Machine Learning

CS 485

Shai Ben-David

LaTeXed by Sibelius Peng

Preface

Disclaimer Much of the information on this set of notes is transcribed directly/indirectly from the lectures of CS 485 during Fall 2020 as well as other related resources. I do not make any warranties about the completeness, reliability and accuracy of this set of notes. Use at your own risk.

Since the course is online, we are watching recordings from a previous offering. Videos are available on https://www.newworldai.com/understanding-machine-learning-course/. The textbook for this course is available at http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning.

Some notations:

- $D[A]$ denotes the probability hitting the set A.
- $Ch(A)$ is the convex hull of the set A.

For any questions, send me an email via https://notes.sibeliusp.com/contact/.

You can find my notes for other courses on https://notes.sibeliusp.com/.

Sibelius Peng

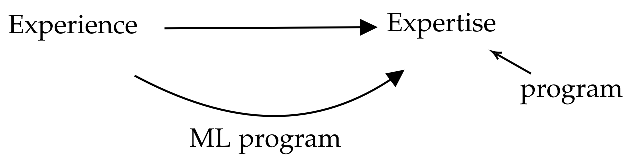
Contents

1 Introduction

Reading

Up to page 41 of the textbook.

What is learning?



Process takes us from experience and leads us to expertise. Expertise would be another program that can do something you need expertise to do. For example, develop a spam filter. The outcome program is the spam filter.

1.1 Learning in Nature

Bait Shyness: It's difficult to poison the rats with the bait food. The rats will find that the shape might be different. If they take a bite and feel sick, they will immediately associate the sickness with the food and then never touch it again. It's a clear example of learning from a single experience.

Spam filters: Inputs are emails which are labeled.

$$(\text{email1, spam}), (\text{email2, not spam}), \dots$$

Then we have to come up with the program which filters the spam. The simplest way is to **memorize** all the emails that are spam. So what's wrong with such a program?

It does not generalize. We want **generalization**. Memorization is not enough. Generalization is sometimes called **inductive reasoning**: take previous cases and try to extend it to something new.

Pigeon Superstition: discovered by Skinner in 1947. He took a collection of pigeons and put them in the cage. Also he put different kinds of toys. Above the cage, there is some mechanism that can spread grains. Something interesting happens. When the birds get hungry, they pick around for worms. Suddenly there's a spread of food. The birds start to learn: maybe the toys the bird is picking at that particular moment had some influence on getting food. So the next time the bird is hungry, the bird is more likely to pick on this toy than others. Then the next time food spreads, it reinforces what the bird did. After several times, the birds are completely devoted to some specific toys.

This is silly generalization. For rats, it's important generalization making them survive.

Garcia 1996, looks at the rats again. He gave the rats the poisoned bait which smelled and looked exactly like the usual food they get. Then the question: does the rat learn the connection between sickness and the poisoned food? Rats fail to associate the bell ringing with the poison effect. Here note that unlike the Pavlov's dog experiment which did repeatedly many times, the rat only has one chance to learn.

The key point here is prior knowledge: the rat already knows the shape and smell of the food through generation. Why have this limitation, why not paying attention to everything? In terms of rats, if they feel sick, every experience/feed is special, then the rat don't know what to associate to. Therefore, the prior knowledge is very important.

If we have little prior knowledge, we need a lot of training. If we have much prior knowledge, maybe we can do without much experiences. ML is living somewhere between these two.

Why do we need Machine Learning?

1. Some tasks that we (animals) can carry out may be too complex to program. E.g., Spam filter, driving, speech recognition.
2. Tasks that require experience with amounts of data that are beyond human capabilities. E.g., ads placement, genetic data.
3. Adaptivity.

1.2 Many types of machine learning

1. Supervised vs. Unsupervised. Supervised: spam filter. Unsupervised: outlier detection, clustering.

   There's also an intermediate scenario called reinforcement learning.
2. Batch vs. Online. Batch: get all training data in advance. Online: need to response as you learn.
3. Cooperative → indifferent → adversarial. Teacher.
4. Passive vs. Active learner.

1.3 Relationships to other fields

AI: two important differences: We are going beyond what human/animals can do, not try to imitate; This area is rigorous, mathematical, nothing like "happens to be".

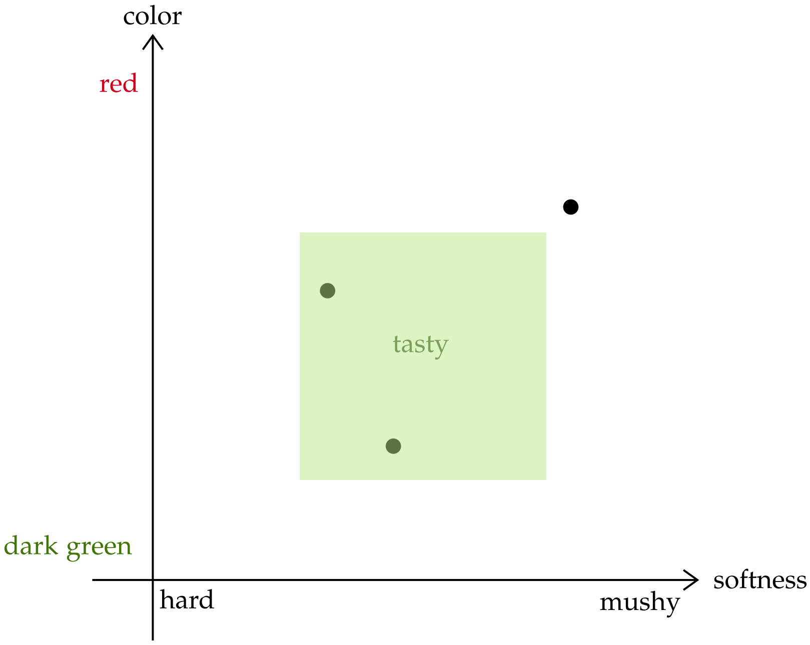
Also we need: Algorithms & complexity, statistics, linear algebra, combinatorics, optimization. However, there's something different with statistics in several ways.

1. algorithmic statistics
2. Distribution free. No clue on how spam is generated.
3. finite samples.

Outline of the course

- Principles: supervised, batch, ...
- Algorithmic paradigms.
- Other types of learning.

1.4 Papaya Tasting



Each papaya corresponds to a coordinate (c, s).

Training data: $(x_1, y_1), \ldots (x_m, y_m) = S$, where $x_i \in \mathbb{R}^2$ and $y_i \in \{T, N\}$

Domain set: $[0, 1]^2$

Label set: $\{T, N\}$

Output: $f : [0, 1]^2 \to \{T, N\}$. Prediction rule.

Assumption about data generation:

1. Training data are randomly generated.
2. Reliability by rectangles. See the picture above.

Measure of success: probability of my predictor f to err on randomly generated papaya.

2

A Gentle Start

2.1 Formal model for learning

In the context of papaya example last time,

Domain set X	$[0,1]^2$
Label set $\mathcal{Y}$	$\{T,N\}$
Training input (sample) $S=((x_1,y_1),\ldots,(x_m,y_m))$	set of already tasted papayas
Learners output $h : X \to Y$	Prediction rule for tasteness
The quality of such h is determined with respect to some data generating distribution and labeling rule	$L_{D,f}(h) = D[\{x : h(x) \neq f(x)\}]$ where L stands for loss, D is the distribution of papaya generated in the world, f is the function to determine the true tastefulness of papaya. So it determines the probability that our hypothesis h fails

The goal of the learner is given S to come up with h with small loss.

2.2 Empirical Risk Minimization

Basic learning strategy: **empirical risk minimization** (ERM) which minimizes the empirical loss.

We define **empirical loss (risk)** over a sample S:

$$L_S(h) = \frac{|\{i : h(x_i) \neq y_i\}|}{|S|}$$

A very simple rule for finding h with small empirical risk (ER)

$$h_S(x) \triangleq \begin{cases} y_i & \text{if } x = x_i \text{ for some } (x_i, y_i) \in S \\ N & \text{otherwise} \end{cases}$$

Then $L_S(h_S) = 0$.

Although the ERM rule seems very natural, without being careful, this approach may fail miserably. It **overfit**s our sample.

To guard against overfitting we introduce some prior knowledge (Inductive Bias).

There exists a good prediction rule that is some axis aligned rectangle. Let H denote a fixed collection of potential (candidate) prediction rules, i.e.,

$$H \subseteq \{f : X \to Y\} = X^Y,$$

and we call it **hypothesis class**. Then we have a revised learning rule: ERM_H - "pick $h \in H$ that minimizes $L_S(h)$", i.e.,

$$\text{ERM}_H(S) \in \underset{h \in H}{\text{argmin}}\{L_S(h)\}$$

Theorem 2.1

Let X be any set, $Y = \{0, 1\}$, and let H be a finite set of functions from X to Y. Assume:

1. The training sample S is generated by some probability distribution D over X and labeled some $f \in H$, and elements of S are picked i.i.d.[a]
2. **Realizability assumption**: $\exists h \in H$ such that $L_{D,f}(h) = 0$

Then ERM_H is guaranteed to come up with an h that has small true loss, given sufficiently large sample S.

[a] identically and independently distributed

Remark:
This is quite different from hypothesis testing. Unlike hypothesis testing, here we have assumptions after seeing the data. We are developing theories based on the data, and here H is a finite set.

Proof:
Confusing samples are those on which ERM_H may come up with a bad h.

Fix some success parameter $\varepsilon > 0$, and the set of confusing S's is

$$\{S : L_{D,f}(h_S) > \varepsilon\}$$

We wish to upper bound the probability of getting such a bad sample.

$$D^m[\{S|_X : L_{D,F}(h_S) > \varepsilon\}]$$

where $S|_X = x_1, \ldots, x_m$.

Consider $H_B = \{h \in H : L_{D,f}(h) > \varepsilon\}$ which is the set we want to avoid.

The misleading samples is the set of samples that may lead to an out come in H_B, formally:

$$M = \{S|_X : \exists h \in H_B \text{such that } L_S(h) = 0\}$$

We claim that $\{S|_X : L_{D,f}(h_S) > \varepsilon\} \subseteq M$. The former set is the cases that we select bad hypothesis, and M is the cases there exist bad hypothesis. So it is a subset. We might not have selected a bad hypothesis from M, then we cannot put an equal between these two sets. We want to upper bound a probability of the set we defined. Therefore, it suffices to upper bound $D^m(M)$.

$$D^m(M) = D^m\left[\bigcup_{h \in H_B} : \{S|_X : L_S(h) = 0\}\right]$$

Now we need two basic probability rules:

1. The union bound: For any two events A, B and any probability distribution P, $P(A \cup B) \leq P(A) + P(B)$.
2. If A and B are independent events then $P(A \cap B) = P(A) \cdot P(B)$.

For any fixed $h \in H_B$. Let us upper bound $D^m[\{S|_X : L_S(h) = 0\}]$. For a single one, the probability of h is doing wrong on X is at least ε, then

$$\begin{aligned} D^m[\{S|_X : L_S(h) = 0\}] &= D^m[\{S_X : h(x_1) = f(x_1) \wedge \cdots \wedge h(x_m) = f(x_m)\}] \\ &\leq (1-\varepsilon)^m \end{aligned}$$

Then we conclude

$$D^m[\text{Bad } S] \leq D^m(M) = D^m[\cup_{h \in H_B}(L_S(h) = 0)] \leq |H_B| \cdot (1-\varepsilon)^m \leq |H| \cdot (1-\varepsilon)^m$$

Then ERM_H has small probability of failure as $m \to \infty$. □

Note that here we call it paradigm not algorithm since it doesn't tell you which H to pick.

This time we use a different notation:

$$\Pr_{S \sim D^m}[L_{D,f}(\text{ERM}_H(S)) > \varepsilon] \leq |H| \cdot (1-\varepsilon)^m$$

for every $\varepsilon \geq 0$ and for every m, where

$$L_{D,f}(h) = \Pr_{x \sim D}[h(x) \neq f(x)].$$

Trust that for every $1 > \varepsilon > 0$, m

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

Thus the probability of making an error is going down exponentially fast in the sample size.

Also recall the proof idea:

- Step 1: For any given $h : X \to \{0, 1\}$, $\Pr_{S \sim D^m}[L_S(h) = 0]$ is small and getting smaller with m, provided that $L_{D,f}(h) > \varepsilon$. We are looking at samples that make h look good in spite of h being bad.
- Step 2: Take union over all $h \in H$.

3

A Formal Learning Model

3.1 A formal notion of learnability

PAC learnability

We say that a class of predictors H is **PAC learnable** (Probably Approximately Correct) if there exists a function $m_H : (0,1) \times (0,1) \to \mathbb{N}$ such that there exists a learner

$$A : \bigcup_{m=1}^{\infty} (X \times \{0,1\})^m \to \{f | f : X \to \{0,1\}\}$$

that for every D probability distribution over X and every $f \in H$ and every $\varepsilon, \delta > 0$

$$\Pr_{S \sim D^m, f} [L_{D,f}(A(S)) > \varepsilon] < \delta$$

for every $m \geq m_H(\varepsilon, \delta)$.

Here we can think of ε as an accuracy parameter and δ as confidence parameter.

If we rephrase Theorem 2.1, we get

Theorem

Every finite H is PAC learnable with $m_H(\varepsilon, \delta) \leq \frac{\ln |H| + \ln(1/\delta)}{\varepsilon}$. Furthermore, any ERM$_H$ learner will be successful.

