Introduction to Machine Learning **Fall Semester, 2016/7**

Lecture 2: November 6, 2016

Lecturer: Yishay Mansour *Scribe: ym*

2.1 The PAC Model and Model Selection

In this lecture we introduce the PAC model. The PAC learning model is one of the important learning model. PAC stands for *Probably Approximately Correct*, our goal is to learn a hypothesis from a hypothesis class such that in high confidence we will have a small error rate (approximately correct). We start the lecture with an intuitive example to explain the idea behind the PAC model and continue to formalize the PAC model, and give a few examples. A main part of the lecture would be generalization bounds.

2.2 An intuitive example

Suppose we want to predict a 'typical person'. Our input is a sample of different attributes of people which includes their height and weight and their label: whether it is a typical person ('+') or not ('-'). Let H be our hypothesis class. In our example, H will be the set of all possible rectangles on a two-dimensional plane which are axis-aligned (not rotated). One example of hypothesis $h \in H$ can be: mark a person which denotes as (height, weight) to be a typical one ('+') if its description is in the range of:

$$1.60 \leq height \leq 1.90, 60 \leq weight \leq 90$$

Assuming the real target function is a rectangle (this assumption will be addressed later on). Our goal is to find the best rectangle R' that approximates the target rectangle target R. In general, we will try to learn an accurate predictor which will optimize our accuracy.

Generally, in the PAC model, we do not impose any assumptions on the underlying distribution of the examples other than that such a distribution exists and the examples are independently and identically distributed (i.i.d) according to that distribution. If the test examples were taken from a different distribution than the training examples, there is no reason to expect to learn a low error hypothesis. This critical assumption is at the base of almost all the machine learning approaches. The learning problem we described can be viewed as follows:

- Goal: Learn rectangle R.

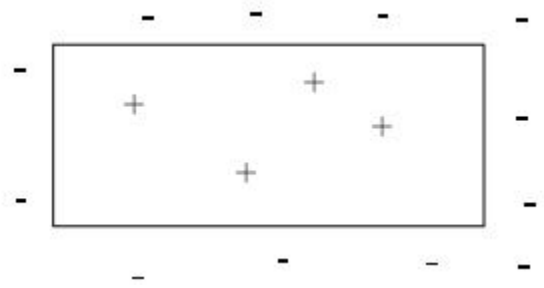

Figure 2.1: A rectangle with positive and negative examples

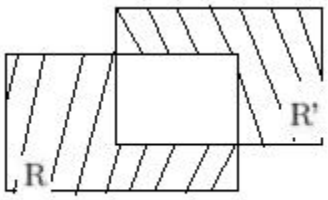

Figure 2.2: R and R' areas including two different error spaces. In our example, since $R' \subseteq R$ there exists only error of $(R - R')$.

- Input: Examples based on data set and their label: $\langle (x, y), +/- \rangle$.
- Output: R', a good approximation rectangle of the real target rectangle R. (An example of R', see Figure 2.1)

2.2.1 A Good Hypothesis

Our goal is to find a hypothesis that will have a small error rate, smaller than an input parameter ε. Let $R \Delta R'$ be the error of R' in respect to the real target rectangle R. This can be defined by two separate areas: $(R - R') \cup (R' - R)$ (where $(R - R')$ are *false negative* and $(R' - R)$ are *false positive*) as shown in Figure 2.2.

Then, our goal can be defined as to find $R' \in H$ such that with probability of at least $1 - \delta$ (confidence) such that,

$$\Pr[error] = \mathcal{D}(R \Delta R') \leq \varepsilon,$$

where we assume that $R \in H$.

