richardwu.ca

CS 485/685 Course Notes

Machine Learning: Statistical and Computational Foundations

Shai Ben-David • Winter 2019 • University of Waterloo

Last Revision: April 4, 2019

Table of Contents

Abstract

These notes are intended as a resource for myself; past, present, or future students of this course, and anyone interested in the material. The goal is to provide an end-to-end resource that covers all material discussed in the course displayed in an organized manner. These notes are my interpretation and transcription of the content covered in lectures. The instructor has not verified or confirmed the accuracy of these notes, and any discrepancies, misunderstandings, typos, etc. as these notes relate to course's content is not the responsibility of the instructor. If you spot any errors or would like to contribute, please contact me directly.

1 January 8, 2019

1.1 What is machine learning?

In machine learning, we aim to construct a program that takes as input **experiences** and produces as output **expertise**, or what we have learned from the experience.
We can then apply the **expertise** to produce useful programs such as a spam filter.
An example of learning in nature is **bait shyness**: rats who become sick from eating poisoned bait will become more cautious of food of similar characteristic in the future. Since rats will become more cautious of bait in the future, a delayed poison mechanism (rat is poisoned only 2 days after consuming the bait) is necessary for effective bait by de-associating poison from the bait.
Another example is an experiment called **pigeon superstition** by Skinner (1947): pigeons are starved in a cage with various objects. At random intervals, food is dispersed to satiate the pigeons. Eventually, each pigeon develops a "superstition": they each associate one arbitrary behaviour (e.g. a specific object or a specific movement) that results in food being dispersed.
On the contrary, Garcia (1996) tried a similar experiment to bait shyness with rats where poisoned and un-poisoned bait were identical in characteristic. Whenever a rat approached poisoned bait, a stimulus (e.g. bell ringing, electric shock) was applied to the rat. Surprisingly, the rats did not associate the arbitrary stimulus to the poisoning. This is contrary to the pigeon superstition: this can be explained by evolution (future generations are those that can become aware of poisonous bait) and the fact that rats have **prior knowledge** that poisoning comes from the bait itself, not some arbitrary stimulus.

1.2 Why do we need machine learning?

We desire machines to perform learning because machines can **process lots of data** and are (generally) **fast**.
We desire machines to *learn* because some tasks are simply to complex to hardcode in (e.g. image recognition). Some tasks we do not fully understand how to solve with hardcoded rules. Furthermore, learning allows adaptivity where the machine can constantly learn from new experiences and inputs.

1.3 Types of machine learning

Supervised and unsupervised Machine learning can be generally classified as either **supervised** or **unsupervised**.

Supervised learning takes labelled examples as experience and tries to re-produce these labels on future examples by learning rules. Spam detection may be supervised learning.

Unsupervised learning does not require labelled training data. Examples of unsupervised learning is outlier detection and clustering.

Semi-supervised learning takes as input both labelled and unlabelled data and sits between supervised and unsupervised.

Reinforcement learning also sits between supervised and unsupervised: the machine knows only the rules of the environment and takes actions until a reward (i.e. label) is produced. The machine then learns to label intermediary actions to the final reward produced in the episode (sequence of actions that resulted in the reward).

Passive and active We can also distinguish between **passive** and **active** learning: the former simply takes observed data whereas the latter involves actively performing experiments and interpreting the consequences of the experiments.

Teacher Machine learning can be guided by a "teacher" i.e. how the random sample used as input is generated. Teachers may be **indifferent**, **helpful** or **adversarial**. Helpful teachers produce hints and try to guide the program in the right direction whereas adversarial tries to fool the program.

Batch and online **Batch** learning is learning from a relatively large corpus of data before producing expertise. In contrast **online** learning requires the program to learn as experience is streamed and may result in more mistakes being made.

2 January 10, 2019

2.1 Components of a model

For example sakes, suppose we observe a number of papayas and assign them a score $\in [0, 1]$ for color and hardness. We then label each one as either tasty or not tasty. Using our observations, we would like to predict in the future the tastiness of papayas based on their color and hardness score.

Input The **input** to our learner consists of three parts:

Domain set (X) It is the set of our explanatory variates, in this case $[0, 1] \times [0, 1]$ corresponding to the color score and the hardness score.

Label set (Y) It is the set of our labels: tasty and not tasty.

Training set Our observations i.e. $\{(x_1, y_1), \ldots, (x_m, y_m)\} \subset X \times Y$

Output The **output** of our learner is a **prediction rule** $h : X \to Y$ i.e. the function we learn that maps our papaya scores to a label.

Simple generating model There exists some underlying (unknown) generating process of the population we are interested in (papayas). There is some unknown **probability distribution** D **over** X and some unknown **labelling rule** $f : X \to Y$.

Together (D, f) describes the generation of papayas.

Success measure We define some metric to measure how well our learner learns the underlying generating model. For example

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

We would like to minimize $L_{(D,f)}(h)$ to find the optimal learner h.

2.2 Empirical Risk Minimization (ERM)

As a first strategy, a learner can employ **Empirical Risk Minimization (ERM)** whereby it picks an h that minimize the errors on the *training sample*.
There is however an issue with this strategy: suppose we learn a rule where we label tasty for papayas with scores that exactly match our tasty papayas' scores and not tasty for everything else. That is

$$h_S(x) = \begin{cases} \text{tasty} & \text{if } (x, \text{tasty}) \in S \\ \text{not tasty} & \text{otherwise} \end{cases}$$

We define the **empirical loss (risk)** over a sample $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$

$$L_S(h) = \frac{|\{i \mid h(x_i) \neq y_i\}|}{m}$$

Note the above strategy give us exactly zero empirical error on our sample set S for any generating process but is obviously not a very robust strategy as it **overfits** to our sample.
Suppose the generating process is such that D is the uniform distribution over $[0,1] \times [0,1]$ and let

$$f(x_1, x_2) = \begin{cases} \text{tasty} & \text{if both coordinates in } [0,1,0.9] \\ \text{not tasty} & \text{otherwise} \end{cases}$$

Note that the empirical risk is $L_S(h) = 0$, but the risk on our population is $L_{(D,f)}(h) = (0.8)^2 = 0.64$ (since we would be predicting incorrectly for infinitely many points in the region $[0.1, 0.9] \times [0.1, 0.9]$).

2.3 Introducing prior knowledge with inductive bias

For the papaya example above, we could incorporate some prior knowledge such that tasty papayas belong in some rectangular region of color and hardness scores (which we must learn). We could have also assumed the tasty papayas belong in some linear halfspace, or some arbitrary region that we can learn.
More formally, prior knowledge are a set of rules H (set of functions from X to Y) assumed by the learner to contain a good predictor: H is a **hypothesis class**.
Reformulating our previous ERM strategy, ERM_H picks $h \in H$ that minimizes empirical risk over the training set, that is we pick h^* where

$$h^* \in \text{argmin}_{h \in H} L_S(h)$$

Under the following assumptions ERM_H has good success guarantees:

Assumption 1 (Realizability) $\exists h \in H$ such that $L_{(D,f)}(h) = 0$

Assumption 2 S is picked iid by D and labelled by f (i.e. our sample is representative)

3 January 15, 2019

3.1 Finite hypothesis classes

Theorem 3.1. Let H be a **finite set** of predictors (our hypothesis class). Assume our two assumptions from above hold. Then every ERM_H learning rule is guaranteed to converge to a zero-loss predictor as the sample size tends to infinity.

Namely for every ERM_H learner A and every $\epsilon > 0$

$$\Pr_{S \sim X \times f}\left[L_{(D,f)}(A(S)) > \epsilon\right] \to 0$$

as $|S| \to \infty$ (this is exactly convergence in probability where $\Pr\left[L_{(D,f)}(A(S))\right] \to 0$).

Proof. Let B_ϵ denote the set of all hypotheses in H that have error $> \epsilon$ i.e.

$$B_\epsilon = \{h \in H \mid L_{(D,f)}(h) > \epsilon\}$$

Let our set of "misleading" samples be

$$M = \{S \mid |S| = m \text{ and } \exists h \in B_\epsilon \text{ s.t. } L_S(h) = 0\}$$

(samples where we have a zero empirical loss but has $> \epsilon$ loss on the true population: misleading because it tricks us that the $h \in B_\epsilon$ re-constructs f with zero error when in fact it does not).
Note that

$$\Pr_S\left[L_{(D,f)}(A(S)) > \epsilon\right] \le \Pr\left[S \in M\right]$$

that is the probability that our sample does not perform better than ϵ on our true population is bounded by the probability of picking a misleading sample (this is not just an equality since we also have samples S where $L_{(D,f)}(A(S)) > \epsilon$ and $L_S(A) > 0$).

Lemma 3.1. We claim

$$\Pr_{|S|=m}\left[S \in M\right] \le |H|(1-\epsilon)^m$$

Proof. Consider any $h \in B$ (where obviously $h \neq f$). There exists a "disagreement" region D where for any x, $h(x) \neq f(x)$ (either $h(x) = +, f(x) = -$ or $h(x) = -, f(x) = +$).
For our sample S of size m, we know that $S \subseteq D^c$ (our sample cannot be in the disagreement region since $L_S(h) = 0$ i.e. our sample is perfect; empirical loss is zero so it must agree with f).
Note that since $L_s(h) > \epsilon$ (i.e. h disagrees with f on a region of proportion at least ϵ, our D), then the region where h and f agree is at most of proportion $1 - \epsilon$ (i.e. D^c).
Choose m sample points iid from D^c is thus

$$\Pr_S\left[L_S(h) = 0\right] \le (1-\epsilon)^m$$

□

Lemma 3.2 (Union bound)**.** Given two set of events A, B we know $P(A \cup B) \le P(A) + P(B)$.

Note that $\Pr[S \in M] = \Pr[\text{for some } h \in B, L_S(h) = 0]$ is the union of all misleading hypotheses $h \in B$, thus

$$\begin{aligned}\Pr\left[\text{for some } h \in B, L_S(h) = 0\right] &\le \sum_{h \in B} \Pr(L_S(h) = 0) \\ &= |B|(1-\epsilon)^m \\ &< |H|(1-\epsilon)^m\end{aligned}$$

Note that $1 - \epsilon \le e^{-\epsilon}$ thus $\Pr[S \in M] \le |H|e^{-\epsilon m}$ which goes to 0 as $m \to \infty$ as desired. □

4 January 17, 2019

4.1 Probably Approximately Correct (PAC) learning

In our previous theorem with ERM_H with inductive bias we showed it could do well but only under **strong assumptions**.
Our goal is to prove similar guarantees but with more realistic/relaxed assumptions. Specifically, we would like to relax our assumption that there exists a *deterministic* f that generates the true distribution D (the labels Y) over domain X.
That is, the relax model given domain of instances X and label set Y, the data is generated by a probability distribution D over $X \times Y$. We denote our set of predictors $h : X \to Y$ as H.
Our relaxed model defines the empirical loss as

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

and the true loss over our probability distribution D over $X \times Y$

$$L_D(h) = \Pr_{(x,y)\sim D}[h(x) \neq y] = D(\{(x,y) \mid h(x) \neq y\})$$

Definition 4.1 (PAC learnable (Leslie Valiant 1984))**.** A *hypothesis class* H is **PAC learnable** if there exists a function $m_H(\epsilon, \delta) : (0,1) \times (0,1) \to \mathbb{N}$ and a learner A (map from labelled samples to functions $h : X \to Y$) such that for every $\epsilon, \delta \in (0,1)$ for every distribution D over X and every labelling function $f \in H$, if $m' \geq m(\epsilon, \delta)$ and a labelled sample $S = \{(x_1, f(x_1)), \ldots, (x_m, f(x'_m))\}$ generated iid according to D and labelled by f, then

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

That is: the error is bounded by ϵ (approximately), and the probability of error is bounded by δ (probably) for some large enough sample size.

Remark 4.1. The number of required samples is determined regardless of D and f.

Some weaknesses of this definition:

- Realizability assumption ($h \in H$ such that $L_{(D,f)}(h) = 0$): the learner has strong prior knowledge.
- The labelling rule is *deterministic*: the label of any x is fully determined by X.
- The training distribution and test distribution are the sample: this may be unobtainable in some cases.

4.2 Finite hypothesis H is PAC learnable

Theorem 4.1. Any finite H is PAC learnable.