Last time we have showed

$$\Pr_{S \sim D^m, f} [L_{D,f}(A(s)) > \varepsilon] \leq |H| \cdot e^{-\varepsilon m} \leq \delta$$

Then take ln,

$$\ln |H| - \varepsilon m \leq \ln(\delta)$$

Then we get results as desired.

Strength of the PAC definition is that we can guarantee the number of needed examples (for training) regardless of the data distribution D and of which $f \in H$ is used for labeling. We call this is a "**Distribution free guarantee**".

Weakness: If only works if the labeling rule f comes from H.

Relaxation The data is generated by some probability distribution D over $X \times Y$.

We still wish to output a labeling rule $h : X \to Y$.

Assume $Y = \{0, 1\}$, we claim the best predictor h should be

$$h^*(x) = \begin{cases} 1 & \text{if } D((x,1)|x) \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

which is called the Bayes rule. The problem is we *do not* know D. We only see a sample (generated by D).

Note that $L_D(h^*)$ in some cases may be high. If it's a coin flip, and the data generating process is completely random, then this rate will be half.

3.2 A More General Learning Model

Now redefine successful learning to have only a relative error guarantee.

Agnostic PAC learnability

A class of predictors H is **agnostic PAC learnable** if there exist some function $m_H(\varepsilon, \delta) : (0,1) \times (0,1) \to \mathbb{N}$, and a learner A (taking samples, outputting predictors) such that for every D over $X \times Y$ and every $\varepsilon, \delta > 0$

$$\Pr_{S \sim D^m}[L_D(A(S)) > \min_{h \in H}(L_D(h)) + \varepsilon] < \delta$$

whenever $m \geq m_H(\varepsilon, \delta)$.

Weaker notion of learner's success defined relative to some "benchmark" class of functions H. h is **ε-accurate** with respect to D, H if $L_D(h) \leq \min_{h' \in H}(L_D(h')) + \varepsilon$.

Some other learning tasks

1. **Multiclass prediction.**

   The set of labels Y could be larger than just two elements. For example, {Politics, Sports,Entertainment, Finance, . . .}

2. **Real valued prediction (Regression).**

   The set of labels is the real line. For example, predict tomorrow's max temperature.

3.3 More general setup for learning

- Domain set Z
- Set of models M
- Loss of some model: on a given instance z: $\ell(h, z)$

The data is generated by some unknown distribution over Z and we aim to find the best model for that distribution.

$$L_D(h) = \mathbb{E}_{z \sim D}\ell(h, z)$$

So far,

- $Z = X \times \{0, 1\}$

- M: functions from X to $\{0,1\}$
- $\ell(h, \underbrace{(x,y)}_{z}) = \begin{cases} 1 & \text{if } h(x) \neq y \\ 0 & \text{if } h(x) = y \end{cases}$

$L_D(h)$ coincides with our previous definition $L_D(h) = \Pr_{(x,y)\in D}(h(x) \neq y)$

Let's consider our previous examples with this new setting.

1. Binary label prediction.

   $$Z = X \times \{0,1\}, \quad \ell(h,(x,y)) = \begin{cases} 1 & \text{if } h(x) \neq y \\ 0 & \text{if } h(x) = y \end{cases}$$

2. Multiclass prediction.

   $Z = X \times Y$ where Y is the set of topics from the previous example.

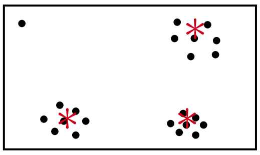
   $$\ell(h,(x,y)) = \begin{cases} 1 & \text{if } h(x) \neq y \\ 0 & \text{if } h(x) = y \end{cases}$$

   These are called 0-1 loss for the obvious reason.

3. Regression (Predicting temperature)

   $Z = X \times \mathbb{R}, \quad \ell(h,(x,t)) = (h(x) - t)^2$ which is called square loss

4. Representing data by k codewords.



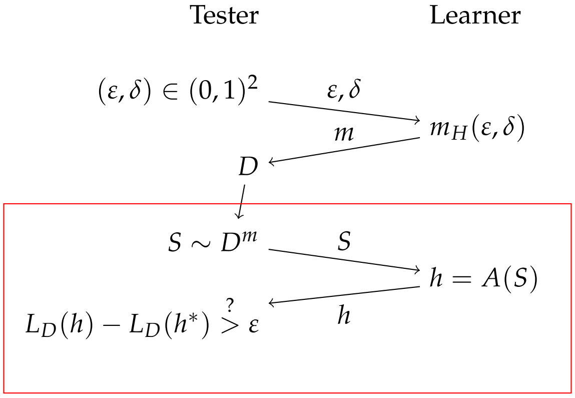
distributions of audio signals

$Z = \mathbb{R}^d \qquad M =$ vectors of k members of $\mathbb{R}^d \qquad h = (c_1 \ldots c_k)$

$\ell((c_1 \ldots c_k), z) = \min_{1\leq i\leq k} \|c_i - z\|^2$

3.4 Agnostic PAC-Learning as a game

Fix the domain set X and hypothesis class H.


Tester
Learner
$(\varepsilon,\delta) \in (0,1)^2$
ε, δ
$m_H(\varepsilon,\delta)$
m
D
$S \sim D^m$
S
$h = A(S)$
h
$L_D(h) - L_D(h^*) \overset{?}{>} \varepsilon$

How often the learner fails? more than δ?

where $h^* = \operatorname{argmin}_{h\in H} L_D(h)$.

4

Learning via Uniform Convergence

epsilon-representative sample

A sample $S = (z_1 \dots z_m)$ (or $S = (x_1, y_1) \dots (x_m, y_m)$) is ε**-representative** of a class H with respect to a distribution D if

$$\forall h \in H, \quad |L_S(h) - L_D(h)| \leq \varepsilon,$$

where $L_S(h) = \frac{1}{m} \sum_{z \in S} \ell(h, z)$, the Empirical Risk of h.

Note that if S is indeed representative of H with respect to D, then ERM_H is a good learning strategy.

Claim If S is ε-representative of H with respect to D then for any ERM_H function h_S

$$L_D(h_S) \leq \min_{h \in H}(L_D(h)) + 2\varepsilon$$

Proof:

By definition, since S is ε-representative and $h_S \in H$, then $L_D(h_S) \leq L_S(h_S) + \varepsilon$. Since h_S is ERM_H, then $L_S(h_S) \leq \min_{h \in H}[L_S(h)]$. Again since S is ε-representative, we have

$$L_D(h_S) \leq L_S(h_S) + \varepsilon \leq \min_{h \in H}[L_S(h) + \varepsilon] \leq \min_{h \in H}[L_D(h)] + \varepsilon + \varepsilon$$

□

4.1 Finite Classes Are Agnostic PAC Learnable

Next step Show that if a large enough S is picked at random by D then with hight probability, such S will be ε-representative of H with respect to D.

Suggestion Prove an upper bound for sample complexity of a specific algorithm, namely ERM: $A(S) = \operatorname{argmin}_{h \in H} L_D(h)$.

We then present the above claim as a lemma.

Lemma 4.1

If S is ε-rep, then $L_D\left(A^{\text{ERM}}(S)\right) - L_D(h^*) \leq 2\varepsilon$.

sample complexity of uniform convergence

(w.r.t. H) $m_H^{UC}(\varepsilon, \delta)$: the minimum number m such that for every distribution D, if we pick $S \sim D^m$, then with probability at least $1-\delta$, S is ε-representative.

If we have $m_H^{UC}(\varepsilon, \delta)$ samples, then with high probability our sample S is ε-representative $\xrightarrow{\text{lemma}}$ ERM will work → it will be agnostic PAC-learnable.

Corollary 4.2

$$m_H(\varepsilon, \delta) \leq m_H^{UC}(\varepsilon/2, \delta)$$

New goal find upper-bound for $m_H^{UC}(\varepsilon, \delta)$ in the case $|H| < \infty$

Strategy:

- Step 1: for a single hypothesis $h \in H$, bound the number of samples to make sure that $L_D(h) \approx L_S(h)$ with "high probability".
- Step 2: use union bound to bound the probability that "any" of them fails.

Hoeffding's inequality

Assume $\theta_1, \theta_2, \ldots, \theta_m$ are iid random variables with mean μ that take values in $[a, b]$, then

$$\Pr\left[\left|\mu - \frac{1}{m}\sum \theta_i\right| > \varepsilon\right] < 2\exp\left(\frac{-2m\varepsilon^2}{(b-a)^2}\right)$$

Fix some $h \in H$.

$$\Pr[|L_D(h) - L_S(h)| \geq \varepsilon] = \Pr\left[\left|\mathop{\mathbb{E}}_{z\sim D} \ell(h,z) - \frac{1}{m}\sum_{z\in S}\ell(h,z)\right| \geq \varepsilon\right] \leq 2e^{\frac{-m\varepsilon^2}{(1-0)^2}} = 2e^{-m\varepsilon^2}$$

Proof of the main result:

$$\begin{aligned}
\Pr[S \text{ is not } \varepsilon\text{-representative w.r.t. } H] &= \Pr\left[\exists h \in H, \text{ s.t. } |L_D(h) - L_S(h)| > \varepsilon\right] \\
&\leq \sum_{h\in H} \Pr[|L_D(h) - L_S(h)| > \varepsilon] \qquad \text{by union bound} \\
&\leq \sum_{h\in H} 2e^{-m\varepsilon^2} \quad \text{by Hoeffding's ineq} \\
&= |H| \cdot 2e^{-m\varepsilon^2}
\end{aligned}$$

Then what can we say about $m_H^{UC}(\varepsilon, \delta)$?

$$|H| \cdot 2e^{-m\varepsilon^2} < \delta \implies m_H^{UC} > \frac{\ln(2|H|/\delta)}{2\varepsilon^2}$$

Corollary 4.3

$$m_H(\varepsilon, \delta) \leq \frac{2\ln(2|H|/\delta)}{\varepsilon^2}$$

4.2 PAC-learnable infinite class example

Do we have an infinite class that is PAC-learnable?

Let H^{thr} be the class of all thresholds on $[0,1]$, that's

$$H^{thr} = \left\{ h_r : h_r(x) = \begin{cases} 0 & x \le r \\ 1 & x > r \end{cases}, \quad r \in [0,1] \right\}$$

Practical Approach: Discretize

$$H^{thr}_{\alpha} = \left\{ h_r : h_r(x) = \begin{cases} 0 & x \le r \\ 1 & x > r \end{cases}, \quad r \in \left\{ 0, \frac{1}{\alpha}, \frac{2}{\alpha}, \dots, \frac{\alpha - 1}{\alpha}, 1 \right\} \right\}$$

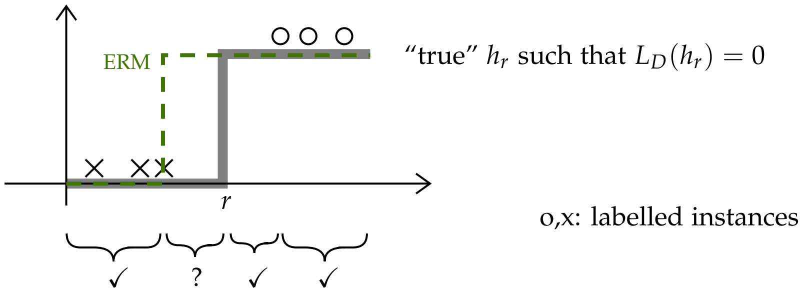
$|H^{thr}_{\alpha}| = \alpha + 1$

In theory, H^{thr}_{α} may not be a good approximation of H^{thr}

$$\min_{h \in H^{thr}} L_D(h) \ll \min_{h \in H^{thr}_{\alpha}} L_D(h)$$

for some specific distribution. Consider the case: $D[\{(\frac{3}{2\alpha}, 0)\}] = D[\{(\frac{7}{2\alpha}, 0)\}] = 0.5$

Is H^{thr} PAC-learnable (realizable case)?



Let A be the ERM algorithm that resolves ties in favor of smaller thresholds.

Let q_ε be the smallest number in $[0,r]$ that satisfies $D_{|_x}\{x \in [q_\varepsilon, r]\} \le \varepsilon$.

Claim If sample $S_{|X}$ contains a point in $[q_\varepsilon, r]$ then $L_D(A(S)) \le \varepsilon$.

Proof:
Let t be such point in S. Then

$$\begin{aligned} L_D(A(S)) &\le D_{|X}\{x : x \in (t, r]\} \\ &\le D_{|X}\{x : x \in (q_\varepsilon, r]\} \quad \text{because } t \ge q_\varepsilon \\ &\le \varepsilon \end{aligned}$$

□

Proof of PAC-learnability of H^{thr}:

$$\begin{aligned} \Pr[L_D(A(S)) > \varepsilon] &\le D^m_{|X}\{s : \nexists x \in S|_X \text{ s.t. } x \in [q_\varepsilon, r]\} \\ &\overset{iid}{\le} \left(D_{|X}\{x : x \notin [q_\varepsilon, r]\} \right)^m \\ &\le (1 - \varepsilon)^m \le e^{-\varepsilon m} \end{aligned}$$

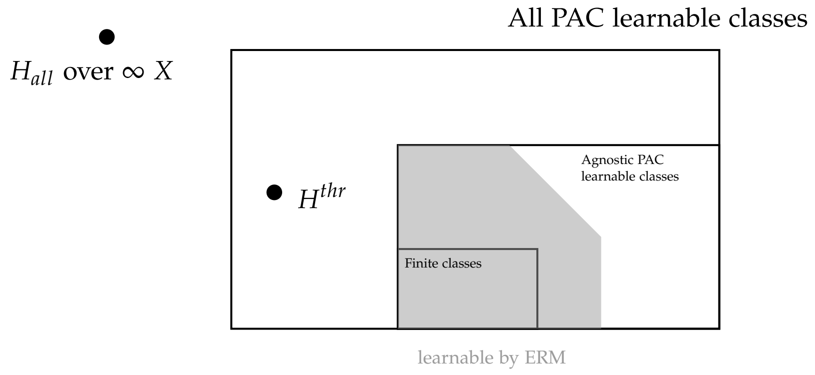
□

Remark:

So H^{thr} is PAC-Learnable with $m_{H^{thr}}(\varepsilon,\delta) \leq \frac{\ln(1/\delta)}{\varepsilon}$.

