Summary of Introduction to Theory of Machine Learning

Jinhong Du

May 18, 2020

Contents

Chapter 1

Introduction

1.1 Statistical Learning Theory

1.1.1 Goals of Machine Learning Theory

Develop and analyze models to understand:

- what kinds of tasks we can hope to learn, and from what kind of data,
- what types of guarantees might we hope to achieve,
- other common issues that arise.

1.1.2 Learning Setting

In *Statistical learning theory (SLT)*, we have the following elements:

- **Data.** The sample $\mathcal{S} = \{(x_i, y_i)\} \subset \mathcal{X} \times \mathcal{Y}$ from the population $\mathcal{D}$ where the labels are produced by some target function $f : \mathcal{X} \mapsto \mathcal{Y}$ in $\mathcal{C}$. We will mainly focus on the case when $\mathcal{Y}$ is a finite set, e.g. $\mathcal{Y} = \{0, 1\}$.
- **Algorithm.** The algorithm optimizes over $\mathcal{S}$ to produce some hypothesis $h \in \mathcal{H}$. Ideally, the error rate on sample $err_{\mathcal{S}}(h)$ is low.
- **Loss.** Goal is for h to do well on new examples also from the probability distribution $\mathcal{D}$ over $\mathcal{X}$, we want the true error rate/true risk/expected loss $err_{\mathcal{D}}(h) = \mathbb{P}_{\mathcal{D}}\{h(x) \neq f(x)\} \leq \epsilon$.

Two big questions in this framework are:

1. **Algorithm design.** (exploitation) How might we automatically generate rules that do well on observed data?
2. **Confidence bound / sample complexity.** (exploration) What kind of confidence do we have that they will do well in the future? For a given learning alg, how much data do we need, and how can we design alg to need less? For the confidence question, we'll need some connection between future data and past data.

1.2 Probably Approximately Correct Learning

1.2.1 Definition

One formal learning model in the statistical learning framework is the *probably approximately correct (PAC)* learning.

Definition 1. (PAC Learnability) *Suppose we have an algorithm for a class of functions C such that*

- *For any given $\epsilon > 0$, $\delta > 0$, any target $f \in C$ and any distribution $\mathcal{D}$, when the sample size $|\mathcal{S}|$ is large enough, the algorithm over $\mathcal{S}$ produces $h \in \mathcal{H}$ of $err_{\mathcal{D}}(h) \leq \epsilon$ with probability at least $1 - \delta$.*
- *Running time $t\left(\frac{1}{\epsilon}, \frac{1}{\delta}, \cdots\right)$, and sample size $s\left(\frac{1}{\epsilon}, \frac{1}{\delta}, \cdots\right)$.*

then this is a PAC-learning algorithm is with running time t and sample size s.

A problem/hypothesis class $\mathcal{H}$ is PAC-learnable if there exists a PAC-learning algorithm for C with t and s being polynomial in relevant params.

We say that learning is *proper* if $\mathcal{H} \subset C$, or just say learning C by $\mathcal{H}$.

1.2.2 Drawbacks

- In the real world, there's never a perfect/consistent function in the class. It is hard to prove guarantees for efficient algorithms on finding near-best function in the PAC-learning framework.
- In the real world, labeled examples may be more expensive than running time.
- "Prior knowledge/beliefs" might be not just over f form of target but other relations to data.
- It doesn't address other kinds of information (cheap unlabeled data, pairwise similarity information).
- It assumes a fixed distribution.

1.3 Decision Lists Algorithm

1.3.1 Definition

Definition 2. (Decision Lists) *A decision list (DL) of length k is of the form:*

```
if $f_1$ then $b_1$
else if $f_2$ then $b_2$
        ⋮
else if $f_k$ then $b_k$
```

where f_i is the ith formula and b_i is the ith boolean for $i \in \{1, \ldots, k\}$. Especially, we will focus on the case when f_i's are formulas that examine the value of a single binary variable.

1.3.2 Analysis of PAC Learnability

Say we suspect there might be a good prediction rule of this form. To design and analyze an algorithm, we can follow these steps:

1. (exploitation) Design an efficient proper algorithm A that will find a length-n DL over n Boolean features with $err_{\mathcal{S}}(h) = 0$ if one exists.
   - Start with empty list.
   - Find if-then rule consistent with data (and satisfied by at least one example).
   - Put rule at bottom of list so far, and cross off examples covered. Repeat until no examples remain.
   - If this fails (gets stuck) then no rule is consistent with the remaining data. So no DL is consistent with remaining data, i.e., no DL is consistent with original data. When there is no decision rule consistent with data, it will become a NP-hard problem to find the decision rule that achieve minimum error.

   The complexity of this algorithm is $O(n \cdot |\mathcal{S}|)$.
2. (exploration) Show the following theorem:

   Theorem 1. *If $|\mathcal{S}| \geq \frac{1}{\epsilon}\left[\ln(|\mathcal{H}|) + \ln\left(\frac{1}{\delta}\right)\right]$, then with probability at least $1-\delta$, for all $h \in \mathcal{H}$ with $err_{\mathcal{D}}(h) > \epsilon$, we have $err_{\mathcal{S}}(h) > 0$, or equivalently,*

   $$\mathbb{P}\{\exists \ \textit{decision list } h \textit{ with } err_{\mathcal{S}}(h) = 0 \textit{ but } err_{\mathcal{D}}(h) > \epsilon\} < \delta.$$

   Proof. Consider some decision rule h with $err_{\mathcal{D}}(h) > \epsilon$, that we're worried they might fool us. Chance that $err_{\mathcal{S}}(h) = 0$ is at most $(1-\epsilon)^{|\mathcal{S}|}$. Let $|\mathcal{H}|$ be the number of DLs over n Boolean features, then $|\mathcal{H}| \leq n! \cdot 2 \cdot 4^n$.
   Using the union bound gives

   $$\mathbb{P}\{\exists \text{ decision rule } h \text{ with } err_{\mathcal{D}}(h) > \epsilon \text{ and } err_{\mathcal{S}}(h) = 0\} \leq |\mathcal{H}|(1-\epsilon)^{|\mathcal{S}|} \leq |\mathcal{H}|e^{-\epsilon|\mathcal{S}|} < \delta$$

   for $|\mathcal{S}| \geq \frac{1}{\epsilon}\left[\ln(|\mathcal{H}|) + \ln\left(\frac{1}{\delta}\right)\right] \asymp \frac{1}{\epsilon}\left[n\ln(n) + \ln\left(\frac{1}{\delta}\right)\right]$. ■

   In particular, the number of examples (confidence/sample-complexity) needs to only be proportional to $\log(|\mathcal{H}|)$. There is nothing special about DLs in our argument, so the results can be applied to other algorithms.
3. If the underlying model f is in fact a DL, then w.h.p. the algorithm's hypothesis will be approximately correct, i.e., A is a good algorithm that produces a hypothesis that is PAC.

Corollary 1. *If computation time is no object, then for any target function $f : \mathcal{X} \mapsto \{0,1\}$, every finite consistent hypothesis class $\mathcal{H}$ is PAC learnable with sample size*

$$|\mathcal{S}| \geq \frac{1}{\epsilon}\left[\ln(|\mathcal{H}|) + \ln\left(\frac{1}{\delta}\right)\right],$$

or equivalently, with size of hypothesis class

$$|\mathcal{H}| \lesssim \epsilon|\mathcal{S}| - \ln\left(\frac{1}{\delta}\right).$$

1.3.3 Decision Trees

However, the *decision trees* over $\{0,1\}^n$ not known to be PAC-learnable. The DL argument works because no matter how large a sample S you have, you will never produce a list longer than length n. For decision trees, the analogous algorithm would produce a tree as big as the sample, and so we would not get our guarantee of low error under $\mathcal{D}$.

Given any data set $\mathcal{S}$, it's easy to find a consistent DT if one exists, since we can fit the data perfectly.

Simple heuristics used in practice (ID3 etc.) don't work for all $f \in \mathcal{C}$ even for uniform $\mathcal{D}$. Would suffice to find the (approximately) smallest DT consistent with any dataset $\mathcal{S}$, but that's NP-hard.

1.4 More Examples

Other classes we can efficiently PAC-learn by DLs:

- AND-functions, OR-functions
- 3-CNF (conjunctive normal form) formulas (3-SAT (propositional satisfiability) formulas), 3-DNF (disjunctive normal form) formulas
- k-Decision lists (each if-condition is a conjunction of size k), k is constant.

Given a data set $\mathcal{S}$, deciding if there is a consistent 2-term DNF formula is NP-complete. It is hard to learn $\mathcal{C}$ by $\mathcal{C}$, but easy to learn $\mathcal{C}$ by $\mathcal{H}$, where $\mathcal{H} = \{2 - \text{CNF}\}$.

1.5 Occam Learning

The succinctness of a concept c in concept class $\mathcal{C}$ can be expressed by the *description length* $size(c)$ of the shortest bit string that can represent c in $\mathcal{C}$. Occam learning connects the succinctness of a learning algorithm's output to its predictive power on unseen data.

Definition 3. (Occam Learning) *Let C and $\mathcal{H}$ be concept classes containing target concepts and hypotheses respectively. Then, for constants $\alpha \geq 0$ and $0 \leq \beta < 1$, a learning algorithm L is an (α, β)-**Occam algorithm** for C using $\mathcal{H}$ iff, given a set $S = \{x_1, \ldots, x_m\}$ of m samples labeled according to a concept $c \in C$, L outputs a hypothesis $h \in \mathcal{H}$ such that*

- *h is consistent with c on S (that is, $h(x) = c(x), \forall x \in S$).*
- *$size(h) \leq (n \cdot size(c))^{\alpha} m^{\beta}$.*

*where n is the maximum length of any sample $x \in S$. An Occam algorithm is called **efficient** if it runs in time polynomial in n, m, and $size(c)$. We say a concept class C is **Occam learnable** with respect to a hypothesis class $\mathcal{H}$ if there exists an efficient Occam algorithm for C using $\mathcal{H}$.*

Theorem 2. *If computation time is no object, then any consistent class of hypotheses is PAC-learnable with sample size*

$$|\mathcal{S}| \gtrsim \frac{1}{\epsilon}\left[size(f) + \ln(size(f)) \ln\left(\frac{1}{\delta}\right)\right].$$

Proof. Let $s_1 = 10$, $\delta_1 = \frac{\delta}{2}$. For $i = 1, \ldots,$, do

- Request $\frac{1}{\epsilon}[s_i + \ln\left(\frac{1}{\delta_i}\right)]$ samples $\mathcal{S}_i$.
- Check if there is a function of size at most s_i consistent with S_i. If so, output it and halt.
- Otherwise, let $s_{i+1} = 2s_i$ and $\delta_{i+1} = \frac{\delta_i}{2}$ and repeat.

Let I be the step when the above precedure stops. Then we will have at most $\sum_{i=1}^{I} \delta_i \leq \delta$ chance of failure. The numebr of total samples we use is

$$\begin{aligned}
\sum_{i=1}^{I} |\mathcal{S}_i| &= \sum_{i=1}^{I} \frac{1}{\epsilon}\left[s_i + \ln\left(\frac{1}{\delta_i}\right)\right] \\
&= \frac{1}{\epsilon}\left[s_1 + 2s_1 + \cdots + 2^I s_1 + \ln\left(\frac{2}{\delta}\right) + \cdots + \ln\left(\frac{size(f)}{\delta}\right)\right] \\
&\leq \frac{1}{\epsilon}\left[(2^I - 1)s_1 + \ln[size(f)] \ln\left(\frac{size(f)}{\delta}\right)\right] \\
&\leq \frac{1}{\epsilon}\left[2size(f) + \ln^2(size(f)) + \ln(size(f)) \ln\left(\frac{1}{\delta}\right)\right] \\
&\lesssim \frac{1}{\epsilon}\left[size(f) + \ln(size(f)) \ln\left(\frac{1}{\delta}\right)\right].
\end{aligned}$$

So this is a generic way to learn any f from $O(size(f))$ samples if ignore computation time. ■

Notice that this gives a different bound with the one in Corollary 1

Chapter 2

Online Learning

2.1 Setting

In online learning setting, We have no assumption about data distribution. Also, we no longer talk about past performance to predict future results, since the data distribution may be varying.