Proof. Recall if H is finite then

$$\Pr_{S\sim(D^m,f)} \left[L_{(D,f)}(A(S)) > \epsilon\right] < |H|e^{-m\epsilon}$$

Our claim holds if $|H|e^{-m\epsilon} \leq \epsilon$. Solving for m

$$m \geq \frac{\ln(|H|) + \ln\left(\frac{1}{\delta}\right)}{\epsilon}$$

□

4.3 Real intervals on real domain is PAC learnable

Theorem 4.2. Let $X = \mathbb{R}$ and H is the class of all real intervals where $H_{int} = \{h_{(a,b)} \mid a \leq b\}$ where

$$h_{(a,b)}(x) = \begin{cases} 1 & x \in [a,b] \\ 0 & \text{otherwise} \end{cases}$$

then H_{int} is PAC learnable.

Proof. Suppose we have a sample generated by $f \in H_{int}$ where all our positive examples and only positive examples lie within an interval of $X = \mathbb{R}$ (since $f \in H_{int}$ so it labels positive examples only in a real interval), e.g.

$$\ldots \quad 0 \quad 0 \quad 1 \quad 1 \qquad 1 \quad \ldots$$

where $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$. Our learner A could be such that $A(S) \in H_{int}$ where

$$a(S) = \min\{x_i \mid (x_i, 1) \in S\}$$
$$b(S) = \max\{x_i \mid (x_i, 1) \in S\}$$

We show that this rule A is a successful PAC learner for H_{int}.
Given ϵ, m let us upper bound the probability that an m-size sample will lead A to out h with $> \epsilon$ error.
Denote $B_\epsilon = \{h \in H \mid L_{(D,f)}(h) > \epsilon\}$ (bad hypotheses) and $M = \{S \mid A(S) \in B\}$ (set of misleading samples).
Note that a sample $S \in M$ is misleading if our minimum and maximum positive samples cover a "small" region of the actual interval specified $f \in H_{int}$ our arbitrary labelling function.
That is

$$\Pr(S \in M) = \Pr\big(S \text{ does not hit intervals } [\min(f), min(h)] \text{ or } [\max(h), \max(f)]\big)$$

S is a misleading sample only if S does not hit either the interval from $\min(f)$ to $\min(f) + weight_D(\epsilon/2)$ or interval form $\max(f) - weight_D(\epsilon/2)$ to $\max(f)$ (where $weight_D(\epsilon/2)$ is defined as the region R immediately to the right/left where $\Pr_D(x \in R) = \epsilon/2$).
That is

$$\Pr(S \in M) \leq \big(1 - \frac{\epsilon}{2}\big)^m + \big(1 - \frac{\epsilon}{2}\big)^m$$

where each sample misses both intervals of weight/probability $\frac{\epsilon}{2}$.
Therefore

$$\Pr_{S \sim (D^m, f)} \big[L_{(D,f)}(A(S)) > \epsilon\big] \leq 2\big(1 - \frac{\epsilon}{2}\big)^m$$

for some $m(\epsilon, \delta)$ such that $2\big(1 - \frac{\epsilon}{2}\big)^m < \delta$, thus H_{int} is PAC learnable. □

5 January 22, 2019

5.1 Agnostic PAC learning (more general learning model)

Our previous definition of PAC learnable is still too unrealistic. Namely we are going to:

1. Remove the *realizability* assumption: we do not require there exist $h \in H$ such that $L_D(h) = 0$ (although we do not necessarily remove the requirement that the labelling rule is in H)

2. Remove the deterministic labelling requirement (allow smae x to show up with different labels)

We then assume there exists some probability distribution D over "abstract" set Z whereby data is generated iid.
Our new notion of loss is we are given some function $l(\text{hypothesis}, z \in Z)$ which is real-valued.
For example we may have for email spam detection:

Example 5.1. Let $Z = X \times \{0,1\}$ where X is the set of emails and $0/1$ denotes not spam or spam.
Let $h : X \to \{0,1\}$ (our hypothesis function).
Let

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

Given ta training sample $S = (z_1, \ldots, z_m)$ and a predictor (hypothesis function) h the **empirical loss** of h on S is now

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m} l(h, z_i)$$

Also we define the **true loss** of h as (for true distribution D over Z)

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

Remark 5.1. In our example with $Z = X \times \{0,1\}$ and $l_{0,1}$ our new definition of empirical and true loss $L_S(h)$ and $L_D(h)$ are equivalent to our definitions in PAC learnable.

However our individual loss function could be arbitrarily defined:

Example 5.2. Suppose we want to predict tomorrow's temperature from today's measurements.
Let $Z = \big(\text{today's measurements} \times \text{tomorrow's temp}\big)$.
Let $h : \text{today's measurements} \to [-50, +50]$.
We define loss as $l(h,(x,y)) = |h(x) - y|$ (L1 norm).

Example 5.3 (K-means clustering)**.** Suppose we would like to pick k locations for a chain of stores in KW.
Each h will represent a set of k potential locations $h = (\mu_1, \ldots, \mu_k)$.
Therefore we let Z = location of customers seeking a store and we define our loss to be

$$l((\mu_1, \ldots, \mu_k), z) = \min_{1\leq i\leq k} |z - \mu_i|$$

i.e. the loss is the L1 distance between a customer z and the closest location μ_i.
Our training data would then be a sample $S = (z_1, \ldots, z_m)$ which is a record of past customers.

We will now explore how learning is achieved under this more general model.
The prior knowledge of the learner is again modeled by a set H of possible predictors (class of hypotheses). Our **input** is a training set $S = (z_1, \ldots, z_m)$ generated iid by some unknown D over Z. The **output** of the learner is a predictor h.
Our goal is the minimize the *true loss* $L_D(h)$.

Remark 5.2. If we know the distribution D over Z then we could solve our problem without learning, but of course all we know is S.

Definition 5.1 (Agnostic PAC learnable)**.** A class H is **agnostic PAC learnable** if $\exists m : (0,1) \times (0,1) \to \mathbb{N}$ and a learner A (mapping S's to h's) such that $\forall\epsilon\forall\delta$, for all distribution D over Z, and for all $m \geq m(\epsilon, \delta)$ we have

$$\Pr_{S\sim D^m}\left[L_D(A(S)) \geq \min_{h\in H} L_D(h) + \epsilon\right] < \delta$$

Remark 5.3. Agnostic PAC learnable is almost identical to PAC learnable except we do not assume a lower bound of 0 on $L_D(A(S))$: instead we lower bound it with $\min_{h\in H} L_D(h)$ plus some small ϵ.
Furthermore we no longer assume a determinisitic f and instead describe a D over Z.

Remark 5.4. More correctly, we require only $\inf_{h\in H} L_D(h)$: we need not require an attainable minimum.

How can we learn in this new model? In many cases ERM_H is still a good strategy.

6 January 24, 2019

6.1 Uniform Convergence Property

Definition 6.1 (ϵ-representative)**.** A sample $S = (z_1, \ldots, z_m)$ is **ϵ-representative of** H with respect to a distribution D if for any $h \in H$ we have $|L_S(h) - L_D(h)| < \epsilon$.

Remark 6.1. Sample S need not be sampled from distribution D: they are independent under the ϵ-representative definition.

Claim. If S is $\frac{\epsilon}{2}$-representative of H wrt D then for any ERM_H learner A

$$L_D(A(S)) \leq \min_{h\in H} L_D(h) + \epsilon$$

This is the **Uniform Convergence Property**.

Proof. Note that for any $h \in H$

$$\begin{aligned} L_D(A(S)) &\leq L_S(A(S)) + \frac{\epsilon}{2} && \frac{\epsilon}{2} - \text{representative} \\ &\leq L_S(h) + \frac{\epsilon}{2} && A(S) \text{ is } ERM_H \\ &\leq L_D(h) + \epsilon && \frac{\epsilon}{2} - \text{representative} \end{aligned}$$

sine this holds for any h then $L_D(A(S)) \leq \min_{h\in H} L_D(h) + \epsilon$. □

6.2 Finite hypothesis classes are agnostic PAC learnable

Claim. Given a **finite** hypothesis class H, H is agnostic PAC learnable.

Question 6.1. For a given distribution D and **finite** hypothesis class H, how do we determine m large enough such that

$$L_D(A(S)) \leq \min_{h\in H} L_D(h) + \epsilon$$

i.e. such that H is agnostic PAC learnable?

Proof. Note that

$$\begin{aligned} & P_{S\sim D^m}\left[\forall h \in H \text{ s.t. } |L_S(h) - L_D(h)| \leq \frac{\epsilon}{2}\right] > 1 - \delta \\ =& D^m\left[\{S \mid \exists h \in H \text{ s.t. } |L_S(h) - L_D(h)| > \frac{\epsilon}{2}\}\right] < \delta \\ \iff & \bigcup_{h\in H} D^m\left[\{S \mid h \in H \text{ s.t. } |L_S(h) - L_D(h)| > \frac{\epsilon}{2}\}\right] < \delta \end{aligned}$$

where $D^m\big[\{S\}\big]$ is the total probability mass of $\{S\}$ in D^m.
Recall $L_S(h) = \frac{1}{m}\sum_{i=1}^m l(h, z_i)$ and $L_D(h) = \mathbb{E}_{Z\sim D}\big[l(h,z)\big]$. Let $\theta_i = l(h, z_i)$ and let $L_D(h) = \mu$, thus $\mathbb{E}(\theta_i) = \mu$.
Note that on the LHS we have

$$\begin{aligned}
&\bigcup_{h\in H} D^m\left[\{S \mid h \in H \text{ s.t. } |L_S(h) - L_D(h)| > \frac{\epsilon}{2}\}\right] \\
\le &\sum_{h\in H} D^m\left[\{S \mid h \in H \text{ s.t. } |L_S(h) - L_D(h)| > \frac{\epsilon}{2}\}\right] \\
\le &\sum_{h\in H} P\left[\left|\frac{1}{m}\sum \theta_i - \mu\right| > \frac{\epsilon}{2}\right] \\
\Rightarrow &2|H|\exp\left(-2m\left(\frac{\epsilon}{2}\right)^2\right) < \delta
\end{aligned}$$

where the second last inequality follows from **Hoeffding's inequality** (assuming $l \in [0,1]$):

Theorem 6.1 (Hoeffding's inequality)**.** Let $\theta_1, \dots, \theta_n$ be random variables where $\mathbb{E}(\theta_i) = \mu$ and $a \le \theta_i \le b$. Then

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

Solving for m we get

$$m \ge \frac{2\log\left(\frac{2|H|}{\delta}\right)}{\epsilon^2}$$

□

Since there exists such a function $m(\epsilon, \delta)$ a finite hypothesis class H is agnostic PAC learnable.

7 January 31, 2019

7.1 Minimal sample size for the class of all functions

Theorem 7.1. If X has size $2m$ then we need $\ge d$ examples to learn the hypothesis class of *all functions* over $X \times \{0,1\}$ to an accuracy of $\frac{1}{4}$ with $\delta \le \frac{1}{4}$.

Proof. Let X be our domain and D a *uniform distribution* over X.
Choose an $f : X \to \{0,1\}$ to label our points. Our learner's input is $\{(x_1, f(x_1)), \dots, (x_m, f(x_m))\}$.
Suppose $m < \frac{|X|}{2}$ i.e. our sample is less than half the size of our domain.
We then have a region $X \setminus S$ where any new point x has probability $\ge \frac{1}{2}$ of being in $X \setminus S$ which our learner is impartial to labelling as either 0 or 1 (since we have no information from our sample).
Furthermore since we have an arbitrary labelling function f, then for every $x \notin S$ we have

$$\Pr\big[A(S) \ne f(x)\big] = \frac{1}{2}$$

or $\Pr_{(D,f)}\big[A(s) \ne f(x)\big] \ge \frac{1}{4}$. □

7.2 No Free Lunch Theorem

Theorem 7.2 (No Free Lunch Theorem)**.** Let A be any learning algorithm for the task of binary classification $(0-1$ loss) over a domain X. Let the sample size m be any number smaller than $\frac{|X|}{2}$. Then there exists a distribution D over $X \times \{0,1\}$ such that:

1. There exists a function $f : X \to \{0,1\}$ with $L_D(f) = 0$
2. With probability of at least $\frac{1}{7}$ over the choice of $S \sim D^m$ we have $L_D(A(S)) \geq \frac{1}{8}$

i.e. this this theorem states that there exists a task for any learner A that it fails on, but which there is another learner that can successfully learn it. A trivial successful ERM learner would be one with $H = \{f\}$ or more generally an ERM with finite hypothesis class whose size satisfies $m \geq 8\log\left(\frac{7|H|}{6}\right)$.

