Recitation on Generalization Bounds

ESE 546

November 2020

1 Introduction

In this recitation we will take a step back from both deep learning architectures as well as learning algorithms for training those architectures, and discuss about learning in an abstract, formal way. In order to do this, we need to introduce a "learning model", ie. a formal description of what constitutes learning, in a way that makes our assumptions and desiderata explicit. We will mainly focus on binary classification, as this is the prototypical example for supervised learning.

The main contributors of this discussion will be: Leslie Valiant and Vladimir Vapnik. The first is a Harvard professor that invented the most popular learning model called *Probably Approximately Correct Learning* (PAC-learning). The second is a Russian statistician that had already developed a theory (*VC-theory*) that provided a definitive answer on the class of functions that were learnable under the PAC model. His coming to US, literally altered the field of Machine learning. You may know him as an inventor of the SVMs too.

2 PAC learning

As we have seen many times until now, the main task in machine learning is to exploit the available (relatively few) supervised data, in order to construct a model that generalizes well, ie. has good performance, outside of the training data. We usually know the *instance space* $\mathcal{X}$ the data live in (eg. 28×28 grayscale images) and we search over a *hypothesis space* $\mathcal{H}$ (eg. a specific neural net architecture) using the available data, in order to find a good *hypothesis* $h \in \mathcal{H}$. But why should we expect this hypothesis to work on brand-new data? To justify that, we need to make some assumptions about the data acquisition process. The main assumption in this model of learning is that:

The Nature provides i.i.d. samples $x \in \mathcal{X}$ from some (unknown) distribution $\mathcal{D}$.
There is some (unknown) "true labeler" $c(x)$, that annotates those samples.

Note the implicit assumption following from the fact that there is some latent data distribution, ie. *both our training and our test data are samples from the same distribution*. This assumption is lifted in other learning paradigms like eg. transfer learning. Moreover, since there is a "target" function, this is what our learning algorithm should aim for. We will also assume for our subsequent analysis that $c(x) \in \mathcal{C}$, where $\mathcal{C}$ is a known function class (called *concept class*) and that $\mathcal{H} = \mathcal{C}$. Lifting those assumptions will not change our discussions qualitatively, but we are going to do it to see what happens quantitatively in the following sections. The fact that the target concept is deterministic as well as the fact that the data are "clean" (no noise; no adversaries) are two more assumptions that someone might consider extending.

The learning algorithm L (assume deterministic; not important for now), will take as input some annotated i.i.d. data $\mathrm{S} = \{< x_i, c(x_i) >\}_{i=1}^{m}$ and output a hypothesis $h : \mathcal{X} \to \{0, 1\} \in \mathcal{H}$. We talked about "good performance on brand new data" and to quantify that, we define the error of a hypothesis as: $e(h) = \mathbb{E}_{x \sim \mathcal{D}}[1_{h(x) \neq c(x)}] = \mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq c(x)]$ (0-1 loss/accuracy).

What is reasonable to expect from our learning algorithm?

1. We should expect it to "work" for all $c(x) \in \mathcal{C}$ and any $\mathcal{D}$ on $\mathcal{X}$. These conditions, especially the second one, might seem strict but it is a "safety" condition since both the target and distribution are unknown. It is what we call "worst-case analysis" in computer science.
2. What does "work" mean above? It means mainly 2 things:

- We can accept outputting hypotheses with some error $\epsilon \in [0, 1/2)$ for a given amount of data, but this error should decrease as we get more data. That is the "Approximate" part of learning.
- We should not forget that $h = h(S)$ and S is a random variable. So the above statement holds probabilistically over the random datasets S drawn. There is always some probability that we sample very non-representative samples from $\mathcal{D}$, in which case we cannot expect to output a good hypothesis (error could be random or worse). This probability of "failure" $\delta \in [0, 1/2)$ should also become smaller with the amount of data provided. This is the "Probabilistic" part of learning.