2.2 Mistake-Bound Model

2.2.1 Definition

We first consider a binary prediction problem. We can view learning as a sequence of stages. In each stage, algorithm is given x, asked to predict $f(x)$, and then is told correct value.

We make no assumptions about the order of examples. The samples may not be iid. Our goal is to bound total number of mistakes.

Definition 4. (Mistake-Bound Models) *Algorithm A learns class C with mistake bound M if A makes at most M mistakes on any sequence of examples consistent with some $f \in C$.*

In this setting, can no longer talk about how much data do we need for the algorithm to converge. Instead, we try to bound the mistake in terms of size of examples $|\mathcal{S}|$ and complexity of target $size(f)$.

Definition 5. (Mistake-Bound Learnability) *A hypothesis class C is learnable in a mistake-bound model if there exists an algorithm with both mistake bound and running time per stage are $poly(|\mathcal{S}|, size(f))$.*

Example 1. (Disjunction) Suppose features are boolean: $\mathcal{X} = \{0,1\}^n$. The target f is an `OR` function, like $x_3 \vee x_9 \vee x_{12}$. We can find an on-line strategy that makes at most n mistakes.

1. Start with $h(x) = x_1 \vee x_2 \vee \cdots \vee x_n$. {features in h} $\supset$ {features in f} is invariant.
2. h will not make mistakes on samples with positive labels.

3. If h make mistake on a sample x with a negative label, then we discard features in h set to 1 in x.
4. Maintains invariance and decreases $|h|$ at least by 1. So at most n mistakes total.

No deterministic algorithm can do better than this algorithm since the samples could be adversarial.

2.2.2 Properties

MB Learnability And Conservative Algorithm

Definition 6. (Conservative Algorithm) *An algorithm is conservative if it only changes its state/parameters wher it makes mistake.*

Theorem 3. *If the hypothesis class C is learnable with mistake bound M, then it is learnable by a conservative algorithm.*

Proof. We take a generic algorithm A. Then we create a new conservative A' by running A, but remain the same state if no mistake is made.

We will still have at most M mistakes, because A still sees a legal sequence of examples. ■

MB Learnability And PAC Learnability

Theorem 4. *If the hypothesis class C is mistake-bound learnable, then it is also PAC learnable. Or equivalently, if the algorithm A learns the hypothesis class C with mistake-bound M, then A is also the PAC-learning algorithm for C.*

Proof. (Window algorithm) For any $\epsilon, \delta > 0$, define $s = \frac{1}{\epsilon} \ln\left(\frac{M}{\delta}\right)$. We run a conservative algorithm A and get a sequence of produced hypotheses: $h = (h_1, h_2, \ldots, h_M, h_{M+1})$, since h can make at most M mistakes. For each one, if it is consistent with consecutive s examples, then stop. If $err_{\mathcal{D}}(h_i) > \epsilon$, then the probability we stopped with only s samples using h_i was at most $\frac{\delta}{M}$.

Then we have

$$
\begin{aligned}
&\mathbb{P}\{err_{\mathcal{D}}(h) > \epsilon \text{ and we stop with only } Ms \text{ samples using } h\} \\
\leq &\sum_{i=1}^{M} \mathbb{P}\{err_{\mathcal{D}}(h_i) > \epsilon \text{ and we stop with only } s \text{ samples using } h_i\} \\
&\leq M(1-\epsilon)^{\frac{1}{\epsilon}\ln\left(\frac{M}{\delta}\right)} \\
&\leq M e^{-\ln\left(\frac{M}{\delta}\right)} = \delta,
\end{aligned}
$$

i.e., if we have sample size $|\mathcal{S}| \geq \frac{M}{\epsilon} \ln\left(\frac{M}{\delta}\right)$, then with probability at least $1-\delta$, we have $err_{\mathcal{D}}(h) < \epsilon$. This also means that we are fooled by any of the hypotheses $f \in C$ with probability at most δ. Therefore, A is PAC-learning for C.

We can actually get a better bound of $O\left(\frac{1}{\epsilon}\left[M + \ln\left(\frac{1}{\delta}\right)\right]\right)$, using the fact that most bad hypotheses will not take fully $\frac{1}{\epsilon}\ln\left(\frac{1}{\delta}\right)$ time to be discovered. ■

Example 2. (End-Point Estimation) Say we view each sample as an integer between 0 and $2n-1$. Let $C = \{[0,a] : a < 2n\}$ (e.g., device fails if it gets too hot). In a PAC model we could just pick any consistent hypothesis since the samples $\mathcal{S}$ is available. However, this doesn't work in a MB model, since samples may be revealed in sequence adversarially. For example, when the samples are $1, 2, \ldots$, every time we make a guess based on previous samples, it would be incorrect.

This example shows that MB learnability requires more than PAC learnability.

A good way for a MB model to work is to use the bisection algorithm.

2.2.3 Algorithms with Unbounded Computation Time

Algorithm 1. (The Halving Algorithm)

1. Let $\mathcal{H} = C$ where at least one consistent hypothesis h exists.
2. Predict with the majority of the $|\mathcal{H}|$ consistent experts in the class $\mathcal{H}$ until we make a mistake.
3. At least half of the experts in $\mathcal{H}$ will also make a mistake; remove them from the class and renew the class $\mathcal{H}$.
4. Repeat the above procedure 2-3.

Since the Halving Algorithm assumes that one of the experts is perfect, there will be always at least one hypothesis in $\mathcal{H}$. The algorithm makes at most $\log_2(|C|)$ mistakes.

If we have a prior $\boldsymbol{p}$ (e.g., $\boldsymbol{p} = \mathbf{1}$) over functions in C, then we can weight the vote according to p, which makes at most $\log_2\left(\frac{1}{p_f}\right)$ mistakes, where p_f is the proportion of p with respect to the targe function f.

If we have a randomized target function f from C, which was really chosen according to $\boldsymbol{p}$, then the expected number of mistakes is at most $\sum_{h\in C} p_h \log_2\left(\frac{1}{p_h}\right)$, which is the entropy of distribution $\boldsymbol{p}$.

Noted that the Halving Algorithm is not necessary optimal when samples are adversarial. The optimal algorithm throws out sets with larger mistake bounds, while the Halving Algorithm throws out larger sets.

2.3 Regret-Bound Model

2.3.1 Definition

When there is no perfect model that make no mistake, we can use *regret bound*, a generalization of mistake bound, to assess a model. That is we can show that our algorithm does nearly as well as best predictor in some class.

Definition 7. (No-Regret Algorithm) *A no-regret algorithm has average regret that is sublinear in the total number of round T as $T \to \infty$.*

2.3.2 Combining Expert Advice

Let's look at an online binary prediction problem, e.g., to predict if the stock market go up or down. Also, we have n experts for advice.

We can do nearly as well as best in hindsight, which is trivial in PAC (i.i.d.) setting. If one expert is perfect, can get at most $\log_2(n)$ mistakes with the Halving algorithm. But what if none is perfect? Can we do nearly as well as the best one in hindsight?

One strategy is to use *Iterated Halving Algorithm.* It is same as before, but once we've crossed off all the experts, we restart from the beginning. This makes at most $\log_2(n)(m+1)$ mistakes, where m is the number of mistakes of the best expert in hindsight.

Another strategy is to give weights to experts. The intuition is that making a mistake doesn't completely disqualify an expert. So, instead of crossing off, we just lower its weight.

Algorithm 2. (Weighted Majority Algorithm)

1. Start with all experts having weight 1.
2. Make a prediction based on weighted majority vote.
3. Penalize mistakes by cutting weight in half.
4. Repeat steps 2-3.

Theorem 5. *Running the Weighted Majority Algorithm on the binary prediction problem yields*

$$M_t \leq 2.4(m_t + \log_2 n),$$

where n is the number of experts, M_t is the number of mistakes we've made up to time t, m_t is the number of mistakes the best expert has made up to time t, and W_t is the total weight with $W_0 = n$.

Proof. After each mistake, W_t drops by at least 25%, since the majorities (more than a half of experts) make a mistake. So, after M_t mistakes, W_t is at most $n\left(\frac{3}{4}\right)^{M_t}$. The weight of the best expert is $\frac{1}{2^{m_t}}$. So,

$$\frac{1}{2^{m_t}} \leq n\left(\frac{3}{4}\right)^{M_t},$$

which impies

$$M_t \leq 2.4(m_t + \log_2 n)$$

■

This is not good enough. We can further imporve the result. First, instead of taking majority vote, we use weights as probabilities. Second, we reduce the weight by ϵ instead of $\frac{1}{2}$.

Algorithm 3. (Randomized Weighted Majority Algorithm)

1. Start with all experts having weight 1.
2. Calculate the probability distribution for the experts, $\tilde{w}_{t,i} = \frac{w_{t,i}}{\sum_{i=1}^{n} w_{t,i}}$ for $i = 1, \dots, n$. Make a prediction based on the probability an expert which is randomly selected from the probability distribution. (An equivalent way is to predict 1 with probability $\sum_{\{i:i\text{th expert predict } 1\}} \frac{w_{t,i}}{\sum_{i=1}^{n} w_{t,i}}$)
3. Penalize mistakes by reduce weight by ϵ.
4. Repeat steps 2-3.

Theorem 6. *Running the Randomized Weighted Majority Algorithm on the binary prediction problem until time T yields*

$$M < \frac{1}{\epsilon}\left[-m\ln(1-\epsilon) + \ln(n)\right] \approx \left(1 + \frac{\epsilon}{2}\right)m + \frac{1}{\epsilon}\ln(n),$$

where n is the number of experts, M is the expected number of mistakes we make up to time T, and m is the number of mistakes the best expert make up to time T. Thus, the regret bound is given by

$$M - m \leq 2\left(T\ln(n)\right)^{\frac{1}{2}},$$

which is sublinear in T as $T \to \infty$.

Proof. Say at time t we have fraction F_t of weight on experts that made mistake. So, we have probability F_t of making a mistake, and we remove an ϵF_t fraction of the total weight. The final weight W satisfies

$$W = n\prod_{t=1}^{\infty}(1 - \epsilon F_t).$$

So

$$\ln(W) = \ln(n) + \sum_{t=1}^{\infty}\ln(1 - \epsilon F_t) \leq \ln(n) - \epsilon\sum_{t=1}^{\infty}F_t = \ln(n) - \epsilon M,$$

where M is the expected number of mistakes. If best expert makes m mistakes in total, then $\ln(W) > \ln[(1 - epsilon)^m]$. So, $\ln(n) - \epsilon M > m\ln(1-\epsilon)$, which yields

$$M < \frac{1}{\epsilon}\left[-m\ln(1-\epsilon) + \ln(n)\right] \approx \left(1 + \frac{\epsilon}{2}\right)m + \frac{1}{\epsilon}\ln(n).$$

If we assume that $m \geq \ln(n)$ and set $\epsilon = \left(\frac{\ln(n)}{m}\right)^{\frac{1}{2}}$, then

$$M \leq m + 2(m\ln(n))^{\frac{1}{2}} \leq m + 2\left(T\ln(n)\right)^{\frac{1}{2}}.$$

So the regret bound is $M - m \leq 2\left(T\ln(n)\right)^{\frac{1}{2}}$, which is sublinear in T as $T \to \infty$. ■

If the experts are multiple actions and each has a loss/cost in $\{0, 1\}$ at each time, we can pick an expert with probability proportional to its weight instead of picking a prediction with prob proportional to its weight. Same analysis applies.