Proof. Let $C \subseteq X$ be of size $2m$.
The intuition is that any learner that has observed only half of the instances of C has no information regarding the labels in the rest of C. Therefore there exists some "reality" i.e. some target function f that would always contradict labels assigned by $A(S)$ on unobserved instances.
Note that there are $T = 2^{2m}$ possible functions/labellings from C to $\{0,1\}$. Denote these functions as $f_1, \ldots, f_T$.
Let D_i be distributions over $C \times \{0,1\}$ where

$$D_i((x,y)) = \begin{cases} \frac{1}{|C|} & \text{if } y = f_i(x) \\ 0 & \text{otherwise} \end{cases}$$

That is $L_{D_i}(f_i) = 0$ (wrong labels have probability 0).
We show that for any algorithm A with sample size m it holds that

$$\max_{i\in[T]} \mathbb{E}_{S\sim D_i^m}\left[L_{D_i}(A(S))\right] \geq \frac{1}{4}$$

that is there exists some $f : X \to \{0,1\}$ and distribution D where $L_D(f) = 0$ and

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

if the above holds then our claim holds.
Note that there are $k = (2m)^m$ possible sequences of m examples from C. Denote these sequences as $S_1, \ldots, S_k$.
Also if $S_j = (x_1, \ldots, x_m)$ then we denote $S_j^i = \{(x_1, f_i(x_1)), \ldots, (x_m, f_i(x_m))\}$ (sample labelled by f_i).
If the distribution is D_i then A can receive training sets $S_1^i, \ldots, S_k^i$. Note that all these samples have equal probability of being sampled (because of D_i's definition) thus

$$\mathbb{E}_{S\sim D_i^m}\left[L_{D_i}(A(S))\right] = \frac{1}{k}\sum_{j=1}^{k} L_{D_i}(A(S_j^i))$$

Using the fact that maximums $\geq$ averages $\geq$ minimums we have

$$\begin{aligned}\max_{i\in[T]} \frac{1}{k}\sum_{j=1}^{k} L_{D_i}(A(S_j^i)) &\geq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{k}\sum_{j=1}^{k} L_{D_i}(A(S_j^i)) \\ &= \frac{1}{k}\sum_{j=1}^{k}\frac{1}{T}\sum_{i=1}^{T} L_{D_i}(A(S_j^i)) \\ &\geq \min_{j\in[k]}\frac{1}{T}\sum_{i=1}^{T} L_{D_i}(A(S_j^i))\end{aligned}$$

Let us fix some $j \in [k]$ (fix some sample). Let $S_j = (x_1, \dots, x_m)$ and let $v_1, \dots, v_p$ be the examples in $C \setminus S_j$. Note that $p \geq m$ (since we only have half of C). Therefore for every $h : C \to \{0,1\}$ and every i

$$\begin{aligned}L_{D_i}(h) &= \frac{1}{2m}\sum_{x\in C} \mathbb{1}_{[h(x)\neq f_i(x)]} \\ &\geq \frac{1}{2m}\sum_{r=1}^{p} \mathbb{1}_{[h(v_r)\neq f_i(v_r)]} \\ &\geq \frac{1}{2p}\sum_{r=1}^{p} \mathbb{1}_{[h(v_r)\neq f_i(v_r)]}\end{aligned}$$

Therefore we have

$$\begin{aligned}\frac{1}{T}\sum_{i=1}^{T} L_{D_i}(A(S_j^i)) &\geq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{2p}\sum_{r=1}^{p} \mathbb{1}_{[A(S_j^i)(v_r)\neq f_i(v_r)]} \\ &= \frac{1}{2p}\sum_{r=1}^{p}\frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{[A(S_j^i)(v_r)\neq f_i(v_r)]} \\ &\geq \frac{1}{2}\min_{r\in[p]}\frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{[A(S_j^i)(v_r)\neq f_i(v_r)]}\end{aligned}$$

For any $r \in [p]$ (any point not in our sample) we can partition $f_1, \dots, f_T$ into $T/2$ disjoint pairs $(f_i, f_{i'})$ such that for $c \in C$ $f_i(c) \neq f_{i'}(c)$ if and only if $c = v_r$ (they only differ labelling on one point v_r). Since for this pair we must have $S_j^i = S_j^{i'}$ (points in sample must all be the same) then

$$\mathbb{1}_{[A(S_j^i)(v_r)\neq f_i(v_r)]} + \mathbb{1}_{[A(S_j^{i'})(v_r)\neq f_{i'}(v_r)]} = 1$$

(at most one is labelled incorrectly by A), thus we have

$$\frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{[A(S_j^i)(v_r)\neq f_i(v_r)]} = \frac{1}{2}$$

thus combining everything our claim holds. □

8 February 5, 2019

8.1 Summary of PAC learnability

We have proved PAC learnabiliy for a number of hypothesis classes, namely:

- Every finite H
- The set of intervals on $\mathbb{R}$

We also note that every ERM_H is a *good PAC learner*.
Some classes we've seen that are PAC *unlearnable*:

- If X has size $\geq 2d$ then we need $\geq d$ examples to learn the class of *all functions* to accuracy $\frac{1}{4}$ with $\delta \leq \frac{1}{4}$

8.2 Infinite domain on class of all functions

A corollary to our theorem before regarding sample sizes $< \frac{|X|}{2}$:

Corollary 8.1. If X is infinite then the class of all functions (from X to $\{0,1\}$) is *not PAC learnable.*
That is, without inductive bias (prior knowledge) we cannot learn from an infinite domain.

Proof. Assume for contradiction that H_{All} is PAC learnable.
Namely $\exists$ learner A and $m(\epsilon,\delta)$ such that for all D over $X \times \{0,1\}$ and $\forall \epsilon, \delta > 0$ on sample size $\geq m(\epsilon,\delta)$ we have

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

Consider the number $m(0.1, 0.1)$. Pick a $W \subseteq X$ of size $> 2m(0.1,0.1)$.
Pick D to be uniform over W where for all $x \in X$ we have

$$D(x) = \begin{cases} \frac{1}{|W|} & \text{if } x \in W \\ 0 & \text{if } x \notin W \end{cases}$$

Since H contains every function over W from our theorem above we require $> \frac{|W|}{2}$ for $\epsilon \leq \frac{1}{8}, \delta \leq \frac{1}{7}$, but we promised that $m(0.1,0.1)$ should suffice, thus we have a contradiction. □

8.3 Shattering

So when exactly does ERM_H succeed? We saw that $H^{\mathbb{R}}_{intervals}$ has a good ERM_H learner but we also saw that $H^{\mathbb{R}}_{finite}$ where

$$H^{\mathbb{R}}_{finite} = \{f : \mathbb{R} \to \{0,1\} \mid f^{-1}(1) \text{ is finite}\}$$

would not succeed.
In 1970 Vapnik-Chervonenkis and in 1989 EBHW both measured teh compleity of class H that fully determines whether H is learnable.
We begin with a few definitions:

Definition 8.1 (Shattering)**.** A class of functions H (from X to $\{0,1\}$) **shatters** $W \subseteq X$ if **for every** $f : W \to \{0,1\}$ there is some $h \in H$ such that for every $x \in W$: $h(x) = f(x)$.
That is: for any possible labelling of X (which we have $2^{|X|}$) there exists some $h \in H$ that can produce the same labelling.

Example 8.1. Let $X = \mathbb{R}$ and $W = \{a, b, c\}$ where $a < b < c$.
Does $H_{intervals}$ shatter W? **No**. Consider

$$f(x) = \begin{cases} 1 & \text{if } x \in \{a, b\} \\ 0 & \text{if } x = c \end{cases}$$

Does H_{finite} shatter W? **Yes**, since for every labelling function over $\{a, b, c\}$ (we have 2^3 such labelling functions) there exists $h \in H_{finite}$ that corresponds to every such labelling function.

Example 8.2. Let $X = [0, 1]^2$ (unit square). Let $W = \{(0.2, 0.2), (0.3, 0.3), (0.4, 0.1)\}$.
Does $H_{rectangles}$ shatter W? Yes (we can clearly see we can draw rectangles around any subset of the points). However if W are three collinear points e.g. $\{(0.1, 0.1), (0.2, 0.2), (0.3, 0.3)\}$ then the corrsponding labelling $1 - 0 - 1$ would not be shatter-able by $H_{rectangles}$.

9 February 7, 2019

9.1 V-C dimension

Definition 9.1 (V-C dimension)**.** Given a class H the **V-C dimension of** H is the size of the **largest set** that H shatters, that is

$$VC(H) = \max_{|A|}\{H \text{ shatters } A\}$$

It is ∞ if H shatters aribtrarily large W's.

Remark 9.1. We can represent functions $f : X \to \{0, 1\}$ as subsets of X where for a given function f we have

$$S_f = \{x \in X \mid f(x) = 1\}$$

Similarly the converse holds: given $B \subseteq X$ consider

$$f_B(x) = \begin{cases} 1 & \text{if } x \in B \\ 0 & \text{if } x \notin B \end{cases}$$

Thus we have a 1-1 correspondence.

Remark 9.2. H shatters A if

$$\{B \mid B \subseteq A\} = \{h \cap A \mid h \in H\}$$

that is: H shatters A if we can produce label every subset of A with 1 (and all else as 0).

Remark 9.3. For the hypothesis class of all functions H_{all} note that $|H_{all}^X| = 2^{|X|}$. Furthermore note that the collection of all subsets of $B \subseteq X$ is $|\{B \mid B \subseteq X\}| = 2^{|X|}$.
Since both sets have the same cardinality and the collection of $B \subseteq X$ is maximal in XS then $VC(H_{all}) = \infty$.

For example, the V-C dimensions of the classes:

- $VC(H_{intervals}^{\mathbb{R}}) = 2$
- $VC(H_{finite}^{\mathbb{R}}) = \infty$
- $VC(H_{rect}^{\mathbb{R}^2}) \geq 4$: note that we can easily draw 4 points which we can shatter with rectangles.

Claim. H_{rect} cannot shatter *any* set of 5 points ($VC(H_{rect}^{\mathbb{R}^2}) \leq 4$).

Proof. Let A be any set of > 4 points. Pick the 4 points in A that are farthest left, right, up and down i.e. let $B = \{top_A, bot_A, left_A, right_A\}$.

Note that we can never have an $h \in H_{rect}$ that *exactly labels* only B since h would pick all of A if h contains B. □

Therefore $VC(H_{rect}^{R^2}) = 4$.

9.2 Hyper-rectangle class

We extend our rectangle examples in $\mathbb{R}^1$ and $\mathbb{R}^2$ to any $\mathbb{R}^d$ for $d \in \mathbb{N}$:

Definition 9.2 (Hyper-rectangle class)**.** The **class of hyper-rectangles** is defined as

$$H_{rect}^{\mathbb{R}^d} = \{h_{\{(a_1,b_1),\dots,(a_d,b_d)\}} \mid a_1, b_1, a_2, b_2, \dots, a_d, b_d \in \mathbb{R}\}$$

For any set of intervals $\{(a_1, b_1), (a_2, b_2), \dots, (a_d, b_d)\}$ (min and max bounds for every dimension d) let

$$h_{\{(a_1,b_1),\dots,(a_d,b_d)\}} = [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_d, b_d]$$

define a **hyper-rectangle** in $\mathbb{R}^d$.
Then for all $X = (x_1, \dots, x_d)$ we have

$$h_{\{(a_1,b_1),\dots,(a_d,b_d)\}}(x_1, \dots, x_d) = \begin{cases} 1 & \text{if for all } i \leq d, a_i \leq x_i \leq d_i \\ 0 & \text{otherwise} \end{cases}$$

Claim. We claim $VC(H_{rect}^{\mathbb{R}^d}) \geq 2d$.