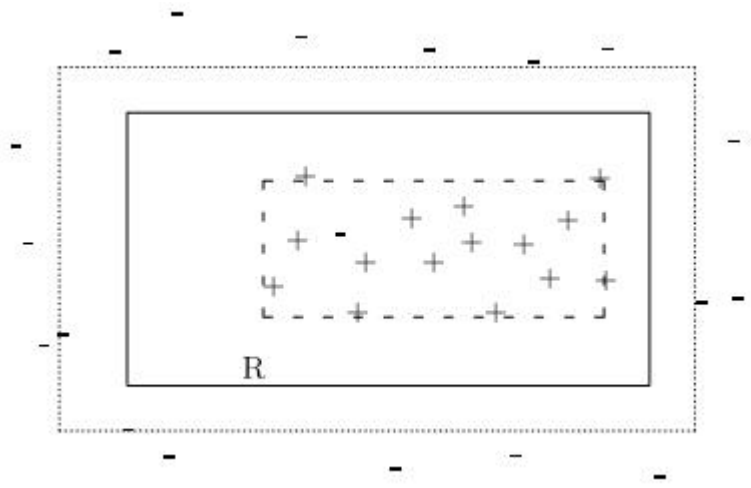

Figure 2.3: The smallest R_{min} (dashed line) and largest R_{max} (dotted line) rectangles border the rectangles which are consistent with the examples. The area between them represents the error region.

2.2.2 Learning Strategy

Let $S = \{\langle(x_1, y_1, b_1)\rangle, ..., \langle(x_m, y_m, b_m)\rangle\}$ be the sample. Intuitive, we would like a rectangle that will be *consistent*, i.e., zero error on the training data S. This is possible since we assumed that there exists a rectangle $R \in H$ (the target rectangle) that labels correctly any example. Even with the restriction to output a consistent rectangle $R' \in H$ we have a lot of freedom. We can choose any rectangle from the minimal size (R_{min}) up to the maximal size (R_{max}), as shown in Figure 2.3. We will show later, that it does not matter which consistent rectangle the algorithm returns.

2.3 A formal Presentation of the PAC Model

2.3.1 Preliminaries

- The goal of the learning algorithm is to learn an unknown target concept out of a known concept class, for example to learn a particular rectangle out of the set of all rectangles.
- Learning occurs in a stochastic setting. Examples from the target rectangle are drawn randomly according to a fixed, unknown probability distribution and are i.i.d.
- We assume that the training and testing samples are generated by the same unknown probability distribution.

- The algorithm should be efficient: the sample size $m(\varepsilon, \delta)$ required for obtaining small error (ε) with high confidence ($1 - \delta$) is a function of $\frac{1}{\varepsilon}$ and $\ln \frac{1}{\delta}$. To be efficient we like to have $m(\varepsilon, \delta)$ to be polynomial in $\frac{1}{\varepsilon}$ and $\ln \frac{1}{\delta}$. Also, the algorithm runs in time polynomial in the sample size.

2.3.2 Definition of the PAC Model

Let the set X be the *instance space* or *example space*. Let $\mathcal{D}$ be any fixed probability distribution over the instance space X. A *concept class* over X is a set

$$\mathcal{C} \subseteq \{c \mid c : X \to \{0, 1\}\}.$$

Let c_t be the *target concept*, $c_t \in \mathcal{C}$. Let and $h \in \mathcal{H}$ be the *learned hypothesis*, where the *hypothesis class* $\mathcal{H}$. Generally, $\mathcal{H}$ may be a different concept class than $\mathcal{C}$. We will define the *error* of h with respect to the distribution $\mathcal{D}$ and the target concept c_t as follows:

$$error(h) = \Pr_{\mathcal{D}}[h(x) \neq c_t(x)] = \mathcal{D}(h \Delta c_t(x))$$

Let $EX(c_t, \mathcal{D})$ be a procedure (we will sometimes call it an *oracle*) that runs in a unit time, and on each call returns a labeled example $\langle x, c_t(x) \rangle$, where x is drawn independently from $\mathcal{D}$. In the PAC model, the oracle is the *only* source of examples for the learning algorithm.