Furthermore, if losses/costs are in $[0,1]$, we can modify the update rule by $w_i \leftarrow w_i(1-\epsilon c_i)$. The fraction of w_t we removed from the system is $\frac{\sum_{i=1}^{n} w_i \epsilon c_i}{\sum_{j=1}^{n} w_j} = \epsilon \sum_{i=1}^{n} p_i c_i$ where $\sum_{i=1}^{n} p_i c_i$ is the expected cost at time T. The analysis is very similar to case of $\{0,1\}$.

Chapter 3

Classic Algorithms

3.1 The Perceptron Algorithm

3.1.1 Perfectly Linearly Separable Case

To learn a large-margin linear separator in $\mathbb{R}^d$ in an online setting:

- Examples arrive one at a time.
- Given x, predict label y.
- Told correct answer.

Our goal is to bound number of mistakes under assumption there exists w^* such that $w^* \cdot x \geq 1$ on positives and $w^* \cdot x \leq -1$ on negatives.

Since $\frac{w^* \cdot x_0}{\|w^*\|}$ is the distance of x_0 to hyperplane $w^* \cdot x = 0$, our assumption is equivalent to assuming that there exists a separator of margin $\gamma = \frac{1}{\|w^*\|}$.

Algorithm 4. (Perceptron Algorithm)

1. Initialize $w = 0$. Predict positive if $w \cdot x > 0$, else predict negative.
2. Mistake on positive: $w \leftarrow w + x$.
3. Mistake on negative: $w \leftarrow w - x$.

Theorem 7. *The perceptron algorithm makes at most $\|w^*\|^2 \max_{x \in \mathcal{S}}\{x^2\}$ mistakes if there exists w with $w^* \cdot x \geq 1$ on all positives and $w^* \cdot x \leq -1$ on all negatives. If points all lie in a ball of radius R centered at the origin, then mistake bound is at most $\frac{R^2}{\gamma^2}$.*

Proof. We prove for the special case when points all lie in a ball of radius R centered at the origin. Consider $w \cdot w^*$ and $\|w\|$. Since

$$(w \pm x) \cdot w^* = w \cdot w^* \pm x \cdot w^* \geq w \cdot w^* + 1,$$

each mistake increases $w \cdot w^*$ by at least 1. Since

$$(w \pm x) \cdot (w \pm x) = w \cdot w \pm 2(w \cdot x) + x \cdot x \leq w \cdot w + R^2,$$

each mistake increases $w \cdot w$ by at most R^2. So, after M mistakes, $w^2 \le MR^2$, so $w \le \sqrt{M}R$.

Since $\frac{w \cdot w^*}{\|w^*\|} \le \|w\|$ by Cauchy-Schwarz inequality, we have $\frac{M}{\|w\|} \le \sqrt{M}R$. So $M \le \|w^*\|^2 R^2$. ■

Algorithm 5. (Lower Bound) *In general it is not possible to get at most $\frac{R^2}{\gamma^2}$ mistakes with a deterministic algorithm.*

Proof. Consider $\frac{R^2}{\gamma^2}$ coordinate vectors scaled to length R,

$$w^* = \frac{1}{R}\left(\pm x_1 \pm x_2 \pm \cdots \pm x_{\frac{R^2}{\gamma^2}}\right),$$

so that $|w^* \cdot x_i| = 1$ for $i = 1, \ldots, \frac{R^2}{\gamma^2}$. Then we can force all mistake with an adversarial opponent.

The margins are γ as desired since $\|w^*\| = \frac{\sqrt{\frac{R^2}{\gamma^2}}}{R} = \frac{1}{\gamma}$. ■

3.1.2 Imperfectly Linearly Separable Case

In this case, a mistake could cause $|w \cdot w^*|$ to drop. The hinge-loss of w^* on x with label y is

$$l_{\text{hinge}}(y, w^* \cdot x) = \begin{cases} \max\{0, 1 - w^* \cdot x\} & \text{, if } y = 1 \\ \max\{0, 1 + w^* \cdot x\} & \text{, if } y = 0 \end{cases}$$

Theorem 8. *On any sequence of examples $\mathcal{S}$, the Perceptron algo makes at most*

$$\min_{w^*}[\|w^*\|^2 R^2 + 2L_{hinge}(w^*, \mathcal{S})]$$

mistakes, where $L_{hinge}(w^*, \mathcal{S}) = \sum_{(x,y)\in\mathcal{S}} l_{hinge}(y, w^* \cdot x)$.

Proof. Since

$$(w \pm x) \cdot w^* = w \cdot w^* \pm x \cdot w^* = w \cdot w^* + 1 - (1 \mp x \cdot w^*) \ge w \cdot w^* + 1 - \max\{0, 1 \mp x \cdot w^*\},$$

after making M mistakes, we have $w \cdot w^* \ge M - L_{\text{hinge}}(w^*, \mathcal{S})$. Still, $\|w\|^2 \le MR^2$. Sicne $(w \cdot w^*)^2 \le \|w\|^2 \|w^*\|^2$. Then

$$(M - L_{\text{hinge}}(w^*, \mathcal{S}))^2 \le MR^2 \|w^*\|^2,$$

i.e.,

$$M \le R^2\|w^*\|^2 + 2L_{\text{hinge}}(w^*, \mathcal{S}) - \frac{1}{M} L_{\text{hinge}}(w^*, \mathcal{S}) \le R^2 \|w^*\|^2 + 2L^2_{\text{hinge}}(w^*, \mathcal{S}).$$

■

3.1.3 Kernel Functions

If the data is not perfect linearly separable, we can also use kernel functions to make it linearly separable in the high dimensional space.

Definition 8. (Kernel Function) *A pairwise function $K(x, x') : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is a kernel if there exists a function $\phi : \mathcal{X} \mapsto \mathcal{V}$ from input space to a new space (of possibly much higher dimension) such that $K(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{V}}$.*

An explicit representation for ϕ is not necessary, as long as $\mathcal{V}$ is an inner product space.

If we can modify Perceptron Algorithm so that only interacts with data via taking dot-products, and then replace $\langle x, x' \rangle$ with $\langle \phi(x), \phi(x') \rangle_{\mathcal{V}}$, then algorithm will act as if data was in higher-dimensional space $\mathcal{V}$. Since the weight vector in the Perceptron Algorithm is always a sum of previous examples (or their negations), e.g., $w = x_1 + x_3 - x_6$ where x_i is the ith sample. So, to predict on new x, just compute $w \cdot x' = x_1 \cdot x' + x_3 \cdot x' - x_6 \cdot x'$. Now replace dot-product with kernel.

3.2 Support Vetcor Machines

3.2.1 Perfectly Linearly Separable Case

In the batch (PAC) setting, we are given $\mathcal{S}$ up front. Let's just solve for w^* of largest margin. ("realizable case")

$$\begin{aligned} &\min \|w\|^2 \\ &s.t.\ y_i(w \cdot x_i) \geq 1 \text{ for all } (x_i, y_i) \in \mathcal{S} \end{aligned}$$

3.2.2 Imperfectly Linearly Separable Case

We introduce slack variables and consider the soft-margin SVMs.

$$\begin{aligned} &\min \|w\|^2 + C \sum_i \xi_i \\ &s.t.\ y_i(w \cdot x_i) \geq 1 - \xi_i \text{ for all } (x_i, y_i) \in \mathcal{S} \\ &\quad \xi_i \geq 0 \text{ for all } i \end{aligned}$$

where ξ_i's are slack variables and C is a known constant.

The first term times R^2 is roughly an upper bound on the amount of overfitting. The second term is the hinge loss, which is an upper bound on empirical 0/1-loss. Together, the objective function is proportional to rough upper-bound on true error.

This is the *primal* form of SVM. While we can kernelize SVM in its *dual* form.

3.2.3 Lagrangial Dual

Consider an optimization problem of the form: minimize a convex function in some variables, subject to linear constraints on these variables.

We can think it as a game. For each constraint j, the opponent gets to choose $\alpha_j \geq 0$ linear penalty. We choose setting of variables and want to minimize cost. While the opponent wants to maximize.

If we have to go first, clearly we should pick optimal feasible point (else opponent will assign infinite penalty to any violated constraint).

If the opponent goes first, it can assign penalties such that we can do no better. (No "duality gap".) This relies on convexity of the cost function. The opponent's optimization problem is called the *dual problem*.

Let w denote strategy of us (*primal variables*) and let α denote strategy of the opponent (*dual variables*). The Lagrangian is the total cost $L(w, \alpha)$ paid by us. The opponent's optimization problem is

$$\max_{\alpha} \min_{w} L(w, \alpha)$$
$$s.t.\ \alpha_j \geq 0\ \forall\ j$$

The dual problem of SVM is given by

$$\max_{\alpha_1, \alpha_2} \min_{w, \xi} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i + \sum_i \alpha_{1i}[1 - \xi_i - y_i(w_i \cdot x_i)] - \sum_i \alpha_{2i}\xi_i$$
$$s.t.\ \alpha_{1i}, \alpha_{2i} \geq 0 \text{ for all } i$$

Now, let's think about a specific ξ_i. Contribution is $\xi_i(C - \alpha_{1i} - \alpha_{2i})$. The opponent had better set $\alpha_{1i} + \alpha_{2i} = C$, else we can make this $-\infty$. So, replace α_{2i} with $C - \alpha_{1i}$, let $\alpha_i = \alpha_{i1}$, and have constraint $0 \leq \alpha_i \leq C$. Then the dual problem simplifies to

$$\max_{\alpha} \min_{w} \frac{1}{2}\|w\|^2 + \sum_i \alpha_i(1 - y_i(w \cdot x_i))$$
$$s.t.\ 0 \leq \alpha_i \leq C \text{ for all } i$$

We can solve inner minimization by setting gradient to 0, which yields $w = \sum_i \alpha_i y_i x_i$. Plug in to further simplify the dual problem:

$$\max_{\alpha} \min_{w} \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$
$$s.t.\ 0 \leq \alpha_i \leq C \text{ for all } i$$

Notice this is kernelizable. Hence, we can run SVMs with any kernel using the dual and replacing $x_i \cdot x_j$ with $K(x_i, x_j)$.

Chapter 4

Overfitting

4.1 Uniform Convergence

4.1.1 Chernoff and Hoeffding bounds

Consider m flips of a coin of bias p. Let N be the observed numebr of heads. Let $\epsilon, \alpha \in [0, 1]$.

Hoeffding bounds:

- $\mathbb{P}(\frac{N}{m} > p + \epsilon) \leq e^{-2m\epsilon^2}$ for $\epsilon \in [0, 1]$.
- $\mathbb{P}(\frac{N}{m} < p + \epsilon) \leq e^{-2m\epsilon^2}$ for $\epsilon \in [0, 1]$.

Chernoff bounds:

- $\mathbb{P}(\frac{N}{m} > p(1 + \alpha)) \leq e^{-\frac{1}{3}mp\alpha^2}$ for $\alpha > 0$.
- $\mathbb{P}(\frac{N}{m} < p(1 - \alpha)) \leq e^{-\frac{1}{2}mp\alpha^2}$ for $\alpha \in (0, 1]$.

The direct application of the Chernoff and Hoeffding bounds are as follows.