Proof. Let

$$A = \{(1, 0, \dots, 0), (-1, 0, \dots, 0), (0, 1, 0, \dots, 0), (0, -1, 0, \dots, 0), \dots, (0, \dots, 0, 1), (0, \dots, 0, -1)\} \subseteq \mathbb{R}^d$$

where A is the set of one-hot vectors in $\mathbb{R}^d$ and their negations.
More compactly if $l_i^d = (0, \dots, 0, 1, 0, \dots, 0)$ where only the ith dimension of l_i is 1 then

$$A = \{l_i \mid 1 \leq i \leq d\} \cup \{-l_i \mid 1 \leq i \leq d\}$$

Given any $B \subseteq A$ let h_B be the hyper-rectangle $[a_1, b_1] \times \dots \times [a_d, b_d]$ such that for every $1 \leq i \leq d$:

if both l_i and $-l_i \in B$	$a_i = -2$	$b_i = 2$
if $l_i \in B, -l_i \notin B$	$a_i = 0$	$b_i = 2$
if $l_i \notin B, -l_i \in B$	$a_i = -2$	$b_i = 0$
otherwise	$a_i = 0$	$b_i = 0$

Note that the boundaries for dimension i must always include 0 since all points $l_j, -l_j$ where $j \neq i$ have their ith coordinate as 0. □

Claim. We claim $VC(H_{rect}^{\mathbb{R}^d}) \leq 2d$.

Proof. The proof follows similarly from the $\mathbb{R}^2$ case.
Given any $A \subseteq \mathbb{R}^d$ where $|A| > 2d$ we pick subset $B \subseteq A$ where for every $1 \leq i \leq d$, we pick *a point* t_i and s_i with maximum and minimum value, respectively, in the ith coordinate. Note that $|B| = 2d$ so $B \neq A$.
For every rectangle h that includes all members of B we have $A \subseteq h$ (h bounds all of A), so h cannot cut B from A thus h cannot shatter A. □

9.3 Bounding V-C dimension of a class

We notice that for $H_{rect}^{\mathbb{R}^2}$ and $H_{rect}^{\mathbb{R}^d}$ we proved minimum and upper bounds of V-C dimension in the following way:

Minimum bound $\exists A \forall B \subseteq A$ such that $\exists h \in H$ that can cut B.

Then $VC(\cdot) \geq |A|$.

Maximum bound $\forall A \exists B \subseteq A$ such that $\nexists h \in H$ that can cut B.

Then $VC(\cdot) < |A|$.

We provide another example:

Example 9.1. Let $X = \mathbb{N}$ and $H_5^{\mathbb{N}} = \{A \subseteq N \mid |A| = 5\}$ (all functions that mark exactly 5 points as 1).

Claim. $VC(H_5^{\mathbb{N}}) \geq 5$.

Proof. Pick $A = \{6, 7, 8, 9, 10\}$. For any $B \subseteq A$, let

$$h_B = B \cup (5 - |B|) \text{ points above } 10$$

where $|h_B| = 5$. Clearly $h_B \cap A = B$ so h_B cuts B and thus shatters A. □

Claim. $VC(H_5^{\mathbb{N}}) \leq 5$.

Proof. For every A of size > 5 let $B = A$. No members of H include all points in B so A is not shattered. □

It follows $VC(H_5^{\mathbb{N}}) = 5$.

9.4 Size of hypothesis class from V-C dimension

Claim. For every H and every d if $VC(H) \geq d$ then $|H| \geq 2^d$.

Proof. There is some A of size d shattered by H so for every $B \subseteq A$ there is a corresponding $h_B \in H$.
Since $|A| \geq d$, then A has $\geq 2^d$ subsets thus $|H| \geq 2^d$. □

Note that the converse does not hold. Consider the following:

Example 9.2. For example, the hypothesis class of intervals on $\mathbb{R}$ obviously has size ∞ ($|H_{int}| = \infty$) but $VC(H_{int}) = 2$.

10 February 14, 2019

10.1 Fundamental theorem of statistical learning

Theorem 10.1 (Fundamental theorem of statistical learning)**.** The follow statements are equivalent for every class H:

1. H has the uniform convergence property
2. Every ERM_H learner is a successful PAC agnostic PAC learner
3. H is agnostic PAC learnable
4. Every ERM_H is successful PAC learnable for H
5. H is PAC learnable
6. $VC(H)$ is finite

We have shown that $1 \Rightarrow 2$, $2 \Rightarrow 3$, $1 \Rightarrow 4$, $4 \Rightarrow 5$.
To show $5 \Rightarrow 6$ is equivalent to showing:

Claim. If $VC(H) = \infty$ then H is not PAC learnable.

Proof. Proof is basically applying the No Free Lunch Theorem. Recall that the NFL theorem states that if there is a domain subset $W \subseteq X$ of size d such that H *contains all functions from* W *to* $\{0,1\}$ then H shatters W. To PAC learn H to $\delta = \frac{1}{8}, \epsilon = \frac{1}{8}$ we need $\geq \frac{d}{2}$ sample size.
In other words: if H shatters a set of size d then $m_H^{PAC}(\frac{1}{8}, \frac{1}{8}) \geq \frac{d}{2}$.

Corollary 10.1. If $VC(H) = \infty$ then $m_H^{PAC}(\frac{1}{8}, \frac{1}{8})$ is not any finite number (since H shatters an arbitrarily large W).
Therefore H is not PAC learnable.

□

10.2 Shatter function and Sauer's Lemma

We now show $6 \Rightarrow 1$ in order to prove the fundamental theorem of statistical learning holds.

Definition 10.1 (Shatter function)**.** The **shatter function** of a class H is a function $\pi_H : \mathbb{N} \to \mathbb{N}$ defined by

$$\pi_H(m) = \max_{|A|=m} |H_A|$$

where H_A are the functions in H restricted to $A \subseteq X$ i.e. $H_A = \{h_{|A} \mid h \in H\}$ where $h_{|A}$ is the function from A to Y such that for every $x \in A$ we have $h_{|A}(x) = h(x)$.

Some observations about the shatter function π_H:

1. For every m (and every H) note that $\pi_H(m) \leq 2^m$.
2. If $VC(H) \geq m$ then $\pi_H(m) = 2^m$.
3. If $\pi_H(m) < 2^m$ then $VC(H) < m$.

   Note if $\pi_H(m) < 2^m$ then there are no A of size m does H gets all of its behaviours. That is H shatters no set of size m, which implies H shatters no set of size $\geq m$, which implies $VC(H) < m$.

We also require the following lemma and corollary:

Lemma 10.1 (Sauer-Shelah-Perles-Vapnik-Chervonenkis lemma)**.** (AKA **Sauer's or Sauer-Shelah lemma**). For every H and every m

$$\pi_H(m) \leq \sum_{i=0}^{VC(H)} \binom{m}{i} = |\{B \subseteq A \mid |B| \leq d\}|$$

Note that $\binom{m}{i} \leq m^i$.
The RHS is exactly the number of subsets of A of at most size d.

Corollary 10.2. If $VC(H) = d$ then for all m we have $\pi_H(m) \leq m^d$.

Remark 10.1. We have $\pi_H(m)$ where $VC(H) = d$ is bounded by both the functions 2^m and m^d.
For a fixed d we eventually have $m^d << 2^m$.
That is: $\pi_H(m)$ grows exponentially until it reaches $m = VC(H)$, then $\pi_H(m)$ becomes bounded by a polynomial m^d.

Corollary 10.3. The number of linearly separable subsets of m points is at most m^3.

Proof. Consider Hs^2 the set of linear partitions of $\mathbb{R}^2$.
We claim $VC(Hs^2) = 3$. Clearly it is easy to show there exists an arrangemnet of 3 points we can shatter with linear halfspaces.
We omit the proof that Hs^2 cannot shatter any 4 points.
By Sauer's lemma it follows $\pi_{Hs^2}(m) \leq m^3$. □

Let's take $m = 1000$ for example. The number of functions on m points in $\mathbb{R}^2$ to $\{0, 1\}$ is 2^{1000}. By the above corollary we know there can be at most $1000^3 \approx 2^{30}$ linear halfspace functions, a very small fraction of possible functions.

11 February 26, 2019

11.1 Extended Sauer's Lemma

Lemma 11.1 (Extended (Sauer's) Lemma)**.** For every set $A \subseteq X$

$$|H_A| \leq |\{B \subseteq A \mid H \text{ shatters } B\}|$$

Remark 11.1. This Extended Lemma implies the Sauer Lemma since if H shatters B and $VC(H) = d$ then this implies $|B| \leq d$.

Remark 11.2. Note the inequality says that the number of subsets that H cuts is *fewer than* the number of subsets H shatters: this is counterintuitive since shattering a set seems harder than cutting a set.

Example 11.1. Let us look at $H_{intervals}$ and the inequality. Suppose $A = \{1, 2, 3, 4, 5\}$.
Note that

$$H_{intervals,A} = \{h_{|A} \mid h \in H\} = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\},$$
$$\{1, 2, 3\}, \{2, 3, 4\}, \{3, 4, 5\}, \{1, 2, 3, 4\}, \{2, 3, 4, 5\}, \varnothing, \{1, 2, 3, 4, 5\}\} \quad (11.1)$$

where $|H_A| = 16$.
Note that $H_{intervals}$ can shatter any sets of size $0, 1, 2$, thus $|\{B \subseteq A \mid H \text{ shatters } B\}| = \binom{5}{2} + \binom{5}{1} + \binom{5}{0} = 16$.
Therefore the Extended Lemma does indeed hold for $H_{intervals}$.

11.2 VC bound on union of classes

How might we use Sauer's Lemma to characterize the behaviour of classes H?

Question 11.1. Let H_1, H_2 be two classes over X and assume $VC(H_1) = VC(H_2) = d$. Can we bound $VC(H_1 \cup H_2$?
Let m be the size of a set shattered by $H_1 \cup H_2$. Let A be a subset of size m that $H_1 \cup H_2$ shatters. Then

$$|\{h_{|A} \mid h \in H_1\} \cup \{h_{|A} \mid h \in H_2\}| = |\{h_{|A} \mid h \in H_1 \cup H_2\}| = 2^m$$

where the last equality holds since h shatters A.
Note that

$$\begin{aligned}|\{h_{|A} \mid h \in H_1\} \cup \{h_{|A} \mid h \in H_2\}| &\leq |\{h_{|A} \mid h \in H_1\}| + |\{h_{|A} \mid h \in H_2\}| \\ &\overset{Sauer's lemma}{\leq} m^d + m^d\end{aligned}$$

therefore $2^m \leq 2m^d$ or $m \leq 1 + d \log m$ (we can actually show that $m \leq d \log d$).
This implies that m cannot be too large.

11.3 Finite VC implies uniform convergence property

We now use Sauer's Lemma to prove $6 \Rightarrow 1$ of the fundamental theorem of statistical learning.
Recall: H has the **uniform convergence property** if there exists a function $m_H(\epsilon, \delta)$ such that for every distribution D over $X \times \{0,1\}$ and $\epsilon, \delta > 0$, if $m \geq m_H(\epsilon, \delta)$ then

$$\Pr_{S \sim D^m}\left[S \text{ is } \epsilon - \text{representative of } H \text{ wrt } D\right] > 1 - \delta$$

A sample S is ϵ-representative of H wrt D if $\forall h \in H$

$$|L_S(h) - L_D(h)| \leq \epsilon$$

Proof. Idea: Let D be any distribution over $X \times \{0,1\}$ and S a D-sample of size m such that $m >> VC(H)$.
For each $h \in H$, we wish to show that $|L_S(h) - L_D(h)| \leq \epsilon$.
If we fix h, then Hoeffding's Lemma guarantees that

$$\Pr\left[|L_S(h) - L_D(h)| > \epsilon\right] \leq 2e^{-2m\epsilon^2}$$

Thus the probability this will hold for all $h \in H$ is $\leq |H| \cdot 2e^{-2m\epsilon^2}$.
We only care about h's behaviour on S thus we may replace $|H|$ with $|\{h_{|S} \mid h \in H\}| \leq m^d$ by Sauer's Lemma thus we have

$$\Pr\left[|L_S(h) - L_D(h)| > \epsilon\right] \leq 2m^d e^{-2m\epsilon^2}$$

We can simply choose m large enough such that $2m^d e^{-2m\epsilon^2} < \delta$. □

12 February 28, 2019

12.1 Fundamental theorem of PAC learning

Theorem 12.1. H is learnable $\iff VC(H) < \infty$.
Alternatively we have a quantitative version: Let H be any class of finite VC dimension. Let $m_H(\epsilon, \delta)$ be the sample size needed to learn H to accuracy $< \epsilon$ with probability $\geq 1 - \delta$. Then for some constants c_1, c_2 we have for the

realizable PAC setting

$$c_1 \frac{VC(H) + \log\left(\frac{1}{\delta}\right)}{\epsilon} \leq m_H(\epsilon, \delta) \leq c_2 \frac{VC(H) + \log\left(\frac{1}{\delta}\right)}{\epsilon}$$

and for the **agnostic PAC setting**

$$c_1 \frac{VC(H) + \log\left(\frac{1}{\delta}\right)}{\epsilon^2} \leq m_H^A(\epsilon, \delta) \leq c_2 \frac{VC(H) + \log\left(\frac{1}{\delta}\right)}{\epsilon^2}$$

Remark 12.1. As we require a better accuracy ($\epsilon \to 0$) or higher probability ($\delta \to 0$) then we require more samples.

