Introduction to Machine Learning (67577)

Recitation 9
Ensemble Methods and Boosting

Second Semester, 2021

Contents

1 Weak Learnability

Given some distribution $\mathcal{D}$ and a set of *iid* samples over it $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$ we would like to find a good classifier that achieves minimum loss. As we restrict ourselves to some hypothesis class $\mathcal{H}$, clearly, the best classifier we could do is:

$$h^* = \underset{h\in\mathcal{H}}{argmin} L_{\mathcal{D}}(h)$$

As we do not know $\mathcal{D}$ we are not able to calculate $L_{\mathcal{D}}(h)$, and we will need to find other ways to solve/approximate the problem. At this point in the course, all we can do is either find a classifier $\hat{h}$ that minimizes $L_S(\hat{h})$ or replace our hypothesis class with one with a smaller VC-dimension. We will see such tradeoff in the example below (in subsection 1.1).
Another problem to address is the computational complexity. Assume we know how to find the best classifier, but finding it is NP-Hard (with respect to the number of samples, m, and the approximation and accuracy parameters, $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$). Can we call this classifier a good classifier? We'll discuss this in the second example (in section 1.2).

1.1 The Bias-complexity trade-off

Let $h^* := argmin_{h\in\mathcal{H}} L_{\mathcal{D}}(h)$ and $h_S = \mathcal{A}(S)$ be the output of a learning algorithm, then:

$$L_{\mathcal{D}}(h_S) = \underbrace{L_{\mathcal{D}}(h^*)}_{\varepsilon_{approximation}} + \underbrace{L_{\mathcal{D}}(h_S) - L_{\mathcal{D}}(h^*)}_{\varepsilon_{estimation}}$$

In many cases, the two parts above are coming one on the expense of the other. This cause a problem called the *Bias-Complexity Tradeoff*.

■ **Example 1.1** You're already familiar with linear classifiers, i.e. halfspaces, but this is a subfamily of **polynomial classifiers**. Given a polynomial $p : \mathbb{R}^d \to \mathbb{R}$, the classifier $h_p : \mathbb{R}^d \to \{\pm 1\}$ is defined as

$$h_p(x) = sign(p(x)).$$

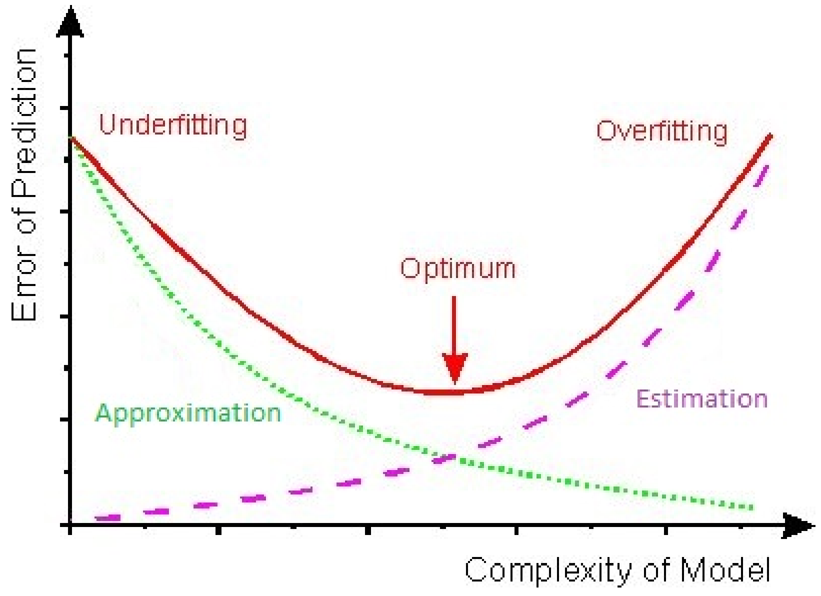
Denote the hypothesis class

$$\mathcal{H}_r = \{h_p : p \text{ is a polynomial of degree at most } r\}.$$

Note: $\mathcal{H}_0$ is the trivial all-zeros or all-ones classifier, and $\mathcal{H}_1$ is the linear classifier.
Now we get to the following question: How to choose r? ■

