high probability. Recall that our goal is to minimize $err(h)$ over all possible $h \in C$ but we can access only $\hat{err}(h)$. If we can guarantee that the two are close to each other *for every* $h \in C$, then minimizing $\hat{err}(h)$ over all $h \in C$ will approximately minimize $err(h)$. Put formally, in order to ensure that L learns the class C, we must show that

$$prob\left[\max_{h \in C} |\hat{err}(h) - err(h)| < \epsilon\right] > 1 - \delta$$

In other words, we need to show that the empirical errors converge (at high probability) to the true errors *uniformly* over C as $m \to \infty$. If that can be guaranteed, then with (high) probability $1 - \delta$, for every $h \in C$,

$$err(h) - \epsilon < \hat{err}(h) < err(h) + \epsilon.$$

So, since the algorithm L running on training set Z returns $h = L(Z)$ which minimizes the empirical error, we have:

$$err(L(Z)) \leq e\hat{r}r(L(Z)) + \epsilon = \min_h e\hat{r}r(h) + \epsilon \leq Opt(C) + 2\epsilon,$$

which is what is needed in order that L learns C. Thus, what is left is to prove the following claim:

Claim 3

$$prob\left[\max_{h \in C} |e\hat{r}r(h) - err(h)| \geq \epsilon\right] \leq 2|C|e^{-2\epsilon^2 m}$$

Proof: We will use the union bound. Finding the maximum over C is equivalent to taking the union of all the events:

$$prob\left[\max_{h \in C} |e\hat{r}r(h) - err(h)| \geq \epsilon\right] = prob\left[\bigcup_{h \in C} \{Z : |e\hat{r}r(h) - err(h)| \geq \epsilon\}\right],$$

using the union-bound and Claim 2, we have:

$$\leq \sum_{h \in C} prob\left[|e\hat{r}r(h) - err(h)| \geq \epsilon\right] \leq |C| 2e^{(-2\epsilon^2 m)}.$$

□

Finally, given that $2|C|e^{-2\epsilon^2 m} \leq \delta$ we obtain the sample complexity:

$$m_0 = \frac{2}{\epsilon^2} \ln \frac{2|C|}{\delta}.$$

This discussion is summarized with the following theorem (Anthony-Bartlett, pp. 21):

Theorem 2 *Let C be a finite set of functions from X to $Y = \{0, 1\}$. Let L be an algorithm such that for any m and for any training set $Z = \{(\mathbf{x}_i, y_i)\}$, $i = 1, ..., m$, then the hypothesis $L(Z)$ satisfies:*

$$e\hat{r}r(L(Z)) = \min_{h \in C} e\hat{r}r(h).$$

Then L is a learning algorithm for C with sample complexity $m_0 = \frac{2}{\epsilon^2} \ln \frac{2|C|}{\delta}$.

Note that the main difference with the realizable case (Theorem 1) is the larger $1/\epsilon^2$ rather than $1/\epsilon$. The realizable case requires a smaller training set since we are estimating a random quantity so the smaller the variance the less data we need.