We will not prove here, but H^{thr} is agnostic PAC-Learnable.

4.3 Summary


All PAC learnable classes
H_{all} over ∞ X
H^{thr}
Agnostic PAC learnable classes
Finite classes
learnable by ERM

Agnostic PAC learnable is stronger than PAC learnable because within agnostic, we require for every distribution, you will be able to get close to the best classifier with respect to that distribution. In PAC, we only require this will hold for the distributions for which one of the element of H is a perfect classifier.

Finally, learnable by ERM, agnostic learnable, PAC learnable are going to be the same family of classes.

Example: non-ERM learnable class

Let $X = \mathbb{R}$. Let $H_{finite} = \{h_A : A \text{ is a finite subset of } \mathbb{R}\}$ where $h_A = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$

Let $H = H_{finite} \cup \{h_{one}\}$

Claim H_{finite} is not learnable (PAC) by ERM.

Proof:

Let P be the uniform distribution over $[0,1]$. Pick as a labeling rule the all-1 function. We wish to show ERM may fail on this challenge. Pick any sample size m, and $S \sim P^m$, then

$$S = ((x_1, 1), \ldots, (x_m, 1))$$

An ERM algorithm may now pick h_A for $A = \{x_1, \ldots, x_m\}$, then $L_S(h_A) = 0$, but $L_P(h_A) = 1$. h_A fails on every test point $x \notin A$. □

However $H_{thr} = \{h_x : x \in \mathbb{R}\}$ where $h_x(y) = \begin{cases} 1 & y \leq x \\ 0 & \text{otherwise} \end{cases}$

Under this setting, ERM is a successful PAC learner.

5

The Bias-Complexity Tradeoff

Tradeoff between "approximation error" and "estimation error".

$$\varepsilon_{app} = \min_{h \in H} L_D(h)$$
$$\varepsilon_{est} = L_D(h_S) - \varepsilon_{app}$$

NFL shows for large class, estimation error is large; for small class, approximation error is large.

5.1 The No-Free-Lunch Theorem

Theorem 5.1: No-Free-Lunch

Let X be a domain of size n i.e., $|X| = n$). Let H_n^{all} be the set of all possible labelings, i.e., $H_n^{all} = \{h : X \to \{0,1\}\}$ ($|H_n^{all}| = 2^n$). NFL proves that

$$m_{H_n^{all}}\left(\frac{1}{8}, \frac{1}{7}\right) \geq \frac{n}{2}$$

Proof:
Either see textbook or Lecture 8. □

Corollary 5.2

$$m_{H_\infty^{all}}\left(\frac{1}{8}, \frac{1}{7}\right) \geq \infty$$

Therefore, the class of labeling functions over an infinite domain is not PAC-Learnable.

Intuition Assume that $|S| = \frac{n}{2}$. Assume $D_{|n}$ is uniform over X. Then the learner can find out about the labels of points in $S_{|X}$. But for the points are not in the sample, it cannot do better than random guess. (Because every labeling is possible on them) Therefore, it fails on at least $\frac{1}{2}$ of the points in $X \setminus S_{|X}$ (in expectation). It will fail on 25% of the points (expected)

$$\mathop{\mathbb{E}}_{S \sim D^m} L_D(A(S)) \geq \frac{1}{4}$$

6

The VC-Dimension

6.1 Infinite-Size Classes Can Be Learnable

See section 4.2

6.2 The VC-Dimension

shatter

Let H be a class of $\{0,1\}$ functions over some domain X, and let $A \subseteq X$. H **shatters** A if for every $g : A \to \{0,1\}$, $\exists h \in H$ such that for any $x \in A$, $h(x) = g(x)$.

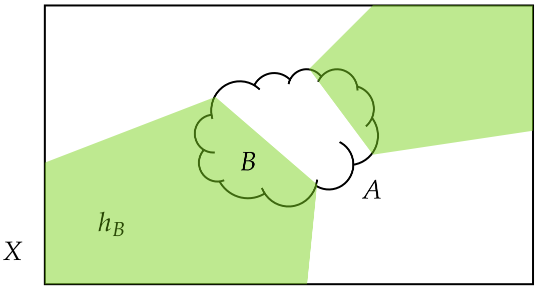
Example:

Let $X = \mathbb{R}$. Consider H_{thr}.

$A = \{7, 10\}$. We have 4 possible g's over A. However, we cannot get $g(7) = 0, g(10) = 1$ from H_{thr}. Thus H does not shatter A.

Note that there is equivalence between functions from X to $\{0,1\}$ and subsets of X. Given $h : X \to \{0,1\}$, define $A_h = \{x \in X : h(x) = 1\}$. Or going backwards, given $A \subseteq X$, define $h_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$

In terms of subsets H is a collection of subsets of X: A is shattered by H if for every $B \subseteq A$, $\exists h_B \in H$ such that $B = h_B \cap A$.



Example:

Let $X = \mathbb{R}^2$. Let $H = \{B_{(x,r)} : x \in \mathbb{R}^2, r \in \mathbb{R}^+\}$ where $B_{(x,r)} = \{y : \|y - x\| \leq r\}$

Any A of size 2 is shattered. Any set A consisting of 3 non-colinear points is shattered by H. Any set A consisting of 3 colinear points is not shattered by H.

VC-dimension

$$\mathrm{VCdim}\,(H) := \max\{|A| : A \text{ is shattered by } H\}$$

and it is ∞ if no maximal such A exists.

6.3 Examples

1. $\mathrm{VCdim}\,(H_{thr}) = 1$

   Proof:
   We have showed any set A of size ≥ 2 is not shattered by H_{thr}. So $\mathrm{VCdim}\,(H_{thr}) \leq 1$. The set $\{1\}$ is shattered by H_{thr}, therefore $\mathrm{VCdim}\,(H_{thr}) \geq 1$. Therefore $\mathrm{VCdim}\,(H_{thr}) = 1$. □

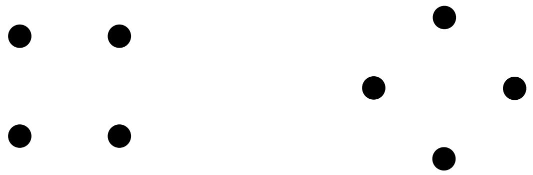
2. $\mathrm{VCdim}\left(H_{finite}\right) = \infty$

   Proof:
   Every finite A is shattered by H_{finite} because $\forall B \subseteq A$, $h_B \in H_{finite}$ and $h_B \cap A = B$. □

3. Let $X = \mathbb{R}$, $H_{interval} = \{h_{a,b} : a, b \in \mathbb{R}, a < b\}$ where $h_{a,b}(x) = \begin{cases} 1 & x \in [a,b] \\ 0 & \text{otherwise} \end{cases}$

   Every $A \subseteq R$ of size 2 is shattered by $H_{interval}$, then $\mathrm{VCdim}\,(H_{interval}) \geq 2$. Any ≥ 3 point set is not shattered so $\mathrm{VCdim}\,(H_{interval}) = 2$.

4. Axis Aligned Rectangles. $H_{rect} = \{h_{(a,b,c,d)} : a, b, c, d \in \mathbb{R}\}$. $X = \mathbb{R}^2$. See the precise definition in 6.3.3 of textbook.



not shattered

shattered

Claim $\mathrm{VCdim}\,(H_{rec}) \leq 4$.

Proof:
Given any set $A \subseteq \mathbb{R}^2$, let x_ℓ^A be the leftmost point of A, x_r^A be the rightmost point of A, x_b^A be the lowest point of A, x_t^A be the highest point of A. Every rectangle $h \in H_{rec}$ that captures $\{x_\ell^A, x_r^A, x_b^A, x_t^A\}$ captures all of A. If $|A| \geq 5$, A contains some $x^* \notin \{x_\ell^A, x_r^A, x_b^A, x_t^A\}$, we cannot get $B = \{x_\ell^A, x_r^A, x_b^A, x_t^A\}$. □

6.4 Some basic properties of VCdim

1. $\mathrm{VCdim}\,(H) \leq \log_2(|H|)$ or $|H| \geq 2^{\mathrm{VCdim}(H)}$

   It takes $2^{|A|}$ h's to shatter a set A.

   This ineq can be not tight. For example, $|H_{thr}| = \infty \gg 2^1$

2. If $H_1 \subseteq H_2$, then $\mathrm{VCdim}\,(H_1) \leq \mathrm{VCdim}\,(H_2)$

Lemma 6.1

If H has infinite VCdim then H is not PAC learnable.

Proof:

Follows from the NFL.

For the sake of contradiction, assume such H is PAC learnable, then for some $m_H : (0,1)^2 \to \mathbb{N}$, some A, for every distribution P over X and every $f \in H$ and every $\varepsilon, \delta > 0$,

$$\Pr_{S \sim P^m, f}[L_{P,f}(A(S)) > \varepsilon] < \delta$$

whenever $m \geq m_H(\varepsilon, \delta)$.

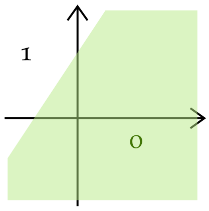
Consider $m_H(\frac{1}{8}, \frac{1}{8})$. By VCdim $(H) = \infty$, $\exists W \subseteq X$ such that H shatters W and $|W| > 2m_H(\frac{1}{8}, \frac{1}{8})$. H induces every possible function from W to $\{0,1\}$. But by the NFL theorem, in such case,

$$m_H(1/8, 1/8) \geq \frac{|W|}{2} > m_H(1/8, 1/8)$$

contradiction. □

6.5 Another example of VCdim

A practical class H: the class of linear predictors over $\mathbb{R}^n$



The class HS^n. Let $X = \mathbb{R}^n$. Given some vector $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Let

$$h_{w,b}(x) = \text{sign}(\langle w, x\rangle + b) = \begin{cases} +1 & \sum_i^n w_i x_i + b \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Then

$$HS^n = \{h_{w,b} : w \in \mathbb{R}^n, b \in \mathbb{R}\}$$

Let us restrict our attention to "homogeneous" linear classifiers $\{h_{w,0} : w \in \mathbb{R}^n\}$. Then what is VCdim $\left(HS_0^n\right)$? We claim that VCdim $\left(HS_0^n\right) = n$ and VCdim $(H^n) = n + 1$ for every n.

Proof:

- VCdim $\left(HS_0^n\right) \geq n$.

Consider the points $\{e_1, \ldots, e_n\}$. Pick $B \subseteq \{e_1, \ldots, e_n\}$. Let $h_B = (w_1, \ldots, w_n)$ where

$$w_i = \begin{cases} +1 & \text{if } e_i \in B \\ -1 & \text{if } e_i \notin B \end{cases}$$

Then $\langle w, e_i\rangle = w_i$. Thus for all $e_i \in \{e_1, \ldots, e_n\}$, then $h_B(e_i) = 1$ if $e_i \in B$ and -1 otherwise.

- VCdim $\left(HS_0^n\right) \leq n$.

We need to prove given any set $A = (x_1, \ldots, x_{n+1})$ in R^n, A cannot be shattered by HS_0^n.

Since the dimension of $\mathbb{R}^n$ is n, for any $n+1$ vectors $x_1, \ldots, x_{n+1}$, there exists coefficients $a_1, \ldots, a_{n+1}$ not all of them zero, such that

$$\sum_{i=1}^{n+1} a_i x_i = 0$$

Let $P \subseteq \{1, \ldots, n+1\}$ be the set of coordinates for which $a_i > 0$, and $N \subseteq \{1, \ldots, n+1\}$ the set of coordinates for which $a_i < 0$. Then

$$\sum_{i=1}^{n+1} a_i x_i = 0 \implies \sum_{i\in P} a_i x_i = \sum_{j\in N} |a_j| x_j$$

Let $B = \{x_i : i \in P\}$. Then $h_B(x_i) = +1$ if $x_i \in B$ and -1 otherwise, then

$$h_B\left(\sum_{i\in P} a_i x_i\right) = \sum_{i\in P} a_i h_B(x_i) > 0$$

and

$$h_B\left(\sum_{j\in N} |a_j| x_j\right) = \sum_{j\in N} |a_j| h_B(x_j) < 0$$

which is a contradiction. □

Lemma 6.2: Radon's theorem

For every n, every $x_1, \ldots, x_{n+2} \in \mathbb{R}^n$, $\exists B \subseteq \{x_1, \ldots, x_{n+2}\}$,

$$Ch(B) \cap Ch(\{x_1, \ldots, x_{n+2}\} \setminus B) \neq \varnothing$$

where Ch is the convex hull.

6.6 The Fundamental Theorem of PAC learning

Theorem 6.3: The Fundamental Theorem of Statistical Learning

For every domain X and every class H of functions from X to $\{0, 1\}$. TFAE:

1. H has the uniform convergence property.
2. ERM is a successful agnostic PAC learner for H.
3. H is agnostic PAC learnable.
4. ERM is a successful PAC learner for H.
5. H is PAC learnable.
6. $\text{VCdim}(H)$ is finite.