In the above example r represents some measure of the complexity of the hypothesis class, $\mathcal{H}_r$. As a matter of fact, you can actually measure the complexity using the VC-dimension. As complexity increases there are two competing effects:

- The *approximation error* decreases - Meaning that as your hypothesis class becomes richer you might find a better hypothesis for explaining the given S. Does not depend on the sample size m and is determined by the hypothesis class chosen.
- The *estimation error* increases - Meaning that as your hypothesis class becomes richer there will be more hypotheses that could be returned from the ERM algorithm and the algorithm is more sensitive to noise in the training set. Increases with the complexity of $\mathcal{H}$ and decreases with m.



Try to think, **How can we tune the complexity of the hypothesis class?** Sometimes the complexity cannot be simply measured by the VC-dimension, however, the effects of increasing it are the same. Throughout the course we have seen several examples for these tuning possibilities: k parameter in k-NN algorithm (k-Nearest-Neighbors), the depth of classification trees and even the number of features used in linear regression. A great explanation can be found **here**. The bias-variance tradeoff demonstrates that in many cases we prefer a simple estimator, with high approximation error, but with low overfitting.

1.2 Computational complexity

(Not to be confused with *sample complexity*!) We define an algorithm as *feasible* or *efficiently computable* if it is polynomial in the size of the input. Learning algorithms are not different, and computational efficiency is a crucial aspect when searching for a hypothesis class. Mainly, there are two issues to consider when analyzing the runtime of a learning algorithm:

- Runtime of finding a good hypothesis; and
- Runtime of predicting the label of a new example given this hypothesis.

An efficient learning algorithm is an algorithm that is polynomial with respect to the following values:

1. m - the sample set size.
2. ε^{-1} - the accuracy parameter.
3. δ^{-1} - the confidence parameter.

Try to explain to yourself why each of them is important for the computation complexity.

■ **Example 1.2 (Face classification)**
Assume: $\mathcal{X} = \{gray - scale\ pictures\ of\ 19 \times 19\ pixels\}$, $\mathcal{Y} = \{-1,1\}$,
$\mathcal{H} = \{f : [256]^{19 \cdot 19} \to \{-1,1\}\}$.
This is huge number of functions so we need some heuristics in order to learn this class. ■

As we can see, some classes are efficiently computable while others are not. In that sense, a simple hypothesis class is preferable.

1.3 Weak Learners

The "weak learners" is the answer for both problems. We can think of weak learners as a "rule of thumb": better than a random guess, but still it cannot output a hypothesis with very low error. Let's examine the weak learners concept.

Definition 1.1 — γ-Weak-Learner. A learning algorithm $\mathcal{A}$ is a γ-weak-learner for a hypothesis class $\mathcal{H}$ if there exists a function $m_{\mathcal{H}} : (0,1) \to \mathbb{N}$ such that:

- For every $\delta \in (0,1)$
- For every distribution $\mathcal{D}$ over $\mathcal{X}$
- For every labeling function $f : \mathcal{X} \to \{\pm 1\}$

if the realizable assumption holds with respect to $\mathcal{H}, \mathcal{D}, f$, then when running the learning algorithm on $m \geq m_{\mathcal{H}}(\delta)$ $i.i.d$ samples generated by $\mathcal{D}$ and labeled by f, the algorithm returns a hypothesis h_S such that

$$\mathbb{P}\left(L_{\mathcal{D},f}(h_S) \leq \frac{1}{2} - \gamma\right) \geq 1 - \delta$$

Definition 1.2 — γ-Weak-Learnability. A class $\mathcal{H}$ is γ-weak-learnable if there exists a γ-weak-learner for that class.

A few remarks:

1. In weak learnability we have a guarantee only for a specific ε as opposed to strong (PAC) learnability, where we required the same guarantee for every $\varepsilon, \delta \in (0,1)$.
2. What does it mean that an algorithm A is a 0.5-weak-learner? It means that it will return the true hypothesis (with probability that we can choose), which is clearly not trivial.