Theorem 9. (Uniform Convergence) *If $|\mathcal{S}| \geq \frac{1}{2\epsilon^2}[\ln(2|\mathcal{H}|) + \ln\left(\frac{1}{\delta}\right)]$, then with probability at least $1 - \delta$, for all $h \in \mathcal{H}$ we have*

$$|err_{\mathcal{D}}(h) - err_{\mathcal{S}}(h)| < \epsilon.$$

Proof. First, fix some $h \in \mathcal{H}$ and let x_j be the indicator random variable for the event that h makes a mistake on the jth example in $\mathcal{S}$. The x_j are independent $\{0, 1\}$ random variables and the probability that x_i equals 1 is the true error of h, and the fraction of the x_j's equal to 1 is exactly the training error of h, i.e. $err_{\mathcal{S}}(h) = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} x_i$.

By union bound and Hoeffding bound, the chance of failure is

$$\mathbb{P}\{|err_{\mathcal{D}}(h) - err_{\mathcal{S}}(h)| > \epsilon,\ \forall\ h \in \mathcal{H}\} \leq 2|\mathcal{H}|\mathbb{P}\left(\frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} x_i > err_{\mathcal{D}}(h) + \epsilon\right) \leq 2|\mathcal{H}|e^{-2|\mathcal{S}|\epsilon^2}.$$

Set $\delta = 2|\mathcal{H}|e^{-2|\mathcal{S}|\epsilon^2}$, we have $|\mathcal{S}| = \frac{1}{2\epsilon^2}[\ln(2|\mathcal{H}|) + \ln\left(\frac{1}{\delta}\right)]$. ■

This bound is worse than previous bound because we are asking for something stronger, which requires the bound holds for any $h \in \mathcal{H}$. The following one get bounds between these two.

Theorem 10. *If $|\mathcal{S}| \geq \frac{6}{\epsilon^2}[\ln(|\mathcal{H}|) + \ln\left(\frac{1}{\delta}\right)]$, then with probability at least $1 - \delta$, for all $h \in \mathcal{H}$ such that $err_{\mathcal{D}}(h) > 2\epsilon$, we have $err_{\mathcal{S}}(h) > \epsilon$, and for all $h \in \mathcal{H}$ such that $err_{\mathcal{D}}(h) < \frac{\epsilon}{2}$, we have $err_{\mathcal{S}}(h) < \epsilon$.*

Proof. By Chernoff bound and union bound, the chance of all $h \in \mathcal{H}$ making errors at least 2ϵ, is at most $|\mathcal{H}|e^{-\frac{1}{3}|\mathcal{S}|\epsilon}$. Set $\delta = |\mathcal{H}|e^{-\frac{1}{3}|\mathcal{S}|}$, we have $|\mathcal{S}| = \frac{3}{\epsilon}[\ln(2|\mathcal{H}|) + \ln\left(\frac{1}{\delta}\right)]$.

Analogously, the chance of all $h \in \mathcal{H}$ making errors at most $\frac{\epsilon}{2}$, is at least $|\mathcal{H}|e^{-\frac{1}{2}|\mathcal{S}|\epsilon}$. Set $\delta = 2|\mathcal{H}|e^{-\frac{1}{2}|\mathcal{S}|}$, we have $|\mathcal{S}| = \frac{2}{\epsilon}[\ln(|\mathcal{H}|) + \ln\left(\frac{1}{\delta}\right)]$.

Let $x_1(h), \ldots, x_n(h)$ be the indicator random variables for the event that h makes a mistake on the jth example in $\mathcal{S}$.

(1) When $err_{\mathcal{D}}(h) > \epsilon$, we have

$$\begin{aligned}\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} x_{h,i} < \epsilon,\ \forall\ h \in \mathcal{H}\right) &\leq \mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} x_{h,i} < \frac{1}{2}\cdot 2err_{\mathcal{D}}(h),\ \forall\ h \in \mathcal{H}\right)\\ &\leq |\mathcal{H}|e^{-\frac{1}{8}n\cdot err_{\mathcal{D}}(h)} \leq |\mathcal{H}|e^{-\frac{1}{8}n\epsilon}.\end{aligned}$$

(2) When $err_{\mathcal{D}}(h) < \frac{\epsilon}{2}$, from Lemma 1, we have

$$\begin{aligned}\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} x_{h,i} > \epsilon,\ \forall\ h \in \mathcal{H}\right) &\leq |\mathcal{H}|\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} x_{h,i} > \epsilon \middle| err_{\mathcal{D}}(h) = \frac{\epsilon}{2}\right)\\ &= |\mathcal{H}|\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} x_i > 2err_{\mathcal{D}}(h) \middle| err_{\mathcal{D}}(h) = \frac{\epsilon}{2}\right)\\ &\leq |\mathcal{H}|e^{-\frac{1}{6}n\epsilon}.\end{aligned}$$

Another way to see this is given some h of true error less than $\frac{\epsilon}{2}$, define a new h' that is identical to h except that it has an additional error region of probability mass $\frac{\epsilon}{2} - err_{\mathcal{D}}(h)$ that raises its true error to exactly $\frac{\epsilon}{2}$. Now, we apply Chernoff to h' and we notice that h is guaranteed to have empirical error no greater than that of h', since h' makes a mistake on every example that h does.

Setting $\max\{|\mathcal{H}|e^{-\frac{1}{8}n\epsilon}, |\mathcal{H}|e^{-\frac{1}{6}n\epsilon}\} \leq \delta$, we have

$$n \geq \frac{3}{err_{\mathcal{D}}(h)}\left[\ln(\mathcal{H}) + \ln\left(\frac{1}{\delta}\right)\right].$$

■

Lemma 1. *Suppose that $X \sim Binomial(n, p)$, then the probability $\mathbb{P}(X \geq k)$ is increasing in p for $k \in \{0, 1, \ldots, n\}$ and n fixed.*

Proof 1. One way to flip a coin of bias p is to pick a uniform random number r in [0,1]: if $r < p$ then output heads, and if $r \geq p$ then output tails. Consider two coins: one of bias p and one of bias $p' > p$. We can "couple" them by using the same random number r for both coins. That is, to flip both coins m times, we pick m iid random numbers $r_1, r_2, \ldots, r_m$, and use these m numbers for both coins. Each coin by itself got flipped m times independently, but the two coins are coupled: whenever the first coin is heads, the second coin is heads too. So, the chance that the 2nd coin has k or more heads must be at least as large as the chance that the first coin has k or more heads. ■

Proof 2. Let $f_k(p) = \sum_{i \geq k} \binom{n}{i} p^i (1-p)^{n-i}$, then

$$\begin{aligned}
f_k'(p) &= \sum_{i \geq k} \binom{n}{i} [ip^{i-1}(1-p)^{n-i} - (n-i)p^i(1-p)^{n-i-1}] \\
&= n \sum_{i \geq k} \binom{n-1}{i-1} p^{i-1}(1-p)^{n-i} - n \sum_{i \geq k} \binom{n-1}{i} p^i (1-p)^{n-i-1} \\
&= n \left[\sum_{i \geq k} \binom{n-1}{i-1} p^{i-1}(1-p)^{n-i-1} \right] \cdot (1-p) - n \sum_{i \geq k} \binom{n-1}{i} p^i (1-p)^{n-i-1} \\
&= n \sum_{i \geq k} \binom{n-1}{i-1} p^{i-1}(1-p)^{n-i-1} - n \sum_{i \geq k} \binom{n-1}{i-1} p^i (1-p)^{n-i-1} - n \sum_{i \geq k} \binom{n-1}{i} p^i (1-p)^{n-i-1} \\
&= n \sum_{i \geq k} \binom{n-1}{i-1} p^{i-1}(1-p)^{n-i-1} - n \sum_{i \geq k} \binom{n}{i} p^i (1-p)^{n-i-1} \\
&= n \sum_{i \geq k} \binom{n-1}{i-1} p^{i-1}(1-p)^{n-i-1} \left(1 - \frac{np}{i}\right)
\end{aligned}$$

If $k \geq \lceil np \rceil$, then $1 - \frac{np}{i} \geq 1 - \frac{np}{k} \geq 0$ for $i \geq k$ and thus $f_k'(p) \geq 0$.

If $k < \lceil np \rceil$, then the term $p^{i-1}(1-p)^{n-i-1}\left(1 - \frac{np}{i}\right)$ is negative for $i \leq k$. Thus, $f_k'(p) \geq f_{k-1}'(p) \geq \cdots \geq f_0'(p) \equiv 0$ since $f_0(p) \equiv 1$.

So $f_k'(p)$ is always non-negative for fixed k and n. ■

4.2 VC-Dimension

If many hypotheses in $\mathcal{H}$ are very similar, we shouldn't have to pay so much.

Example 3. *Consider the class* $\mathcal{H} = \{[0,a] : 0 \leq a \leq 1\}$. *Define* α_ϵ *and* α_ϵ' *such that* $\mathbb{P}([\alpha_\epsilon, \alpha]) = \mathbb{P}([\alpha, \alpha_\epsilon]) = \epsilon$. *It is enough to get at least one example in each interval, so just need* $(1-\epsilon)^{|\mathcal{S}|} \leq \frac{\delta}{2}$ *for each interval, i.e.* $|\mathcal{S}| \geq \frac{1}{\epsilon} \ln\left(\frac{2}{\delta}\right)$.

We define $\mathcal{H}[\mathcal{S}]$ to be the set of all different ways to label points in $\mathcal{S}$ using concepts in $\mathcal{H}$, and $\mathcal{H}[m]$ to be the maximum $|\mathcal{H}[\mathcal{S}]|$ over datasets $\mathcal{S}$ of m points.

Theorem 11. *For any class* $\mathcal{H}$, *distribution* $\mathcal{D}$, *if*

$$|\mathcal{S}| = m > \frac{2}{\epsilon}\left[\log_2(2\mathcal{H}[2m]) + \log_2\left(\frac{1}{\delta}\right)\right],$$

then with probability $1-\delta$, all $h \in \mathcal{H}$ with error at least ϵ are inconsistent with data.

Proof. Given a set $\mathcal{S}$ of m examples, define $A_{\mathcal{S}}$ be the event that exists $h \in \mathcal{H}$ with $err_{\mathcal{D}}(h) \geq \epsilon$ but $err_{\mathcal{S}}(h) = 0$. We want to show $\mathbb{P}_{\mathcal{S}\sim\mathcal{D}^m}(A_{\mathcal{S}}) \leq \delta$.

Now, consider drawing two sets $\mathcal{S}, \mathcal{S}'$ of m examples each. Let $B_{\mathcal{S},\mathcal{S}'}$ be the event that exists $h \in \mathcal{H}$ with $err_{\mathcal{S}'}(h) \geq \frac{\epsilon}{2}$ but $err_{\mathcal{S}}(h) = 0$. We claim $\mathbb{P}_{\mathcal{S},\mathcal{S}'\sim\mathcal{D}^m}(B_{\mathcal{S},\mathcal{S}'}) \geq \frac{1}{2}\mathbb{P}_{\mathcal{S}\sim\mathcal{D}^m}(A_{\mathcal{S}})$. This is because that $\mathbb{P}(B) \geq \mathbb{P}(A)\mathbb{P}(B|A)$ and $\mathbb{P}(B|A) \geq \frac{1}{2}$ by Chernoff bound so long as $m \geq \frac{8}{\epsilon}$. So it is suffices to show $\mathbb{P}_{\mathcal{S},\mathcal{S}'\sim\mathcal{D}^m}(B_{\mathcal{S},\mathcal{S}'}) \leq \frac{\delta}{2}$.