Proof:
See 6.5 of textbook or Lecture 8.

Hard part is 6 → 1. □

Quantitative statement of the fundamental theorem

There exists constants c_1, c_2 such that for any class H of finite VCdim, denoting $\text{VCdim}(H) = d$ we get: $\forall \varepsilon, \delta > 0$

1. $c_1 \frac{d+\ln(1/\delta)}{\varepsilon} \leq m_H^{PAC}(\varepsilon, \delta) \leq c_2 \frac{d\ln(1/\varepsilon)+\ln(1/\delta)}{\varepsilon}$
2. $c_1 \frac{d+\ln(1/\delta)}{\varepsilon^2} \leq m_H^{AgPAC}(\varepsilon, \delta) \leq c_2 \frac{d+\ln(1/\delta)}{\varepsilon^2}$
3. $c_1 \frac{d+\ln(1/\delta)}{\varepsilon^2} \leq m_H^{UC}(\varepsilon, \delta) \leq c_2 \frac{d+\ln(1/\delta)}{\varepsilon^2}$

   $c_1 \frac{d+\ln(1/\delta)}{\varepsilon^2} \leq m_H^{ERM}(\varepsilon, \delta) \leq c_2 \frac{d+\ln(1/\delta)}{\varepsilon^2}$

Proof:
See Lecture 10. □

Why do we need so many examples for learning in the agnostic case?

Consider a situation: given two points x, y in domain. Define a probability distribution by picking each of x, y with equal probability $1/2$, and $P(1|x) = P(0|y) = \frac{1}{2} + \varepsilon$, $P(0|x) = P(1|y) = \frac{1}{2} - \varepsilon$.

Let $H =$ all $0 - 1$ functions over $\{0, 1\}$, then best predictor in this case is $(1, 0)$. This is also the Bayes predictor. In order to detect if a coin has bias $\frac{1}{2} + \varepsilon$ towards H or $\frac{1}{2} + \varepsilon$ towards T, one needs $\sim \frac{1}{\varepsilon^2}$ coin tosses (as ε gets small)

6.7 The Sauer's Lemma

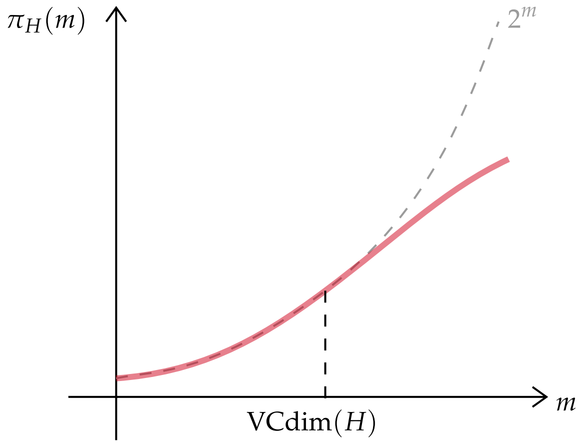
shatter function

Given a class of binary-valued functions, H, over some domain set X, define the **shatter function**, $\pi_H : \mathbb{N} \to \mathbb{N}$, for any $m \in \mathbb{N}$

$$\pi_H(m) = \max_{A \subseteq X : |A| = m} \{|\{h \cap A : h \in H\}|\}$$

Simple observations:

1. For any H and any m, $\pi_H(m) \leq 2^m$
2. If H shatters a set of size m, then $\pi_H(m) = 2^m$.
3. If $\text{VCdim}(H) < m$, then $\pi_H(m) < 2^m$.



Lemma 6.4: Sauer

For every class H and every d if $\text{VCdim}(H) = d$ then, for all m

$$\pi_H(m) \leq \sum_{i=0}^{d} \binom{m}{i} \leq m^d$$

Proof:
See textbook or Lecture 11. □

Corollary 6.5

The number of partitions realizable by a linear half-space is $\leq m^{n+1}$ as $\text{VCdim}(HS^n) = n + 1$.

Consequences of the Sauer's Lemma

1. The vast majority of partitions of a subset of $\mathbb{R}^n$ cannot be realized by a linear separator: $m^{n+1} \ll 2^m$.
2. Upper bounding $m_H^{PAC}(\varepsilon, \delta)$.
3. Upper bounding the VCdim of $H_1 \cup H_2$ in terms of VCdim (H_1) of VCdim (H_2).

   Assume $H_1 \cup H_2$ shatters some A of size m, $|\{h \cap A : h \in H_1 \text{ or } h \in H_2\}| \geq 2^m$. On the other hand,

$$|\{h \cap A : h \in H_1 \text{ or } h \in H_2\}| \leq |\{h \cap A : h \in H_1\}| + |\{h \cap A : h \in H_2\}|$$
$$\overset{Sauer}{\leq} m^{\text{VCdim}(H_1)} + m^{\text{VCdim}(H_2)} \qquad m \text{ cannot be too big}$$

Upper bounding the sample complexity $m_H^{UC}(\varepsilon, \delta)$. We wish to show that for sufficiently large m depending only on ε, δ, VCdim (H), $\Pr[\exists h \in H \ \ |L_S(h) - L_P(h)| > \varepsilon] < \delta$, samples $S \sim P^m$. If we fix h, we know $\Pr_{S \sim P^m}[L_S(h) - L_P(h) > \varepsilon] < 2e^{-m\varepsilon^2}$ by Hoeffding ineq. Using the union bound over all $h \in H$, we got $|H| \cdot 2e^{-m\varepsilon^2}$.

First Idea: we only care about behaviours of h on the sample S. The number of $\{h \cap S : h \in H\}$ is always finite, bounded by Sauer's lemma. We get $\frac{2m^d}{e^{2m\varepsilon^2}} \to 0$ as $m \to \infty$. Illegal move: choose functions/class based on the sample.

The double sample argument. Some good resources:

- https://cse.buffalo.edu/~hungngo/classes/2011/Fall-694/lectures/vc-theorem.pdf
- https://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0218.pdf

I may evaluate the error of any function in H_S by using a fresh (independent of S) sample T,

$$\Pr_{S \sim P^m} \underbrace{[\exists h \in H \ \ |L_S(h) - L_P(h)| > \varepsilon]}_{B_S} \leq 2 \Pr_{\substack{S \sim P^m \\ T \sim P^m}} \underbrace{[\exists h \in H \ \ |L_S(h) - L_T(h)| > \varepsilon]}_{B_{S,T}}$$

Pick any subset $A \subseteq X$ of size $2m$. We bound $\Pr_{S,T \sim P^m}[B_{S,T} | S, T \subseteq A]$. For this evaluation, it suffices to consider H_A - the set of possible behaviours of H on A. Then

$$\Pr_{S,T \sim P^m} \leq (2m)^d \cdot 2e^{-2m\varepsilon^2}$$

Big conclusion: A class H is learnable if and only it has VCdim.

How useful/relevant is such a paradigm?

Argument 1: Sometimes a fixed, small VCdim, H is all we really care about.

However, there are situations in which our goal is to minimize our prediction error regardless of any class H. Next step: Extend learning beyond finite VC classes.

7

Nonuniform Learnability

Recall our definition of Agnostic PAC learnability.

non-uniformly learnable

A hypothesis class H is **non-uniformly learnable** if there exists a learning algorithm A, a function $m_H(\varepsilon, \delta, h)$ $(m_H : (0,1)^2 \times H \to \mathbb{N})$, such that for every P over X, every $\varepsilon, \delta > 0$, every $h \in H$, if $m \geq m_H(\varepsilon, \delta, h)$, then for $S \sim P^m$, it is with probability $> 1 - \delta$

$$L_P(A(S)) \leq L_P(h) + \varepsilon$$

7.1 Characterizing Nonuniform Learnability

Theorem 7.1

A class H is non-uniformly learnable if and only if there are classes $\{H_n\}_{n \in \mathbb{N}}$ each having finite VCdim, s.t.,

$$H = \bigcup_{n=1}^{\infty} H_n$$

Two obvious preliminary questions for Theorem 7.1:

1. Is there a class that is NUL, but not PAC learnable?
2. Is there a class that is not NUL?

Example:

Let $X = \mathbb{R}$. Let H be the class of all threshold polynomial functions. For each polynomial $p = a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d$. Let $h_p(x) = \text{sign}(p(x))$.

Let H be the class of all such polynomial predictors h_p.

Claim 1 $H = \bigcup_{n=0}^{\infty} H_n$ where $H_n = \{h_p : p \text{ is a polynomial of degree } \leq n\}$.

H_1 gives just threshold functions. H_2 gives intervals and complement of intervals.

Claim 2 Each H_n has a finite VCdim and hence is A-PAC learnable.

Conclusion: H is NUL. Because it has infinite VCdim, H is not PAC learnable.

Example:

Let $X = \mathbb{R}$ and for each k let $H_k = \{h : \mathbb{R} \to \{0,1\} : h(x) = 1 \text{ for at most } k \text{ many } x\text{'s}\}$. For example, $H_1 =$ The class of singletons. Then it can be proved that VCdim $\left(\cup_{k=1}^{\infty} H_k\right) = \infty$ and for every k, VCdim $(H_k) = k$. Thus $H = \cup_{k=1}^{\infty} H_k$ is NUL, but not PAC learnable.

Lemma 7.2

For every infinite set X, the class of all $\{0,1\}$ valued functions over X cannot be covered by any $\bigcup_n H_n$ where each H_n has a finite VCdim.

So we didn't trivialize non-uniform learnability.

Lemma 7.3

If H is NUL, then there exist classes H_n such that

1. $H = \bigcup_{n=1}^{\infty} H_n$,
2. Each H_n has a finite VCdim (and therefore, A-PAC).

Proof:

Given a NUL H, we know from the definition of NUL that there exists $m_H(h, \varepsilon, \delta)$ such that ...

For every n, define a class H_n:

$$H_n = \left\{ h \in H : m_H\left(h, \frac{1}{8}, \frac{1}{8}\right) \leq n \right\}$$

Claim These H_n meets requirements above.

1. Given any $h \in H$, let $n = m_H\left(h, \frac{1}{8}, \frac{1}{8}\right)$, so $h \in H_n$.
2. For every n, VCdim $(H_n) \leq 2n$. Otherwise, pick a subset $A \subseteq X$, $|A| > 2n$ such that H_n shatters A. Then NFL theorem tells us that we need $> m$ size training sample to learn the class of all functions over A with accuracy $\frac{1}{8}$, confidence $\frac{1}{8}$. But since A is shattered by H_n, H_n contains all functions over A. However, by definition of H_n it can be $(1/8, 1/8)$ learnt from n-size samples.

□

The positive direction of the characterization of NUL

Lemma 7.4

If for every n, H_n is learnable, then $H = \bigcup_{n=1}^{\infty} H_n$ is NUL.

Proof:

By the fundamental theorem, each H_n has the uniform convergence property. Namely, for every H_n there is a function $m_n^{UC}(\varepsilon, \delta)$ such that for any $\varepsilon, \delta > 0$, if $m > m_n^{UC}(\varepsilon, \delta)$ for every P,

$$\Pr_{S \sim P^n}\left[\exists h \in H_n : |L_S(h) - L_p(h)| > \varepsilon\right] < \delta$$

In other words, the event in the bracket: S is not ε-representative for P, H_n. □

We add a new component to our discussion - weighting function. $w : \mathbb{N} \to [0,1]$. $w(n)$ is the "weight" we assign to the class H_n. We will require that $\sum\limits_{n=1}^{\infty} w(n) \leq 1$

7.2 Structural Risk Minimization

The idea will be, given a sample S, we wish to pick $h \in H$ that minimizes

$$L_S(h) + \text{penalty that grows as } w(n) \text{ shrinks for the minimal } n \text{ for which } h \in H_n$$

Theorem 7.5

Let $H = \bigcup\limits_{n=1}^{\infty} H_n$ such that each H_n has uniform convergence property with a function $m_n^{UC}(\varepsilon, \delta)$ and let w be any weighting function with $\sum\limits_{n=1}^{\infty} w(n) \leq 1$. Then for every probability distribution P, and $h \in H$, every sample size m, and every $\delta > 0$,

$$\Pr_{S \sim P^n}\left[\exists h \in H : |L_S(h) - L_p(h)| > \varepsilon_{n(h)}\right] < \delta$$

where $n(h)$ is $\min_n\{h \in H_n\}$ and $\varepsilon_n = \min_\varepsilon$ such that $m > m_n^{UC}(\varepsilon, w(n) \cdot \delta)$

(See alternative statement on Theorem 7.4 in textbook)

Proof idea:
Step 1: For every n,

$$\Pr_{S \sim P^n}\left[\exists h \in H_n : |L_S(h) - L_p(h)| > \varepsilon_n\right] < \delta \cdot w(n)$$

Step 2: Union bound over all n's

$$\Pr_{S \sim P^n}\left[\exists h \in \bigcup H_n : |L_S(h) - L_p(h)| > \varepsilon_n\right] < \sum_n \delta \cdot w(n) \leq \delta$$

□

A-PAC		Non uniform learnability
Convergence bound (relates $L_S(h)$ to $L_D(h)$)	$\forall, \forall, \forall$, if $m > \cdots$ $\lvert L_S(h) - L_P(h)\rvert > \varepsilon$ w. p. $< \delta$	$H = \cup_n H_n$, each H_n has UC with $m_n(\varepsilon, \delta)$, $w : \mathbb{N} \to [0,1]$, $\sum_n w(n) \leq 1$, $\forall P, \forall m, \delta$ $\Pr_{S \sim P^m}\left[\exists h \in H : \lvert L_S(h) - L_P(h)\rvert > \varepsilon_{n(h)}(m, w(n(h))\delta)\right] < \delta$
Learner bound its error/loss	If A is an ERM learner, then $\forall, \forall, \forall h \in H$, $L_P(A(S)) \leq L_P(h) + \varepsilon$ for $m > m_H(\frac{\varepsilon}{2}, \delta)$ w.p. $1 - \delta$	Structural Risk Minimization Let $A(S)$ be any $h \in H$ that minimizes this bound $L_S(h) + \varepsilon_{n(h)}(m, w(n(h))\delta)$.