Theorem 1.1 Weak learnability implies PAC learnability.

Proof. The fundamental theorem of learning states that if a hypothesis class $\mathcal{H}$ has a VC-dimension d, then the sample complexity of PAC learning $\mathcal{H}$ satisfies $m_{\mathcal{H}}(\cdot, \delta) \geq C_1 \frac{d + \log(1/\delta)}{\varepsilon}$, where C_1 is a constant. Applying this with $\varepsilon = 1/2 - \gamma$ we immediately obtain that if $d = \infty$ then $\mathcal{H}$ is not γ-weak-learnable.

■

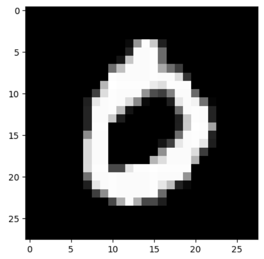
This implies that from the statistical perspective (i.e., if we ignore computational complexity), weak learnability is also characterized by the VC dimension of $\mathcal{H}$ and therefore is just as hard as PAC (strong) learning. However, when we do consider computational complexity, the potential advantage of weak learning is that maybe there is an algorithm that satisfies the requirements of weak learning and can be implemented efficiently.

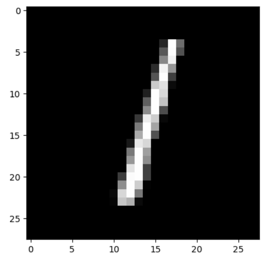
■ **Example 1.3** [Digits Classification] Given 28×28 images of 0 and 1 and:

$$f = \begin{cases} 1 & x \text{ is image of } 1 \\ -1 & x \text{ is image of } 0 \end{cases}$$

A good "rule of thumb" is:

- "Pixels in the middle are darker than the pixels in the middle-right $\Rightarrow$ image of 1".





■

How can we extend this observation into a hypothesis class? Define the hypothesis $h_{i,j}$ to be 1 iff the (i,j)th pixel is higher than 127, and

$$\mathcal{H} = \{h_{i,j} : 1 \leq i,j \leq 28\} \cup \{-h_{i,j} : 1 \leq i,j \leq 28\}.$$

After we'll discuss on AdaBoost, we'll see that this hypothesis class is not bad at all.

2 Boosting the Confidence: Committee Methods

So given some weak learner, we would like to find some practical strategy to improve its confidence. To do so we can create ensembles of classifiers, that is a collection of classifiers where the final prediction is based on all of them, which will improve the performance. The next examples shows the power of "the wisdom of the crowd" when it comes to learn a new problem.

Lemma 2.1 Let $X_1,\ldots,X_T \overset{iid}{\sim} Ber(p)$ taking values in $\{\pm 1\}$ and denote $X=\sum X_i$. The probability of the committee making the correct decision is $\mathbb{P}(X>0)$.

Proof. As the committee decides by a majority vote then the probability of the committee deciding right is the same as the probability of having more members deciding right than members deciding wrong:

$$\mathbb{P}(\text{Committee decides right}) = \mathbb{P}(|\text{Decided right}| > |\text{Decided wrong}|)$$

W.o.l.g, suppose the true answer is $+1$. As each random variable takes a value in $\{\pm\}$ we could express the collective vote as $X = sign(\sum X_i)$. If the committee decided right, then there are more members that voted right and $\sum X_i > 0$ which means that $X=1$. On the other hand, if the committee decided wrong, then more members voted wrong and $\sum X_i <= 0$ which means that $X=-1$. So, we conclude that:

$$\mathbb{P}(\text{Committee decides right}) = \mathbb{P}(X>0)$$

■

Lemma 2.2 Let $X_1,\ldots,X_T \overset{iid}{\sim} Ber(p)$ taking values in $\{\pm 1\}$ with $p>0.5$ and denote $X=\sum X_i$. The probability of the committee deciding correctly is bounded below by $1-\exp\left(-\frac{T}{2p}\left(p-\frac{1}{2}\right)^2\right)$.