To sum everything up:

Definition: Let $\mathcal{C}$ be a concept class over $\mathcal{X}$. We say that $\mathcal{C}$ is *PAC-learnable* if there exists an algorithm L with the following property: for every concept $c \in \mathcal{C}$, for every $\epsilon, \delta \in [0, 1/2)$, if L is given access to $m = m(\epsilon, \delta)$ i.i.d. samples from D, and their labels from c, then it outputs a hypothesis $h_S \in \mathcal{C}$ such that: $\mathbb{P}_{S \sim \mathcal{D}^m}[e(h_S) < \epsilon] \geq 1 - \delta$.

Remark: Two key requirements that we also expect from the algorithm L are:

1. (Statistical) $m = m(\epsilon, \delta)$ is a decreasing function of both arguments, however as ϵ, δ grow smaller we should expect a reasonable behavior from the amount of samples L can request, eg. polynomial in $1/\epsilon, 1/\delta$. The minimum m to "PAC-learn" $\mathcal{C}$ is called *sample complexity* of $\mathcal{C}$.
2. (Computational) L should also run in polynomial time w.r.t. $1/\epsilon, 1/\delta$ and w.r.t. $size(c)$. This is to avoid algorithms that search over exponentially large Hypothesis classes in a brute force way, as well as to avoid computational blow up when for example we request half the error of a previous run.

Observe that PAC learning assumes nothing about how L will choose to use the data. We will only talk about *Empirical Risk Minimization* here, but other learning rules are not forbidden.

Example (Learning Monotone Boolean Formulae):

- $x = (x_1, \cdots, x_n) \in \mathcal{X} = \{0, 1\}^n$, $c(x)$ is a conjunction of some $\{x_i\}_{i=1}^n$. Eg. $c(x) = x_1x_3x_4$. So, $c(10011) = 0, c(11110) = 1$, etc.
- $\mathcal{H}$ is also a class function containing all possible conjuctions of $\{x_i\}_{i=1}^n$. How big is $\mathcal{H}$? Each literal can either exist in the conjunction or not. So, $|\mathcal{H}| = 2^n$. This is exponential in n, so we cannot brute force search all hypothesis, we need to cut down the space by a more clever algorithm.
- Since $c(x) \in \mathcal{H} \implies \min_{h \in \mathcal{H}} 1/m \sum_{i=1}^n I[h(x^{(i)}) \neq c(x^{(i)})] = 0$, but for a fixed amount of data there is some probability that many hypotheses have zero training error, even if the expected error is high. As we expect, with more i.i.d. samples from the data distribution this is less and less probable.
- Imagine some algorithm L that does the following:
  1. It starts from the hypothesis $h_0(x) = x_1 \cdots x_n$, of all literals present.
  2. For every positively labeled sample $< (x_1, \cdots, x_n), 1 >$ it deletes all x_i's that are 0.
- **Note**: (From recitation) Mind that since $c(x) \in \mathcal{H}$ it is not possible to see an example: $< (1, \cdots, 1), 0 >$.
- What kind of errors does this algorithm make? If some literal x_i was deleted, it is because it had the value $x_i = 0$ on a positively labeled sample. So, it should be deleted otherwise the hypothesis will output 0. So, the only risk is if we output a hypothesis with more literals present than those in $c(x)$. So, the output $h(x)$ can only err on positive examples of $c(x)$, never on negative. Only false negatives! We intuitively can see why requesting more samples, diminishes the probability of this event happening.
- $p_i = \mathbb{P}_{x \sim D}[c(x) = 1, x_i = 0 \; in \; x]$. $e(h) \leq \sum_{x_i \in h} p_i$.

- If some p_i is small, then it does not contribute much to the error. If some p_i is large then we make sure to see enough samples so that we remove that x_i from h. After all, it only takes one appearance of this event to delete this x_i, and the event has probability p_i which is large.

- Rigorously, if all x_i in h have $p_i < \epsilon/n$, then: $e(h) < \epsilon$. On the other hand, if some x_i has $p_i > \epsilon/n$, then the probability of having this x_i in h, is the probability, the event of p_i never happens in the draw of m samples. But this new probability is smaller than $1 - \epsilon/n$. And the event will never happen in m i.i.d. draws with probability smaller than $(1 - \epsilon/n)^m \leq e^{-m\epsilon/n}$. Using the union bound, since there are at most n literals in h, the probability that there is at least one such "bad event" is at most $ne^{-m\epsilon/n}$.