Note that when n gets larger, ε is small. For two reasons. First, w will be small because the summation of w is ≤ 1, then far members in the sequence should be small. Then multiply by δ, we get even smaller, thus high confidence. Second, m is fixed. I cannot guarantee for a small ε, ε should be very loose.

A SRM algorithm is given some training sample S and a confidence parameter δ and picks h to minimize this error bound. $L_S(h)$ is how well you do with the sample, $\varepsilon_{n(h)}(m, w(n(h))\delta)$ is the penalty for picking a complex assumption.

Theorem 7.6

Given any $H = \bigcup_n H_n$, where each H_n has the uniform convergence property with some $m_n(\varepsilon, \delta)$ and given any $w : \mathbb{N} \to [0,1]$ such that $\sum_n w(n) \leq 1$, and SRM algorithm is a successful Non-Uniform learner with sample complexity

$$m_H(h, \varepsilon, \delta) = m_{n(h)}\left(\frac{\varepsilon}{2}, w(n(h)) \cdot \delta\right)$$

Proof:
See textbook. □

SRM algorithm

prior knowledge: $H = \bigcup_n H_n$ where H_n has uniform convergence with m_n^{UC} and $w : \mathbb{N} \to [0,1]$ where $\sum_n w(n) \leq 1$.

$$\begin{aligned} \varepsilon_n(m, \delta) &:= \min\{\varepsilon \in (0,1) : m_n^{UC}(\varepsilon, \delta) \leq m\} \\ n(h) &:= \min\{n : h \in H_n\} \end{aligned}$$

input: training set $S \sim D^m$, confidence δ

output: $h \in \text{argmin}_{h \in H}[L_S(h) + \varepsilon_{n(h)}(m, w(n(h)) \cdot \delta)]$

7.3 Minimum Description Length (MDL) Learning

Skipped. Lecture 14, 54:30 → Lecture 15, 43:30.

8

The Runtime of Learning

8.1 Computational Complexity of Learning

Two kind of resources: Information (training sample size), computation (for how long will our algorithm run, once it has sufficient information?)

Runtime of my algorithm: Asymptotic behavior, count computational steps rather than time.

For combinatorial tasks: shortest path on a graph (s, t), or sorting. Here $O(n)$, n is the size of the input, but it is not clear what is the input for learning. What should play the role of the input size parameter in learning?

Answer 1: sample size $|S|$. The problem here is that this approach allows cheating. For example, suppose the algorithm takes 1000 steps and an input size of 10 and 1000 both take the same number of steps. Sometimes more samples will make the problem easier.

Answer 2: $f(\varepsilon, \delta)$. Then we really want to have running time $poly(\frac{1}{\varepsilon}, \frac{1}{\delta}, \underset{\substack{\uparrow \\ \text{hypothesis complexity}}}{n})$

One more issue to take care of: what is the output of a learning algorithm? It's a function $h : X \to \{0, 1\}$. Input: S, ε, δ. Output: Use ERM on S. Thus we define the runtime of a learning algorithm L as

$$\max\left\{\begin{array}{lcl} \text{time it} & & \text{time it takes} \\ \text{takes } L & & \text{that } h \text{ to output} \\ \text{to output} & , & \text{a label on} \\ \text{some } h & & \text{any given } X \end{array}\right\}$$

We need to make sure our output can be efficiently used.

8.2 Examples of the computational complexity of some concrete tasks

Learning axis aligned rectangles in $\mathbb{R}^d$

Consider the complexity of ERM learning. Recall VCdim $(H_d) = 2d$, then

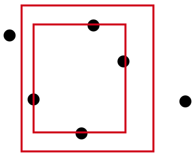
$$m_{H_d}(\varepsilon, \delta) \sim c\frac{2d + \ln(2/\delta)}{\varepsilon^2}$$

Then our algorithm:

⓪ Pick an $m_H(\varepsilon, \delta)$ size sample S

① implement ERM(S)

It suffices to consider rectangles that have points of S on every boundary edge. Every such rectangle is determined by $\leq 2d$ points from S. There are $\leq m^{2d}$ such tuples. For example, in $\mathbb{R}^2$, these two rectangles give the same error.



Thus runtime $\sim \left[c\frac{2d+\ln(2/\delta)}{\varepsilon^2}\right]^{2d}$.

Conclusion: For every fixed dimension d, ERM_{H_d} can be implemented in time $poly(1/\varepsilon, 1/\delta)$, therefore, we have efficient learning. However, as a function of d, my runtime is exponential.

To prove a positive result:

- Step 1: Upper bound needed sample size.

  We know that VCdim $\left(Rec^d\right) = 2d$ and $m_{H^d_{Rec}}(\varepsilon, \delta) \leq c\frac{2d+\ln(2/\delta)}{\varepsilon^2}$

- Step 2: Upper bound the time needed to compute $\text{ERM}_{H^d_{Rec}}(m)$.

Algorithm: Let $A_1, \ldots, A_t, \ldots, A_{m^{2d}}$ be a list of all subsets of S of size $2d$.

For $1 \leq t \leq m^{2d}$, let R_t be the minimum axis aligned rectangle that contains A_t, and compute $L_S(R_t)$.

Output: some $R_t \in \text{argmin}(L_S(R_i))$

Correctness: For every $h \in H_{Rec^d}$, for every S, $\exists R_t$ such that $L_S(R_t) \leq L_S(h)$

Runtime: $\underset{\substack{\uparrow \\ \text{per iteration}}}{m} \cdot m^{2d} = m^{2d+1} = \left[c\frac{2d+\ln(1/\delta)}{\varepsilon}\right]^{2d+1}$ which is not polytime in d

Boolean queue example

$X = \{0,1\}^d$. Let H^d_{con} = Boolean conjunctions over $\{0,1\}^d$

Consider the variables $p_1, \ldots, p_d$. A *literal* is a variable or its negation $(p, \neg p)$. A *conjunction* $\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_k$ where each ℓ_j is a literal. More compact notation

$$\bigwedge_{j=1}^{k} \ell_{i_j}$$

What is the computational complexity of learning H^d_{con}? Consider ERM learning.

- Step 1: Upper bound $m_H(\varepsilon, \delta)$.

  Recall that for any class H, VCdim $(H) \leq \log(|H|)$. We have $|H| \leq 2^{2d}$ because either $p/\neg p$ is in the conjunction or not. We also have $|H| \leq 3^d$ (include p, $\neg p$ or no p).

  Then $m_H(\varepsilon, \delta) \leq c\frac{2d+\ln(1/\delta)}{\varepsilon^2}$

- Step 2: Given a sample S of size m. How much time (computational) is needed to compute $\text{ERM}_H(S)$?

Let $h_0 = p_1 \wedge \neg p_1 \wedge p_2 \wedge \neg p_2 \wedge \cdots \wedge p_d \wedge \neg p_d$ which is all $2d$ literals.

$$L_S(h_0) = \frac{|\{h(x_i, y_i) : y_i = 1\}|}{m}$$

Given h_t, define h_{t+1} as follows: consider the t's examples in S (x_t, y_t),

$$h_{t+1} = \begin{cases} h_t & h_t(x_t) = y_i \\ h_t \setminus \{\text{literals that has a conflict with } (x_t, y_t)\} & h_t(x_t) \neq y_i \end{cases}$$

h_{t+1} is the most demanding conjunction that accepts the + labeled examples among $(x_1, y_1), \dots, (x_t, y_t)$.

Then h_{m+1} is consistent with S (assuming some conjunction has zero error over S). Then this is correct in the realizable case.

Runtime: $m \cdot 2d = c\dfrac{2d + \ln(1/\delta)}{\varepsilon^2} \cdot 2d$ which is polytime of d.

3-term DNF

$X = \{0,1\}^d$, $H =$ 3-term DNF's over X. Each $h \in H$ has the form $h = A_1 \vee A_2 \vee A_3$ where each A_i is a conjunction.

- First step: estimating $m_H(\varepsilon, \delta)$

  $|H^d_{3T-DNF}| \leq (3^d)^3 = 3^{3d}$ then $m_H(\varepsilon, \delta) \leq c\dfrac{d + \ln(1/\delta)}{\varepsilon^2}$

- How hard it is to compute $\text{ERM}_H(S)$ over an m size sample? This is NP-hard even in the realizable case.

Non ERM algorithm: Note that each $h = A_1 \vee A_2 \vee A_3$ is equivalent to

$$h' = \bigwedge_{\substack{u \in A_1 \\ v \in A_2 \\ w \in A_3}} (u \vee v \vee w)$$

Define new $(2d)^3$ variables: x_{uvw} will represent $u \vee v \vee w$. Such conjunctions can be learned in time $poly(1/\varepsilon, 1/r, 8d^3)$. Here we learn in a bigger class $Conj^{8d^3}$.

Proper learning - output $h \in H$
Unrestricted learning[1] - output any h.

8.3 Hardness of Learning

What do we mean by computational hardness?

NP-hardness: Unless there is a big surprise in math, there is no polynomial time algorithm that solves the problem for *all* inputs.

Some examples of NP-hard learning problems:

1. H^d_{Rec} if we wish the algorithm to be also polynomial in d.

   Also NP hard in realizable and proper case.

2. Learning half-spaces (linear separators).

   Easy - realizable case. NP hard in the agnostic case (proper). Hard also for unrestricted learning.

Intersection of k half-spaces: NP hard in the realizable case as soon as $k \geq 3$

Cryptographic hardness

One-way functions are functions that are easy to compute $f(x)$ from x, but hard to compute x from $f(x)$.

[1] or improper learning in the textbook

Crypto is often based on the assumption that there are one ways functions in the sense that no polytime algorithm can invert them.

f is a *trapdoor function* if 1. f is one way; 2. For every n, there is a key s_n such that it is easy to invert $f(x)$ on all inputs $x \in \{0,1\}^n$, given s_n.

Let $\mathcal{F}_n$ be a class of trapdoor functions over $\{0,1\}^n$, i.e., $\mathcal{F}_n = \{f_{s_n} : s_n \text{ is a key}\}$

Learn $= \{f^{-1} : f \in \mathcal{F}_n\}$. $|\mathcal{F}_n| \leq 2^{poly(n)}$, then VCdim $(\mathcal{F}_n) = poly(n)$.

9 Linear Predictors

In this chapter we will study the family of linear predictors, one of the most useful families of hypothesis classes. Many learning algorithms that are being widely used in practice rely on linear predictors, first and foremost because of the ability to learn them efficiently in many cases. In addition, linear predictors are intuitive, are easy to interpret, and fit the data reasonably well in many natural learning problems. [1]

See section 6.5 for a brief introduction to the halfspaces.

First, we define the class of affine functions as

$$L_d = \{h_{w,b} : w \in \mathbb{R}^d, b \in \mathbb{R}\},$$

where

$$h_{w,b}(x) = \langle w, x \rangle + b + \left(\sum_{i=1}^{d} w_i x_i\right) + b.$$

It will be convenient also to use the notation

$$L_d = \left\{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle + b : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\right\}.$$

9.1 Halfspaces

The first hypothesis class we consider is the class of halfspaces, designed for binary classification problems, namely, $X = \mathbb{R}^d$ abd $Y = \{\pm 1\}$. The class of halfspaces is defined as follows:

$$HS_d = \operatorname{sign} \circ L_d = \left\{\mathbf{x} \mapsto \operatorname{sign}\left(h_{\mathbf{w},b}(\mathbf{x})\right) : h_{\mathbf{w},b} \in L_d\right\}$$

So it returns $\operatorname{sign}(\langle w, x \rangle + b)$.

We have shown in section 6.5 that $\operatorname{VCdim}(HS_d) = d + 1$ although we only showed the homogeneous case.

We then show two solutions to find an ERM halfspace in the realizable case.

9.1.1 Linear Programming

Let $S = \{(x_i, y_i)\}_{i=1}^m$ be a training set of size m. Since we assume the realizable case, an ERM predictor should have zero errors on the training set. That is, we are looking for some vector $w \in \mathbb{R}^d$ for which $\operatorname{sign}(\langle w, x_i \rangle) = y_i$ for all i. Equivalently, $y_i \langle w, x_i \rangle > 0$ for all i.

[1] This chapter hasn't been covered explicitly in any video lectures. Therefore I copied the introduction text from the textbook.

Let w^* be a vector satisfies this condition (it exists due to realizability). Define $\gamma = \min_i (y_i \langle w^*, x_i \rangle)$ and let $\overline{w} = \frac{w^*}{\gamma}$. Therefore, for all i we have

$$y_i \langle \overline{\mathbf{w}}, x_i \rangle = \frac{1}{\gamma} y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle \geq 1$$

We have thus shown that there exists a vector that satisfies

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1, \quad \forall i = 1, \dots, m$$

And clearly, such a vector is an ERM predictor.

To solve it, we can set a dummy objective and use the constraints above.

9.1.2 Perceptron for Halfspaces

A different implementation of the ERM rule is the Perceptron algorithm of Rosenblatt (Rosenblatt 1958).