Proof. W.l.o.g let us assume that the correct answer is $+1$ and denote $X=\sum_{i=1}^{T} X_i$. We are therefore interested in bounding $\mathbb{P}(X>0)$ from below. We will achieve this by bounding $\mathbb{P}(X\le 0)$ from above. Notice that for any $a>0$ it holds that:

$$\mathbb{P}(X\le 0) = \mathbb{P}(-aX \ge 0) = \mathbb{P}\left(e^{-aX} \ge e^0\right)$$

Now, using Markov's inequality

$$\mathbb{P}(X\le 0) \le \mathbb{E}\left[e^{-aX}\right] = \mathbb{E}\left[e^{-a\sum_i X_i}\right] \overset{iid}{=} \mathbb{E}\left[e^{-aX_1}\right]^T$$

Notice that as $X_1 \sim Ber(p)$ over $\{\pm 1\}$ it holds that:

$$\mathbb{E}\left[e^{-aX_1}\right] = pe^{-a} + (1-p)e^a = e^a\left(1-p+pe^{-2a}\right) \le e^{a-p+pe^{-2a}}$$

where the last inequality is because $1+x\le e^x$. Next, we use the inequality $x\ln(x) \ge \frac{x^2}{2}-\frac{1}{2} \quad x\in(0,1)$. For a selection of $a=\frac{1}{2}ln(2p)$ which is positive for $p>0.5$ we get that:

$$\begin{aligned}
\mathbb{P}(X\le 0) &\le \mathbb{E}\left[e^{-aX_1}\right]^T && \le \exp\left(T\left(a-p+pe^{-2a}\right)\right) \\
&= \exp\left(T\left(\tfrac{1}{2}ln(2p)-p+\tfrac{1}{2}\right)\right) && = \exp\left(Tp\left(-\tfrac{1}{2p}ln\left(\tfrac{1}{2p}\right)-1+\tfrac{1}{2p}\right)\right) \\
&\le \exp\left(Tp\left(\tfrac{1}{2}-\tfrac{1}{2(2p)^2}-1+\tfrac{1}{2p}\right)\right) && = \exp\left(-\tfrac{Tp}{2}\left(\tfrac{1}{4p^2}-\tfrac{1}{p}+1\right)\right) \\
&= \exp\left(-\tfrac{Tp}{2}\left(\tfrac{1}{2p}-1\right)^2\right) && = \exp\left(-\tfrac{T}{2p}\left(p-\tfrac{1}{2}\right)^2\right)
\end{aligned}$$

Finally, we conclude that:

$$\mathbb{P}(X>0) = 1-\mathbb{P}(X\leq 0) \geq 1-\exp\left(-\frac{T}{2p}\left(p-\frac{1}{2}\right)^2\right)$$

■

The above bound teaches us that for a committee of “non-dump“ members (i.e. $p > 1/2$) the probability of being wrong decays with a rate of $\mathcal{O}\left(\frac{1}{e^T}\right)$.Relating this to our learning scheme, it means that the confidence, $1-\delta$, in our prediction increases dramatically as T increases.

