Module 5: An Introduction to the Statistical Learning theory

we will consider the supervised learning problem, but the framework can also be extended to unsupervised learning.

A formal framework for learning theory:

(1) Data is assumed to be generated according to a distribution $(x, y) \sim D(x, y)$. (x: vector of features; y: label) The input domain is denoted by X ($x \in X$), the label domain is denoted Y (i.e $y \in Y$), and D is a distribution over $X \times Y$.

(2) The learner's output: We are looking for predictive relations $h: X \longrightarrow Y$ with low prediction error. Formally speaking, for each function $h: X \longrightarrow Y$ we define its error as:

prediction error » generalization error » true error

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

D: distribution of data

$$= \mathbb{E}_{(x,y)\sim D}\left[\mathbb{1}\{h(x)\neq y\}\right]$$

Ideally, we'd like to find a predictor h that has the

Smallest true error, i.e our gold standard is to solve

$$\text{Minimize}_{h} \; \mathcal{L}_D(h) \qquad (1)$$

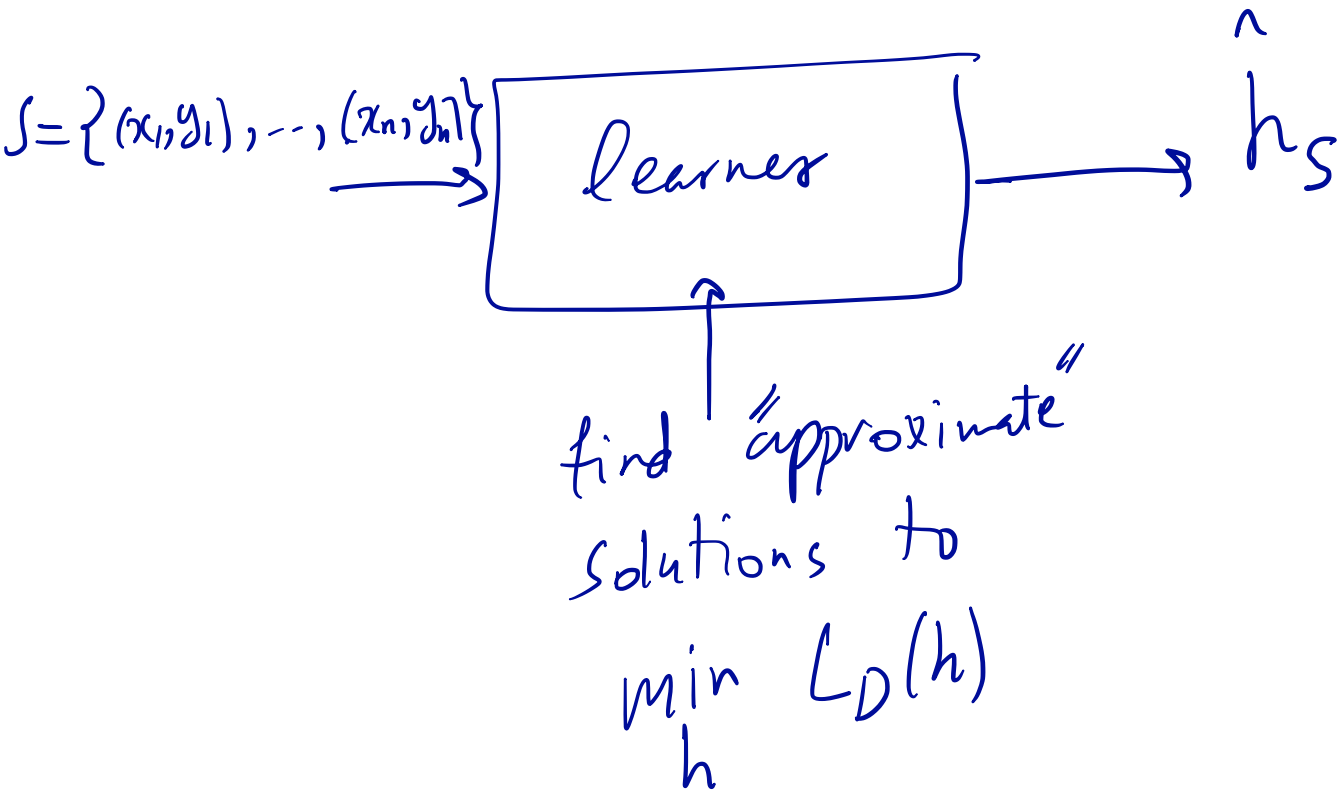
$\hookrightarrow \mathcal{L}_D(h) = E_{(x,y)\sim D}[\#\{h(x) \neq y\}]$

However, this task is impossible since D is unknown.

(3) the learner's input: Training data! The only information that the learner has about the data distribution is a set of training samples (data points), $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$

where each (x_i, y_i) is

Generated i.i.d. according to the distribution D. Hence, the learner has to find "approximate" solutions to problem (1) using the training data set.



$(x_i, y_i) \overset{iid}{\sim} D$

4) Empirical Risk Minimization (ERM):

$L_D(h)$ ⟶ find an unbiased estimator

↑ Can't compute exactly

$$L_D(h) = \mathop{\mathbb{E}}_{(x,y)\sim D}\left[\mathbb{1}\{h(x) \neq y\}\right]$$

unbiased estimate

$$L_S(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{h(x_i) \neq y_i\}$$

Note: $\mathbb{E}_S[L_S(h)] = L_D(h)$

Empirical risk minimization (ERM) is the task of finding a predictor that minimizes the training error:

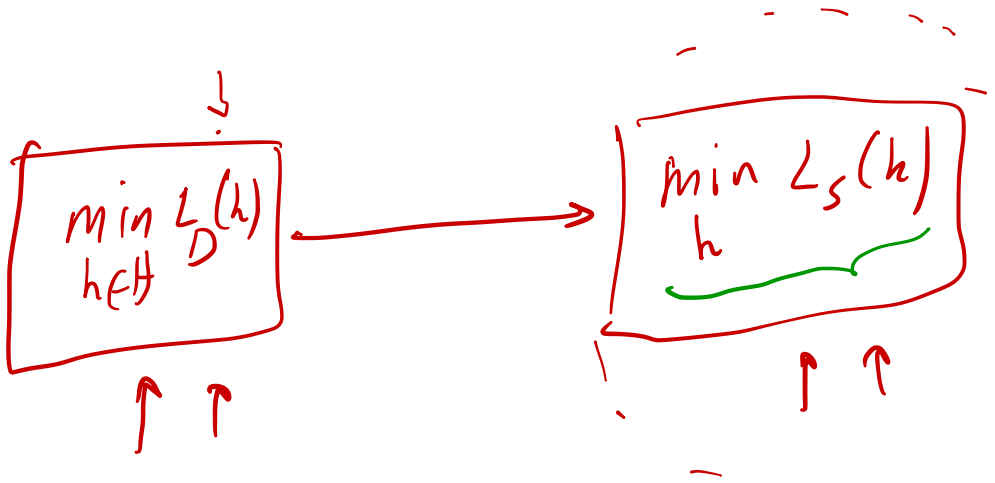
$$\min_h L_S(h) \qquad (2)$$

$\min_h L_D(h)$ $\xrightarrow{\text{instead}}$ $\min_h L_S(h)$

($\min_h L_D(h)$: we can't do)

$$L_D(h) \xrightarrow{\text{instead}} L_S(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{h(x_i) \neq y_i\}$$

Lecture 23:



Claim:

$$\min_h L_S(h) = 0$$

$$L_S(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{h(x_i) \neq y_i\}$$

Define $\tilde{h} : X \longrightarrow Y$ as follows:

$$\tilde{h}(x) : \begin{cases} y_i & \text{if } x = x_i \\ 0 & \text{if } x \in \{x_1, \ldots, x_n\} \end{cases}$$

It's easy to see that

$$L_S(\tilde{h}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{\tilde{h}(x_i) \neq y_i\}$$

$$= 0$$

$L_S(\tilde{h}) = 0 \longrightarrow$

$$\min_h L_S(h) = 0$$

5) Although ERM is natural, it can fail miserably if we are not careful: It can "overfit" easily.