- If this bad event never happens, as we analyzed the error is less than ϵ. Of course, such a bad event happening would be devastating. For some distributions it could lead the error up to 1. However, in our setting we can accept that, as long as it happens rarely, namely with probability at most δ. So if, $ne^{-m\epsilon/n} < \delta \iff m \geq \frac{n}{\epsilon} \ln \frac{n}{\delta}$, we guarantee to meet the PAC criteria, of error less than ϵ, with probability at least $1 - \delta$.

- Both sample complexity and time are polynomial as needed so the class of Monotone Boolean Formulae is (ϵ, δ)-PAC learnable.

3 Union and Chernoff Bounds

Two very important results from probability theory that we will use or already used are the *Union Bound* and *Chernoff Bound*.

3.1 Union Bound (or Boole's Inequality)

For any countable set of events, $\{A_1, \cdots, A_n, \cdots\}$,

$$\mathbb{P}[\bigcup_i A_i] \leq \sum_i \mathbb{P}[A_i]$$

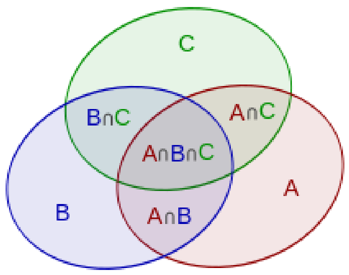
This is a rather loose, but sometimes useful, upper bound and is (mostly) embedded in the assumptions of what we call "probability measure" in probability theory (σ-subadditivity). By this I mean that it can be used without any assumptions, in practice.

If we want a better approximation of the union, and we know more about the problem at hand, we can use the Bonferroni inequalities. By the inclusion-exclusion principle,

For finite set of events, $\{A_1, \cdots, A_n\}$,

$$\mathbb{P}[\bigcup_{i=1}^{n} A_i] = \sum_{i=1}^{n} \mathbb{P}[A_i] - \sum_{1 \leq i < j \leq n} \mathbb{P}[A_i, A_j] + ... + (-1)^{n-1}\mathbb{P}[A_1, A_2, \cdots, A_n]$$

1. We can get better approximations of the union, if we use the first $k \leq n$ terms above.
2. If we stop at odd k, we get an upper bound. If we stop at even k we get a lower bound. The error of the approximation is decreasing with k.



1. Question: Where was this inequality used in the example above.
2. Exercise: Try to prove that $\mathbb{P}[\bigcap\limits_{i=1}^{n} A_i] \geq 1 - \sum_{i=1}^{n} \mathbb{P}[A_i^c]$

3.2 Measure Concentration Inequalities

Let $X_1, \cdots, X_n$ be a sequence of i.i.d. random variables. We focus on the case of Bernoulli random variables where $\mathbb{P}[X_i = 1] = p$. The question is to: "estimate the bias p of the coin from samples". In other words, we need to quantify how close is the "estimator" $\hat{p} = \sum_{i=1}^{n} X_i$ to p. You already may see the connection with Machine learning through the Population and the Empirical Risk in binary classification.

What do we know from the laws of large numbers as $n \to \infty$?

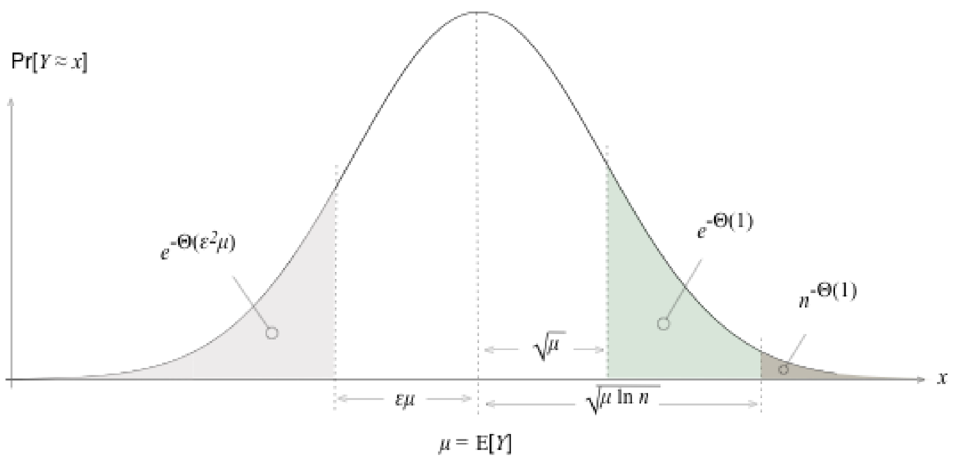
- *Weak Law*: $\forall \epsilon > 0,\ \mathbb{P}[|\hat{p} - p| > \epsilon] \to 0$, as $n \to \infty$.
- *Strong Law*: $\mathbb{P}[\hat{p} \xrightarrow{n \to \infty} p] = 1$.