Definition Let $\mathcal{C}$ and $\mathcal{H}$ be concept classes over X. We say that $\mathcal{C}$ is *PAC learnable by* $\mathcal{H}$ if there exists an algorithm A with the following property: for every concept $c_t \in \mathcal{C}$, for every distribution $\mathcal{D}$ on X, and for all $0 < \varepsilon, \delta < \frac{1}{2}$, if A is given access to $EX(c_t, \mathcal{D})$ and inputs ε and δ, then with probability at least $1 - \delta$, A outputs a hypothesis concept $h \in \mathcal{H}$: If $c_t \in \mathcal{H}$ (Realizable case), then h is satisfying

$$error(h) \leq \varepsilon$$

If $c_t \notin \mathcal{H}$ (Non-realizable), then h is satisfying

$$error(h) \leq \varepsilon + \min_{h' \in \mathcal{H}} error(h')$$

We say that $\mathcal{C}$ is *efficiently PAC learnable*, if A runs in time polynomial in $\frac{1}{\varepsilon}$, $\ln \frac{1}{\delta}$, n and m, where n is the size of the instance from X and m the size of the target function (for example, the size can be the number of bits that are needed to encode it). Implicit, we mean that it is "easier" to learn a "simpler" target function.

2.4 Finite Hypothesis Class

In this section, we show how to learn a good hypothesis from a finite hypothesis class $\mathcal{H}$. The idea is to bound the probability that a hypothesis h is ε-*bad*, where a hypothesis h is ε-*bad* if $error(h) > \varepsilon$.

2.4.1 The Realizable case ($c_t \in \mathcal{H}$)

Generally, given $m(\varepsilon, \delta)$ example, we will request our algorithm A to find an h which is *consistent* with the training sample S, i.e., classifies all the examples in S correctly, which means that $\forall x \in S$ we have $h(x) = c_t(x)$. The algorithm A will succeed to learn if h is not ε-bad, i.e., $error(h) < \varepsilon$. We note that at least one *consistent* hypothesis exists because we assume that $c_t \in \mathcal{H}$ and clearly $error(c_t) = 0$.

We bound the probability of algorithm A to return h that is ε-bad by bounding by δ the probability that some consistent hypothesis is ε-bad. To start, fix an h which is ε-bad:

$$\Pr[h \ \ is \ \ \varepsilon\text{-bad} \ \& \ h(x_i) = c_t(x_i) \ for \ 1 \leq i \leq m] \leq (1-\varepsilon)^m < e^{-\varepsilon m},$$

where the first inequality is derived from the fact that since h is ε-bad we have $\Pr[h(x) = c_t(x)] \leq 1 - \varepsilon$ and the fact that the examples are sampled i.i.d from distribution $\mathcal{D}$. The second inequality follows since $1 - \epsilon \leq e^{-\epsilon}$ for any ϵ. Now we bound the probability of failure of algorithm A, as follows:

$$\begin{aligned}
\Pr[\text{A returns an } \varepsilon\text{-bad hypothesis}] &\leq \Pr[\exists h \in \varepsilon\text{-bad} \ \& \ h(x_i) = c_t(x_i) \ \forall \ 1 \leq i \leq m] \\
&\leq \sum_{h \in \varepsilon\text{-bad} \& h \in H} \Pr[h(x_i) = c_t(x_i) \ \forall \ 1 \leq i \leq m] \\
&\leq |\{h \ : \ h \ is \ \varepsilon\text{-bad} \ \& \ h \in H\}|(1-\varepsilon)^m \\
&\leq |\mathcal{H}|(1-\varepsilon)^m < |\mathcal{H}|e^{-\varepsilon m},
\end{aligned}$$

where the first inequality follows from the union bound, the second inequality follows from the analysis for a fixed ε-bad hypothesis, and the third inequality is immediate since,

$$|\{h \ : \ h \ is \ \varepsilon\text{-bad} \ \& \ h \in \mathcal{H}\}| \subseteq \mathcal{H}.$$

In order to satisfy the condition for PAC learning, we bound the failure probability by δ:

$$|\mathcal{H}|e^{-\varepsilon m} \leq \delta,$$

which lower bounds the sample size,

$$m \geq \frac{1}{\varepsilon} \ln \frac{|\mathcal{H}|}{\delta}.$$

We note that for a finite class of hypothesis $\mathcal{H}$, any consistent algorithm A PAC learns $\mathcal{C} = \mathcal{H}$ using H. As expected the bound tells us that the larger the hypothesis class $\mathcal{H}$, the larger the sample we require. However it is not clear how to implement the consistent algorithm efficiently, and in general it will not be efficient. Also, the above analysis only works for finite classes.