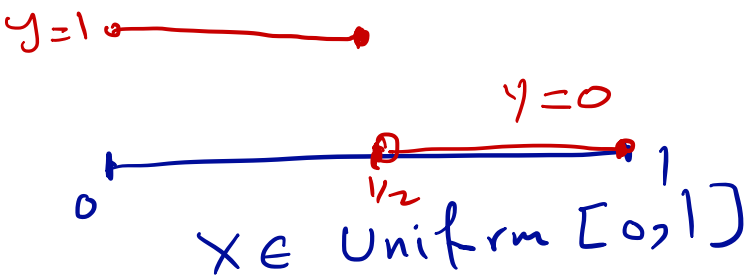
The minimizer of

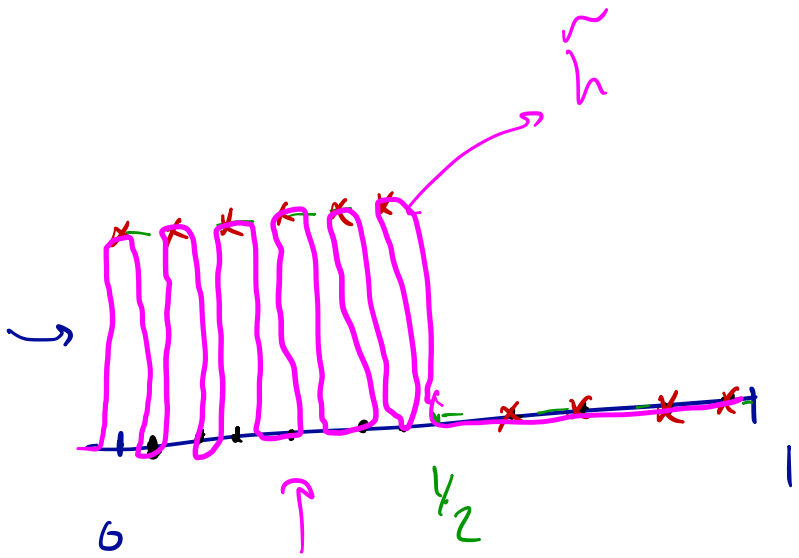
$$\min_h L_S(h) \qquad (2)$$

is always zero.

Example:

Distribution of data:





prediction on unseen data here is very bad

$$\min_h L_S(h)$$

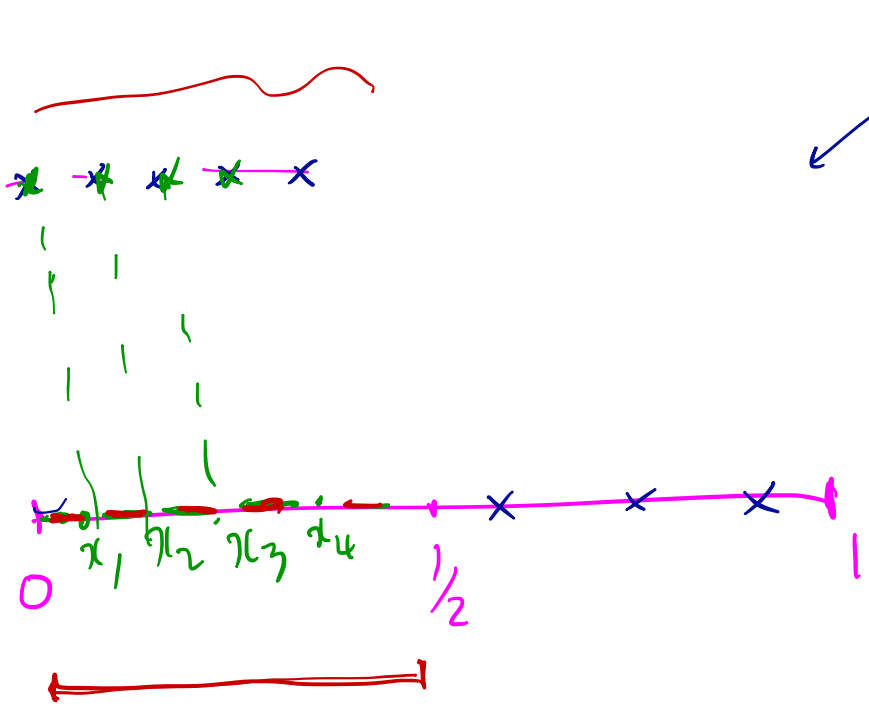
h

$$\left\{ \begin{array}{l} L_S(\tilde{h}) = 0 \\ L_D(\tilde{h}) \cong 0.5 \end{array} \right.$$

$$\min_h L_D(h)$$

$$\min_h L_S(h) \to 0$$

$h \uparrow \tilde{h}$

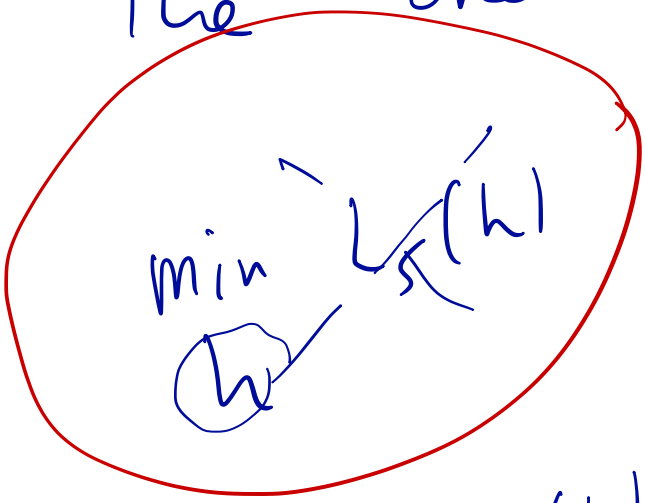


$$\tilde{h} = \begin{cases} h(x_i) = y_i \\ h(x) = 0 \end{cases} \qquad x \in \{x_1, \dots x_n\}$$

$$\begin{cases} L_S(\tilde{h}) = 0 \\ L_{(P,f)}(\tilde{h}) = \underset{(x,y)\sim D}{\mathbb{P}}[h(x) \neq y] \end{cases}$$

$$= 1/2$$

6) We need to search for conditions under which there is a guarantee that ERM does not overfit, namely conditions under which when the ERM predictor has a good performance with respect to the training data, it is also likely to perform well over the underlying (true) distribution