Remark 12.2. As we increase our class complexity/size e.g. adding more variates to our model in H then $VC(H)$ increases requiring a larger sample size.

As we increase the complexity of our class H we have a lower $\min_{h \in H} L_D(h)$, but we require more examples. This is the **bias complexity tradeoff**.
Recall in the NFL theorem we showed (in the realizable case) that $m_H(\epsilon = \frac{1}{8}, \delta = \frac{1}{7}) \geq \frac{d}{2}$ with a *uniform distribution*. How does this bound depend on ϵ, δ in general? How does this bound change for the agnostic setup?

Dependence of ϵ We show the inverse relation with ϵ.

Let $VC(H) = d$ and let $W \subseteq X$ be a set of size d shattered by H.

Define a probability distribution over W as follows: pick some $x_0 \in W$ for every $x \in X$. We let

$$D_\epsilon(x) = \begin{cases} 1 - \epsilon & \text{if } x = x_0 \\ \frac{\epsilon}{d-1} & \text{if } x \in W \setminus \{x_0\} \\ 0 & \text{if } x \notin W \end{cases}$$

By the NFL argument, a sample S that misses $\geq \frac{1}{2}$ of the points in $W \setminus \{x_0\}$ then for every learner and some $h \in H$ the learner will have an expected error $\frac{1}{2}$ on every point in $W \setminus S$ i.e. an expected total error of

$$L_{(D,h)}(A(S)) \geq \frac{1}{4}\epsilon$$

since the total weight on our points in $W \setminus \{x_0\}$ is ϵ.

To get an expected error $< \frac{1}{4}\epsilon$ we require an S that hits $W \setminus \{x_0\}$ at least $\frac{d-1}{2}$ times. A sample of size m is expected to hit $W \setminus \{x_0\}$ $m \cdot \epsilon$ times. Therefore

$$m(\frac{\epsilon}{4}, \cdot) \geq \frac{d-1}{2\epsilon}$$

Agnostic dependence on ϵ We sketch why in the agnostic case we have a $\frac{1}{\epsilon^2}$ relationship.

We make an analogy with flipping coins. Suppose our task to predict heads or tails.

For an unbiased coin $\min_{h \in H} L_D(h) = 0.5$.

However suppose we have a biased coin where either $P(\text{heads}) = \frac{1}{2} + \epsilon, P(\text{tails}) = \frac{1}{2} - \epsilon$ OR $P(\text{heads}) = \frac{1}{2} - \epsilon, P(\text{tails}) = \frac{1}{2} + \epsilon$. In this case $\min_{h \in H} L_D(h) = \frac{1}{2} - \epsilon$.

The best learner is still ERM, and we can show that in order to estimate within ϵ we require more tosses where $m_H^A(\epsilon, \cdot) \propto \frac{1}{\epsilon^2}$.

12.2 Issues with Agnostic PAC

While the agnostic PAC model is well understood it is not practically satisfactory.
Consider a class H of half spaces. We note from agnostic PAC we can guarantee $L_D(A(S)) \leq \min_{\text{half spaces } h} L_D(h)+\epsilon$ with a probability $\geq 1-\delta$.
However, we note that $\min_{\text{half spaces } h} L_D(h)$ may be very high relative to the true error (class with all functions): that is we may have **very high bias** in our class. The weakness of the bound given by PAC learnability is that our error is never below $\min_{\text{half spaces } h} L_D(h)$ with high probability.
How do we overcome this high bias in practice? We might use a class with very large VC dimension i.e. $VC(H)=\infty$ e.g. deep neural networks.

13 March 5, 2019

13.1 Learning half spaces

Keeping in mind what we know about the weakness of (agnostic) PAC learnability, let $X=\mathbb{R}^n$.
What classes of H do we use to learn over $\mathbb{R}^n$? Let us look at Hs^n or half spaces in $\mathbb{R}^n$.
Note that if $h_w \in Hs^n$ then it can be characterized by

$$h_w(x) = \text{sign}(\sum_{i=0}^{n} w_i x_i)$$

where $w=(w_0,\dots,w_n)$ and $x=(x_1,\dots,x_n)$. That is for a given x if $w_1x_1+\ldots+w_nx_n+w_0 \leq 0$ we label it as $-$'ve and if $w_1x_1+\ldots+w_nx_n+w_0 > 0$ we label it as $+$'ve.
We note that $VC(Hs^n)=n+1$.
What about polynomials of degree 2 Pol_2^n: will that allow a smaller error? We have $w_1x_1^2+w_2x_1x_2+\ldots$. We note that it is easy to show $VC(Pol_2^1)=3$ (polynomialso f degree 2 over $\mathbb{R}^1$).

Claim. $VC(Pol_d^1)=d+1$.

Claim. $VC(Pol_d^n)=f(d,n)$ for some function f over d and n.

Question 13.1. What is the VC(polynomials of all degrees over $\mathbb{R}$)?

Answer. It is ∞ since for any subset S of size m we can always find some polynomial that shatters it e.g. Pol_m.

Recall the fundamental theorem of PAC learning: H is PAC learnable if and only if $VC(H)<\infty$: our previous observation highlights a tradeoff between learning higher degree polynomials and achieving a lower lower bound on $L_D(h)$ and ensuring that our class H is still PAC learnable.
Let us define an even *more relax version* of PAC learnability.

13.2 Non-uniform PAC learnability

Definition 13.1 (Non-uniform PAC learnable)**.** A class of predictors H is **non-uniformly PAC learnable** if there exists a learner A and a function $m' : H \times (0,1) \times (0,1) \to \mathbb{N}$ such that for every probability distribution P over $X \times \{0,1\}$, every $\epsilon, \delta > 0$ and **every** $h \in H$, if $m > m'(h,\epsilon,\delta)$ then over any iid P-samples

$$\Pr\left[L_P(A(S)) \leq L_P(h)+\epsilon\right] > 1-\delta$$

Remark 13.1. Compared to our definition of agnostic PAC learnability, we now allow m to change and depend on an $h \in H$ instead of fixing it for $\min_{h\in H} L_P(h)$.

Theorem 13.1. A class H is non-uniformly PAC learnable if and only if there are classes $\{H_n \mid n \in \mathbb{N}\}$ such that

1. $H = \bigcup_{n=1}^{\infty} H_n$
2. Each H_n has a finite VC dimension.

Remark 13.2. This theorem is analgous to the fundamental theorem of PAC learning but we only require H be a union of finite VC dimension classes.

Remark 13.3. $\{H_n\}$ need not be disjoint.

Proof. **Forwards direction** Assume H is non-uniformly PAC learnable and let $m(h, \epsilon, \delta)$ be the function guaranteed by the definition. Pick $\epsilon = \frac{1}{8}, \delta = \frac{1}{8}$ and define for all n

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

clearly $H = \bigcup_n H_n$.

We now show each H_n has finite VC dimension. By the NFL we know that H_n

Reverse direction See below.

□

Corollary 13.1. Every PAC learnable H is also non-uniformly PAC learnable.

Claim. The class of all polynomial threshold functions over $\mathbb{R}$ (Pol^1) is non-uniformly PAC learnable.

Proof. Note that if $h \in Pol^1$ then $h(x) = \text{sign}(p(x))$ for some polynomial p of degree d.
Then

$$Pol^1 = \bigcup_{d \in \mathbb{N}} Pol^1_d$$

where $VC(Pol^1_d) = d + 1$. □

Claim. Let H_{finite} be the class of all functions $f : \mathbb{R} \to \{0, 1\}$ such that $f^{-1}(1)$ is finite.
H_{finite} is non-uniformly PAC learnable.

Remark 13.4. Note that $VC(H_{finite} = \infty$, so H_{finite} is not PAC learnable. On the other hand $H_{finite} = \bigcup_{n \in \mathbb{N}} H_{n-ones}$ where $H_{n-ones} = \{f : f^{-1}(1) \leq n\}$ and $VC(H_{ones}) = n$.

14 March 7, 2019

14.1 Finding ϵ instead of m

Rather than asking "given ϵ, δ, what should m be?" we can ask "given m, δ what accuracy can we promise?"
We define

$$\epsilon_h(m, \delta) = \min\{\epsilon \mid m(h, \epsilon, \delta) \leq m\}$$

With this change of notation, we get that non-uniformly PAC learnability is equivalent to: given a sample size m, with probability $\geq 1 - \delta$, $\forall h \in H$ we have

$$L_P(A(S)) \leq L_P(h) + \epsilon_h(m, \delta)$$

14.2 Non-uniformly PAC learnable theorem reverse proof

Recall our previous theorem:

Theorem 14.1. H is non-uniformly PAC learnable if $\exists H_1, H_2, \dots, H_n, \dots$ such that

1. $H = \bigcup_n H_n$
2. Each H_n has finite VC dimension

Proof. We now prove the reverse direction.
Since each H_n has a finite VC dimension, each H_n has the uniform convergence property: that is for some function $m^{UC}_{H_n}(\epsilon, \delta)$ for every P over $X \times \{0,1\}$ if $m > m^{UC}_{H_n}(\epsilon, \delta)$ then $\forall h \in H_n$ we have $\Pr\left[\Big|L_P(h) - L_S(h)\Big| < \epsilon\right] \geq 1 - \delta$ for $S \sim P^m$.
Thus S is ϵ-representative for H.
Note that uniform convergence guarantees that ERM_H works for large enough sample sizes.
Let $w : \mathbb{N} \to [0,1]$ such that $\sum_{n=0}^{\infty} w(n) \leq 1$ (we choose some weight function for our H_n's such that the total weight is ≤ 1).
Define, for a given $n \in \mathbb{N}$

$$\epsilon_n(m, \delta) = \min\{\epsilon \mid m^{UC}_{H_n}(\epsilon, w(n)\delta) \leq m\}$$

it follows that $\forall h \in H_n$ and for sample sizes of m we have

$$\Pr\left[|L_P(h) - L_S(h)| \leq \epsilon_n\right] \geq 1 - w(n)\delta$$

and so the probability of failure for a given $n \in \mathbb{N}$ and H_n is $< w(n)\delta$.
Thus for all H_n, $n \in \mathbb{N}$ and for all $h \in H_n$ (where $n(h) = \text{argmin}_n\{h \in H_n\}$), we have by union bound

$$\begin{aligned}\Pr\Big[\bigcup_{n \in \mathbb{N}} |L_P(h) - L_S(h)| \geq \epsilon_{n(h)}\Big] &\leq \sum_{n=0}^{\infty} w(n)\delta \\ &\leq \delta \sum_{n=0}^{\infty} w(n) \\ &\leq \delta\end{aligned}$$

□

Remark 14.1. We have just shown that if $H = \bigcup_n H_n$ and each H_n has finite VC dimension then for some function $\epsilon(h, m, \delta)$ for sample size of m we have $\forall h \in H$

$$\Pr\left[L_P(h) - L_S(h) \leq \epsilon(h, m, \delta)\right] \geq 1 - \delta$$

where $\epsilon(h, m, \delta) = \epsilon_{n(h)}(m, w(n)\delta)$ from our notation above.

14.3 Structural risk minimization (SRM)

As an implication of the theorem above, given a sample $S \sim P^m$, how can we pick a good hypothesis h?
By the last result, for all $h \in H$ we have with high probability

$$L_P(h) \leq L_S(h) + \epsilon(h, m)$$

Remark 14.2. Note that to pick an h such that we are minimizing $L_P(h)$, we need to ensure **both** $L_S(h)$ and $\epsilon(h, m)$ are both small.
The $eps(h, m)$ term is the **regularization term**.

This method of minimizing $L_P(h)$ as above is called **Struturcal Risk Minimization (SRM)**.

Example 14.1. As a neural network's size increases, $\epsilon(h, m)$ tends to increase for a given m for each h in the larger neural network.
So we sometimes sacrifice a slightly larger $L_S(h)$ but a lower $\epsilon(h, m)$ be choosing a simpler neural network or function.

14.4 Minimal description length (MDL) principle

We present an example of structural risk minimization.