2.4.2 The Non-realizable case ($c_t \notin \mathcal{H}$)

In this case, we need to relax our goal, since a small error hypothesis might not exist in H. As we defined before, our goal is to learn according to the "best" hypothesis in the hypothesis class (i.e., with the minimal error). Let h^* be the hypothesis with the minimal error, i.e., for every $h \in \mathcal{H}$:

$$0 < error(h^*) \leq error(h)\,.$$

Let $\beta = error(h^*) = \min_{h \in \mathcal{H}} error(h)$. Our goal is to find hypothesis h that will achieve:

$$error(h) \leq \beta + \varepsilon\ .$$

Note that this is a generalization of our previous system (in which $\beta = 0$).

Given $m(\varepsilon, \delta)$ training examples S, we define $\widehat{error}(h)$ to be the empirical error of h on sample S. Formally:

$$\widehat{error}(h) = \frac{1}{m} \sum_{i=1}^{m} I(h(x_i) \neq c_t(x_i)),$$

where I is the indicator function.

We also need to define the algorithm, since now it may be impossible to choose a consistent h. Algorithm A will return a hypothesis $\bar{h}$ that will have the minimal empirical error. That is, $\bar{h} = \operatorname{argmin}_{h \in \mathcal{H}} \widehat{error}(h)$ (If there exists more than one hypothesis, the algorithm will pick one of them.) This algorithm is called *Empirical Risk Minimization (ERM)*.

We will now bound the sample size required for obtaining a good hypothesis using ERM. We want to choose a sample size $m(\epsilon, \delta)$ such that for any hypothesis, the difference between the true and the observed error is small. That is, with probability at least $1 - \delta$:

$$\forall h \in \mathcal{H}, |\widehat{error}(h) - error(h)| \leq \frac{\varepsilon}{2}.$$

If we have a uniform good error estimation, we can derive easily that using the hypothesis $\bar{h}$ from algorithm ERM, we obtain:

$$error(\bar{h}) \leq \widehat{error}(\bar{h}) + \frac{\varepsilon}{2} \leq \widehat{error}(h^*) + \frac{\varepsilon}{2} \leq error(h^*) + \varepsilon,$$

where the first and third inequality hold since the difference between the true and the observed error is small (up to $\frac{\varepsilon}{2}$), and the second inequality holds since ERM selects the hypothesis with the minimal empirical error.

To conclude our bound for the sample size we need to bound the probability of failure in estimating $error(h)$. We will use Chernoff bound[1] and derive that,

$$\Pr[\ |\widehat{error}(h) - error(h)| \geq \frac{\varepsilon}{2}] \leq 2e^{-2(\frac{\varepsilon}{2})^2 m}$$

We can use Chernoff bounds since the m training examples are drawn i.i.d from the same distribution $\mathcal{D}$. We can get a bound on the sample size by requiring the probability of failure over $\mathcal{H}$ to be smaller than δ:

$$\Pr[\exists h \in \mathcal{H}\ :\ |\widehat{error}(h) - error(h)| \geq \frac{\varepsilon}{2}] \leq 2|\mathcal{H}|e^{-2(\frac{\varepsilon}{2})^2 m} \leq \delta\,,$$

hence

$$m \geq \frac{2}{\varepsilon^2} \ln \frac{2|\mathcal{H}|}{\delta}\,,$$

which establishes a bound on the sample size for PAC learning when $c_t \notin \mathcal{H}$. Note that the sample size depends on $\frac{1}{\varepsilon^2}$ instead of $\frac{1}{\varepsilon}$ in the realizable case ($c_t \in \mathcal{H}$). This results from the difference between requiring a single counter-example to disqualify a hypothesis (since for some hypothesis $error(h) = 0$), to the requiring many examples to disqualify a hypothesis, by estimating the observed error.