$\min_{h} L_S(h)$ → $\min_{h \in \mathcal{H}} L_S(h)$

$\min_{h} L_D(h)$

restrict the class of functions (complexity)

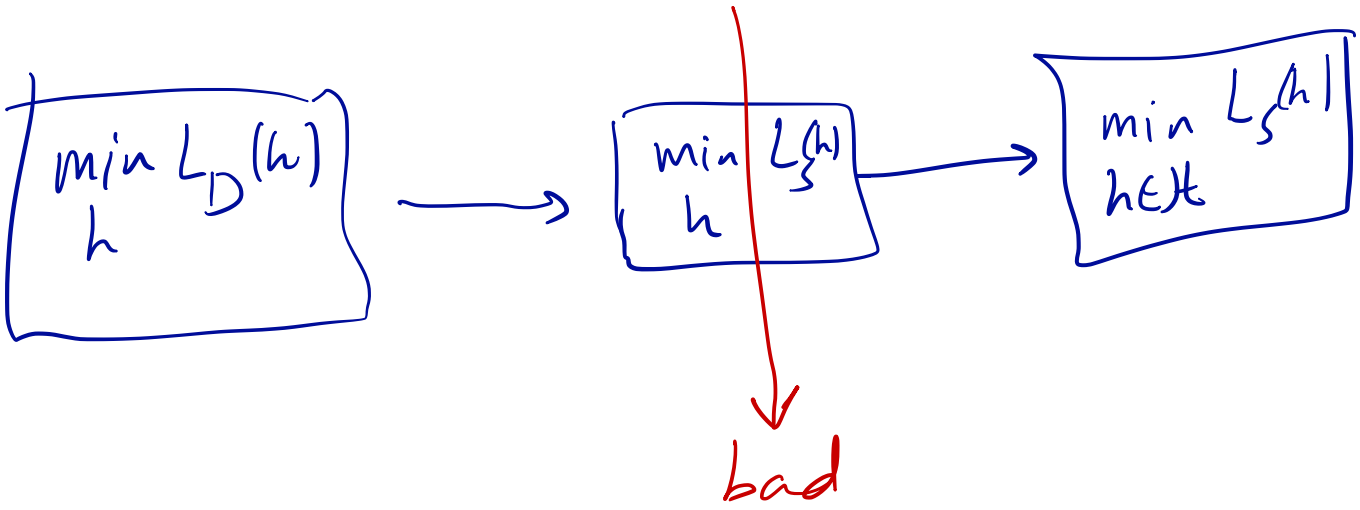
↓

prevent overfitting

A common solution is to apply ERM over a restricted set of functions (classifiers). Formally the learner should choose in advance (before seeing the data) a set of functions (classifiers). This set is called "the predictor class" or "the function class" or "the hypothesis class" and is denoted by $\mathcal{H}$. Each $h \in \mathcal{H}$ is a function from $\mathcal{X}$ to $\mathcal{Y}$.

And the restricted ERM problem ($ERM_{\mathcal{H}}$) is:

minimize $L_S(h)$.
$h \in \mathcal{H}$



So we are biasing the learner towards a particular set of predictors.

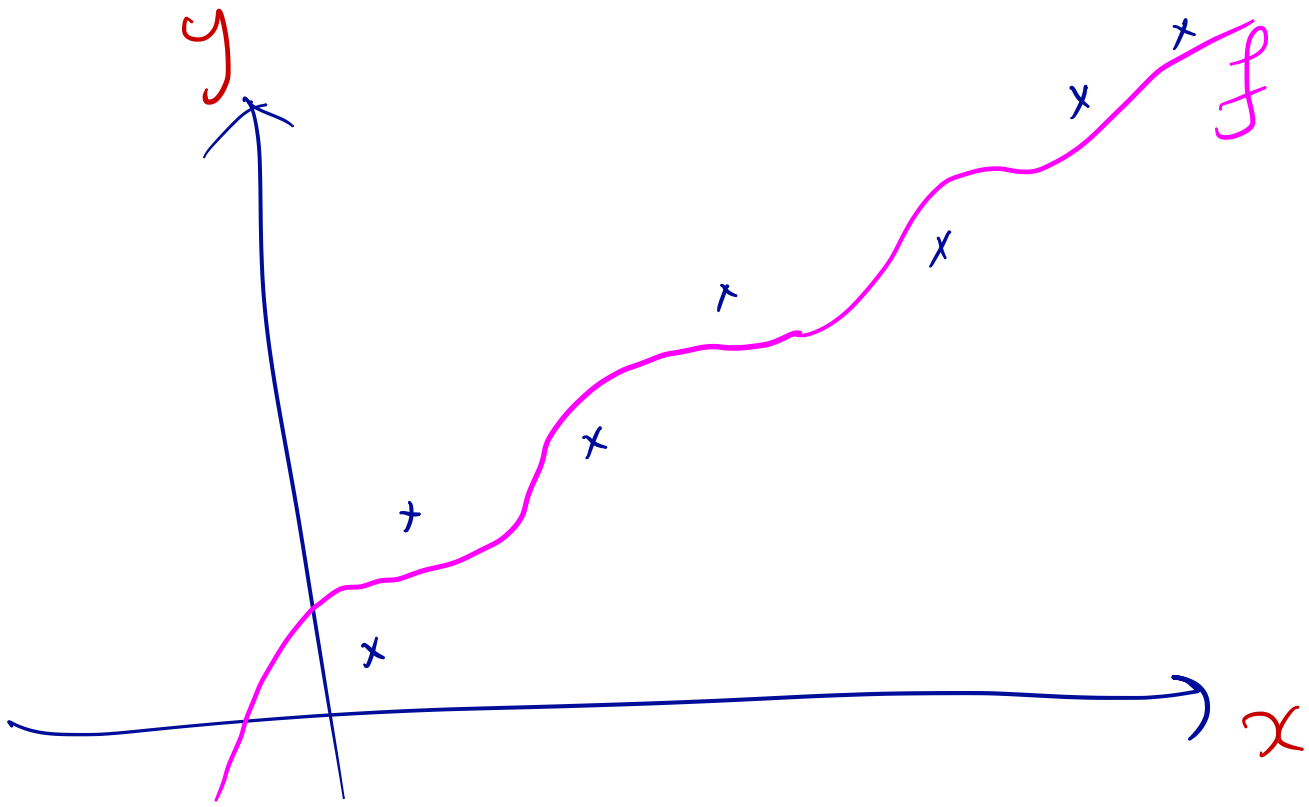
To explain things further, let's look at an example about regression (even though it is not classification, it will help us understand the concepts better).

$$y = f(x) + \epsilon$$

ϵ: independent gaussian $N(0, \sigma^2)$

We don't know f and we'd like to estimate the predictive relation between x and y using training data.