But we need more than that. We need a "rate of convegence":

- *Central limit Theorem*: $\sqrt{n}(\hat{p} - p) \xrightarrow[d]{n \to \infty} \mathcal{N}(0, \sigma^2 = p(1-p))$, or $\mathbb{P}[|\hat{p} - p| > t] \leq 2e^{-nt^2/2\sigma^2}$. "Bounds on the tails of the normal distribution"
- *Chernoff Bound* Observe that $\sigma^2 \leq 1/4$.

$$\mathbb{P}[|\frac{1}{n}\sum_{i=1}^{n} X_i - p| > \epsilon] \leq 2e^{-2n\epsilon^2}$$

.



Question: How many "coin flips" would you need in order to achieve error smaller than a given ϵ with confidence bigger than some given δ? (Using the Chernoff bounds above). Do you see any patterns with the bounds on sample complexity from the PAC learning example above?

The measure concentration theorems extend to nearly any distribution. The most popular inequalities in this field are: Markov Inequality, Chebyshev Inequality and Chernoff Bounds (and Hoeffding's Inequality as an important special case). They are written in terms of increasing tightness, but also of increasing assumptions of what we need to know in order compute them. In general, Markov's inequality only needs the expectation, Chebyshev's Inequality needs the variance too, while Chernoff bounds usually need the whole moment generating function. They are all applications of Markov's Inequality on higher order statistics. The value of Chernoff bounds is increasingly more important when we talk about distributions of few sufficient statistics, like the Bernoulli distribution (or any exponential distribution).

4 Generalization Bounds and Uniform convergence

Let's now lift the assumption that $c(x) \in \mathcal{C}$, after all, even if there exists such true deterministic $c(x)$ we can never be sure that is inside let's say the class of neural networks of specific architecture that we fit the data. This model is called *Agnostic PAC learning.* We will stay in the confinements of Emprirical Risk Minimization where we are provided with some samples $S = \{< x_i, y_i) >\}_{i=1}^m$ from a distribution $\mathcal{D}$ and we output the hypothesis h with the minimum training error ie.:

$$\hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} 1_{h(x_i) \neq y_i} \xrightarrow[h \in \mathcal{H}]{\min} h_{ERM} \in \mathcal{H}$$

The corresponding generalization error (0-1 loss) is:

$$R(h) = \mathbb{E}_{(X,Y) \sim \mathcal{D}}[1_{h(X) \neq Y}] \xrightarrow[h \in \mathcal{H}]{\min} h^* \mathcal{H}$$

Note: (from recitation) We should not confuse the *Bayes optimal predictor* with h^*. The latter is the predictor with minimum generalization error inside our class $\mathcal{H}$, while the former is the predictor with minimum generalization error (unconstrained).

Although we do not have a target c now, nor a target training error 0 like before, we would still like to use samples, in order to increase our confidence that the training error of the ERM predictor, is a good estimator of the generalization error. We will do that using some *Generalization Bound* of the form:

$$|\hat{R}(h_{ERM}) - R(h_{ERM})| < \textit{something} \xrightarrow[m \to \infty]{} 0 \qquad \text{w.h.p.}$$

However, we want more than that! We want to somehow reason about the validity of the ERM framework in the first place. Does it pick a "good" enough hypothesis from the possible hypothesis in $\mathcal{H}$? Namely, we would like to make sure that with enough data, choosing the best classifier in the training error would induce a generalization error that is not much worse than the generalization error of the best possible classifier $h^* \in \mathcal{H}$.