Now, consider a third experiment. Draw a set $\mathcal{S}''$ of $2m$ examples, then randomly partition into $\mathcal{S}$ and $\mathbb{S}'$ of m each. Let $B_{\mathcal{S}'',\mathcal{S},\mathcal{S}'}$ be the event that exists $h \in \mathbb{H}$ with $err_{\mathcal{S}'}(h) \geq \frac{\epsilon}{2}$ but $err_{\mathcal{S}}(h) = 0$. We claim that $\mathbb{P}_{\mathcal{S}''\sim\mathcal{D}^{2m}}(B_{\mathcal{S}'',\mathcal{S},\mathcal{S}'}) = \mathbb{P}_{\mathcal{S},\mathcal{S}'\sim\mathcal{D}^m}(B_{\mathcal{S},\mathcal{S}'})$. So it is suffices to show $\mathbb{P}_{\mathcal{S}''\sim\mathcal{D}^{2m}}(B_{\mathcal{S}'',\mathcal{S},\mathcal{S}'}) \leq \frac{\delta}{2}$.

We will actually prove that for any $|\mathcal{S}''| = 2m$ (not necessary to be iid), $\mathbb{P}_{\mathcal{S}''\sim\mathcal{D}^{2m}}(B_{\mathcal{S}'',\mathcal{S},\mathcal{S}'}) \leq \frac{\delta}{2}$. Now that $\mathcal{S}''$ is fixed, at most $\mathcal{H}[2m]$ labelings to worry about. For each one, show that the chance of being perfect on $\mathcal{S}$ but error at least $\frac{\epsilon}{2}$ on $\mathcal{S}'$ is low (over the random partition into $\mathcal{S}$, $\mathcal{S}'$). Then apply union bound.

So, fix some labeling $h \in \mathcal{H}[\mathcal{S}'']$. Can assume h makes at least $\frac{\epsilon m}{2}$ mistakes in $\mathcal{S}''$ (else probability of bad event is 0).

When we split $\mathcal{S}''$ into $\mathcal{S}, \mathcal{S}'$, what's the chance all these mistakes go into $\mathcal{S}'$? Let's partition $\mathcal{S}''$ by first randomly pairing the points together $(a_1, b_1), \ldots, (a_m, b_m)$. Then for each pair i, flip a coin: if heads, $a_i \to \mathcal{S}, b_i \to \mathcal{S}'$; if tails, $a_i \to \mathcal{S}', b_i \to \mathcal{S}$. If there is any i s.t. h makes mistakes on both a_i and b_i then the chance is 0; else the chance (over the random coin flips) is at most $2^{-\frac{m\epsilon}{2}}$.

The overall failure probability is at most $\mathcal{H}[2m]2^{-\frac{m\epsilon}{2}} \leq \frac{\delta}{2}$. ■

$\mathcal{H}[m]$ is sometimes hard to calculate exactly, but can get a good bound using "VC-dimension", which is roughly the point at which $\mathcal{H}$ stops looking like it contains all functions.

4.2.1 Shattering

Definition 9. (Shattering) *A set of points $\mathcal{S}$ is shattered by a class of functions $\mathcal{H}$ if there are concepts in $\mathcal{H}$ that label $\mathcal{S}$ in all of the $2|\mathcal{S}|$ possible ways.*

Definition 10. (Vapnik-Chervonenkis-dimension) *The VC-dimension of a binary hypothesis class $\mathcal{H}$ is the size of the largest set of points that can be shattered by $\mathcal{H}$.*

So, if the VC-dimension is d, that means there exists a set of d points that can be shattered, but no set of $d+1$ points can be shattered. Furthermore, $VCdim(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$.

Lemma 2. (Sauer's Lemma) *If $VCdim(\mathcal{H}) = d$, then*

$$\mathcal{H}[m] \leq \sum_{i=0}^{d} \binom{m}{i} = O(m^d).$$

Proof. We prove by induction on $d = VCdim(\mathcal{H})$. The cases when $m \le d$ is trivially true. So the result holds for $d = 1$. For $d > 1$, we assume that for any $k < d$, the result always holds.

First, note that $\sum_{i=0}^{d} \binom{m}{i} = \sum_{i=0}^{d} \binom{m-1}{i} + \sum_{i=1}^{d} \binom{m-1}{i-1}$. If we have a set $\mathcal{S}$ of m examples, pick $x \in \mathcal{S}$, we call $h, h' \in \mathcal{H}[\mathcal{S}]$ such that they differ only on x as twins. Since $\mathcal{H}[\mathcal{S} \setminus \{x\}]$ has at most $\sum_{i=0}^{d} \binom{m-1}{i}$ labelings by induction.

Then $\mathcal{H}[\mathcal{S}]$ is larger than $\mathcal{H}[\mathcal{S} \setminus \{x\}]$ by the number of twins. Let $\mathcal{T} = \{h \in \mathcal{H}[\mathcal{S}]$ that labels x negative but has a twin that labels x positive$\}$. Then $\mathcal{H}[\mathcal{S}] - \mathcal{H}[\mathcal{S} \setminus \{x\}] = |\mathcal{T}|$.

Since $VCdim(\mathcal{H}) \ge VCdim(\mathcal{T})+1$, we have $VCdim(\mathcal{T}) \le d-1$. So $|\mathcal{T}| \le \mathcal{T}[m-1] \le \sum_{i=0}^{d-1} \binom{m-1}{i}$. ■

Corollary 2. *For any class $\mathcal{H}$ with $d = VCdim(\mathcal{H})$ and distribution $\mathcal{D}$, if*

$$|\mathcal{S}| = O\left(\frac{1}{\epsilon}\left[d \log\left(\frac{1}{\epsilon}\right) + \log\left(\frac{1}{\delta}\right)\right]\right),$$

then with probability $1-\delta$, all $h \in \mathcal{H}$ with error at least ϵ are inconsistent with data.

Theorem 12. *For any hypothesis $h \in \mathcal{H}$, there exists a distribution $\mathcal{D}$ and target $f \in \mathcal{H}$ such that if $|\mathcal{S}| < \frac{VCdim(\mathcal{H})-1}{8\epsilon}$, then $\mathbb{E}[err_{\mathcal{D}}(h)] > \epsilon$.*

Proof. Consider $d = VCdim(\mathcal{H})$ shattered points. Define distribution $\mathcal{D}$ with probability $1-4\epsilon$ on one point and $\frac{4\epsilon}{d-1}$ on the rest.

Pick a random labeling of the d points as the target f. As the VC-dimension is d, f is in $\mathcal{H}$. Then

$$\begin{aligned}
\mathbb{E}[err_{\mathcal{D}}(h)] &= \frac{1}{2}\mathbb{P}(h \text{ make mistakes on test points}) \\
&\ge \frac{1}{2}\mathbb{P}(\text{test points not in } \mathbb{S}) \\
&\ge \frac{1}{2} \cdot 4\epsilon \cdot \left(1 - \frac{4\epsilon}{d-1}\right)^{|\mathcal{S}|} \\
&\ge 2\epsilon\left(1 - \frac{|\mathcal{S}|4\epsilon}{d-1}\right) \\
&\ge 2\epsilon\left(1 - \frac{1}{2}\right) \\
&= \epsilon
\end{aligned}$$

■

We can even generalize Theorem 11

Theorem 13. *For any class $\mathcal{H}$ and distribution $\mathcal{D}$, if*

$$m = |\mathcal{S}| \ge \frac{8}{\epsilon^2}\left[\ln(\mathcal{H}[2m]) + \ln\left(\frac{2}{\delta}\right)\right],$$

then with probability at least $1-\delta$, all $h \in \mathcal{H}$ have $|err_{\mathcal{D}}(h) - err_{\mathcal{S}}(h)| \le \epsilon$.

Proof. Same as for Theorem 11 except for definition of $B_{\mathcal{S}'',\mathcal{S},\mathcal{S}'}$, the event that there exists $h \in \mathcal{H}$ with $|err_{\mathcal{D}}(h) - err_{\mathcal{S}}(h)| \geq \frac{\epsilon}{2}$. We want to show for any $|\mathcal{S}''| = 2m$, $\mathbb{P}_{\mathcal{S}''\sim\mathcal{D}^{2m}}(B_{\mathcal{S}'',\mathcal{S},\mathcal{S}'}) \leq \frac{\delta}{2}$.

Fix $h \in \mathcal{H}[\mathcal{S}'']$, pairing $(a_1, b_1), \ldots (a_m, b_m)$. Say there are m' indices i s.t. only one of $h(a_i)$ and $h(b_i)$ is a mistake. The probability that h is bad over coin-flip experiment is probability that get $|\#heads - \#tials| \geq \frac{\epsilon m}{2}$ in $m' \leq m$ flips. We can view it as ratio being off from expectation by at least $\frac{\epsilon m}{4m'}$ and apply Hoeffding. ∎

4.3 Rademacher Complexity

These bounds are nice but have two drawbacks we'd like to address:

1. Computability/estimability: say we have a hypothesis class that we don't understand well. It might be hard to compute or estimate $H[m]$.
2. Tightness: Our bounds have two sources of loss. One is we did a union bound over labelings of the double-sample $\mathcal{S}''$, which is overly pessimistic if many are very similar to each other. A second is that we did worst-case over $\mathcal{S}''$, whereas we would rather do expected case, or even have a bound that depends on our actual training set.

We will be able to address both, at least in the uniform convergence case.

Rather than writing m as a function of ϵ, write ϵ as function of m, e.g., we would write Theorem 11 as:

For any class $\mathcal{H}$ and distribution $\mathcal{D}$, with high probability all h in $\mathcal{H}$ satisfy

$$err_{\mathcal{D}}(h) \leq err_{\mathcal{S}}(h) + \sqrt{\frac{8\ln\left(\frac{2H[2m]}{\delta}\right)}{m}}.$$

Definition 11. (Rademacher Complexity) *For a given set of data $\mathcal{S} = (x_1, y_1), \ldots, (x_m, y_m)$ and class of functions $\mathcal{H}$, the empirical rademacher Complexity of $\mathcal{H}$ is:*

$$R_{\mathcal{S}}(\mathcal{H}) = \mathbb{E}_{\sigma}\left[\max_{h\in\mathcal{H}} \sum_{i}^{m} \sigma_i h(x_i)\right]$$

where $\sigma = (\sigma_1, \ldots, \sigma_m)$ is a random $\{-1, +1\}$ labeling.

The distributional rademacher complexity of $\mathcal{H}$ is $R_{\mathcal{D}}(\mathcal{H})$.

How big can $R_{\mathcal{S}}(\mathcal{H})$ be? Class $\mathcal{H}$ produces labelings $h_1, \ldots, h_{|\mathcal{H}[\mathcal{S}]|}$ of $\mathcal{S}$. For each such labeling h_i, the probability that its correlation with σ is more than 2ϵ is at most $e^{-2m\epsilon^2}$ by Hoeffding bounds. Setting this to $\frac{\delta}{|\mathcal{H}[\mathcal{S}]|}$, with high probably all $h \in \mathcal{H}$ have correlation with σ at most $2\sqrt{\frac{\ln\left(\frac{|\mathcal{H}[\mathcal{S}]|}{\delta}\right)}{2m}}$. So $R_{\mathcal{S}}(\mathcal{H})$ cannot be larger.

Lemma 3. (McDiarmid's Inequality) *If $X_1, \ldots, X_m$ are independent random variables and $\phi(x_1, \ldots, x_m)$ is a real-valued function such that*

$$|\phi(x_1, \ldots, x_i, \ldots, x_m) - \phi(x_1, \ldots, y_i, \ldots, x_m)| \leq L_i |x_i - y_i|,$$

then

$$\mathbb{P}\left(\phi(\boldsymbol{X}) > \mathbb{E}[\phi(\boldsymbol{X})] + \epsilon\right) \le e^{-2\frac{\epsilon^2}{\sum_{i=1}^m L_i^2}}.$$

If $X_i \in [0,1]$ and $\phi(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^m x_i$, then $L_i = \frac{1}{m}$ and we recover the Hoeffding bound.