Algorithm 1: Batch Perceptron

```
Input: A training set $\left((x_i, y_i)\right)_{i=1}^{\infty}$
$w^1 := \mathbf{0}$
for $t = 1, 2, \dots$ do
    if $\exists i$ such that $y_i \langle w^t, x_i \rangle \leq 0$ then
        $w^{t+1} = w^t + y_i x_i$
    else
        return $w^t$
```

Since

$$y_i \left\langle \mathbf{w}^{(t+1)}, \mathbf{x}_i \right\rangle = y_i \left\langle \mathbf{w}^{(t)} + y_i \mathbf{x}_i, \mathbf{x}_i \right\rangle = y_i \left\langle \mathbf{w}^{(t)}, \mathbf{x}_i \right\rangle + \|\mathbf{x}_i\|^2$$

Hence, the update of the Perceptron guides the solution to be “more correct” on the i-th example.

Then theorem 9.1 in textbook ensures the correctness of the algorithm.

Here $B = \min\{\|w\| : \forall i \in [m],\ y_i \langle w, x_i \rangle \geq 1\}$

Remark:
The Perceptron is simple to implement and is guaranteed to converge. However, the convergence rate depends on the parameter B, which in some situations might be exponentially large in d. In such cases, it would be better to implement the ERM problem by solving a linear program, as described in the previous section. Nevertheless, for many natural data sets, the size of B is not too large, and the Perceptron converges quite fast.

9.1.3 The VC Dimension of Halfspaces

Theorem 9.1

The VC dimension of the class of homogenous halfspaces in $\mathbb{R}^d$ is d.

Proof:
This has been proved in section 6.5. □

Theorem 9.2

The VC dimension of the class of homogenous halfspaces in $\mathbb{R}^d$ is $d + 1$.

Proof:

See textbook. The trick is to use reduction.

Let $w' = (b, w_1, \ldots, w_d) \in \mathbb{R}^{d+1}$ and let $x' = (1, x_1, \ldots, x_d) \in \mathbb{R}^{d+1}$. Therefore,

$$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \langle \mathbf{w}', \mathbf{x}' \rangle$$

□

10

Boosting

10.1 Weak Learnability

What if we settle for "weak learning"? In Rob Schapire's PhD thesis, it does not make life easier.

The way to show it was by designing an algorithm that given access to a weak learner, outputs a strong learner.

gamma-weak learnable

A class H is **γ-weakly learnable** (for some $\gamma \in (0, 1/2)$) if there exists a learner A and a function $m_H^\gamma(\delta)$ such that for every probability distribution D over X and every $f \in H$ on a sample S generated by (D, f) of size $> m_H^\gamma(\delta)$ with probability $> 1 - \delta$

$$L_{(D,f)}(A(S)) < \frac{1}{2} - \gamma$$

The question that we focused on is the *existence* of efficient weak learners - running in time $\text{poly}(\frac{1}{\delta}, m)$.

Example: Weak Learning of 3-Piece Classifiers Using Decision Stumps

Let H be the class of 3-partitions of $\mathbb{R}$.

$$H_3 = \{h^+_{r_1,r_2}, h^-_{r_1,r_2} : r_1 < r_2 \in \mathbb{R}\}$$

where

$$h^+_{r_1,r_2}(x) = \begin{cases} +1 & \text{if } x < r_1 \text{ or } x > r_2 \\ -1 & \text{if } r_1 \leq x \leq r_2 \end{cases} \qquad h^-_{r_1,r_2}(x) = -h^+_{r_1,r_2}(x)$$

Claim H_3 is $\frac{1}{6}$ weakly learnable by ERM over Decision stumps (threshold functions).

Proof:

For every distribution D over $\mathbb{R}$, and every $f \in H_3$, there exists a threshold function h such that $L_{(D,f)}(h) \leq \frac{1}{3} = \frac{1}{2} - \frac{1}{6}$.

First note that for each of the three regions of f $\exists h$ threshold that errors only on this region. Second note that for any D over r and any $f \in H_3$ there is a region of f that has D-weight at most $\frac{1}{3}$. □

Theorem 10.1

For any $\gamma \in (0, 1/2)$, a class is γ-weakly learnable if only if it has finite VCdim.

Proof:

It's clear that finite VCdim $\implies$ Weakly learnable.

Assume infinite VCdim, then $m(\frac{1}{2} - \gamma, \delta) > c\frac{\mathrm{VCdim}(H)+\ln(1/\delta)}{\varepsilon}$ which is infinite.

So it doesn't give us any new classes, it is just the matter of computational efficiency. □

We will focus on weak learners of the type ERM_B for some "basic" class B. Recall our previous example:

Example:

$H = H_3$, $B = H_{thr}$ and the claim: for evert sample S labeled by a function $f \in H_3$, there exists some $h \in B$ such that $L_S(h) \leq \frac{1}{3}$.

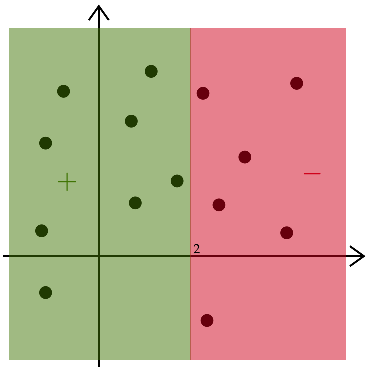
Corollary 10.2

ERM_B is a γ-weak learner for H_3, where $\gamma < \frac{1}{6}$.

Given $\gamma < \frac{1}{6}$ let ε be such that $\gamma + \varepsilon < \frac{1}{6}$, let $m(\delta) = m_B(\varepsilon, \delta)$. It will guarantee that

$$L_{(D,f)}(\mathrm{ERM}_B(S)) \leq L_S(\mathrm{ERM}_B(S)) + \varepsilon \leq \frac{1}{6}$$

In practice, the most popular weak learner class is H^d_{DS}, which is the class of decision stumps over $\mathbb{R}^d$.



$h_{1,2,-1}$

Every $h \in H^d_{DS}$ is determined by three parameters $(i, r, \pm 1)$,

$$h_{i,r,+1}(x) = \begin{cases} -1 & x_i < r \\ +1 & x_i \geq r \end{cases}$$

where $x = (x_1, \ldots, x_d)$.

Claim For every d the class H^d_{DS} is efficiently learnable, more concretely, $\mathrm{ERM}_{H^d_{DS}}$ can be implemented in time $\tilde{O}(md)$ where m is the sample size.

Proof:

Input $S = ((x_1, y_1), \ldots, (x_m, y_m))$ where each x_i is a vector in $\mathbb{R}^d$ and $y_i \in \{+1, -1\}$

For $1 \leq i \leq d$, for each $1 \leq j \leq m$, consider $h_{i,x_j(i),+}, h_{i,x_j(i),-}$. Compute $L_S(h)$ for each such h, output the minimizer of this loss. Need to evaluate $2(m+1)d$ hypotheses h, now we need m checks to evaluate $L_S(h)$ for each h because if we order them, if we move our h over a point (by one step), we just need to check one point's labeling (err → non-err or non-err → err). We need $d(m \log m)$ to order in the dimension, then we check in $2(m+1)d$. Then total is $d(m \log m) + 2(m+1)d \in \tilde{O}(md)$. □

10.2 The Boosting algorithm paradigm

Input:

- a labeled sample $S = (x_1, y_1), \dots, (x_m, y_m)$
- Some Weak Learner (ERM_B)
- T - # of iterations

For each iteration t, we will fix a probability distribution D_t over $(x_1, \dots, x_m)$. We define

$$D_1 := \left(\frac{1}{m}, \dots, \frac{1}{m}\right)$$

We get D_{t+1} by first applying $h_t = WT(D_t, S)$ (weak teacher/learner) and increasing the D probability of each x_i on which h_t errors and decreasing the probability of each x_i on which h_t predicts y_i.

Then we get $h_1, h_2, \dots, h_t, h_T$. Output $\text{sign}\left(\sum_{t=1}^{T} w_t h_t\right)$.

Note that we define error $\varepsilon_t = \sum_{i=1}^{m} D_i^{(t)} \mathbb{1}_{[h(x_i) \neq y_i]}$

For the AdaBoost algorithm

$$D_{t+1}(x_i) = \frac{D_t(x_i) e^{-w_t y_i h_t(x_i)}}{Normalizer}$$

where $w_t = \frac{1}{2} = \frac{1}{2}\log(\frac{1}{\varepsilon_t} - 1)$. Weight is inversely proportional to error.

Analyzing the Boosting algorithm

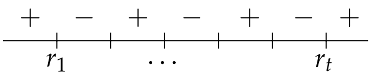
Let us consider the class of outputs of Boosting. When our weak learner is ERM_B, the output $L(B,T) = \{\text{sign}(\sum_{t=1}^{T} w_t h_t) : w_i \in \mathbb{R}, h_i \in B\}$

Demonstration of the richness of $L(B,T)$

Let $B = H_{DS}^1$. Claim: $L(B,T)$ contains all functions (over $\mathbb{R}$) that have $\leq T$ segments.

Proof:

Consider $f =$



for every $f(x) = \text{sign}(\sum w_t h_t)$ where $w_1 = 0.5$, and $w_t = (-1)^t$ for $t > 1$, and $h_t = Thr(r_t)$. □

The empirical error of Boosting

Theorem 10.3

If WT is a γ-weak learner for the sample S (for every distribution D over S, $WT(D,S)$ has error $\leq \frac{1}{2} - \gamma$ with respect to (D,S)), then for every T, then

$$L_S(h_s) = L_S\left(\text{sign}\left(\sum_{t=1}^{T} w_t h_t\right)\right) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{[h_s(x_i) \neq y_i]} \leq \exp(-2\gamma^2 T)$$

What do we gain (or lose) by picking large T?

We know from before, if $h \in H$ and VCdim (H), for every data distribution P, iid S from P,

$$L_P(h) \le L_S(h) + \sqrt{\frac{\text{VCdim}\,(H) + \ln(1/\delta)}{|S|}}$$

Bounding $L_S(h)$ by Theorem 10.3

Bounding the VCdim of the class H from which the Boosting output comes

Recall the class of all potential outputs after T steps is $L(B,T) = \{\text{sign}(\sum_{t=1}^{T} w_t h_t) : w_i \in \mathbb{R}, h_i \in B\}$

Theorem 10.4

For every class B of finite VCdim $(B) = d$ and every T (assume $T, d \ge 3$),

$$\text{VCdim}\,(L(B,T)) \le (d+1)T(3\log((d+1)T) + 2)$$

Proof:
Assume $L(B,T)$ shatters some set A of size m. In that case,

$$\left|\{h \cap A : h \in L(B,T)\}\right| \ge 2^m$$

By Sauer's lemma, $|\{h \cap A : h \in B\}| \le (em/d)^d \le m^d$ if $d \ge 3$. The number of T combinations of such subsets is at most $(m^d)^T = m^{dT}$. Then

$$\left|\{h \cap A : h \in L(B,T)\}\right| \le \underset{\substack{\uparrow \\ \text{picking } h_i\text{'s}}}{m^{dT}} \cdot \overset{\substack{\text{linear predictor} \\ \downarrow}}{m^T}$$

Then $m^{dT} m^T \ge 2^m$. Then we are done if we sub the bound in. □

10.3 Face detection algorithm based on Boosting

Inputs are images gray scale with 24×24 pixels. The class of basic classifiers B. Every $h \in B$ is determined by a rectangle in the image and one of four types A, B, C, D (img from textbook).

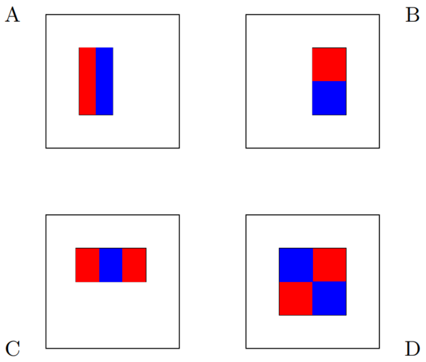


Figure 10.1 The four types of functions, g, used by the base hypotheses for face recognition. The value of g for type A or B is the difference between the sum of the pixels within two rectangular regions. These regions have the same size and shape and are horizontally or vertically adjacent. For type C, the value of g is the sum within two outside rectangles subtracted from the sum in a center rectangle. For type D, we compute the difference between diagonal pairs of rectangles.

Embed the image in $24^4 \cdot 4$ length vector. See the textbook for details.

11

Model Selection and Validation

In the previous chapter we have described the AdaBoost algorithm and have shown how the parameter T of AdaBoost controls the bias-complexity tradeoff. But, how do we set T in practice? More generally, when approaching some practical problem, we usually can think of several algorithms that may yield a good solution, each of which might have several parameters. How can we choose the best algorithm for the particular problem at hand? And how do we set the algorithm's parameters? This task is often called **model selection**. [1]

[1] This chapter hasn't been covered explicitly in any video lectures. Therefore I copied the introduction text from the textbook.

12

Convex Learning Problems

So far we have discussed: Sample complexity and Computational complexity: How can we overcome the computational hardness of learning?

Answer 1: Boosting

Answer 2: Characterize a big family of efficiently learnable task.

convex set

A set $X \subseteq \mathbb{R}^n$ is **convex** if for every $x, y \in X$ and $0 \leq \alpha \leq 1$, $\alpha x + (1-\alpha)y \in X$.

Example:

Ball

Linear half-spaces:

$$L^+_{w,b} = \{\bar{x} \in \mathbb{R}^d : \langle \bar{x}, w \rangle + b \geq 0\}$$
$$L^-_{w,b} = \{\bar{x} \in \mathbb{R}^d : \langle \bar{x}, w \rangle + b \leq 0\}$$

Some nice properties

Proposition 12.1

If A, B are convex, then so is $A \cap B$.