A sufficient condition to achieve that, is to make sure that, for *all* hypothesis in $\mathcal{H}$ we can "trust" the training error, meaning that it is not very far away (same for all hypotheses) from the corresponding generalization error. This is called *Uniform Convergence.*

More rigorously since the samples are i.i.d. and the loss is 0-1, we can use the Chernoff bound from the previous section for all hypotheses $h \in \mathcal{H}$ (using the same samples m). Then,

$$\forall h \in \mathcal{H},\ \mathbb{P}[|\hat{R}(h) - R(h)| > \epsilon] \leq 2e^{-2m\epsilon^2}$$

Assuming a finite class $\mathcal{H} = \{h_1, \cdots, h_n\}$, we can bound the probability that ANY training error is far away from its corresponding generalization error, using the union bound:

$$\mathbb{P}[\exists h \in \mathcal{H} : |\hat{R}(h) - R(h)| > \epsilon] \leq \sum_{i=1}^{n} \mathbb{P}[|\hat{R}(h_i) - R(h)| > \epsilon] \leq \sum_{i=1}^{n} 2e^{-2m\epsilon^2} = 2ne^{-2m\epsilon^2}$$

If we bound this probability of the above "bad" event happening by a given δ we get that:

$$\mathbb{P}[\forall h \in \mathcal{H} : |\hat{R}(h) - R(h)| < \epsilon] \geq 1 - \delta$$

Solving for m, to find the *sample complexity* we get:

$$m \geq \frac{1}{2\epsilon^2} \ln \frac{2n}{\delta} \tag{1}$$

Remark: Observe how this bound changed from the example in PAC-learning. mainly we need $\mathcal{O}(\frac{1}{\epsilon})$ times more samples to get the Uniform Convergence result.

And how is the above result useful? As we analyzed above we need to find a relation between the generalization bounds of our ERM predictor and the best possible predictor inside the hypothesis space.

$$R(h_{ERM}) \leq \hat{R}(h_{ERM}) + \epsilon \leq \hat{R}(h^*) + \epsilon \leq R(h^*) + 2\epsilon, w.p. \geq 1 - 2\delta$$

where the first inequality is from the Chernoff bound on h_{ERM}, the second because h_{ERM} has the property that it is the best predictor w.r.t. the training error and the third from Uniform Convergence, as the ϵ error holds for every hypothesis in $\mathcal{H}$, with this amount of samples. Solving for ϵ in (1) we get:

$$R(h_{ERM}) \leq R(h^*) + 2\sqrt{\frac{1}{2m}\ln\frac{2|\mathcal{H}|}{\delta}} \qquad \text{w.p.} \geq 1 - 2\delta \tag{2}$$

Numerical Example: For the monotone Boolean conjunction example:

- $|\mathcal{H}| = 2^n$. Assume the feature vector has dimension $n = 1000$.
- Set $\delta = 0.001$.
- Remember: $R(h^*)$ is not zero anymore, since we removed the assumption that $c(x) \in \mathcal{C}$. Also, we do not run the algorithm L from above, but another L' (ERM algorithm) that outputs h_{ERM} with the smallest training error.
- Then using m samples we get:
  - $m = 1000$: $R(h_{ERM}) \leq R(h^*) + 1.41$ (vacuous/non-informative bound)
  - $m = 10000$: $R(h_{ERM}) \leq R(h^*) + 0.44$ (informative only if $R(h^*)$ is small as well; expressive hyp. class).
  - $m = 1000000$: $R(h_{ERM}) \leq R(h^*) + 0.04$

5 VC-dimension

Both the example hypothesis class in PAC learning and in the Uniform Convergence were finite, and in the example of boolean formulae, the number of literals n, which may be considered as a measure of the complexity of the class appeared also in the sample complexity. What if the concept class is infinite? (Then, (2) is vacuous). Can we still learn it using finitely many samples? Like the class of linear threshold functions, or the class of neural networks of some fixed architecture? Are they PAC learnable? What are the limits of learning? This question is answered by the VC-theory. Actually VC-theory transcents PAC-learning but we are only going to discuss about one aspect of it in the confinements of PAC-learning. We are going to assign a "complexity" number, for every hypothesis class $\mathcal{H}$.