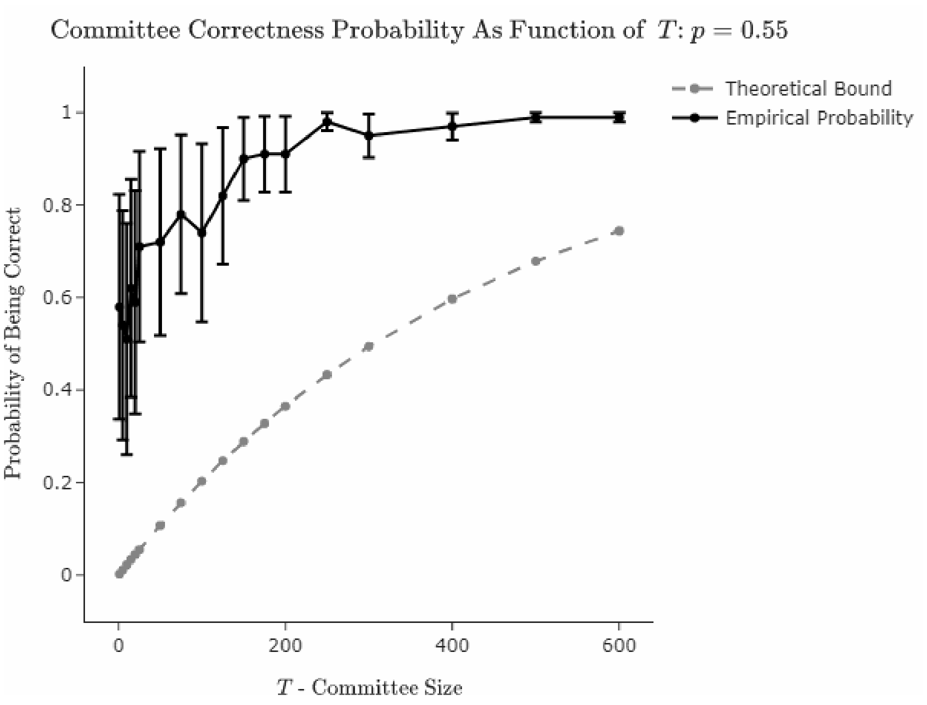


Figure 2: Committee Decision - Correctness Probability: *Theoretical bounds and empirical results as function of T, p.* *Chapter 5 Examples - Source Code*

Next, let us calculate what is a typical decision ($\mathbb{E}(X)$) and how consistent is it ($Var(X)$)?

2.1 Uncorrelated Predictors

Exercise 2.1 Let $X_1,\ldots,X_T \overset{iid}{\sim} Ber(p)$ taking values of $\{\pm 1\}$ with $p > 0.5$. What is the expectation and variance of $X = \frac{1}{T}\sum_{i=1}^{T} X_i$?

Solution: We begin with calculating the expectation and variance of each committee member:

$$\begin{aligned}
\mathbb{E}[X_i] &= 1\cdot\mathbb{P}(X_i=1)+(-1)\cdot\mathbb{P}(X_i=-1) \\
&= 2p-1 \\
Var(X_i) &= \mathbb{E}\left[(X_i-\mathbb{E}[X_i])^2\right] \\
&= p(1-(2p-1))^2+(1-p)(-1-(2p-1))^2 \\
&= 4p(1-p)^2+4p^2(1-p) \\
&= 4p(1-p)
\end{aligned}$$

Then, for X:

$$\begin{aligned}
\mathbb{E}[X] &= \tfrac{1}{T}\textstyle\sum_i \mathbb{E}[X_i] = 2p-1 \\
Var(X) &= \tfrac{1}{T^2}Var\left(\textstyle\sum_i X_i\right) \overset{iid}{=} \tfrac{1}{T^2}\textstyle\sum_i Var(X_i) = \tfrac{4}{T}p(1-p)
\end{aligned}$$

■

Therefore, when using a committee of independent members the expectation of decision remains the same while decreasing the variance at a rate of $\mathcal{O}\left(\frac{1}{T}\right)$. In other word, we are able to keep the same accuracy while increasing the confidence.

2.2 Correlated Predictors

In practice, however, committee members rarely vote independently. So let us assume that each two members are correlated with equal correlation $\rho\in[0,1]$.

Lemma 2.3 Let $X_1,\ldots,X_T$ be a set of identically-distributed real-valued random variables such that: $Var(X_i)=\sigma^2$ and $corr(X_i,X_j)=\rho$, $i\neq j$. The variance in the commitee's decision is given by $\rho\sigma^2+\frac{1}{T}(1-\rho)\sigma^2$.