Corollary 12.2

For every $A \subseteq \mathbb{R}^n$, there exists a set $Ch(A)$ which is convex, $A \subseteq Ch(A)$, and for any convex $B \supseteq A$, $Ch(A) \subseteq B$. Thus $Ch(A)$ is the minimal convex set containing A.

$$Ch(A) = \bigcap \{B : B \text{ convex and } A \subseteq B\}$$

convex function

A function $f : \mathbb{R}^d \to \mathbb{R}$ is **convex** if for all x, y, and $0 \leq \alpha \leq 1$,

$$f\big(\alpha x + (1-\alpha)y\big) \leq \alpha f(x) + (1-\alpha) f(y)$$

Proposition 12.3

$f : \mathbb{R}^d \to \mathbb{R}$ is convex function if and only if

$$\text{epigraph}(f) = \{(x, y) : y \geq f(x)\} \subseteq \mathbb{R}^{d+1}$$

is a convex set.

12.1 Properties of convex functions

local minimum

Given a function $f : \mathbb{R}^d \to \mathbb{R}$, a point u is a **local minimum** of f if there exist some $0 < r$ such that for all $v \in B(u, r)$, $f(u) \leq f(v)$.

Proposition 12.4

If f is convex then any local minimum of f is also a global (true) minimum.

Proof:

Let f be convex and u a local min of f. Let v be any other point in $\mathbb{R}^d$. For some $r > 0$, u is a min among points in $B(u, r)$.

Pick some $\alpha > 0$ such that $\alpha v + (1 - \alpha)u \in B(u, r)$.

By the local convexity, we have

$$\begin{aligned} f(u) &\leq f(\alpha v + (1 - \alpha)u) && \text{by local convexity} \\ &\leq (1 - \alpha)f(u) + \alpha f(v) && \text{by convexity of } f \\ &= f(u) - \alpha f(u) + \alpha f(v) \end{aligned}$$

which implies $f(u) \leq f(v)$. □

Proposition 12.5

Let $f : \mathbb{R} \to \mathbb{R}$ and assume that f', f'' exists. TFAE:

- f is convex
- f' non-decreasing
- $f'' \geq 0$

Corollary 12.6

$f(x) = C$, $f(x) = x$, $f(x) = x^2$ are all convex.

Some closure properties of convex functions

Proposition 12.7

If f, g are both convex, then so is $\max(f, g)(x) = \max\{f(x), g(x)\}$.

Use epigraphs of f and g to prove.

Corollary 12.8

$|x|$ is a convex function.

Proposition 12.9

If $f_1, \dots, f_n$ are all convex, and $w_1, \dots, w_n \geq 0$, then

$$g(x) = \sum_{i=1}^{n} w_i f_i(x)$$

is also convex.

Proposition 12.10

For every convex $g : \mathbb{R} \to \mathbb{R}$, the function $f(w) = g(\langle w, x\rangle + y)$ $(f : \mathbb{R}^d \to \mathbb{R})$ for every $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$ is also convex.

See the alternative statement in Claim 12.4 from the textbook.

> Proof:
> Just evaluate f at $w = \alpha u + (1-\alpha)v$ for every $u, v \in \mathbb{R}^d$ and use the linearity of $\langle w, x\rangle$ and convexity of g. □

In the context of linear regression, we are given a collection of pairs $(x, y) \in \mathbb{R}^d \times \mathbb{R}$. A common loss for linear regression is a search for the best linear function that predicts y from x:

$$\ell(w, (x, y)) = (\langle w, x\rangle - y)^2$$

which is called the square loss.

Corollary 12.11

For any (x, y) the square loss is a convex function of w.

12.2 Convex Learning Problems

convex learning problem

A learning problem, (H, Z, ℓ) is called **convex** if hypothesis class H (set of models) is convex and for all $z \in Z$, $\ell(\cdot, z)$ is a convex function $(\mathbb{R}^d \to \mathbb{R})$.

Here Z is a space of instances, ℓ is loss function, $\ell(h, z) \in \mathbb{R}^+$. In particular, we focused on the binary prediction with 0-1 loss case. $Z = X \times \{0, 1\}$ and

$$\ell(h, (x, y)) = \begin{cases} 0 & \text{if } h(x) = y \\ 1 & \text{if } h(x) \neq y \end{cases}$$

> How to overcome the computational complexity of learning tasks? Since even the most basic learning paradigm is computationally infeasible for many classes of interest.

> We are facing optimization problems: given S and H, find $h \in H$ that minimizes $L_S(h)$ over H.
>
> A major source of computational difficulty solving such tasks is the existence of local minima.
>
> We defined the motion of convex functions and showed that such functions do not have local minima (unless these are global minima).

Continue from last time...

Convex optimization problems: given some domain set $W \subseteq \mathbb{R}^d$, and a function $f : \mathbb{R}^d \to \mathbb{R}$. The task is to find some $w \in W$ such that for all $u \in W$, $f(w) \leq f(u)$. Such a problem is called a convex optimization problem if W is convex set and f is a convex function.

Recall our general learning framework: a triple (H, Z, ℓ). H a class of models, Z a domain set, ℓ loss function $H \times Z \to \mathbb{R}^+$. Given such a triplet and some unknown probability distribution P over Z, the goal of the learner is to use an iid sample S generated by P to find $h \in H$ with small expected loss:

$$L_P(h) = \mathop{\mathbb{E}}_{z \sim P} \ell(h, z)$$

> Example: Binary classification with 0-1 loss
>
> $Z = (X \times \{0, 1\})$, H is a class of functions $h : X \to \{0, 1\}$, and $\ell^{0-1}(h, (x, y)) = \begin{cases} 0 & \text{if } h(x) = y \\ 1 & \text{if } h(x) \neq y \end{cases}$

> Example: Linear regression
>
> $H =$ linear functions $h : \mathbb{R}^d \to \mathbb{R}$ where $h_{w,b}(x) = \langle w, x \rangle + b$.
>
> $Z = \mathbb{R}^{d+1} = \mathbb{R}^d \times \mathbb{R}$
>
> $\ell(h_{w,b}, (x, y)) = \big(\langle w, x \rangle + b - y \big)^2$ square loss

> Example: k-means clustering
>
> $Z \subseteq \mathbb{R}^n$, $H = \underbrace{\mathbb{R}^n \times \cdots \times \mathbb{R}^n}_{k}$, $\ell((\mu_1, \ldots, \mu_k), x) = \min_{1 \leq i \leq k} (x - \mu_i)^2$

A learning problem (H, Z, ℓ) is convex if

1. H is convex subset of $\mathbb{R}^d$ for some d.
2. For every $z \in Z$, the function $\ell(\cdot, z)$ is a convex function.

Note that $H \subseteq \mathbb{R}^d$ is convex as a subset of $\mathbb{R}^d$, also can be viewed as parameter space. For example, every $h \in H$ is a homogeneous linear function $h_w(x) = \langle w, x \rangle$.

0-1 classification with a class of half-spaces as well as linear regression can be viewed as having $H \subseteq \mathbb{R}^d$.

Proposition 12.12

If ℓ is a convex loss, and H is a convex subset of $\mathbb{R}^d$, then for every sample S, $\mathrm{ERM}_H(S)$ is a convex optimization problem.

Given S, H, ℓ, the ERM task is to find $h \in H$ that minimizes $L_S(h) = \frac{1}{|S|} \sum_{z \in S} \ell(h, z)$. Just recall that if for all $i \in [1, m]$, $f_i(w)$ is a convex function, then for any $a_1, \ldots, a_m \geq 0$, $f(w) = \sum_{i=1}^{m} a_i f_i(w)$ is also convex.

Denote $S := (z_1, \ldots, z_m)$, then $L_S(h) = \sum_{i=1}^{m} \frac{1}{m} \ell(h, z_i)$

Example: Linear regression

Allowing our parameter space to be $\mathbb{R}^d$ (or $\{w \in \mathbb{R}^d : \|w\| \leq 1\}$)

$\ell(w,(x,y)) = (\langle w, x\rangle - y)^2$ is convex for every (x,y)

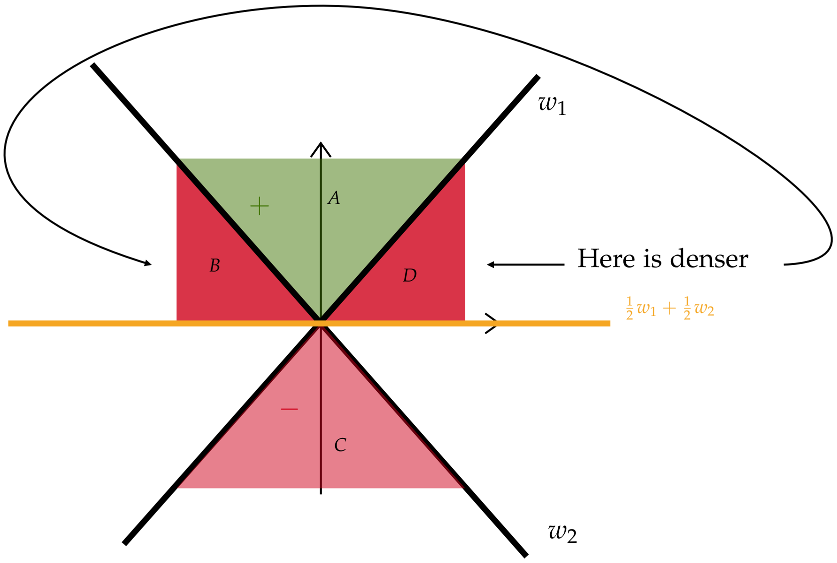
Conclusion: Linear regression with square loss is a convex learning problem.

Example: Learning linear classifiers with the 0-1 loss

Not convex.

Recall, $\ell^{0-1}(w,(x,y)) = \begin{cases} 1 & \text{if } \operatorname{sign}(\langle w, x\rangle) \neq y \\ 0 & \text{otherwise} \end{cases}$

We want to see $\ell(\alpha w_1 + (1-\alpha) w_2, (x,y)) \overset{?}{\leq} \alpha \ell(w_1,(x,y)) + (1-\alpha)\ell(w_2,(x,y))$



w_1 mispredicts on B, w_2 mispredicts on D. $\frac{1}{2}w_1 + \frac{1}{2}w_2$ mispredicts on B and D.

$\ell_S(\frac{1}{2}w_1 + \frac{1}{2}w_2) = \frac{|B|+|D|}{m}$ and $\ell_S(w_1) = \frac{|B|}{m}, \ell_S(w_2) = \frac{|D|}{m}$. Just assume $|B| = |D| = \frac{m}{4}$.

Then $\ell_S(\frac{1}{2}w_1 + \frac{1}{2}w_2) = \frac{1}{2} > \frac{1}{2}\ell_S(w_1) + \frac{1}{2}\ell_S(w_2) = \frac{1}{4}$, thus not convex.

Conclusion: the 0-1 loss is not convex. Furthermore, $\text{ERM}_H(S)$ is not a convex optimization problem.

12.2.1 Learnability of Convex Learning Problems

Note that this subsection is not covered in youtube lectures.

Maybe all convex learning problems over $\mathbb{R}^d$, are learnable? The example below shows the answer is negative, even when d is low. There is no contradiction to VC theory since VC theory only deals with binary classification while here we consider a wide family of problems. There is also no contradiction to the "discretization trick" (in chapter 9) as there we assumed that the loss function is bounded and also assumed that a representation of each parameter using a finite number of bits suffices.

Example: Nonlearnability of Linear Regression Even If $d = 1$

Let $H = \mathbb{R}$ and $\ell(w,(x,y)) = (wx - y)^2$ (homogeneous case). Let A be any deterministic algorithm. Then see the textbook for details: we pick two distributions and use the fact that A is deterministic.

In summary, we have shown that for every A there exists a distribution on which A fails, which implies that the problem is not PAC learnable.

A possible solution to this problem is to add another constraint on the hypothesis class. In addition to the convexity requirement, we require that H will be bounded; namely, we assume that for some predefined scalar B, every hypothesis $w \in H$ satisfies $\|w\| \leq B$.

Boundedness and convexity alone are still not sufficient for ensuring that the problem is learnable, as the following example demonstrates.

> Example:
>
> As in the previous example, consider a regression problem with squared loss. However, let $H = \{w : |w| \leq 1\} \subset \mathbb{R}$ be a bounded hypothesis class. Then similarly, we choose two different distributions such that A must fail on either of them.

This example shows that we need additional assumptions on the learning problem, and this time the solution is in Lipschitzness or smoothness of the loss function. This motivates a definition of two families of learning problems, convex-Lipschitz-bounded and convex-smooth-bounded, which are defined later. [1]

12.3 Surrogate loss functions

Focus on learning with linear functions. $H \subseteq \mathbb{R}^d$, a loss ℓ is a **convex surrogate loss** if

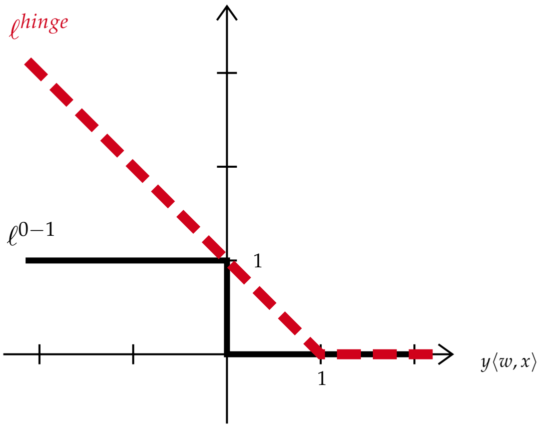
1. $\ell(\cdot, (x, y))$ is a convex function for every (x, y).
2. $\ell(w, (x, y)) \geq \ell^{0-1}(w, (x, y))$ for all $w \in H$, all (x, y).