Theorem 14. *For any class $\mathcal{H}$, distrib $\mathcal{D}$, if $\mathcal{S} \sim \mathcal{D}^m$ then with prob at least $1-\delta$, all $h \in \mathcal{H}$ satisfy*

$$\begin{aligned} err_{\mathcal{D}}(h) &\le err_{\mathcal{S}}(h) + R_{\mathcal{D}}(\mathcal{H}) + \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{m}} \\ &\le err_{\mathcal{S}}(h) + R_{\mathcal{S}}(\mathcal{H}) + 3\sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{m}}. \end{aligned}$$

Proof. Define $MAXGAP(\mathcal{S}) = \max_{h\in\mathcal{H}}[err_{\mathcal{D}} - err_{\mathcal{S}}(h)]$. Think of $MAXGAP(\mathcal{S})$ as ϕ in McDiarmid, then with probability at least $1-\frac{\delta}{2}$, $MAXGAP(\mathcal{S}) \le \mathbb{E}_{\mathcal{S}}[MAXGAP(\mathcal{S})] + \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2m}}$. So, it suffices to show $\mathbb{E}_{\mathcal{S}}[MAXGAP(\mathcal{S})] \le R_{\mathcal{D}}(\mathcal{H})$.

Similarly, with probability at least $1-\frac{\delta}{2}$, $R_{\mathcal{S}}$ is within $2\sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2m}}$ of $R_{\mathcal{D}}(\mathcal{H})$.

We rewrite $err_{\mathcal{D}}(h)$ as $\mathbb{E}_{\mathcal{S}'}[err_{\mathcal{S}'}(h)]$ where $\mathcal{S}'$ is ghost sample. Then

$$\begin{aligned} \mathbb{E}_{\mathcal{S}}[MAXGAP(\mathcal{S})] &= \mathbb{E}_{\mathcal{S}}\left[\max_{h\in\mathcal{H}}(\mathbb{E}_{\mathcal{S}'}[err_{\mathcal{S}'}(h)] - err_{\mathcal{S}}(h))\right] \\ &\le \mathbb{E}_{\mathcal{S},\mathcal{S}'}\left[\max_{h\in\mathcal{H}}(err_{\mathcal{S}'}(h) - err_{\mathcal{S}}(h))\right] \end{aligned}$$

Let $\mathcal{S} = \{(x_1,y_1),\ldots,(x_m,y_m)\}$, $\mathcal{S}' = \{(x_1',y_1'),\ldots,(x_m',y_m')\}$, and $err_{x_i}(h) = \mathbb{1}_{\{h(x_i)\neq y_i\}}$, then

$$\mathbb{E}_{\mathcal{S}}[MAXGAP(\mathcal{S})] \le \mathbb{E}_{\mathcal{S},\mathcal{S}'}\left[\max_{h\in\mathcal{H}}\frac{1}{m}\sum_{i=1}^m(err_{x_i}(h) - err_{x_i'}(h))\right]$$

Now, like in the VCdim proof, let's flip a coin σ_i for each i to decide whether or not to swap (x_i,y_i) and (x_i',y_i') before taking the max.

$$\begin{aligned} &\mathbb{E}_{\mathcal{S},\mathcal{S}',\sigma}\left[\max_{h\in\mathcal{H}}\frac{1}{m}\sum_{i=1}^m\sigma_i(err_{x_i}(h) - err_{x_i'}(h))\right] \\ \le&\mathbb{E}_{\mathcal{S},\sigma}\left[\max_{h\in\mathcal{H}}\frac{1}{m}\sum_{i=1}^m\sigma_i err_{x_i}(h)\right] - \mathbb{E}_{\mathcal{S}',\sigma}\left[\min_{h\in\mathcal{H}}\frac{1}{m}\sum_{i=1}^m\sigma_i err_{x_i'}(h)\right] \\ \le&2\mathbb{E}_{\mathcal{S},\sigma}\left[\max_{h\in\mathcal{H}}\frac{1}{m}\sum_{i=1}^m\sigma_i err_{x_i}(h)\right]. \end{aligned}$$

We are looking at the correlation of the losses of h with σ, rather than the correlation of h with σ.

To fix these, suppose we cheated by changing the definition of $R_{\mathcal{D}}(\mathcal{H})$ so that σ is a random $\{-1,1\}$ multiplier applied to the true labels rather than a random $\{-1,1\}$ labeling. Is that cheating?

$$\begin{aligned}
R_{\mathcal{D}}(\mathcal{H}) &= \mathbb{E}_{\mathcal{S},\sigma}\left[\max_{h\in\mathcal{H}}\frac{1}{m}\sum_{i=1}^{m}\sigma_i y_i h(x_i)\right] \\
&= \mathbb{E}_{\mathcal{S},\sigma}\left[\max_{h\in\mathcal{H}}\frac{1}{m}\sum_{i=1}^{m}\sigma_i\left(1-2err_{x_i}(h)\right)\right] \\
&= \mathbb{E}_{\mathcal{S},\sigma}\left[\frac{1}{m}\sum_{i=1}^{m}\sigma_i+\max_{h\in\mathcal{H}}\frac{1}{m}\sum_{i=1}^{m}\left(-\sigma_i 2err_{x_i}(h)\right)\right] \\
&= 2\mathbb{E}_{\mathcal{S},\sigma}\left[\max_{h\in\mathcal{H}}\frac{1}{m}\sum_{i=1}^{m}(-\sigma_i)err_{x_i}(h)\right].
\end{aligned}$$

■

Chapter 5

Boosting

5.1 Boosting

Definition 12. (Weak Learner) *Algorithm $\mathcal{A}$ is a weak-learner with edge γ for class $\mathcal{H}$ if: for any distribution $\mathcal{D}$ over examples labeled by some target $f \in \mathcal{H}$, with high probability $\mathcal{A}$ produces a hypothesis h with $err_{\mathcal{D}}(h) \leq \frac{1}{2} - \gamma$.*

We can ignore δ here since it can be handled easily (HW2).

We will end up running $\mathcal{A}$ for T times producing hypotheses $h_1, \ldots, h_T$ and combining them into a single rule. By Problem 3 on HW3, the set of such combinations has VC-dim $O(Td\log(Td))$. This will allow us to do all this on a sample of size $\tilde{O}\left(\frac{TD}{\epsilon}\right)$ where $\tilde{O}$ notation hides logarithmic factors.

We will draw a training sample $\mathcal{S}$ of size $m = \tilde{O}\left(\frac{TD}{\epsilon}\right)$. Assume that given any weighting of the points in $\mathcal{S}$, $\mathcal{A}$ will return a hypothesis h of error at most $\frac{1}{2} - \gamma$ over the distribution induced by that weighting. (ignoring δ)

We will show that we can produce h with $err_{\mathcal{S}}(h) = 0$ for $T = \left(\frac{\log m}{\gamma^2}\right)$. Then we just need $m >> \frac{d \log m}{\epsilon\gamma^2} \approx \frac{d \log\left(\frac{d}{\epsilon\gamma}\right)}{\epsilon\gamma^2}$.

Algorithm 6. (Adaboost-Light)

```
1. Given labeled sample $\mathcal{S} = \{x_1, \ldots, x_m\}$, initialize each example $x_i$ to have weight $w_i = 1$. Let
   $\mathbf{w} = (w_1, \ldots, w)$.
2. For $t = 1, \ldots, T$ do:
   a. Call $\mathcal{A}$ on the distribution $\mathcal{D}_t$ over $\mathcal{S}$ induced by $\mathbf{w}$.
   b. Receive hypothesis $h_t$ of error at most $\frac{1}{2} - \gamma$ over $\mathcal{D}_t$.
   c. Multiply the weight of each example misclassified by $h_t$ by $\alpha = \frac{\frac{1}{2}+\gamma}{\frac{1}{2}-\gamma}$. Leave the other
      weights alone.
3. Output the majority-vote classifier $MAJ(h_1, \ldots, h_T)$. Assume $T$ is odd so no ties.
```

Theorem 15. *For Adaboost-Light Algorithm, $T = O\left(\frac{\log m}{\gamma^2}\right)$ is sufficient such that $err_{\mathcal{S}}(MAJ(h_1, \ldots, h_T)) = 0$.*

Proof. First, if $MAJ(h_1, \ldots, h_T)$ makes a mistake on any x_i then its final weight must be greater than $\alpha^{\frac{T}{2}}$. Let W_t be total weight after update t and $W_0 = m$. By the weak-learning assumption, h_t has error at most $\frac{1}{2} - \gamma$ on $\mathcal{D}_t$. So, at most $\frac{1}{2} - \gamma$ fraction of weight multiplied by α. So

$$W_{t+1} \leq \left[\alpha\left(\frac{1}{2} - \gamma\right) + \left(\frac{1}{2} + \gamma\right)\right] W_t = (1 + 2\gamma)W_t.$$

So if $err_{\mathcal{S}}(MAJ(h_1, \ldots, h_T)) > 0$, then $\alpha^{\frac{T}{2}} \leq W_T \leq (1 + 2\gamma)^T m$ and $1 \leq \alpha^{-\frac{T}{2}}(1 + 2\gamma)^T m$.

Substituting $\alpha = \frac{\frac{1}{2}+\gamma}{\frac{1}{2}-\gamma}$, we get

$$1 \leq (1 - 2\gamma)^{\frac{T}{2}}(1 + 2\gamma)^{\frac{T}{2}} m = (1 - 4\gamma^2)^{\frac{T}{2}} m \leq e^{-2\gamma^2 T} m.$$

Once $T > \frac{\ln m}{2\gamma^2}$, we have $e^{-2\gamma^2 T} m < 1$, i.e., $err_{\mathcal{S}}(MAJ(h_1, \ldots, h_T)) = 0$.

More generally, after any T steps, the fraction of mistakes is at most $e^{-2\gamma^2 T} m$. ■

Theorem 16. *Given a weak-learner $\mathcal{A}$ with edge γ for class $\mathcal{H}$, we can produce an algorithm $\mathcal{A}'$ that achieves a PAC guarantee for class $\mathcal{H}$ (with high probability produces hypothesis with error at most ϵ) using $O\left(\frac{1}{\gamma^2} \log \frac{1}{\epsilon}\right)$ calls to $\mathcal{A}$. $\mathcal{A}'$ is efficient if $\mathcal{A}$ is.*

Chapter 6

Statistical Query Models

6.1 Random Classification Noise Models

When there is no perfect predictor, by Hoeffding/Chernoff bounds, minimizing training error will approximately minimize true error: just need $O\left(\frac{1}{\epsilon^2}\right)$ samples versus $O\left(\frac{1}{\epsilon}\right)$.

However, it is NP-hard to find an approximately best conjunction woth polynomial-time algorithms. Indeed, we can do other things, like minimize hinge-loss, but this may be a big gap with respect to error rate of 0-1 loss.

One way to make progress is to make assumptions on the noise in the data, e.g., random classification noise model.

Definition 13. (Random Classification Noise Models) *In the random classification noise model, we have a target function $f \in C$ from a PAC model, but assume labels from noisy channel. Specifically, we assume a noisy oracle $\mathbb{E}X^{\eta}(f, \mathcal{D})$ where η is the noise rate. $l(x) = f(x)$ with probability $1 - \eta$ and $l(x) = 1 - f(x)$ with probability η.*

We say that algorithm $\mathcal{A}$ PAC-learns C from random classification noise if for any $f \in C$, any distribution $\mathcal{D}$, any $\eta < \frac{1}{2}$, $\epsilon, \delta > 0$, given access to $\mathbb{E}X^{\eta}(f, \mathcal{D})$, $\mathcal{A}$ finds a hypothesis h that is ϵ-close to f, with probability at least $1 - \delta$. We want time poly $\left(\frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{1-2\eta}, n, size(f)\right)$.