1. We say that the set $S = \{x_1, \cdots, x_m\}$ is *shattered* by the concept class $\mathcal{C}$, if we can achieve every possible labelling out of the 2^m labellings using some concept $c \in \mathcal{C}$.
2. The size of the largest set S that can be shattered by $\mathcal{C}$ is called the VC-dimension of the class $\mathcal{C}$. It is a measure of the complexity/expressiveness of the class, by counting how many classification "behaviors" the class can express.

   Note:

   - If you find any configuration of n points in space, such that, when you assign any labeling to those points, you can find some function inside your class $\mathcal{C}$, that can realize this labeling, then: $VC(\mathcal{C}) \geq n$.
   - If for all possible configurations of $n+1$ points, you can always find some assignment of the labels, such that no function $c \in \mathcal{C}$ can realize this labelling, then: $VC(\mathcal{C}) < n+1$.
   - If you prove both the above then: $VC(\mathcal{C}) = n$.
3. Examples:
   - N-dim Linear Threshold Functions: VC-dim = N+1.

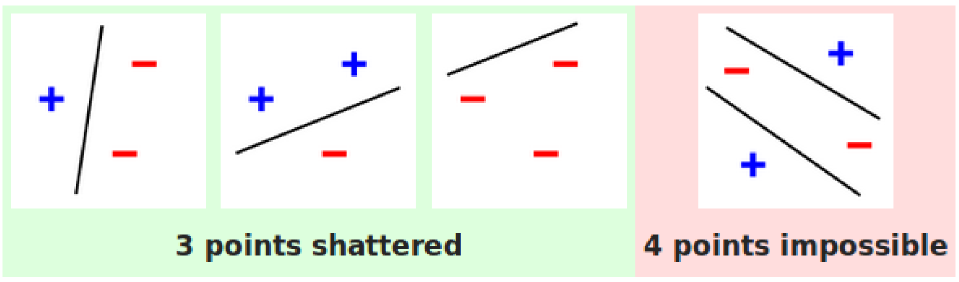


Figure 1: N=2: See that for the lower bound, we found some configuration of the 3 points, such that a linear threshold function always separates the points consistently with the labels; for any possible labeling. 3 such labelings are shown, convince yourselves that it can be done for all 8 cases. Observe that we cannot do the same for 4 points. In the figure above one such unrealizable configuration is given (With the "XOR" labeling). To prove the upper bound we need to talk about ANY configuration though. See that the only other case for 4 points, is that one point is inside the convex hull generated from the other 3. Find the labeling that cannot be obtained with linear classifiers in this case.

- 2 dimensional axis aligned rectangles: VC-dim = 4 (exercise)
- Monotone Boolean Formulae: VC-dim=n (exercise).
- If VC-dim is finite then, always VC-dim $\leq \ln |\mathcal{H}|$ (exercise).
- If our concept class includes classifiers of the form: $sgn(sin(\omega x))$, where ω is a learned parameter, then VC-dim $= \infty$.
- A feedforward network with N weights and sign activation function: VC-dim $\in \mathcal{O}(N \log N)$

If VC-dimension of a class $\mathcal{H}$ is finite (VC-dim $= d < \infty$) then this class has the uniform convergence property. And thus it is (agnostically) PAC-learnable (and ERM is a PAC learning algorithm for $\mathcal{H}$) with sample complexity:

$$m \in \mathcal{O}(\frac{d \ln(d/\epsilon) + \ln(1/\delta)}{\epsilon^2})$$

If the class has infinite VC-dimension then it is not PAC-learnable neither it has uniform convergence property.

The generalization bounds for (even infinite) classes with finite VC-dimension change to:

$$R(h_{ERM}) \leq R(h^*) + 2\sqrt{\frac{1}{m}[VC(\mathcal{H})(\log \frac{2m}{VC(\mathcal{H})} + 1) + \log \frac{4}{\delta}]} \qquad \text{w.p.} \geq 1 - \delta$$