The revised learning problem: find $h \in H$ that minimizes $L_P^{surrogate}(h)$

Hinge loss

$$\ell^{hinge}(w, (x, y)) = \max\left(0, 1 - y\langle w, x\rangle\right)$$

$y\langle w, x\rangle$ is positive if and only if $h_w(x) = y$.



Proposition 12.13

$\ell^{hinge}\langle\cdot, (x, y)\rangle$ is convex for all x, y.

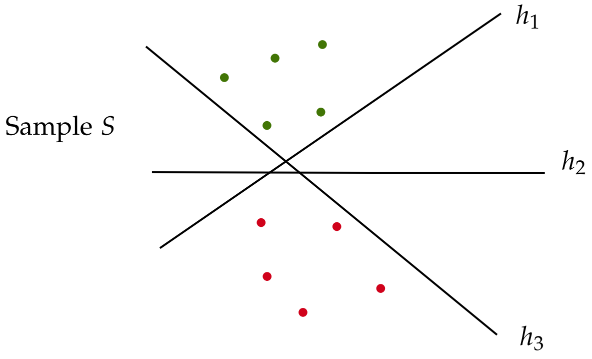
Use max of two convex is convex.

[1] actually no, as this is not required for final.

Proposition 12.14

$\ell^{hinge}(w,(x,y)) \geq \ell^{0-1}(w,(x,y))$.

The margin of a linear classifier



$$h \in \operatorname{argmin} L_S^{0-1}(h)$$

Clearly h_2 is the best among these 3.

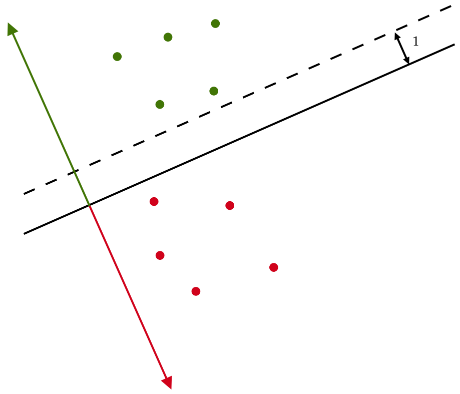
Given a linear function (w,b), $f(x) = \langle w,x\rangle + b$, and a point x, define the **margin** of f with respect to x_0 by $\min\{\|x_0 - u\| : \langle w,u\rangle + b = 0\}$

Proposition 12.15

For every $x \in \mathbb{R}^d$, $w \in \mathbb{R}^d$, $b \in \mathbb{R}$, if $\|w\| = 1$, then $margin(w,b)$ with respect to x equals $|\langle w,x\rangle + b|$.

The projection of x on to the line is $(\langle w,x\rangle + b)w$, and the perp $v = x - (\langle w,x\rangle + b)w$.

Proof:
Do algebra or check lecture 22. □



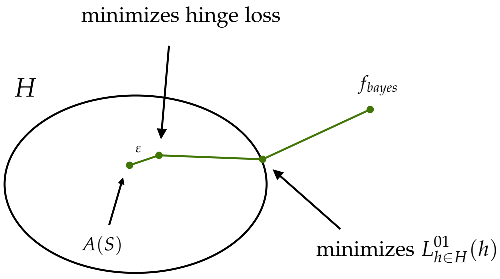
Then go back to hinge loss. It penalizes you for correct labeled points if they are too close to the line. Once it goes enough above the margin, not penalized. For mispredicted points, penalty grows as we are getting away from the line.

Then our learning algorithms will aim to minimize the hinge loss rather than 0-1 loss.

Assume we output a classifier h,

$$L_P^{0-1}(h) \leq L_P^{hinge}(h) \underset{\substack{\uparrow \\ \text{successful} \\ \text{learning}}}{\leq} \min_{h\in H}\left(L_P^{hinge}(h)\right) + \varepsilon = \underbrace{\min_{h\in H}\left(L_P^{hinge}(h)\right) - \min_{h\in H} L_P^{01}(h)}_{\substack{\text{Due to algorithmic} \\ \text{complexity consideration}}} + \underbrace{\min_{h\in H} L_P^{01}(h)}_{\text{approx. err}} + \underbrace{\varepsilon}_{\text{generalization err}}$$

$\varepsilon \to 0$ as $|S| \to \infty$.



The most common learning paradigm for linear classifiers is SVM.

$$A(S) = \text{argmin}\left[\lambda\|w\| + L^{hinge}(w,s)\right]$$

maximize the margin. This is convex.

12.4 Lipschitzness

Not in any videos.

lipschitzness

Let $C \subseteq \mathbb{R}^d$. A function $f : \mathbb{R}^d \to \mathbb{R}^k$ is **ρ-Lipschitz** over C if for every $w_1, w_2 \in C$ we have $\|f(w_1) - f(w_2)\| \leq \rho\|w_1 - w_2\|$.

Intuitively, a Lipschitz function cannot change too fast.

12.5 Smoothness

smoothness

A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is **β-smooth** if its gradient is β-Lipschitz; namely, for all v, w we have $\|\nabla f(v) - \nabla f(w)\| \leq \beta\|v - w\|$.

15

Support Vector Machines

We saw a glimpse in Chapter 12 and Appendix A. Here I'll follow the textbook.

In this chapter and the next we discuss a very useful machine learning tool: the support vector machine paradigm (SVM) for learning linear predictors in high dimensional feature spaces. The high dimensionality of the feature space raises both sample complexity and computational complexity challenges.

In the next chapter we will tackle the computational complexity challenge using the idea of kernels. [1]

15.1 Margin and Hard-SVM

Let $S = \left((x_i, y_i)\right)_{i=1}^{m}$ be training set, $x_i \in \mathbb{R}^d, y_i \in \{\pm 1\}$. We say this training set is **linearly separable** if there exists a halfspace (w, b) such that $y_i = \text{sign}(\langle w, x_i \rangle + b)$ for all i. For any separable training sample, there are many ERM halfspaces. Which one of them should the learner pick? In chapter 12, we know that we should pick h_2. One way to formalize this intuition is using the concept of margin. Hard-SVM is the learning rule in which we return an ERM hyperplane that separates the training set with the largest possible margin.

Claim The distance between a point x and the hyperplane (w, b) where $\|w\| = 1$, is $|\langle w, x \rangle + b|$.

> Proof:
> See the proof of Claim 15.1 in textbook or do algebra. □

Therefore, the Hard-SVM rule is

$$\operatorname*{argmax}_{(\mathbf{w},b):\|\mathbf{w}\|=1} \min_{i\in[m]} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| \quad \text{s.t.} \quad \forall i, y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) > 0$$

If we are in separable case, we can write an equivalent problem:

$$\operatorname*{argmax}_{(\mathbf{w},b):\|\mathbf{w}\|=1} \min_{i\in[m]} y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \tag{*}$$

Next we give another equivalent formulation of the Hard-SVM rule as a quadratic optimization problem.

Algorithm 2: Hard-SVM

```
    Input: $(x_1, y_1), \ldots, (x_m, y_m)$
    Solve: $(\mathbf{w}_0, b_0) = \text{argmin}_{(\mathbf{w},b)} \|\mathbf{w}\|^2$   s.t. $\forall i, y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$
    Output: $\hat{\mathbf{w}} = \frac{\mathbf{w_0}}{\|\mathbf{w_0}\|}, \hat{b} = \frac{b_0}{\|\mathbf{w_0}\|}$
```

[1] from textbook

The lemma 15.2 in textbook shows that the output of Hard-SVM is a solution to (*).

15.1.1 The Homogenous Case

15.1.2 The Sample Complexity of Hard-SVM

16

Kernel Methods

16.1 Embeddings into Feature Spaces

16.2 The Kernel Trick

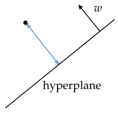
20

Neural Networks

A Lecture 23

This lecture covers briefly on the last few chapters without going much into details. The last two lectures (22, 23) smooth the transition between different chapters as you can see some SVM contents in chapter 12. So I will put all lecture 23 contents here.

We defined margin in lec 22: the distance between the point x and a hyperplane is the distance between x and closest point on the hyperplane. The drawing in lec 22 has crucial mistake.

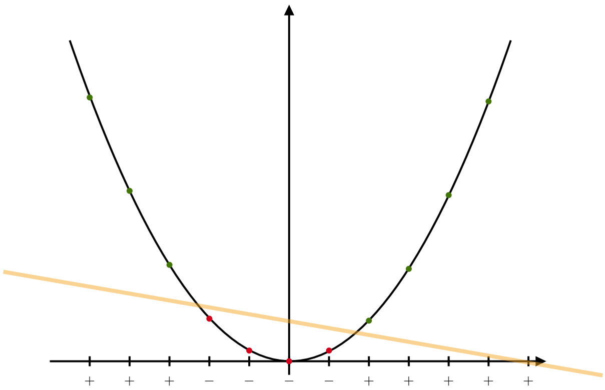


all x on hyperplane (w, b) satisfy $\langle x, w \rangle = 0$. And the distance is $|\langle w, x \rangle + b|$. See claim 15.1 in textbook. The closest point on the hyperplane is given by $v = x - (\langle w, x \rangle + b)w$.

Basic algorithmic tools in ML: we only discussed boosting. The others are kernel methods + SVM; Nearest Neighbor; Neural networks.

SVM (minimize hinge loss), only learns halfspaces. Real training data is not likely to be linearly separable. A learning algorithm, like SVM that outputs a linear classifier will always have error $\geq$ error(best linear classifier).

So we are using the idea of kernel. Let's see a simple example. How can we overcome the limited expressibility of linear classifiers? Let's try to preprocess our data by embedding it into a space which it *is* linearly separable. These points on the line are not linearly separable. After the embedding $x \mapsto (x, x^2)$, they becomes linearly separable.



We can take our data, and embed all negative points to one point, all positive points to another point. This won't work because the embedding relies on the labeling, then we don't know where should the new point be embedded to. In our previous example, it does not depend on the labels.

Is there always an embedding only based on the instances (not on labels) and guarantee every data linearly separable? Even if we require that the embedding does not depend on the labels, we can still (always) have an embedding in which there exists a perfect linear separator between + labeled points and - labeled points. We can label each point of S as a new unit vector in $\mathbb{R}^{|S|}$. We know that

$$L_P(h) \le L_S(h) + \sqrt{\frac{\text{VCdim}\,(H) + \ln(1/\delta)}{m}}$$

where the sample error is zero, and the generalization term which is approximately 1, as $\text{VCdim}\,(H) = m$. The true error is about $0 + 1 = 1$. We get no guarantee on the future points.

Note that we can represent the new points by its distance to the old points.

To get generalization guarantees when our H is the class of linear classifiers over a data representation in some high dimensional space, we use margin based bound instead of vcdim based bound. Namely, if h has margin γ over a sample S, then we can prove that

$$L_P(h) \le L_S(h) + \sqrt{\frac{(\rho/\gamma)^2 + \ln(1/\delta)}{m}}$$

where ρ is the diameter of S. The ρ here ensures that we cannot cheat by scaling S, which will increase ρ and γ at the same rate. Note that this bound depends on the distribution.

The kernel trick: What is the optimization task that we need to solve to find a good linear separator in the space to which S was embedded? We can use the hyperspace w with support vectors which are the points in the sample closest to the halfspaces: $w = \sum \alpha_j f(x_j)$ where f is the embedding. Then

$$\langle w, f(x_i) \rangle = \left\langle \sum_j \alpha_j f(x_j), f(x_i) \right\rangle = \sum_j \alpha_j \left\langle f(x_j), f(x_i) \right\rangle$$

So we can calculate a matrix contains $n \times n$ values: $\langle f(x_j), f(x_i) \rangle$, independent from the dimension we embed the data. This is called kernel matrix. The learning is based only on the kernel matrix, whose entries are $K(x_i, x_j)$. We choose $K : X^2 \to \mathbb{R}$ to reflect our prior knowledge about how likely are x_i, x_j to have the same label.

> Example: Document classification
>
> $$\text{document1} \mapsto (\underset{\substack{\uparrow \\ \text{aardvark}}}{\bullet}, \dots, \underset{\substack{\uparrow \\ \text{chair}}}{\bullet}, \dots, \underset{\substack{\uparrow \\ \text{table}}}{\bullet}, \dots, \underset{\substack{\uparrow \\ \text{zyzzyva}}}{\bullet})$$
>
> This is the vector of whole dictionary and every entry is a count of how often this word occurs in the document. So we represent the document with a bag of word. We can also put a weight on each word according to how informative this word is, like "and", "or" which we don't care about.

RBF kernels for radial basis functions.

k-nearest neighbor learning. k-NN is guaranteed to converge to an optimal classifier for every problem., but the rate of convergence depends on the Lipschitzness of the labelling function. (How likely are two close points to have different labels)

B

Final exam

You should study chapters 1-12, 15, 16 and 20 in the book.

However, you can skip the following subsections:

- The proof of Sauer's lemma in 6.5.1
- The proof of Theorem 7.4 and everything beyond that in Chapter 7.
- Subsection 8.4
- Subsections 9.2, 9.3,
- The proof of Thm 10.2
- 12.1.3, 12.2.2
- 15.2 and everything beyond that in Ch 15
- 16.3
- 20.4, 20.5, 20.6

Index