If h has non-noisy error p, i.e., $\mathbb{P}_{\mathcal{D}}(h(x) \neq f(x)) = p$ then the noisy error rate is $\mathbb{P}_{\mathcal{D}}(h(x) \neq l(x)) = p(1 - \eta) + (1 - p)\eta = \eta + p(1 - 2\eta)$.

Example 4. *To learn monotone OR-functions, assume that η is known and let $p_i = \mathbb{P}(f(x) = 0,\ x_i = 1)$. Any h that includes all x_i such that $p_i = 0$ and no x_i such that $p_i > \frac{\epsilon_i}{n}$ is good. So we just need to estimate p_i to $\pm\frac{\epsilon}{2n}$. Rewrite $p_i = \mathbb{P}(f(x) = 0|x_i = 1)\mathbb{P}(x_i = 1)$ and define $q_i = \mathbb{P}(f(x) = 0|x_i = 1)$. Then $\mathbb{P}_{\eta}(l(x) = 0|x_i = 1) = q_i(1 - \eta) + (1 - q_i)\eta = \eta + (1 - 2\eta)q_i$. So it is enough to approximate $\mathbb{P}_{\eta}(l(x) = 0|x_i = 1)$ to $\pm O\left(\frac{\epsilon}{2n}(1 - 2\eta)\right)$. If the noise rate is not known, we can estimate with smallest value of $\mathbb{P}_{\eta}(l(x) = 0|x_i = 1)$.*

Basic idea of algorithm is:

- See how we can learn in non-noisy model by asking about probabilities of certain events with some slops.
- Try to learn in noisy model by breaking events into:
  - Parts predictably affected by noise.
  - Parts unaffected by noise.

6.2 Statistical Query Models

6.2.1 Definition

Definition 14. *Given a target function $f \in C$, a statistical query is a pair (χ, τ) such that*

- $\chi : \mathcal{X} \times \{0,1\} \mapsto \{0,1\}$,
- $\tau > 0$ *is a tolerance parameter.*

The statistical query oracle returns a value of $\mathbb{E}_{\mathcal{D}}[\chi(x, f(x))]$ with additive error at most τ. We say that f is efficiently SQ-learnable if there exists an algorithm $\mathcal{A}$ such that for any $f \in C$, any probability distribution $\mathcal{D}$, and any $\epsilon > 0$, there is a polynomial $p(\cdot,\cdot,\cdot)$ such that

- *$\mathcal{A}$ makes at most $p\left(\frac{1}{\epsilon}, n, size(f)\right)$ calls to the SQ oracle,*
- *the smallest τ that $\mathcal{A}$ uses satisfies $\frac{1}{\tau} \le p\left(\frac{1}{\epsilon}, n, size(f)\right)$,*
- *the query χ are evaulable in time $p\left(\frac{1}{\epsilon}, n, size(f)\right)$,*
- *$\mathcal{A}$ outputs a hypothesis h satisfying $err_{\mathcal{D}}(h) \le \epsilon$.*

6.2.2 Properties

1. Many algorithms can be simulated with statistical queries.
2. The SQ-model can be automatically converted to work in presence of classification noise.
3. It can give a nice characterization of what can and cannot be learned in it.

6.2.3 SQ-Learnable Implies PAC-Learnable with Noises

Let $\chi(x, l(x)) = 1$ if and only if $x = 1$ and $l(x) = 0$.

How to estimate $\mathbb{P}(\chi(x, f(x)) = 1)$? Let $C = \{x : \chi(x,0) = \chi(x,1)\}$ and $N = \{x : \chi(x,0) \ne \chi(x,1)\}$, then

$$\mathbb{P}(\chi(x, f(x)) = 1) = \mathbb{P}(\chi(x, f(x)) = 1,\ x \in C) + \mathbb{P}(\chi(x, f(x)) = 1,\ x \in N).$$

We can estimate $\mathbb{P}(x \in N)$ and $p_\eta = \mathbb{P}_\eta(\chi(x,l) = 1 | x \in N)$. We want $p = \mathbb{P}(\chi(x, f(x)) = 1 | x \in N)$. Write $p_\eta = p(1-\eta) + (1-p)\eta = \eta + (1-2\eta)p$. So $p = \frac{p_\eta - \eta}{1-2\eta}$. We just need to estimate p_η to additive error $\tau(1-2\eta)$. If we don't know η, we can have guess and check wrapper. So, any statistical query algorithm can automatically be simulated in the presence of random classification noise.

We say that two binary functions f and g are uncorrelated if $\mathbb{P}_{\mathcal{D}}(f = g) = \frac{1}{2}$.

Definition 15. (SQ-dimension) *The SQ-dimension of a class C with respect to $\mathcal{D}$ is the size of the largest set $C' \subset C$ s.t. for all $f, g \in C'$,*

$$\left|\mathbb{P}_{\mathcal{D}}(f = g) - \frac{1}{2}\right| < \frac{1}{|C'|},$$

i.e., the size of largest set of nearly uncorrelated functions in C.

Example 5. (Parity Functions) Let C be the set of parity functions $c(x) = c \cdot x \bmod 2$. Let $\mathcal{D}$ be uniform on $\{0,1\}^n$. Any two parity functions are uncorrelated. So, $SQdim(C) = 2^n$. Any parity function of size $\log_2(n)$ can be described as a size-n decision tree. So, polynomial-sized decision trees are not SQ-learnable either.

Theorem 17. *If $SQdim_{\mathcal{D}}(C) \leq poly(n)$, then we can weak-learn C over $\mathcal{D}$ by SQ algorithms with error rate at most $\frac{1}{2} - \frac{1}{poly(n)}$.*

Proof. Let $d = SQdim_{\mathcal{D}}(C)$. Let $\mathcal{H} \subset C$ be a maximal subset such that for all $h_i, h_j \in \mathcal{H}$, we have

$$\left|\mathbb{P}_{\mathcal{D}}[h_i(X) = h_j(X)] - \frac{1}{2}\right| < \frac{1}{d+1}.$$

So $|\mathcal{H}| < d$. To learn, just try each $h_i \in \mathcal{H}$ and use an SQ model to estimate its error. At least one h_i (or $\neg h_i$) must be a weak predictor. ■

Theorem 18. *If $SQdim_{\mathcal{D}}(C) > poly(n)$, then we cannot weak-learn C over $\mathcal{D}$ by SQ algorithms.*

Proof. To keep things simpler, will change "nearly uncorrelated" to "uncorrelated", i.e., we will assume there are more than poly(n) uncorrelated functions in C.

First we introduce the tool of Fourier analysis of boolean functions. We view function f as a vector of 2^n entries $(\sqrt{\mathcal{D}[x]}f(x))_{x\in\mathcal{S}}$. In other words, the truth-table of f, where entry x is weighted by the square-root of the probability of x. Then $\langle f, f\rangle = 1$ and $\langle f, g\rangle = \mathbb{E}_{\mathcal{D}}[f(X)g(X)] = \mathbb{P}_{\mathcal{D}}(f(X) = g(X)) - \mathbb{P}_{\mathcal{D}}(f(X) \neq g(X))$, which we call the correlation of f and g.

We are in a 2^n – dimensional space, so an orthonormal basis is a set of 2^n orthogonal unit vectors. Let's fix one $\varphi_1, \ldots, \varphi_{2^n}$. Given a vector f, let f_i be the ith entry in the standard basis, $f_i = f(i)\sqrt{\mathcal{D}[i]}$. Then the Fourier coefficient $\hat{f}_i = \langle f, \varphi_i\rangle$ is the ith entry in the φ basis. Consider any Boolean function f. Since it's a unit-lengt vector, this means $\sum_i \hat{f}_i^2 = 1$, which is "Parseval's identity". At most t^2 of the φ_i can have $|\langle f, \varphi_i\rangle| = |\hat{f}_i| \geq \frac{1}{t}$, i.e, any given Boolean function can have correlation at least $\frac{1}{t}$ with at most t^2 functions in an orthogonal set. In particular, any given f can be weakly correlated with at most a polynomial number of them.

If C has $n^{\Omega(1)}$ uncorrelated functions, target is a random one of them, SQs all of form "what is correlation of target with h up to $\pm\frac{1}{poly(n)}$" then with high probability the oracle can always answer

0. It turns out that any SQ can be converted into a portion that looks like this, and a portion that doesn't depend on the target function at all.

Let $\varphi_1, \dots, \varphi_m$ be orthogonal functions in C. Extend arbitrarily to a basis $\varphi_1, \dots, \varphi_{2^n}$. (excess vectors may not be Boolean functions and may not be in C) Now, consider a SQ $\chi : \{0,1\}^n \times \{-1,1\} \mapsto [-1,1]$. Can view this as a vector in 2^{n+1} dimensions. To apply Fourier analysis to this, need to extend our basis to this higher-dimensional space. Define distribution $\mathcal{D}' = \mathcal{D} \times Uniform(\{\pm 1\})$, $\varphi_i(x,y) = \varphi_i(x)$, and $h_i(x,y) = y\varphi_i(x)$. Then, $\varphi_1(x,y), \dots, \varphi_{2^n}(x,y), h_1(x,y), \dots, h_{2^n}(x,y)$ forms a basis of $\mathcal{D}'$.

Now do Fourier decomposition on $\chi(x,y) = \sum_i \alpha_i \varphi_i + \sum_i \beta_i h_i$ where $\sum_i \alpha_1 + \sum_i \beta_i = 1$. So we can write the quantity we care about as:

$$\mathbb{E}_{\mathcal{D}}\left[\chi(x,c(x))\right] = \mathbb{E}_{\mathcal{D}}\left[\sum_i \alpha_i \varphi_i(x) + \sum_i \beta_i h_i(x,c(x))\right] = \sum_i \alpha_i \mathbb{E}_{\mathcal{D}}[\varphi_i(x)] + \sum_i \beta_i \mathbb{E}_{\mathcal{D}}[h_i(x,c(x))].$$

The first term doesn't depend on target at all, which we call $g(\chi, \mathcal{D})$. Recall that c is random from $\{\varphi_1, \dots, \varphi_m\}$, let's say $c = \varphi_{i^*}$. So the second term is β_{i^*}. So with high probability, the oracle can just return $g(\chi, \mathcal{D})$.

If C contains more than $poly(n)$ many uncorrelated functions, then can't learn in SQ model, which holds also for "nearly uncorrelated" as in SQ-dim definition. ■

Chapter 7

Computational Hardness of Learning

7.1 Introduction

We know efficient algorithms for various problems:

- Given a dataset $\mathcal{S}$, find a consistent decision list if one exists.
- Given a dataset $\mathcal{S}$, find a consistent linear threshold function if one exists.
- Given a dataset $\mathcal{S}$, find a consistent disjunction if one exists.

But what about the following?

- Given a dataset $\mathcal{S}$, find a consistent AND of 2 linear threshold functions (intersection of two halfspaces) if one exists.
- Given a dataset $\mathcal{S}$, find a consistent AND of 2 disjunctions (2 clause CNF formula) if one exists.
- Given a dataset $\mathcal{S}$, find a linear threshold function with the fewest mistakes on $\mathcal{S}$.
- Can we get algorithms that are guaranteed to solve these in polynomial-time without assumptions on the distribution, etc.?

The answer is no because they are NP-hard.

7.2 Hardness for Intersection of 2 Halfspaces

Reduction from NP-hard hypergraph 2-coloring problem. Given m subsets $S_1, \ldots, S_m$ of n nodes, color each node Red or Blue such that each S_i has at least one red node and at least one blue node.