2.4.3 Example - Learning Boolean Disjunctions

To demonstrate the PAC model we consider the example of learning Boolean disjunction functions. The problem is defined as follows: Given a set of boolean variables $T = \{x_1, ..., x_n\}$ and a set of literals $L = x_1, \bar{x}_1, \ldots x_n, \bar{x}_n$. we need to learn an OR function over the literals, for example: $x_1 \vee \bar{x}_3 \vee x_5$. Let $\mathcal{C}_{or}$ will be the set of all possible disjunctions. We have that $|\mathcal{C}_{or}| = 3^n$, since for each variable x_i the target disjunction c_t may contain x_i, $\bar{x}_i$, or neither. We will assume $\mathcal{H} = \mathcal{C}$.

ELIM algorithm for learning boolean disjunctions

Given a negative example we can eliminate all literals that evaluate to 1. For example, if given an example 001 (assume 0 was the value of x_1 and x_2 and 1 was the value of x_3) which was negative we can eliminate the literals $\bar{x}_1, \bar{x}_2, x_3$. This is valid since if one of them was in the target disjunction the target function would be positive. The algorithm uses this fact and eliminates all the inconsistent literals. We also notice that the algorithm classifies the

[1] see a summary of concentration bounds in the Appendix

positive examples correctly because the literal in the target disjunction are never eliminated, and we have $c_t \subseteq L_{\text{final}}$.

The algorithm ELIM initializes a set $L = \{x_1, \bar{x}_1, \ldots x_n, \bar{x}_n\}$, and for each negative sample Z it updates $L = L - \{\bar{z}_i | z_i \in Z\}$. From previous sections we can see that the algorithm learns when the sample size:

$$m > \frac{1}{\varepsilon} \ln \frac{|\mathcal{H}|}{\delta} = \frac{1}{\varepsilon} \ln \frac{3^n}{\delta} = \frac{n \ln 3}{\varepsilon} + \frac{1}{\varepsilon} \ln \frac{1}{\delta}$$

2.4.4 Example - Learning parity

Consider the concept class of xor of a subset of n Boolean variables. More formally: A set of variables : $x_1, ..., x_n$. The concepts class is the set of all XOR functions over the variables. Example for a function in $\mathcal{C}_{xor}$ is : $x_1 \oplus x_7 \oplus x_9$.

We notice that $|\mathcal{C}_{xor}| = 2^n$ since any subset yields a function in the class. Allowing literals will increase the size only by a factor of 2 since $\bar{x}_i = x_i \oplus 1$ and $1 \oplus 1 = 0$. Thus, allowing literals will increase the size of $\mathcal{C}_{xor}$ by a factor of 2. The algorithm will regard the samples as a linear equation problem (in Z_2 field) and then solve it using Gauss elimination method. Each sample consists of a bit vector (satisfying the value of the variable) and the classification c_t of this example. This creates a linear equation, for example: $(01101, 1)$ represents the equation: $0 * a_1 + 1 * a_2 + 1 * a_3 + 0 * a_4 + 1 * a_5 = 1(\mod 2)$ (deduced directly from the properties of xor). Each a_i is a binary indicator of x_i in the formula. A solution to all the examples exists (since we want to learn a xor function). Thus, solving the linear equation problem will produce a solution which is consistent with the whole sample The needed sample size m is

$$m > \frac{1}{\varepsilon} \ln \frac{|\mathcal{H}|}{\delta} = \frac{n}{\varepsilon} \ln 2 + \frac{1}{\varepsilon} \ln \frac{1}{\delta}.$$

2.5 Lower bound: No free lunch theorem

We will show that our sampling bound is essentially tight. We will start with the dependency on the size of the hypothesis class $|\mathcal{H}|$. We will fix a hypothesis class that will have all possible functions over a finite domain X. This implies that $|\mathcal{H}| = 2^{|X|}$. We will consider the realizable case, i.e., $c_t \in \mathcal{H}$, and select c_t at random from $\mathcal{H}$. This implies that for any point $x \in X$ we have that $\Pr[c_t(x) = 1] = 1/2$. For the distribution $\mathcal{D}$ we take the uniform distribution over X, i.e., each $x \in X$ has probability $1/|X|$.

Assume that we sample only $0.5 \log_2 |\mathcal{H}| = |X|/2$ examples. We claim that in such a case any algorithm will have expected error of at least $1/4$.