Proof. As X is the average of T identically distributed random variables:

$$Var(X)=Var\left(\frac{1}{T}\sum_i X_i\right)=\frac{1}{T^2}\left[\sum_i Var(X_i)+2\sum_{i<j}Cov(X_i,X_j)\right]$$

Recall that the correlation between two random variables is defined as:

$$corr(A,B):=\frac{Cov(A,B)}{\sqrt{Var(A)\,Var(B)}}$$

and therefore for any $i\neq j$:

$$Cov(X_i,X_j)=corr(X_i,X_j)\sqrt{Var(X_i)\,Var(X_j)}=\rho\sigma^2$$

Plugging this back into the variance:

$$Var(X)=\frac{1}{T^2}\left[T\sigma^2+2\binom{T}{2}\rho\sigma^2\right]=\frac{\sigma^2}{T}+\left(1-\frac{1}{T}\right)\rho\sigma^2=\rho\sigma^2+\frac{1}{T}(1-\rho)\sigma^2$$

■

3 Boosting the Accuracy: AdaBoost

AdaBoost takes advantage of the fact that a weak learner gives the same guarantee for every distribution $\mathcal{D}$. The algorithm explicitly uses this property to force the weak learner to do better on the samples it previously misclassified and this way improve the accuracy. If you want to better understant the way AdaBoost works, **here** you can find a nice demonstration.

Algorithm 1 Adaptive **Boost**ing

```
 1: procedure ADABOOST(training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, base learner $\mathcal{A}$, number of rounds $T$)
 2:     Set initial distribution to be uniform: $\mathcal{D}^{(1)} \leftarrow \left(\frac{1}{m}, \dots, \frac{1}{m}\right)$        ▷ Initialize parameters
 3:     for $t = 1, \dots, T$ do
 4:         Invote base learner $h_t = \mathcal{A}\left(\mathcal{D}^{(t)}, S\right)$
 5:         Compute $\varepsilon_t = \sum \mathcal{D}^{(t)} \mathbb{1}\left[y_i \neq h_t(\mathbf{x}_i)\right]$
 6:         Set $w_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) = \frac{1}{2} \ln\left(\frac{1}{\varepsilon_t} - 1\right)$.
 7:         Update sample weights $\mathcal{D}_i^{(t+1)} = \mathcal{D}_i^{(t)} \exp\left(-y_i \cdot w_t h_t(\mathbf{x}_i)\right), \quad i = 1, \dots, m$
 8:         Normalize weights $\mathcal{D}_i^{(t+1)} = \frac{\mathcal{D}_i^{(t+1)}}{\sum_j \mathcal{D}_j^{(t+1)}} \quad i = 1, \dots, m$
 9:     end for
10:     return $h_S(\mathbf{x}) = sign\left(\sum_{i=1}^{T} w_t h_t(\mathbf{x})\right)$
11: end procedure
```

■ **Example 3.1** Let's take another look at the digit classification task: note that we have some prior knowledge:

- We try to learn a function $f : \{0, ..., 255\}^{19\times 19} \to \{0, 1\}$, which classifies 0/1 images differently.
- We also know something about $\mathcal{D}$: we observe only images of 0/1.

Now consider a weak-learner algorithm which considers only hypotheses that take into account only one pixel from the image. We hope that this class of hypotheses can yield a rule of thumb for every distribution $\mathcal{D}$ over 0/1 images (Note that this is not true for any function and every $\mathcal{D}$). **here** (Please use your huji account) you can see a code example of the 0-1 classification by Adaboost. ■

Now we can finally answer the questions from the beginning:

- **Bias-complexity tradeoff** - We have seen that the more expressive the hypothesis class the learner is searching over, the smaller the approximation error is, but the larger the estimation error becomes. The boosting paradigm allows the learner to have smooth control over this tradeoff. The learning starts with a basic class (that might have a large approximation error), and as it progresses (i.e. as T grows) the class that the predictor may belong to grows richer. In many real world problems we see that for small T, increasing T will reduce the bias dramatically while preserve low variance, thus the MSE will decrease. But when we have very large T, the improvement in the bias becomes minor when comparing to the variance.

- **Computational complexity of learning** - For many interesting concept classes the task of finding an ERM hypothesis may be computationally infeasible. A boosting algorithm amplifies the accuracy of weak learners that come from an easy-to-learn hypothesis class and performs just slightly better than a random guess. When a weak learner can be implemented efficiently, boosting provides a tool for aggregating such weak hypotheses to approximate gradually good predictors for larger, and harder to learn, classes.