$$S = \{(x_i, y_i)\}_{i=1}^{n}$$



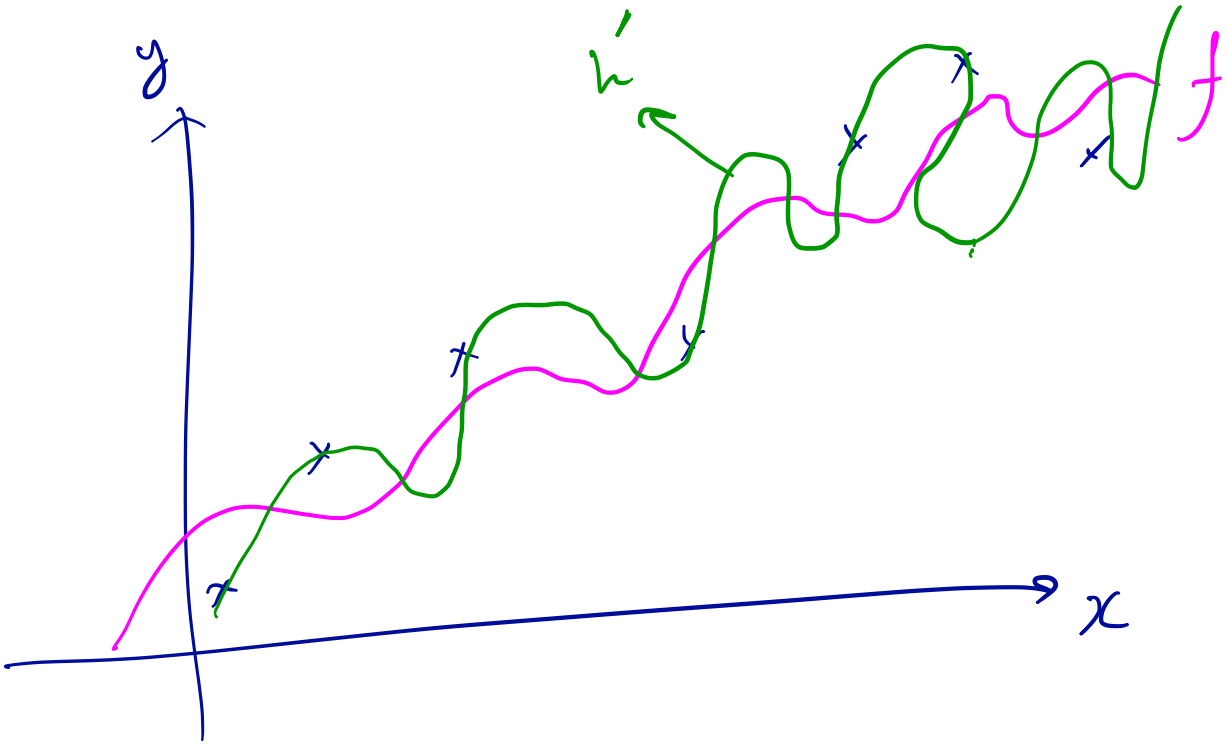
for any $h : X \longrightarrow Y$

$$L_D(h) = \underset{(x,y)\sim D}{\mathbb{E}}\left[(h(x) - y)^2 \right]$$

$$\min_h L_D(h) = \sigma^2$$

$\min_h$ ↳ minimizer $\longrightarrow f$

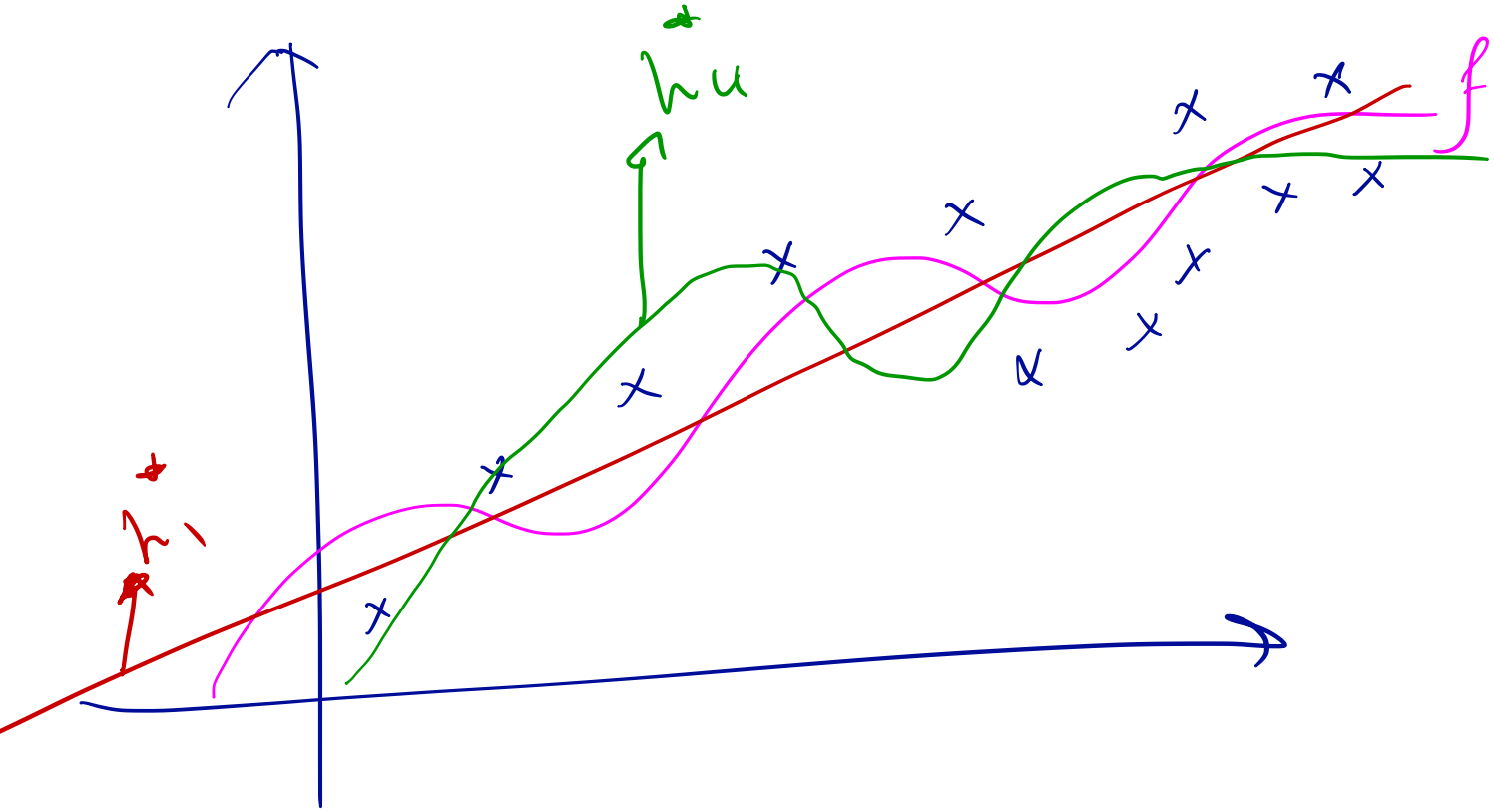
$$L_D(f) = \underbrace{\mathbb{E}\left[(y - f(x))^2 \right]}_{\epsilon^2} = \sigma^2$$



$$\begin{cases} \min_h L_D(h) = 6^2 \\ \min_h L_S(h) = 0 \end{cases} \longrightarrow L_S(h') = 0$$