Since we sample only $|X|/2$ examples, there are at least $|X|/2$ points we did not sample. For any such point $x \in X$, the probability that $c_t(x) = 1$ is exactly $1/2$, regardless of the

sample. This implies that regardless what our hypothesis will predict on x it will make an error with probability exactly $1/2$. Therefore the error rate would be at least $1/4$. (Note that this holds in the non-realizable case also, where the learner can output any hypothesis.)

We can now consider the dependency on the accuracy parameter ϵ and δ. Assume we have only two function $\mathcal{C} = \{c_1, c_2\}$ and $\Pr[c_1(x) \neq c_2(x)] = 4\epsilon$. We set c_t to be either c_1 or c_2, each with probability $1/2$. We will consider an arbitrary $\mathcal{H}$.

Note that for any function h we have either $\Pr[h(x) \neq c_1(x)] > \epsilon$ or $\Pr[h(x) \neq c_2(x)] > \epsilon$. This implies that we cannot have a function h which is good both for c_1 and c_2. Therefore, essentially, any good prediction function h is identifying whether $c_t = c_1$ or $c_t = c_2$.

Assume we have a sample S. If on the entire sample S we have that c_1 and c_2 agree, we clearly cannot distinguish between them. The probability that they will agree on m points is,

$$\Pr[c_1(x_i) = c_2(x_2) : 1 \leq i \leq m] = (1 - 4\epsilon)^m$$

Since in such a case we cannot distinguish whether c_t is c_1 or c_2, it implies that the failure probability is at least $(1 - 4\epsilon)^m$. If we have

$$m < \frac{\log \delta}{\log(1 - 4\epsilon)} \approx \frac{1}{4\epsilon} \log \frac{1}{\delta}.$$

Then,

$$\delta < (1 - \epsilon)^m$$

which means that the algorithm fails with probability larger than δ. This implies that the sample size has to be at least $\Omega(\epsilon^{-1} \log \delta^{-1})$.

2.6 Remark on infinite classes

We will extensively discuss hypothesis classes which have continuous parameters in the remaining of this lecture and next lecture. However, in many cases we can reduce infinite hypothesis classes to finite one by doing discretization of the parameters. Our computer programs never have real numbers, most likely they use 64 bit precision.

Consider for example a hyperplane hypothesis $sign(\sum_{i=1}^{d} a_i x_i - \theta)$. We have $d + 1$ real value parameters, but if we use the computer discretization, it implies that we have only $2^{64(d+1)}$ hypotheses. Our sampling bound would give a dependency of the form $\log |\mathcal{H}| = 64(d + 1)$. This gives also the intuition why should we expect the number of different parameters would influence our sample size.

Using discretization is more of a "trick". In what follow we will give a more interesting and a better methodology to handle infinite size hypothesis classes. We will start by studying a very simple class, and try to give a direct proof. Next lecture we will develop a general methodology to handle those cases.

2.7 Infinite Hypothesis Class

The theoretical approach in this case will be briefly presented later in this lecture and in greater detail in a future talk. Here is another example:

Let X be the interval $[0, 1]$.

Let $\mathcal{H}$ be a concept class over X:

$$\mathcal{H} = \{c_\theta(x) \mid 0 \leq \theta \leq 1\}$$

and

$$c_\theta(x) = \begin{cases} 1 & x \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

Let $c_t(x) = c_{\theta^*}(x)$.

2.7.1 The realizable case $c_t \in \mathcal{H}$

Proof I

Let us choose m examples and define:

$$min\{x \mid c_t(x) = 1\} \; = \; max,$$

and

$$max\{x \mid c_t(x) = 0\} \; = \; min.$$

We known that $\theta^* \in [min, max]$.