Definition 14.1 (Description language)**.** A **description language** for a class H is a function $\wedge : H \to$ all finite binary strings.
This function should satisfy the **prefix-free** requirement: for no two $h, \bar{h} \in H$ we have $\hat{h}$ is a prefix of $\hat{\bar{h}}$.

Theorem 14.2 (Kraft inequality)**.** For every prefix-free language (and every H) we have

$$\sum_{h \in H} \frac{1}{2^{|\hat{h}|}} \leq 1$$

Proof. For intuition, consider the prefix tree (as we move down the tree, we append 0 or 1 to our current string):

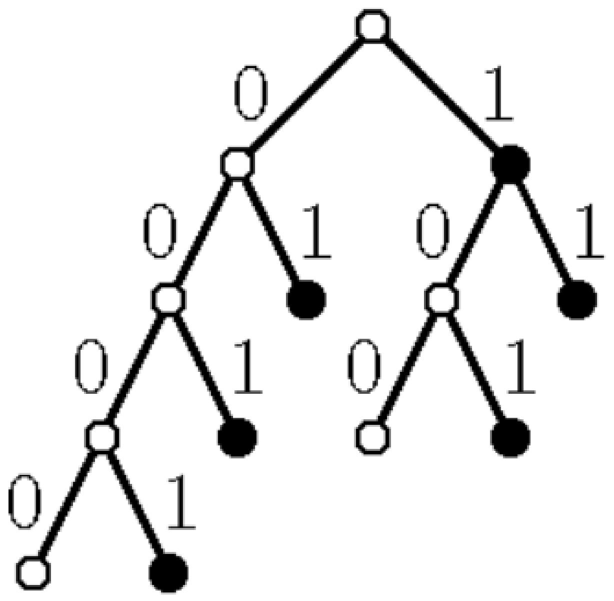

Figure 14.1: Prefix tree: from the root node if we moved left, left and right we get the string 001.

Note that for any level, we have $2^i \times \frac{1}{2^i} = 1$. Furthermore once we choose some string at a given node, all children string in the prefix tree are not allowed since the chosen string would be a prefix of any of the children.

Formally: define a probability distribution over class H as follows: toss an unbiased 0/1 coin until one gets a dsecription string of some h.
It follows that the sum of the probabilities of getting the description string of $h \in H$ is $\sum_{h\in H} \Pr(h) \leq 1$ (law of total probability). Note that it is not exactly $= 1$ since we may end up tossing the coin indefinitely if we never encounter a description match.
Note that for any h we have

$$P(h) = \frac{1}{2^{|\hat{h}|}}$$

by the iid of tossing an unbiased coin. $\square$

Let H be any countable class and $\wedge$ be a description language for H. H is non-uniformly PAC learnable with

$$m(h, \epsilon, \delta) = \frac{|\hat{h}| + \log\left(\frac{1}{\delta}\right)}{\epsilon^2}$$

15 March 12, 2019

15.1 Note on non-uniformly PAC learnability

Recall in the proof for the theorem on non-uniformly PAC learnability that can be guaranteed if we have finite VC dimension H_n's such that $H = \bigcup_n H_n$, we want to pick an h that minimizes

$$L_S(h) + \epsilon_{n(h)}(m, w(n(h))\delta)$$

where $n(h) = \text{argmin}_n\{h \in H_n\}$ and $w : \mathbb{N} \to [0,1]$ such that $\sum_{n=0}^{\infty} w(n) \leq 1$. This is **structural risk minimization (SRM)**.
We can think of $\epsilon_{n(h)}(m, w(n(h))\delta)$ as a penalty term. How do we use this penalty term to our advantage? We note that as $n \to \infty$, then $w(n) \to 0$ (since the sum of $w(n)$ is finite). This will decrease our confidence bound $w(n)\delta$ which requires a larger ϵ and thus a larger penalty term.
We can thus construct our H_n's such that a larger n corresponds to a more "complex" hypothesis class H_n. So while a complex hypothesis h can reduce our empirical risk $L_S(h)$, we can penalize the complexity with the second term. A less complex hypothesis class (smaller n) has a smaller penalization but may have a larger $L_S(h)$.
We thus have some abstract algorithm to penalize complexity.

15.2 SRM using description language

Recall that we cap map each H to a description where $h \to \hat{h}$ ($\hat{h}$ is a binary string). All binary strings are prefix-free and from Kraft's inequality we have $\sum_{h\in H} \frac{1}{2^{|\hat{h}|}} \leq 1$.
Suppose we have some hypothesis class $H = \{h_1, h_2, h_3, \ldots, h_n, \ldots\}$. We define each $H_n = \{h_n\}$ (contains 1 hypothesis). Clearly $VC(H_n) = 0$ (one hypothesis can shatter 0 points). Note that H is non-uniformly PAC learnable.
Let $w(n) = \frac{1}{2^{|\hat{h}_n|}}$. What is $\epsilon_n(m, w(n)\delta)$?

Recall that by definition $\epsilon_n(m,\delta) = \min_\epsilon\{n(\epsilon,\delta) \leq m\}$ and we showed for any hypothesis class

$$\begin{aligned} m(\epsilon_n,\delta) &= \frac{VC(H_n) + \log\left(\frac{1}{w(n)\delta}\right)}{\epsilon_n^2} \\ \Rightarrow \epsilon_n &= \sqrt{\frac{0 - \log w(n) + \log\frac{1}{\delta}}{m}} \\ \Rightarrow \epsilon_n &= \sqrt{\frac{|\hat{h}| + \log\frac{1}{\delta}}{m}} \end{aligned}$$

So the longer the description length a hypothesis h_n has the larger its penalty term ϵ_n.

Remark 15.1. The h we pick to minimize $L_S(h) + \epsilon_n(m, w(n)\delta)$ depends on our **choice of description**, thus we must pick our description before seeing S. That is: the choice of description is where our **prior knowledge** is inserted into our algorithm.

15.3 SRM using prior distribution

Suppose we have some prior distribution over our hypotheses. Let Q be a probability distribution over H. Now define $w(h) = Q(h)$. Since Q is a probability over H we have $\sum_{h\in H} Q(h) = 1$.
Thus we want to pick an h that minimizes

$$L_S(h) + \sqrt{\frac{-\log Q(h) + \log\frac{1}{\delta}}{m}}$$

The quality of our prior knowledge comes through as the fit between the empirical error and the likelihood $Q(h)$ we assigned.

15.4 SRM with finite H

For a finite H we can pick Q such that $Q(h) = \frac{1}{|H|}$. Then we recover our previous bound

$$\epsilon(m,\delta) = \sqrt{\frac{\log|H| + \log\frac{1}{\delta}}{m}}$$

16 March 14, 2019

16.1 Lessons from minimum description length boundb

1. The choice of the description language is up to the learner: different choices results in different algorithms
2. The function $\hat{}: H \to \{0,1\}^n$ defines what will be a "complex" h vs a "simple" h
3. The description language has to be picked before seeing S
4. Occam's Razor: simpler solutions are more likely to be correct than complex ones

16.2 The computational complexity of learning

There are two major resources to consider in machine learning in general:

Information complexity How big should our training data be?

Computational complexity Once we have enough data how much computation is needed to pick h?

We've looked at information complexity, now we look at **computational complexity**.
In usual CS algorithms we measure the computational complexity of a problem as a function of the **input size**. For example sorting for an input of $I = \{x_1, \dots, x_n\}$, we express that it runs in time $f(n)$ if for all inputs I of size $\leq n$ the algorithm gives an answer in $\leq f(n)$ steps.
Contrast the above with algorithms given an input set $S \subseteq X$. The size of S is not a *meaningful parameter* to evaluate the hardness of the task. Rather, our **meaningful parameters are** ϵ, δ**.**

Definition 16.1 (Computational complexity function)**.** A learning task has complexity $f(\epsilon, \delta)$ if for all ϵ, δ it can be solved up to ϵ-accuracy and δ confidence with a learner computing for $\leq f(\epsilon, \delta)$ steps.
For example $f(\epsilon, \delta) \in \text{Polynomial}\big(\frac{1}{\epsilon}, \frac{1}{\delta}\big)$.

Example 16.1. Input: find predictor for temperature at noon tomorrow with $\epsilon = 0.1, \delta = 0.05$.
Output: Collect sample of size 1000 and output h that structurally minimizes $L_S(h) + \epsilon(m, \delta)$.

So in other words, given an input x how much computation is needed to find $h(x)$? Our input is ϵ, δ and our output is some hypothesis h.
A learner thus requires time $f(\epsilon, \delta)$ to learn a task if for every ϵ, δ within $f(\epsilon, \delta)$ steps it outputs a predictor h that meets the (ϵ, δ) requirement and for any input x, $h(x)$ can be computed in time $f(\epsilon, \delta)$.

16.3 Realizable input and ERM_H

For a **realizable input** there are some problems where we can simply use ERM_H.

Example 16.2. Let our hypothesis class H be of rectangles in $\mathbb{R}^2$. As we've previously shown if its realizable then we can use ERM_H efficiently.
Step 1: given a sample S of size m how much computation needed to find h such that $L_S(h) = 0$? We need to find the rightmost, leftmost, topmost and bottommost instance labelled positively in S.
Step 2: what should m be as a function of ϵ, δ? Recall in the realizable PAC learnable case since $VC(H) = 4$ then $m_H(\epsilon, \delta) = \frac{4+\log \frac{1}{\delta}}{\epsilon}$, thus we have

$$f(\epsilon, \delta) \approx 4\left(\frac{4 + \log \frac{1}{\delta}}{\epsilon}\right) \in \text{poly}\left(\frac{1}{\epsilon}, \frac{1}{\delta}\right)$$

16.4 Unrealizable input (agnostic) and ERM_H

What if we ran ERM_H on an unrealizable input?

Example 16.3. Let our hypothesis class H be of rectangles in $\mathbb{R}^2$.
Since we are only interested in the minimum loss over our entire class H, we only need to consider rectangles determined by at most 4 points in S. Thus there are $\approx m^4$ such rectangles. Also for agnostic learnability $m \approx \frac{4+\log \frac{1}{\delta}}{\epsilon^2}$ thus we have

$$f(\epsilon, \delta) \approx \left(\frac{4 + \log \frac{1}{\delta}}{\epsilon^2}\right)^4$$

17 March 19, 2019

17.1 More on computational complexity

We wish to bound the time it takes the learning algorithm to output a hypothesis h and the time to compute h on any X by $f(\epsilon, \delta|X|)$. Ideally f is a polynomial in terms of $\frac{1}{\epsilon}, \frac{1}{\delta}$ and $|X|$ (the dimension i.e. number of features).
Recall that ignoring computational complexity, any learnable class is learnable by an ERM_H algorithm (with no additional sample complexity).
Once computation time is considered, there are cases that non-ERM_H algorithm can run much fast than any ERM_H one.

17.2 CNF classifier and boolean domains

Consider boolean vectors domains e.g. for categorical features we have features that dicrete values "yes" or "no".
Then we have $X = \{0,1\}^n$ where $X = (010110\ldots)$.
A basic class over such boolean domains are Conjunctive Normal Form (CNF) formulas:

Definition 17.1 (CNF formula)**.** Given variables $x_1, \ldots, x_n$ (take on values true/false) and literals $x_i, \ldots, \neg x_j$, then the set of **CNF formulas** are the `AND` of our literals i.e.

$$\phi = x_{i_1} \wedge x_{i_2} \wedge \ldots \wedge \neg x_{i_{k+1}} \wedge \neg x_{i_{k+2}} \wedge \ldots$$

To learn from a class of CNF classifiers:

Input $S = (x_1, y_1), \ldots, (x_m, y_m)$ where for a given example we have $(x_i, y_i) = ((0110\ldots), +)$ (positive label) or $(x_i, y_i) = ((0110\ldots), -)$ (negative label).

Output A CNF formula ϕ that has small error (over the data generating distribution)

Question 17.1. Given $X = \{0,1\}^n$ and $|X| = 2^n$ (n-dimensional domain), what is $VC(CNF_n)$?