Let's now consider restricted ERM objectives:

$$\min_{h \in \mathcal{H}} L_S(h)$$



$ERM_{\mathcal{H}} \longrightarrow \min_{h \in \mathcal{H}} L_S(h)$

e.g. when $\mathcal{H}_1 = \{$ polynomials of degree 1 $\}$

$\hookrightarrow \min_{h \in \mathcal{H}_1} L_S(h) \longrightarrow h_1^*$

$\mathcal{H}_4 = \{$ polynomials of degree 4 $\}$

$\min_{h \in \mathcal{H}_4} L_S(h) \longrightarrow h_4^*$

There are two important points:

(1) since we are restricting our function class, our best bet would be to achieve

$$\min_{h \in \mathcal{H}} L_D(h)$$

which is larger than

$$\min_{h \in \mathcal{H}} L_D(h) \geq \min_{h} L_D(h)$$

$$\min_{h} L_D(h) \neq 0$$

Hence, if the class $\mathcal{H}$ is not rich enough we may have

(underfitting)

$$\min_{h \in \mathcal{H}} L_D(h) \gg \min_h L_D(h)$$

$\Rightarrow$ $\mathcal{H}$ should be sufficiently complex

(2) The minimizer of

$$\min_{h \in \mathcal{H}} L_S(h)$$

is not the same as the minimizer of

$$\min_{h \in \mathcal{H}} L_D(h)$$

In other words if we denote

$$\rightarrow h_S^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}}\ L_S(h)$$

↳ this is computable

then it is possible that

$$L_D(h_S^*) \gg \min_{h \in \mathcal{H}} L_D(h)$$

↑ overfitting

To avoid overfitting in $\mathcal{H}$ we should make sure that we have sufficiently many training data points.

h_S^* → minimizer of $L_S(h)$ within the class $\mathcal{H}$.

since $h^* \in \mathcal{H} \Rightarrow L_D(h^*) \geq \min_{h \in \mathcal{H}} L_D(h)$

$$L_D(h^*) >> \min_{h \in \mathcal{H}} L_D(h)$$

↑ overfitting

---

two potential problems:

(1) $\min_{h \in \mathcal{H}} L_D(h) >> \min_h L_D(h)$

↑ underfitting

we need to increase the complexity of $\mathcal{H}$

→ solution of $ERM_{\mathcal{H}}$

(2) $L_D(h_S^*) >> \min_{h \in \mathcal{H}} L_D(h)$

we need to increase the # of train data points

↓ overfitting

Lecture 24:

$$\min_h L_D(h) \longrightarrow \min_h L_S(h)$$

$L_S(h) \to 0$

$$\tilde{h} = \begin{cases} y_i & x = x_i \\ 0 & x \notin \{x_1, \dots, x_n\} \end{cases}$$

$L_S(\tilde{h}) = 0$

$L_D(\tilde{h})$ very large

$$\min_{h \in \mathcal{H}} L_S(h)$$

1. Since we're restricting our search to $\mathcal{H}$

$$\min_{h \in \mathcal{H}} L_D(h) \geq \min_h L_D(h)$$

2. what we're actually optimizing is

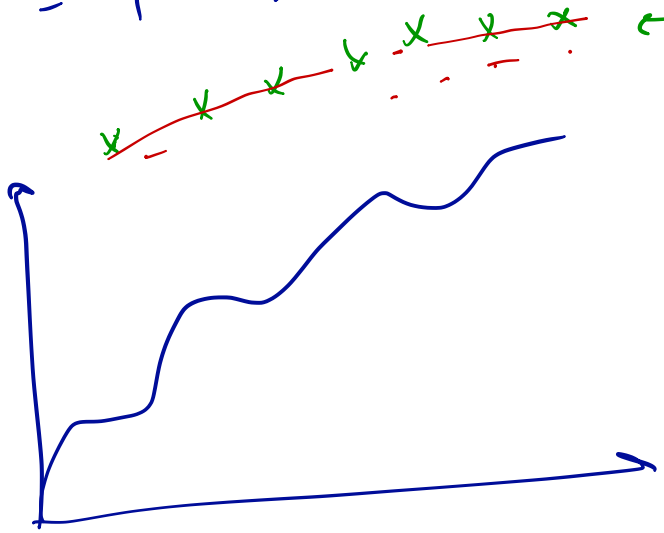
$$\min_{h \in \mathcal{H}} L_S(h) \Longrightarrow$$ Let's denote the minimizer by h_S^*.

$$L_D(h_S^*) \geq \min_{h \in \mathcal{H}} L_D(h)$$

↳ if $L_D(h_S^*)$ is much larger than $\min_{h \in \mathcal{H}} L_D(h)$, then this is called overfitting.

3. All the guarantees and bounds are probabilistic (they'll hold with high probability excluding the unlikely scenarios).

$$y_i = f(x_i) + \epsilon_i$$

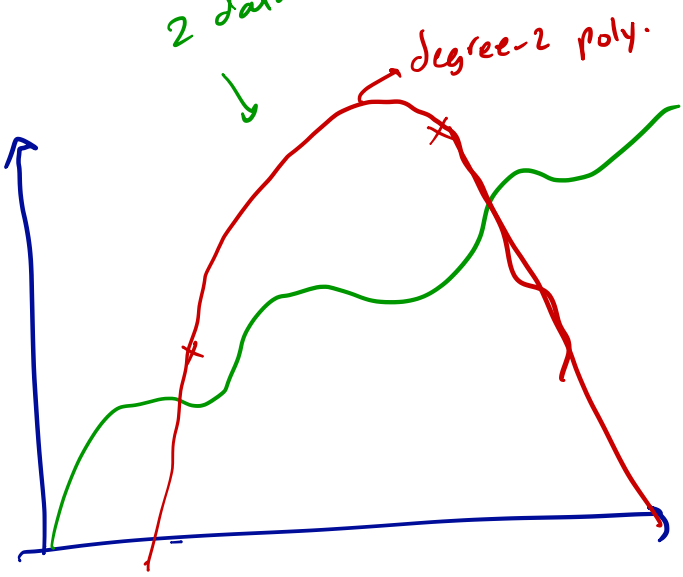


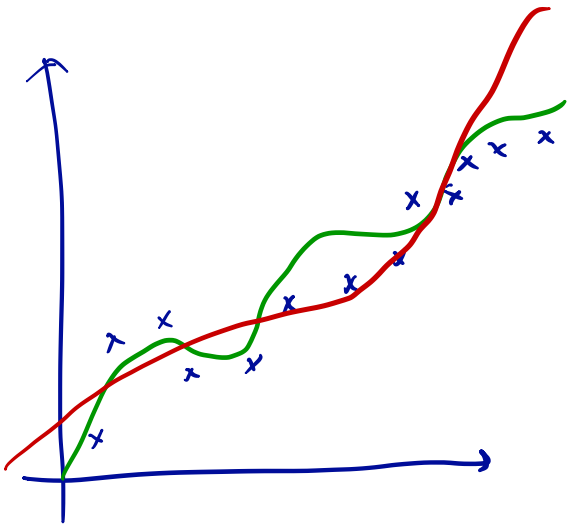
$L_D \rightarrow E_D$

$L_S \rightarrow \frac{1}{n} \sum_i$

$L_S \rightarrow L_D$

$\min_{h \in \mathcal{H}} L_S(h) \rightarrow$ in order to ensure that overfitting does not take place, we need to have a sufficient number of training data points.