We shall choose a value $\theta \in [min, max]$ and return c_θ. It is enough to show that with probability $1-\delta$ the weight of the interval $[min, max]$ under $\mathcal{D}$ is at most ε. This is sufficient because errors will occur only on this interval. Let's analyze the probability that m samples will not be taken from the interval $[min, max]$. Suppose

$$\mathcal{D}([min, max]) \geq \varepsilon \; .$$

The probability that we did not sample in $[min, max]$ is $(1-\varepsilon)^m$, then using the inequality $(1-x) \leq e^{-x}$ we get:

$$(1-\varepsilon)^m \leq e^{-\varepsilon m} \leq \delta$$

so it is enough that:

$$m \geq \frac{1}{\varepsilon} ln \frac{1}{\delta}.$$

This proof is wrong - why?

Note that min and max where determined only by the sample. Therefore, in the sample there is point between them! Consequently we can not make a probabilistic assumption on

the sample, since it is already fixed. To correct we will need to define the events independent of the sample.

Proof II

We will define parameters which do not depend on the sample. Let $max^{'}$ and $min^{'}$ be the points which are $\varepsilon/2$ close to θ^*. Namely, $max^{'} : D[\theta, max^{'}] = \varepsilon/2$ and $min^{'} : D[min^{'}, \theta^*] = \varepsilon/2$. Note that θ is the target we try to learn. The goal is to show that with high probability $(1-\delta)$: $max \in [\theta^*, max^{'}]$, and $min \in [min^{'}, \theta]$. In this case any value between $[min, max]$ is good and could be returned by the algorithm, since now $D([min, max]) < \varepsilon$, and D is the real distribution (not just the sample).

We have that

$$Prob[x_1, x_2, \ldots, x_m \notin [min^{'}, \theta^*]] = (1 - \frac{\varepsilon}{2})^m < e^{\frac{-m\varepsilon}{2}}$$

The failure probability for $[\theta^*, max^{'}]$ is derived similarly. Finally we need that $2e^{\frac{-m\varepsilon}{2}} < \delta$, which implies that we need a sample size $m > \frac{2}{\varepsilon} ln \frac{2}{\delta}$. Recall that $min^{'}, max^{'}$ where chosen with no dependence on the sample, but did depend on the target.

2.7.2 The non-realizable case $c_t \notin \mathcal{H}$

This case corresponds to noisy data or a more complicated labeling function than those represented by $\{c_\theta\}$. In this case there might be multiple best function h which are far from each other. This will require a different approach.

For a sample size m, there are only $m+1$ possible hypotheses according to the intervals between samples (the values of θ within an interval are equivalent in the sense that they produce the same learning error). We can thus choose the hypothesis which will minimize the error on the examples. The main issue is to bound the error of the best hypothesis on the sample compared to the best overall hypothesis.

For a given distribution $\mathcal{D}$, let $0 = z_0 < z_1 \cdots < z_k = 1$ such that $\mathcal{D}([z_i, z_{i+1}]) = \frac{\varepsilon}{4}$ (this is called a $\frac{\varepsilon}{4}$ net). Let h_{alg} be the output of the learning algorithm, h^* an optimal hypothesis, and h_{z_i} the hypothesis with $\theta = z_i$. Our definition of the $\{z_i\}$ implies that for any h there exist z_k such that:

$$|error(h_{z_k}) - error(h)| \leq \frac{\varepsilon}{4}$$

We will need another property, which would apply to the observed error. Given a sample S let $\widehat{error}(h)$ be the observed error of h on the sample. We would like that for any sample S, we have

$$|\widehat{error}(h_{z_k}) - \widehat{error}(h)| \leq \frac{3\varepsilon}{8},$$

Note that if in the interval between z_k and the threshold θ_h of h, we have at most $\frac{3\varepsilon}{8}$ point, then this property will hold.

We will also show that when the sample size is large enough, for every z_i, with a high probability, $|error(h_{z_i}) - \widehat{error}(h_{z_i})| \leq \frac{\varepsilon}{16}$.

Given the above three assumptions we can show that the error of the selected hypothesis h_{alg} is at most ε.