Proof. Consider points in $\mathbb{R}^n$. Label origin positive and each coordinate vector e_j negative. For each set S_i, label vector $\sum_{e \in S_i} e$ for that set as positive. We claim that data is consistent with an intersection of 2 halfspaces if and only if the given instance is 2-colorable.

Define halfspaces $w_1 x_1 + \cdots + w_n x_n \leq \frac{1}{2}$ where $w_j = 1$ if j is red and $w_j = -n$ if j is blue, and $w'_1 x_1 + \cdots + w'_n x_n \leq \frac{1}{2}$ where $w'_j = 1$ if j is blue and $w'_j = -n$ if j is red.

Given halfspaces, call one "Red" and the other "Blue". If e_j is separated from orgin by Red halfspace, then color j Red; if separated by Blue halfspace then color j Blue; if separated by both, pick arbitrarily.

No 1-color set because convex hulls of $\{e_j \in S_i\}$ and $S_i \cup \{0\}$ overlap. ■

7.3 Hardness for Learning 2-Clause CNF

A 2-clause CNF formula is an AND of two OR functions, e.g., $(x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6)$. It is also NP-hard to find a consistent 2-clause CNF if one exists. But, you can learn a 2-clause CNF using a 2-DNF representation (Each AND is called a "term"). Any 2-clause CNF can be multiplied out to a 2-DNF, e.g., $x_1x_4 \vee x_1x_5 \vee x_1x_6 \vee \cdots$. We can learn it by using the "list-and-cross-off" algorithm. But the VC-dimension is now $O(n^2)$.

7.4 Hardness for Learning LTFs with Lowest Empirical Error

We know how to find a consistent LTF when one exists. Can also minimize total hinge loss. But what about minimizing the training error? It turns out this is NP-hard.

Proof. Reduction from the Maximum Independent Set problem in graphs. Origin is positive, each coordinate vector e_j is negative, for each edge (i, j) put a positive at $e_i + e_j$.

Notice that maximum independent set corresponds to largest set of negatives that can be linearly separated from the positives.

This shows the problem "Find the LTF that correctly classifies all positives and makes fewest mistakes on negatives" is NP-hard.

To finish off the argument, just replicate each positive example $n + 1$ times (so min error will only make mistakes on negatives. ■

7.5 Representation-Independent Hardness

These results show hardness for PAC learning using a particular hypothesis class $\mathcal{H}$. (We gave hardness for consistency problem, but can set uniform distribution on the output of the reduction and set $\epsilon < \frac{1}{m}$ with m is the number of points).

Representation-independent: hardness based on complexity of target, allowing learner to use any representation it wants.

Examples:

- Parity functions require $2^{\Omega(n)}$ SQs of tolerance $\frac{1}{poly(n)}$ to learn in SQ model.
- Decision trees, DNF formulas require $n^{O(\log n)}$ SQs of tolerance $\frac{1}{poly(n)}$ to learn in SQ model.
- Hardness was even for doing slightly better than random guessing.

These results don't restrict representation used by learning algorithm, but they do restrict the way the algorithm can interact with the data.

What if we don't want to restrict either one? In this case, can use cryptographic assumptions.

Definition 16. (PseudoRandom Generator) *A function $f : \{0,1\}^n \mapsto \{0,1\}^m$ where $m > n$ is a pseudorandom generator if for any poly-time algorithm $\mathcal{A}$, any constant c,*

$$|\mathbb{P}_{v\sim\{0,1\}^m}[\mathcal{A}(v) = 1] - \mathbb{P}_{x\sim\{0,1\}^n}[\mathcal{A}(f(x)) = 1]| = O\left(\frac{1}{n^c}\right).$$

In other words, no polynomial-time algorithm $\mathcal{A}$ can distinguish pseudorandom strings of length m (the result of running f on random input of length n) and truly random strings of length m.

Classic result: we can construct generator f such that breaking f would give a polynomial-time algorithm for factoring. So, f is a PRG if factoring is hard. (Also known for some other hard problems too).

Any algorithm that can even weak-learn arbitrary $O(\log n)$-depth AND/OR networks over uniform random examples in polynomial time would give a poly-time algorithm for factoring.

High-level idea:

- Think of PRG with significant stretch: $\{0,1\}^n \mapsto \{0,1\}^{poly(n)}$.
- Network has PRG input I built in, computes $f(I)$, and outputs jth bit, where j is given by low-order $\log_2(n^2)$ bits of example x.
- If the algorithm can learn, then it can distinguish PRG output from true random.

We can even extend to pseudo-random functions instead of pseudo-random generators. These are indistinguishable from truly random even with query access.

More recent results are: Under a stronger assumption that $m < n^{\sqrt{k}\log(k)}$ examples over $\{0,1\}^n$ with k bits set to 1 in each, for any constant c, there is no polynomial-time algorithm that given any sample $\mathcal{S}$ of n^c points in $\{-1,+1\}^n$ can with high probability distinguish the case (a) that labels are just uniform random coin flips, versus (b) there exists a parity function of error at most ϵ.

So this is pretty sad. Luckily, real-world problems are much nicer and simple local update algorithms have been very successful.

One major challenge of learning theory is to reconcile the power of deep learning (and the ability to use simple local updates to learn complex representations in general) with these worst-case hardness results. Clearly, the problems where deep learning succeeds are not worst-case, and understanding what makes them easier is a major research area.

Chapter 8

Game Theory

8.1 Introduction

Game theory is a field developed by economists to study social and economic interactions. They wanted to understand why people behave the way they do in different economic situations, effects of incentives, and rational explanation of behavior.

"Game" means the interaction between parties with their own interests. Could be called *interaction theory.*

Important for understanding/improving large systems

- Internet routing, social networks, e-commerce.
- Problems like spam etc.

Setting:

- Have a collection of participants, or *players*.
- Each has a set of choices, or *strategies* for how to play/behave.
- Combined behavior results in *payoffs* (satisfaction level) for each player.

8.2 2-Player Zero-Sum Games

Game defined by matrix with row for each of Row's options and a column for each of Col's options. Matrix $\boldsymbol{R}$ gives row player's payoffs, $\boldsymbol{C}$ gives column player's payoffs, $\boldsymbol{R} + \boldsymbol{C} = 0$.

Minimax optimal strategy is a (randomized) strategy that has the best guarantee on its expected payoff, over choices of the opponent, i.e., the thing to play if your opponent knows you well.

If $\boldsymbol{R} = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$, then the minimax optimal strategy for shooter is 50/50. Guarantees expected payoff at least $\frac{1}{2}$ no matter what goalie does. The minimax optimal strategy for goalie is 50/50. Guarantees expected shooter payoff at most $\frac{1}{2}$ no matter what shooter does.

If $\boldsymbol{R} = \left[\begin{smallmatrix} \frac{1}{2} & 1 \\ 1 & 0 \end{smallmatrix}\right]$, then the minimax optimal strategy for shooter is $(\frac{2}{3}, \frac{1}{3})$. Guarantees expected payoff at least $\frac{2}{3}$ no matter what goalie does. The minimax optimal strategy for goalie is $(\frac{1}{3}, \frac{2}{3})$.

Guarantees expected shooter payoff at most $\frac{2}{3}$.

Theorem 19. (Minimax Theorem) *Every 2-player zero-sum game has a unique value V such that*

- *Minimax optimal strategy for R guarantees R's expected gain at least V.*
- *Minimax optimal strategy for C guarantees C's expected loss at most V.*

Proof. Suppose for contradiction it was false. This means some game G has $V_C > V_R$:

- If Column player commits first, there exists a row that gets the Row player at least V_C.
- But if Row player has to commit first, the Column player can make him get only V_R.

We scale matrix so payoffs to row are in $[-1, 0]$. Let $\delta = V_C - V_R$.

Now, consider playing randomized weighted majority algorithm as Row, against Col who plays optimally against Row's distribution. In T steps, in expectation, the algorithm gets at least the payoff of the best Row in hindsight minus $2(T \log n)^{\frac{1}{2}}$, and the best Row in hindsight gets at most TV_C (best against opponent's empirical distribution). However, the algorithm gets at most TV_R (each time, opponent knows randomized strategy). The gap is δT, which contradicts assumption once $\delta T > 2(T \log n)^{\frac{1}{2}}$, or $T > \frac{4 \log n}{\delta^2}$. ■

This means that it doesn't hurt to publish your strategy if both players are optimal. (Borel had proved for symmetric 5×5 but thought was false for larger games)

Theorem 20. *No-regret strategies will do nearly as well or better against any sequence of opponent plays.*

Do nearly as well as best fixed choice in hindsight, which implies do nearly as well as best distribution in hindsight, and therefore do nearly as well as minimax optimal.

What if two regret minimizers play each other? Then their time-average strategies must approach minimax optimality.

1. If Row's time-average is far from minimax, then Col has strategy that in hindsight substantially beats value of game.
2. So, by Col's no-regret guarantee, Col must substantially beat value of game.
3. So Row will do substantially worse than value.
4. Contradicts no-regret guarantee for Row.

8.3 Boosting And Game Theory

Suppose we have an algorithm $\mathcal{A}$ that for any distribution (weighting function) over a dataset $\mathcal{S}$ can produce a rule $h \in \mathcal{H}$ that gets at most 45% error. Then the Adaboost gives a way to use such an $\mathcal{A}$ to get error which goes to 0 at a good rate, using weighted votes of rules produced.

How can we see that this is even possible? Let's assume the class $\mathcal{H}$ is finite. Think of a matrix game where columns indexed by examples in $\mathcal{S}$, rows indexed by h in $\mathcal{H}$, where $M_{ij} = 1$ if $h_i(x_j)$ is correct, else $M_{ij} = -1$.

Assume for any $\mathcal{D}$ over columns, there exists a row such that the expected payoff is at least 0.1. Minimax implies there exists a weighting over rows such that for every x_i, expected payoff is at least 0.1. So, $\text{sign}(\sum_t a_t h_t)$ is correct on all x_i. Weighted vote has L_1 margin at least 0.1. AdaBoost gives us a way to get this with only access via weak learner. But this at least implies existence.

8.4 General-Sum Games

In general-sum games, we can get win-win and lose-lose situations.

Definition 17. (Nash Equilibrium) *A Nash Equilibrium is a stable pair of strategies (could be randomized). Stable means that neither player has incentive to deviate on their own.*

Theorem 21. (Existence of Nash Equilibrium) *Any general-sum game must have at least one such equilibrium, which might requires randomized strategies (called "mixed strategies").*

Corollary 3. *Existence of Nash Equilibrium implies minimax theorem.*

Proof. Pick some NE and let V be the value to row player in that equilibrium. Since it's a NE, neither player can do better even knowing the (randomized) strategy their opponent is playing. So, they're each playing minimax optimal. ■

What if all players minimize regret? In zero-sum games, empirical frequencies quickly approaches minimax optimal. In general-sum games, does behavior quickly (or at all) approach a Nash equilibrium? After all, a Nash Equilibrium is exactly a set of distributions that are no-regret wrt each other. So if the distributions stabilize, they must converge to a Nash equilibrium.

What can we say? If algorithms minimize "internal" or "swap" regret, then empirical distribution of play approaches correlated equilibrium.

More general forms of regret:

1. External regret.
2. Regret with time-intervals.
3. Internal regret, or swap regret.

Definition 18. (Correlated Equilibrium) *Distribution over entries in matrix, such that if a trusted party chooses one at random and tells you your part, you have no incentive to deviate.*

A Nash equilibrium is two distributions for the row and the column, while correlated equilibrium is a distribution over entries. Thus, Nash equilibrium implies correlated equilibrium.

If all parties run a low swap regret algorithm, then empirical distribution of play is an approximately correlated equilibrium.