$y_i = f(x_i) + \epsilon_i$

$\mathcal{H} = \{$ polynomials of degree 3 $\}$

$\Rightarrow$ overfitting can be avoided provided that we have sufficiently many traing data points.

Hence, the main question is:

How many training data points do we need to make sure that overfitting does not happen?

↳ # of data points that we need will depends on:

$\mathcal{H}$, ϵ , δ

what does "avoiding overfitting" mean?

$h_S^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}}\, L_S(h)$:

what we can do computationally

$\min_{h \in \mathcal{H}} L_D(h)$

the best error that we can hope for.

overfitting is avoided when: $L_D(h_S^*)$ is close to $\min_{h \in \mathcal{H}} L_D(h)$

Overfitting is avoided when

small value

$$\min_{h \in \mathcal{H}} L_D(h) \leq L_D(h_S^*) \leq \min_{h \in \mathcal{H}} L_D(h) + \epsilon$$

With probability $1-\delta$ we have:

$$L_D(h_S^*) - \min_{h \in \mathcal{H}} L_D(h) \leq \epsilon$$

provided that sufficiently many data points are available.

---

if we have more than n_ϵ data points, then with high probability the following is true:

with probability at least $1-\delta$

$$L_D(h_S^*) - \min_{h \in \mathcal{H}} L_D(h) \leq \epsilon$$

n_ϵ depends on $\epsilon, \delta, \mathcal{H}$.

PAC Learning:

PAC = Probably Approximately Correct

Definition (PAC): A function class $\mathcal{H}$ is called PAC learnable if for every $\epsilon, \delta \in (0,1)$, there exists a number $n_0(\epsilon, \delta)$ such that the following holds:

There exist a learning algorithm that for any distribution D over $X \times Y$, by using a set S of $n_0(\epsilon, \delta)$ data points, we obtain a predictor function h_S^* such that with probability $1-\delta$ we have

$$\min_{h \in \mathcal{H}} L_D(h) \leq L_D(h_S^*) \leq \min_{h \in \mathcal{H}} L_D(h) + \epsilon.$$

$\mathcal{H} \longrightarrow \begin{cases} \text{Algorithm} \rightarrow h_S^* \\ n_0(\epsilon, \delta) \end{cases} \Rightarrow \mathcal{H}$ is PAC learnable

---

Let's start with the simplest possible hypothesis class and see what PAC-learning means with that class.

$$\mathcal{H} = \{h_1, h_2, \ldots, h_m\}$$

$\min_{h \in \mathcal{H}} L_D(h)$ $\longrightarrow$ $\min_{h \in \mathcal{H}} L_S(h)$

$\Downarrow$

h_S^* is one of the functions in $\mathcal{H}$ with smallest empirical error.

$\min_{h \in \mathcal{H}} L_S(h) = \min\{L_S(h_1), \ldots, L_S(h_m)\}$

What we'd like to find in this case is a number $n_0(\epsilon, \delta)$ such that for any distribution D over data we have:

with probability $1-\delta$:

$$L_D(h_S^*) \leq \min_{h \in \mathcal{H}} L_D(h) + \epsilon$$

In other words we're asking how large the number of training data points should be s.t. we have w.p. $1-\delta$ that $L_D(h_S^*) \leq \min_{h \in \mathcal{H}} L_D(h) + \epsilon$.

In order to find $n_0(\epsilon, \delta)$ we need to answer two main questions.

(1) Given a fixed function $h: X \to Y$, how many traing data point should we have such that:

with probability $1-\delta$:

$$|L_S(h) - L_D(h)| < \varepsilon$$

$$\left| \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{h(x_i) \neq y_i\} - E_D[\mathbb{1}\{h(x) \neq y\}] \right| < \epsilon$$

what should be $|S| = n_0(\epsilon, \delta)$?

To answer this question we'll use a very important probabilistic tool which is call the Hoeffding's inequality. This inequality has very useful in a variety of applications in data science (both in terms of theory

and algorithm design).

Hoeffding's Inequality:

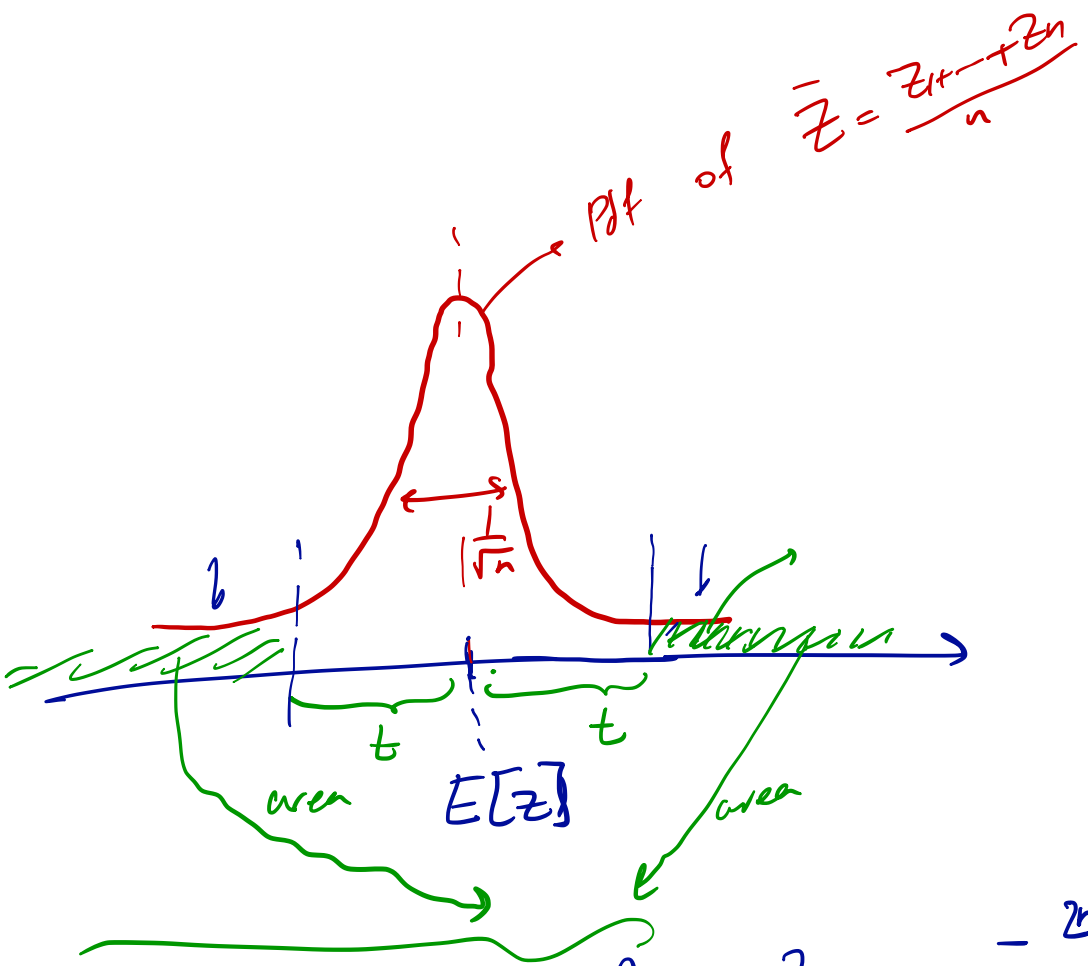
Let $Z_1, Z_2, \ldots, Z_n$ be n iid random variables that take value in the unit interval (i.e. $Z_i \in [0,1]$).