Let h_{z_k} be such that both $|error(h_{z_k}) - error(h_{\text{alg}})| \leq \frac{\varepsilon}{4}$ and $|\widehat{error}(h_{z_k}) - \widehat{error}(h_{\text{alg}})| \leq \frac{3\varepsilon}{8}$. Then we have that

$$\widehat{error}(h_{\text{alg}}) \geq \widehat{error}(h_{z_k}) + \frac{3\varepsilon}{8} \geq error(h_{z_k}) + \frac{5\varepsilon}{8} \geq error(h_{\text{alg}}) + \frac{11\varepsilon}{16}$$

Let h_{z_i} be such that both $|error(h_{z_k}) - error(h^*)| \leq \frac{\varepsilon}{4}$. This implies

$$error(h^*) \geq error(h_{z_i}) + \frac{\varepsilon}{4} \geq \widehat{error}(h_{z_i}) + \frac{5\varepsilon}{16}$$

Since we are using ERM we have, by definition, that $\widehat{error}(h_{\text{alg}}) \leq \widehat{error}(h)$ for any h, and hence $\widehat{error}(h_{\text{alg}}) \leq \widehat{error}(h_{z_i})$. This implies that

$$error(h^*) \geq \widehat{error}(h_{z_i}) + \frac{5\varepsilon}{16} \geq \widehat{error}(h_{\text{alg}}) + \frac{5\varepsilon}{16} \geq error(h_{\text{alg}}) + \frac{5\varepsilon}{16} + \frac{11\varepsilon}{16} = error(h_{\text{alg}}) + \varepsilon$$

We now need to show how do we achieve the desired properties with high probability. For any z_i we have that

$$Pr[\, |error(h_{z_i}) - \widehat{error}(h_{z_i})| \geq \frac{\varepsilon}{16}] \leq 2e^{-2(\frac{\varepsilon}{16})^2 m}$$

Since the number of z_i's is $\frac{4}{\varepsilon}$, the probability for some z_i having an estimation error larger than $\frac{\varepsilon}{16}$ may be bounded by

$$Pr[\exists i \; |error(h_{z_i}) - \widehat{error}(h_{z_i})| \geq \frac{\varepsilon}{16}] \leq \frac{8}{\varepsilon} \cdot e^{-2(\frac{\varepsilon}{16})^2 m} < \delta \,.$$

Next we need to bound the condition on the observed error. For this it is sufficient that we will not sample more than $3\varepsilon/8$ in any interval $[z_i, z_{i+1}]$. Recall that the probability of such an interval is $\varepsilon/4$. Therefore,

$$Pr[|[z_i, z_{i+1}] \cap S| \geq \frac{3}{8}\varepsilon] \leq 2e^{-2(\varepsilon/8)^2 m}$$

and when we sum over all such intervals we get a bound of

$$Pr[\exists |[z_i, z_{i+1}] \cap S| \geq \varepsilon m/8] \leq \frac{8}{\varepsilon} e^{-2(\varepsilon/8)^2 m}$$

Therefore, if we choose a sample whose size is:

$$m \geq \frac{128}{\varepsilon^2}[\ln \frac{8}{\varepsilon} + \ln \frac{2}{\delta}] \,,$$

we are guaranteed that the output hypothesis h_{alg} will have error at most ϵ more than h^* with probability $1 - \delta$.

2.8 Concentration bounds

Here are a few simple concentration bounds from probability theory.

Markov Inequality

Let X be a non-negative random variable (r.v.) and $E[X]$ the *expected value* (mean) of X, then:

$$Pr[X \geq \alpha] \leq \frac{E[x]}{\alpha}$$

2.8.1 Chebyshev Inequality

Suppose that X is arbitrary with expectation $E[X] = \eta$ and variance $Var(X)$, then:

$$Pr[|x - \eta| \geq \beta] \ \leq \ \frac{Var(x)}{\beta^2}$$

2.8.2 Chernoff Inequality

Let the sequence of random variables $X_1, \ldots, X_n$ where $X_i \in \{0, 1\}$ be independent identically distributed (i.i.d.), and $Pr[X_i = 1] = p$, then:

$$Pr[|\sum_{i=1}^{n} \frac{X_i}{n} - p| \ \geq \ \lambda] \ \leq \ 2e^{-2\lambda^2 n}$$

The last inequality is especially interesting, since it gives us a tool to argue "how fast" the "observed mean" would converge to the "true mean". It can be viewed as a "quantitative form of the Law of Large Numbers.