Answer. We note that $|CNF_n| \leq 3^n$ (variable x_i can either be in the CNF, negated, or not in the CNF) so

$$VC(CNF_n) \leq n \log 3 \leq 2n$$

Thus given a ϵ, δ, in the *realizable case* a sample of size

$$c \cdot \frac{n + \log \frac{1}{\delta}}{\epsilon}$$

will suffice to learn from CNF_n.
We now need to find an efficient algorithm that takes $S = (x_1, y_1), \ldots, (x_m, y_m)$ and if some CNF ϕ has $L_S(\phi) = 0$ (i.e. realizable) then the out would be such ϕ.
A candidate CNF learning algorithm:

1. Initialize

$$\phi_0 = x_1 \wedge x_2 \wedge \ldots \wedge x_n \wedge \neg x_1 \wedge \neg x_2 \wedge \ldots \wedge \neg x_n$$

   thus for any $x \in X$ we have $\phi_0(x) = 0$ (the above CNF always evaluates to false).

2. At each step t we have a formula ϕ_t and wish to derive ϕ_{t+1}.

   We consider the t-th positively labelled example in S:

$$s_t = ((010110\ldots), +)$$

We obtain ϕ_{t+1} by deleting from ϕ_t every literal that is violated by the example.

3. At any time step t, the first t positively labelled examples should have the correct label, and since we are in the realizable case, all prior negatively labelled examples should remain negatively labelled.

Let p be the number of positive labelled examples in S. Then ϕ_{p+1} would agree with all instances in S i.e. $L_S(\phi_{p+1}) = 0$.
On S of sie m the algorithm terminates in time $O(m \cdot n)$, so the runtime of our algorithm is

$$f(\epsilon, \delta, n) \leq c \cdot \frac{n + \log \frac{1}{\delta}}{\epsilon} \cdot n \approx \frac{n^2 + \log \frac{1}{\delta}}{\epsilon}$$

17.3 Summary of efficient realizable and agnostic learners

In the **realizable case** the following classes have efficient (polynomial time) ERM_H learners:

- Rectangles in $\mathbb{R}^d$
- Halfspaces in $\mathbb{R}^d$
- CNF_n

Remark 17.1. In the **agnostic case** ERM_H learning of each of the above classes is **NP-hard**.

Definition 17.2 (NP-hard). A problem is **NP-hard** if having a polynomial time solution to this problem implies efficient solutions to all problems in the NP class.
We do not know of any such algorithms and believe no such algoithm exists i.e. $NP \neq P$.

17.4 Computationally hard realizable problems

Question 17.2. Are there any learning problems that are computational hard, even in the *realizable case*?

Answer. Yes. For example:

- Neural networks once the depth ≥ 3 and width ≥ 3.
- Three term DNF formulas.

  Definition 17.3 (DNF formula). Given $X = \{0,1\}^n$ ϕ that is in Disjunctive Normal Form (DNF) is in the form

  $$\phi = \phi_1 \vee \phi_2 \vee \phi_3$$

  where each ϕ_i is a CNF formula.

17.5 DNF realizable

ERM_H over 3-term DNF formulas is NP-hard.
However there is an **efficient** learning algorithm that is not an ERM_H learner.
Note that

$$|\text{3-term } DNF_n| \leq \left(|CNF_n|\right)^3 \leq 3^{3n}$$

thus $VC(\text{3-term } DNF_n) \leq 3n \log 3$.

Note that for every $\phi = \phi_1 \vee \phi_2 \vee \phi_3$, every $x_1, \ldots, x_n$ can be re-written as

$$\begin{aligned}&(l_1 \wedge l_2 \wedge \ldots \wedge l_{pq}) \vee (l'_1 \wedge l'_2 \wedge \ldots \wedge l'_{pq}) \vee (l''_1 \wedge l''_2 \wedge \ldots \wedge l''_{pq})\\ \Rightarrow&(l_1 \vee l'_1 \vee l''_1) \wedge (l_2 \vee '_2 \vee l''_2) \wedge \ldots\\ \Rightarrow&y_1 \wedge y_2 \wedge \ldots\end{aligned}$$

Thus we have an CNF formula. We pick a class $H' \in CNF_n$ such that $H' \geq H$ (H' covers our H for DNF) and $VC(H') < \infty$.

18 March 21, 2019

18.1 Learning is the opposite of cryptography

Definition 18.1 (One-way functions)**.** A **one-way function** is e.g. some $f : \mathbb{N} \to \mathbb{N}$ whereby given x it is easy to compute $f(x)$, but given y it is hard to find x such that $f(x) = y$.

For example, as humans $f(x) = x^5$ is a one-way function.
For machines even and for some M, $(k, l) \to (k^l) \mod M$ is in general a hard problem (for particularly k, l, M i.e. RSA).
A **trapdoor function** is one that has a **secret key** that can invert the function. A **lunch break attack** records a sample of pairs $(f(m_i), m_i)$ where m_i is the input. The lunch break attack then learns from the class $H = \{f_{key} \mid \text{ all possible keys}\}$ and f_{key} is a function over all keys of length l.
Since $|H| \leq 2^l$, then $VC(H) \leq l =$ key length. But we know that in cryptography learning such a function is hard: this tells us that learning functions from even finite VC-dimension H canbe hard.

18.2 Basic learning algorithms

In general, realizable ERM and especially non-ERM algorithms are computationally efficient whereas in the agnostic case it is not.

Linear predictors Given $X = \mathbb{R}^n$, $H = \{h_{W,b} \mid w \in \mathbb{R}^n, b \in \mathbb{R}\}$ such that for every x, w, b we have

$$h_{w,b}(x) = \text{sign}(w \cdot x + b)$$

i.e. $h_{w,b}(x) = 1$ if $\sum_{i=1}^n x_i w_i + b_i > 0$ and -1 otherwise.
Some advantages of linear predictors:

1. Fast to compute/infer at test time
2. Interpretable
3. "Easy to learn"

How do we train/learn linear predictors in the *realizable* case? Given $S = \{(\bar{x}_1, y_1), \ldots, (\bar{x}_m, y_m)\}$ where $\bar{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$, we assume there exists some $h_{w,b}$ such that $L_S(h_{w,b}) = 0$. Our goal is to find such w, b.

Method 1: Linear programming Given a set of linear constraints:

$$\begin{aligned}&a^1_1 x_1 + a^1_2 x_2 + \ldots + a^1_n x_n \geq b_1\\ &\vdots\\ &a^m_1 x_1 + a^m_2 x_2 + \ldots + a^m_n x_n \geq b_m\end{aligned}$$

and a linear objective $u_1x_1 + \ldots + u_nx_n$, we find a vertex $(x_1, \ldots, x_m)$ that satisfies the constraints and maximizes the objective.

Geometrically we have some convex hull C in $\mathbb{R}^n$ space defined by our constraints and we want $x \in C$ such that $u \cdot x$ is maximized.

We can thus transform our learning task (given S find $h_{w,b}$) into a linear programming task since we know there exists w, b such that $L_S(h_{w,b}) = 0$: we find $w_1, \ldots, w_n$ and b satisfying $w \cdot x^i + b > 0$ for $(x^i, 1)$ and $w \cdot x^j + b \leq 0$ for $(x^j, -1)$, thus we have:

$$\begin{aligned} &w_1x_1^i + w_2x_2^i + \ldots w_nx_n^i + b > 0 && (x^i, 1) \in S \\ &w_1x_1^j + w_2x_2^j + \ldots w_nx_n^j + b \leq 0 && (x^j, -1) \in S \end{aligned}$$

Note that we have two issues with the current formulation:

1. We have strict inequalities whereas linear programming requires non-strict inequalities. Since we have finitely many points, we can simplify specify that linear link of the $+1$ points are all $\geq$ than the minimum (and vice versa for -1 points).
2. We do not really have an objective function since we only care that the constraints are satisfied.

Perceptrons A perceptron has a layer of n input neurons for each coordinate x_i, $i = 1, \ldots, n$, and m weights w_i for $i = 1, \ldots, n$.
It then takes the linear combination and checks $\sum x_iw_i \geq b$ or $\sum x_iw_i \leq b$.
In an **online learning** setting, it iterates through every $(\bar{x}^i, y^i)$ by first computing $h(\bar{x}^i$ and comparing it to the true answer y^i.
We initialize before the first iteration w^0, b^0. At each stage t, upon seeing $f(\bar{x}^i)$ we update to w^{t+1} and b^{t+1} based on **update rules** (Rosenblatt 1958). Assuming all b's are 0:

$$w_{t+1} = w_t \text{ if } \operatorname{sign}(w^i \cdot \bar{x}^i) = y^i$$
$$w_{t+1} = w_t + y^i \cdot \bar{x}^i \text{ if } \operatorname{sign}(w^i \cdot \bar{x}^i) \neq y^i$$

This algorithm converges fast whenever there exists w^* such that for all i, $\operatorname{sign}(w^* \cdot \bar{x}^i) = y^i$.
We let B be the minimal norm such that for some w^* we have $\|w^*\| = B$
Thus for all $1 \leq i \leq m$ if we converge we have

$$h(\bar{x}_i)(w^* \cdot \bar{x}^i) \geq 1$$

19 March 26, 2019

19.1 Novikoff's theorem for perceptrons

Theorem 19.1 (Novikoff's theorem)**.** Given input $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ where $x_j \in \mathbb{R}^n, y_i \in \{-1, 1\}$, if S is linearly separable then for some w^* we have $L_S(h_{w^*}) = 0$. Thus for some w^* we have $\forall i\ \langle w^*, x_w\rangle y_i \geq 1$. Let $B_S = \min\{\|w\| \mid \forall i, \langle w, x_i\rangle y_i \geq 1\}$.
The number of updates of the perceptron algorithm on any linearly separable, realizable S is $\leq (RB_S)^2$ where $R = \max_i \|x^i\|$.

19.2 Non-realizable percetron inputs

How do we handle non-linearly seaparable inputs (i.e. not realizable by class of halfspaces Hs^n)?

One trick: change the way data is represented.

Example 19.1. Suppose $x_1 < x_2 < x_3 < x_4 < x_5$ with labels $+, +, -, -, +$. Clearly $\{x_i, y_i\}$ is not linearly separable, however if we map $x_i \to (x_i, x_i^2) \in \mathbb{R}^2$ (map into $\mathbb{R}^2$ as a parabola) then we there exists coefficient vector w such that for all $1 \leq t \leq m$, $\langle w, (x_i, x_i^2) \rangle = y_i$ (where $w = (w_1, w_2)$ and $\langle w, (x_i, x_i^2) \rangle = w_1 x_i + w_2 x_i^2$).
We can view the linear classifier over the new representation as a polynomial classifier over the old representation.
In general we can perform the mapping

$$x \to (x, x^2, x^3, \ldots, x^\alpha) \in \mathbb{R}^\alpha$$

We can also repeat this trick for an $x = (x_1, \ldots, x_m) \in \mathbb{R}^n$ where:

$$x \to (x_1, \ldots, x_n, x_1x_2, x_1x_3, \ldots, x_1x_n, x_2x_3, x_2x_4, \ldots, x_2x_n, \ldots, x_1^2, \ldots x_n^2) \in \mathbb{R}^{n^2}$$

where we have all monomials of degree ≤ 2 over $x_1, \ldots, x_m$.

Some important questions:

1. Can we guarantee that under the new representation S will be linearly separable?
2. Can we control the sample complexity of these new classifiers?
3. What happens to the computational complexity?

To answer the above:

1. By going to a large enough dimension of representation every S can be made separable.

   That is for $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ we can map $X \to (00 \ldots 1000) \in \mathbb{R}^n$ for some large enough n.

   Note that

$$m_H(\epsilon, \delta) \approx \frac{VC(H) + \log\frac{1}{\delta}}{\epsilon^2}$$
$$\Rightarrow \epsilon \approx \sqrt{\frac{VC(H) + \log\frac{1}{\delta}}{m}}$$

   and also for large n, $VC(H) = n$ becomes very big, so sample complexity blows up.

2. It turns out that we can control the generalization error in terms of $B_S = \min\{\|w\| \mid \langle w, x_t \rangle y_i \geq 1\}$, which can be controlled as there exists such linear classifiers with small B_S. The above becomes

$$m_H(\epsilon, \delta) \approx \frac{B_S + \log\frac{1}{\delta}}{\epsilon^2}$$
$$\Rightarrow \epsilon \sim \sqrt{\frac{B_S + \log\frac{1}{\delta}}{m}}$$

20 March 28, 2019

20.1 Weak learnability

We recall that many ERM_H learners in the PAC learnable and agnostic PAC learnable cases are computationally hard.

What if we trade computational hardness with a laxer requirement of accuracy? Given some distribution D and labelling function f, perhaps there exists a **weak learner** that is computationally efficient whose error is better than a *random guess*.

Definition 20.1 (γ-weak learner)**.** A learning algorithm A is a **γ-weak learner** for class H if there exists function $m_H : (0,1) \to \mathbb{N}$ such that for every $\delta \in (0,1)$ and every distribution D over X and labelling function $f : X \to \{\pm 1\}$ the realizable assumption holds when running the algorithm on $m \geq m_H(\delta)$ iid examples.
That is the algorthm returns a hypothesis such that with probability $\geq 1 - \delta$ we have $L_{(D,f)}(h) \leq \frac{1}{2} - \gamma$.

Definition 20.2 (γ-weak learnable)**.** A hypothesis class H is **γ-weak learnable** if there exists a γ-weak learner for that class.

Remark 20.1. We note the almost identical definition to PAC learning which we herein refer to as **strong learning**.
Note that strong learnability requires us find an arbitrarily good classifier whereas weak learnability only requires us find one that has error $\leq \frac{1}{2} - \gamma$.

Note that the sample complexity of PAC learnable H satisfies

$$m_H(\epsilon, \delta) \geq C_1 \frac{VC(H) + \log \frac{1}{\delta}}{\epsilon}$$

for some constant C_1. Applying $\epsilon = \frac{1}{2} - \gamma$ we see if $VC(H) = \infty$ then H is not γ-weak learnable, thus weak learnability is characterized by VC dimensio as well. As noted in the fundamental of PAC learnable so weak learning is just as hard as PAC learning, however we do end up with the upside of computation efficiency.
One approach is to take a "simple" hypothesis class B and apply ERM with respect to B as the weak learning algorithm. We require B to satisfy two requirements:

1. ERM_B is efficiently implementable
2. For every sample labeled by some hypothesis from H, the ERM_B hypothesis will have at most error $\leq \frac{1}{2} - \gamma$

21 April 2, 2019

21.1 Decision stumps

The big issues with linear halfspaces:

1. Expressive power (approximation error) where in $L_P(h) \leq \min_{h \in H} L_P(h) + \epsilon \min_{h \in H} L_P(h)$ is too high.
2. Computational complexity (in the agnostic case)

Let the 3-piece classifier hypothesis class $H_3 = \{h_{(s,t,b)} \mid s \leq t, b \in \{-1, 1\}\}$ where

$$h_{(s,t,b)}(x) = \begin{cases} b & \text{if } x \in [s,t] \\ -b & \text{otherwise} \end{cases}$$

i.e. H_3 can partition the real line into 3 intervals of $-1, 1, -1$ or $1, -1, 1$ labellings.
Let B be the class of **decision stumps** (threshold functions) over $\mathbb{R}$.

Claim. ERM_B is a γ**-weak learner** for H_3.
What γ-value can we guarantee?
There are parts of S to the leftmost, middle, rightmost that are homogeneously labelled. At least one of those parts contains $\leq \frac{1}{3}$ in mass over S. We can thus choose an $h \in B$ that produces errors only on this part.
For such a $h \in B$ we have

$$L_S(h) \leq \frac{1}{3} = \frac{1}{2} - \frac{1}{6}$$

What about $L_{(D,f)}(h)$, the true error?
Sine $VC(B) = 2$ then for $m > \frac{2+\log\frac{1}{\delta}}{\epsilon^2}$ we have $|L_S(h) - L_P(h)| < \epsilon$ with probability $\geq 1-\delta$.
Thus given a $\delta > 0$ we pick $m(\delta)$ such that for all $h \in B$ $|L_S(h) - L_P(h)| < \frac{1}{12}$ with probability $\geq 1-\delta$, and thus for $m > m(\delta)$ we have with probability $\geq 1-\delta$

$$L_{(D,f)}(A(S)) \leq \frac{1}{3} + \frac{1}{12} = \frac{1}{2} - \frac{1}{12}$$

Thus ERM_B is $\gamma = \frac{1}{12}$-weak learnable.

Remark 21.1. In general we can pick any $m(\delta)$ such that we can get any arbitrary γ-weak learnable guarantees up to $\gamma = \frac{1}{6}$.

21.2 Weak learners in practice

We begin with two questions:

1. Which weak learners are used in practice?
2. How can we "boost" weak learners to get small errors?

To answer the first question, we often we use decision stumps as weak learners. For example, across some d-dimensional space e.g. $\mathbb{R}^d$ and for each dimension i, we choose the best decision stump partitioning on the ith dimension by iterating through all m points and choosing the split point that minimizes the error. We thus perform $m \cdot d$ comparisons.

Example 21.1. Suppose we wanted to detect a faces in an image using decision stumps in a 24×24 greyscale image.
Consider all rectangles over the image ($\approx 24^4$). There are four possible labellings of the rectangles (given two decision trumps along each dimension):

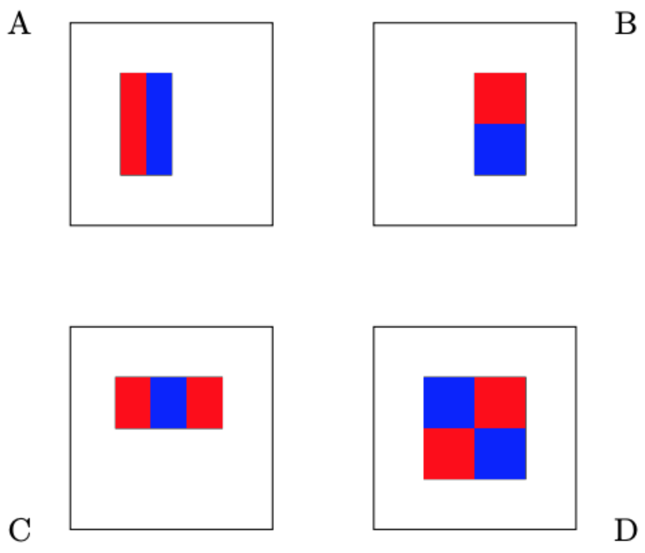

Figure 21.1: Types of rectangle shading. For a given rectangle these are all the same size. Note that the partitioning is exact where we either divide the rectangle exactly in half or in thirds for a given dimension.

Within each rectangle and for each type we associate the value of the sum of the pixels in the red rectangles subtract the sum of the pixels in the blue rectangles.
Then for an image, we produce a $\approx 4 \cdot 24^4$-length vector where we have a value for each of the 4 types for each of the $\approx 24^4$ rectangles.
A stronger learner can then decide which of the rectangle filters (i.e. which features in the vector) are most important in detecting a face by training on a sample of faces. For example, the learner may choose the following two filters for a set of faces:

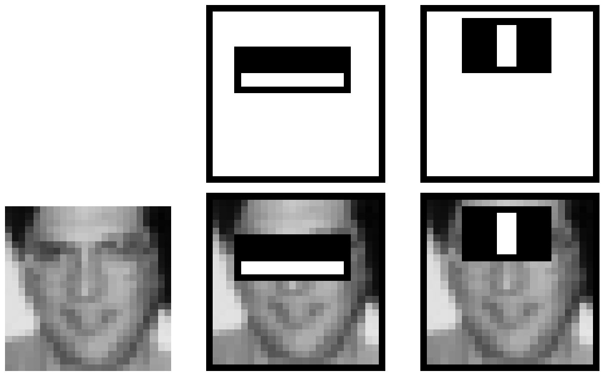

Figure 21.2: First feature measures difference in intensity between eyes and upper cheeks. Second feature compares intensities in eye region vs intensity across bridge of nose.

21.3 Boosting and AdaBoost

We know that PAC learning is computationally infeasible in the *agnostic case*.

We may ask ourselves: maybe weak learning is easier? In 1990, Robert Schapire showed that this is **not** the case.

Theorem 21.1. If you can efficiently γ-weak learn a class H for $\gamma > 0$ then one can efficiently (strong) learn it.

Remark 21.2. The above theorem automatically implies weak learning cannot be easy.

Yoav Freund and Rob Schapire subsequently came up with a **boosting** algorithm called **AdaBoost** in 1996. A boosting algorithm combines weak learners to get a strong learner.

AdaBoost Given a sample S and a weak learner A, we learn a series of hypothesis $h_0, h_1, h_2, \dots$ over the distributions $D_0, D_1, D_2, \dots$ over S where D_0 is a **uniform distribution** over S i.e. $D_0(x_i, y_i) = \frac{1}{m}$ and $h_0 = A(S, D_0)$.
Given h_t, D_t, how do we generate h_{t+1}, D_{t+1}? The idea is we want D_{t+1} to increase the mass $D_{t+1}(x_i, y_i)$ for every i where $h_t(x_i) \neq y_i$ i.e. increase the emphasis on incorrectly labelled points by the previous h_t. h_{t+1} is then the hypothesis $A(S, D_{t+1})$.
Our final hypothesis h_T is defined as

$$h_T = \text{sign}\big(\sum_{t=1}^{T} \alpha_t h_t\big)$$

where the weights α_t per weak learned h_t is **inversely proportional** to $L_{(S,D_t)}(h_t)$.
Note that if A is γ-weak learnable, then we can show

$$L_S(h) \leq \frac{1}{e^{\gamma T}}$$

after T steps where

$$\gamma = \min_{1 \leq i \leq T} \big(\frac{1}{2} - L_S(h_i)\big)$$

Boosting thus solves two issues:

1. Computational complexity of training *expressive* H's: use simple "weak class" of B's that is easy to train
2. h's in B's cannot achieve small train error: boosting solves this

Note that once $L_S(h) < \frac{1}{|S|}$, then $L_S(h) \to 0$, thus in roughly $T \approx \log m$ we get zero train error.

22 April 4, 2019

22.1 Issues with boosting

One issue with boosting is that it tends to overfit: what is the true error $L_P(h)$?

Solution. Note that in order to ensure $|L_S(h) - L_P(h)|$ is small, we recall uniform convergence and the fact that

$$|L_S(h) - L_P(h)| \leq \sqrt{\frac{VC(\bar{H}) + \log \frac{1}{\delta}}{m}}$$

where in our case $\bar{H} = L(B, T)$ is a linaer combination of size T of members of B.
Note that $VC(L(B,T)) \approx T \cdot VC(B)$ so if we control either $VC(B)$ and T (fewer steps) we can reduce the overall $VC(\bar{H})$.

Another issue: in the overall error $L_S(h) = \frac{1}{e^{\gamma T}}$ we have $\gamma = \min\{\frac{1}{2} - L_S(h_i)\} = \min\{\gamma_i\}$: note that this is a min over all h_i. In practice we notice that $\gamma_i \to 0$ so we cannot always achieve $L_S(h) = 0$.

22.2 Generalization and types of errors

As mentioend briefly above, one issue with every learning tool is generalizing well i.e. ensuring $|L_S(h) - L_P(h)|$ is low. Note that in general we have

$$L_P(A(S)) \approx \text{computation error} + |L_P(h) - L_S(h)| + \min_{h \in H} L_P(h)$$

where $absL_P(h) - L_S(h)$ is the **generalization error** and $\min_{h \in H} L_P(h)$ is the **approximation error**.
Note that **computation error** is the error that is a result of getting $A(S)$ as close as possible to $\hat{h}$: the hypothesis that minimizes L_S. This usually involves more computation e.g. iterating over all $h \in H$.
Approximation error is the best possible error we get on the true distribution P within our hypothesis class H.

22.3 Neural networks

A neural network predictor is determined by $< V, E, \sigma, \bar{w} >$ where V, E is the graph (vertices, edges), σ is the activation function (what do nodes compute) and $\bar{w}$ are the weights over edges.
Suppose (V, E) forms a **directed acyclic graph (DAG)**. σ is a function from $\mathbb{R}$ to $\mathbb{R}$ that a node v computes. Then for input $\bar{x}$ and a node v

$$v(\bar{x}) = \sigma\big(\sum_i w_i u_i(\bar{x})$$

where w_i are the weights on the input edges and $u_i(\bar{x})$ are the node activations feeding into v.
Some common activation functions:

1. Threshold function: $\sigma(z) = \text{sign}(z)$
2. Sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$
3. ReLU piecewise linear: $\sigma(z) = \max(0, z)$

How do neural networks handle the fundamental challenges?

1. Expressive power: every function can be computed by a neural entwork with very simple σ (universal approximation theorem).

   If a function can be computed in time $T(m)$ then a neural network of size $T(m)^2$ can compute it.

2. Computational complexity: training is **NP-hard**.

   However during inference, given h and test x computing $h(x)$ is easy.

3. Generalization: For a neural network with graph (V, E) then $VC(NN) \approx |E|$.