Then:

$$\Pr\left\{ \left| \frac{\sum_{i=1}^{n} Z_i}{n} - E[Z_i] \right| \geq t \right\} \leq 2e^{-2nt^2}$$

$Z_1, Z_2, \ldots, Z_n$ iid $Z \longrightarrow \sigma^2 = \mathrm{Var}(Z)$

$$\frac{Z_1 + Z_2 + \cdots + Z_n}{n} \sim E[Z] + \frac{1}{\sqrt{n}} N(0, \sigma^2)$$



$$Pr\{|\bar{Z} - E[Z]| > t\} \leq 2e^{-2nt^2}$$

e.g. $n = 1000$, $t = 0.5$

$$e^{-2 \cdot 1000 \cdot (0.5)^2} = e^{-500} \longrightarrow 0$$

Lecture 25:

$\mathcal{H}$ → PAC learnable:

(1) $n_0(\epsilon, \delta)$

(2) learning algorithm that takes as input a set of f training data points S, and outputs h^*_S.

for any data distribution D, as long as $|S| \geq n_0(\epsilon, \delta)$ then with probability $1-\delta$ we have:

$$L_D(h^*_S) \leq \min_{h \in \mathcal{H}} L_D(h) + \epsilon$$

Example:

$\mathcal{H} = \{h_1, \dots, h_m\}$

(1) specify what $n_0(\epsilon, \delta)$ is.

(2) Algorithm: ERM $\mathcal{H}$:

$$h_S^* = \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h)$$

what is $n_0(\epsilon, \delta)$, s.t.

w.p. $1-\delta$

$$L_D(h_S^*) \leq \min_{h \in \mathcal{H}} L_D(h) + \epsilon$$

Two questions:

(1) Given a fixed function h, how many data points, S, do we need such that

w.p. $1-\delta$

$$|L_S(h) - L_D(h)| < \varepsilon$$

$$L_S(h) = \frac{1}{|S|} \sum_{i=1}^{|S|} \mathbb{1}\{h(x_i) \neq y_i\}$$

$\Updownarrow$ equivalent

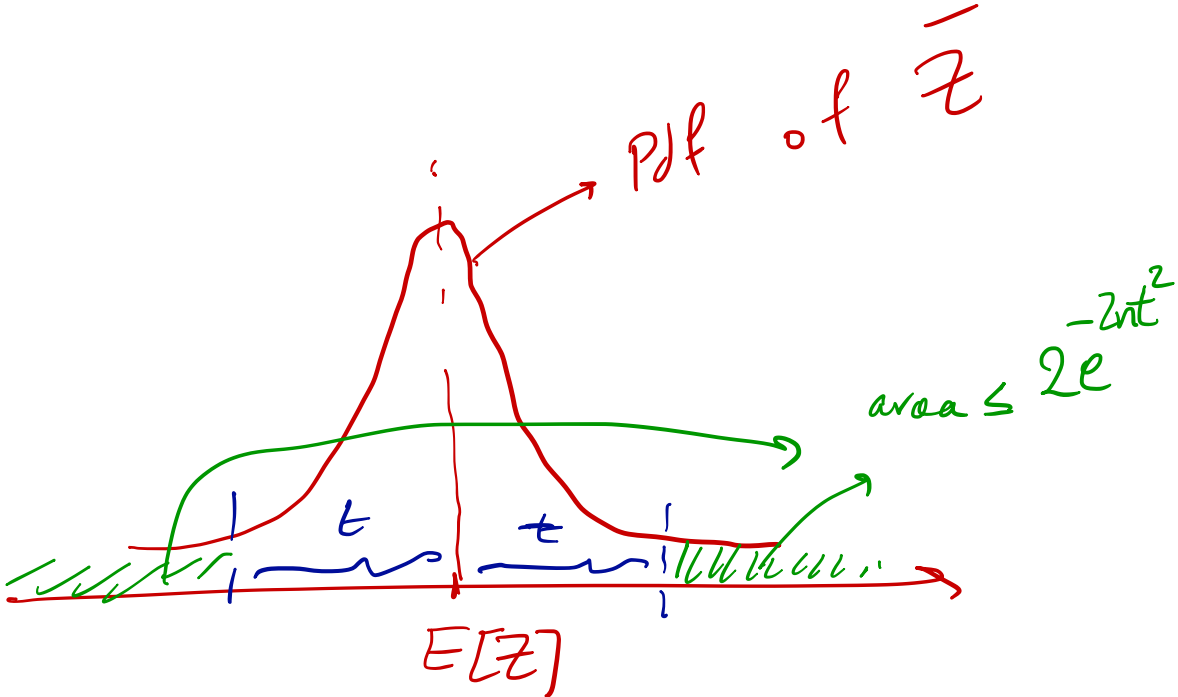
$$\Pr\{|L_S(h) - L_D(h)| \geq \varepsilon\} \leq \delta$$

Recall that Hoeffding's inequality was given as follows:

For any iid random variables $Z_1, Z_2, \dots, Z_n$, s.t. $Z_i \in [0,1]$,

we have

$$\Pr\left\{ \left| \overline{\frac{1}{n}\sum_{i=1}^{n} Z_i} - \overline{E[Z]} \right| > t \right\} \leq 2e^{-2nt^2}$$



$$L_S(h) = \frac{1}{n} \sum_{i=1}^{n} \underbrace{\mathbb{1}\{h(x_i) \neq y_i\}}_{Z_i}$$

$(|S| = n)$

$$L_D(h) = \underbrace{\mathop{E}_{(x,y)\sim D}\left[\mathbb{1}\{h(x) \neq y\}\right]}_{E[Z]}$$

$$L_S(h) = \frac{1}{n} \sum_{i=1}^{n} Z_i = \bar{Z}$$

$$Z_i = \begin{cases} 1 & \text{if } h(x_i) \neq y_i \\ 0 & \text{o.w.} \end{cases} \rightarrow Z_i \in [0,1]$$

$$E[Z_i] = \mathop{E}_{(x_i,y_i)\sim D}\left[\mathbb{1}\{h(x_i) \neq y_i\}\right]$$

$$= L_D(h)$$

Hoeffding:

$$P_r\left\{\left|\underbrace{L_S(h)}_{\frac{1}{n}\sum Z_i} - \underbrace{L_D(h)}_{E[Z]}\right| > \epsilon\right\} \quad \text{for any } \epsilon > 0 \qquad (1)$$

$$\leq 2e^{-2n\epsilon^2}$$

Recall that we're looking for the smallest n s.t. (2)

$$P_r\{|L_S(h) - L_D(h)| > \epsilon\} \leq \delta$$

using (1), and in order to guarantee (2), we can let:

$$2e^{-2n\epsilon^2} \leq \delta \qquad \left(\log\frac{2}{\delta} = -\log\frac{\delta}{2}\right)$$

$$\Rightarrow \boxed{n \geq \frac{1}{2\epsilon^2}\log\frac{2}{\delta}}$$

So the final statement is:

Given any ϵ, δ, as long as the number of training data points is larger than $n_1 = \frac{1}{2\epsilon^2} \log \frac{2}{\delta}$, we have

$$\Pr_{(x,y)\sim D}\left\{ |L_S(h) - L_D(h)| > \epsilon \right\} \leq \delta.$$

(for a fixed function h)

(2) Let's now assume that we have m functions $h_1, h_2, \ldots, h_m$. What is the smallest value $n_0(\epsilon, \delta)$ such that

with probability $1-\delta$: (3)

$$\forall\, i \in \{1, \ldots, m\} \;:\; |L_S(h_i) - L_D(h_i)| < \epsilon .$$

To answer this question, we're going to write an equivalent formulation of relation (3):

Let A_i be the event that

$$|L_S(h_i) - L_D(h_i)| > \epsilon .$$

Then (3) is equivalent to:

$$Pr\{A_1 \cup A_2 \cup A_3 \cdots \cup A_m\} \leq \delta . \quad (4)$$

Remember that we are searching for the smallest value of n such that (4) holds.

$$Pr\{A_1 \cup A_2 \cup \cdots \cup A_m\} \leq \delta$$

To answer this, we're going to use the Union bound:

$$Pr\{A \cup B\} \leq Pr\{A\} + Pr\{B\}$$

$$Pr\{A_1 \cup A_2 \cup \cdots \cup A_m\} \leq Pr\{A_1\} + Pr\{A_2\} + \cdots + Pr\{A_m\}.$$

---

bad event #i : A_i :

$$|L_S(h_i) - L_D(h_i)| > \epsilon$$

We'd like to make sure that none of the bad events, A_i, would take place.

So in order to guarantee

$$Pr\{A_1 \cup A_2 \cup \cdots \cup A_m\} \leq \delta \quad (5)$$

it is sufficient to guarantee that

$$Pr\{A_1\} + Pr\{A_2\} + \cdots + Pr\{A_m\} \leq \delta \quad (6)$$

Note that if (6) holds, then (5) would also hold as a result of the union bound.

Now, to guarantee (6), it is sufficient to choose n large enough s.t. for every h_i

We have

$$Pr\{A_i\} \leq \frac{\delta}{m} \qquad (7)$$

$$\Downarrow$$

$$Pr\{A_1\} + Pr\{A_2\} + \cdots + Pr\{A_m\} \leq \delta$$

$$\Downarrow$$

$$Pr\{A_1 \cup A_2 \cup \cdots \cup A_m\} \leq \delta$$

Now given our answer to question 1, in order to guarantee that $Pr\{A_i\} \leq \frac{\delta}{m}$ we need to

Choose:

if $n \geq n_1(\epsilon, \frac{\delta}{m}) = \underbrace{\frac{1}{2\epsilon^2} \log \frac{2m}{\delta}}_{n_0(\epsilon,\delta)}$

then

$$Pr\{A_i\}$$

$$= Pr\{ |L_S(h_i) - L_D(h_i)| > \epsilon \} \leq \frac{\delta}{m}$$

$\left(n_0(\epsilon,\delta) = n_1(\epsilon, \frac{\delta}{m}) \right)$

Hence,

Statement: If the number of training data points, $|S|$, is larger than $n_0(\epsilon,\delta) = \frac{1}{2\epsilon^2} \log \frac{2m}{\delta}$, then

with probability $1-\delta$ we have

$\forall i \in \{1,\dots,m\}$: $|L_S(h_i) - L_D(h_i)| < \epsilon$.

What we've shown is that we need

$n_0(\epsilon, \delta) = \frac{1}{2\epsilon^2} \log \frac{2m}{\delta}$ data points to guarantee that for all the m functions $h \in \mathcal{H}$ the value of $L_S(h)$ is close to the value of $L_D(h)$:

w.p $1-\delta$:

$$\forall h \in \mathcal{H} : |L_S(h) - L_D(h)| < \epsilon.$$

Now, Let h^*_S be the minimizer of ERM over the class $\mathcal{H}$:

$$h^*_S : \operatorname{argmin}_{h \in \mathcal{H}} L_S(h)$$

($L_S(h^*_S)$ is the smallest among $\mathcal{H}$)

Let $h^*_{true} : \operatorname{argmin}_{h \in \mathcal{H}} L_D(h)$

We will show that if the number of training data points is at least $n_0(\epsilon, \delta)$ then with $1-\delta$ we have

$$L_D(h^*_S) - \min_{h \in \mathcal{H}} L_D(h) \leq 2\epsilon.$$

Hence, $\mathcal{H}$ is PAC-learnable with $n_0(\epsilon, \delta)$ data points.

Let's see why (8) holds.

$$L_D(h^*_S) - L_D(h^*_{true})$$

$$= \underbrace{L_D(h^*_S) - L_S(h^*_S)}_{\leq \epsilon} + \underbrace{L_S(h^*_S) - L_S(h_{true})}_{\leq 0} + L_S(h^*_{true}) - L_D(h^*_{true})$$

(h^*_S is the minimizer of $L_S(\cdot)$)

$$\leq \epsilon + 0 + \epsilon \leq 2\epsilon.$$

Hence, if the number of training data points is at least $n_0(\epsilon, \delta) = \frac{1}{2\epsilon^2} \log \frac{2m}{\delta}$

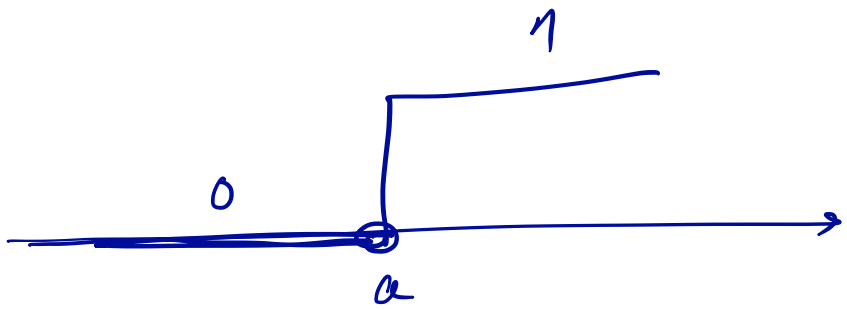
then w.p. $1-\delta$ we have

$$0 \leq L_D(h_S^*) - \min_{h \in \mathcal{H}} L_D(h) \leq 2\epsilon.$$

h_S^*: minimizer of ERM over $\mathcal{H}$.

When $\mathcal{H} = \{h_1, \dots, h_m\}$.

$h_a(x) =$



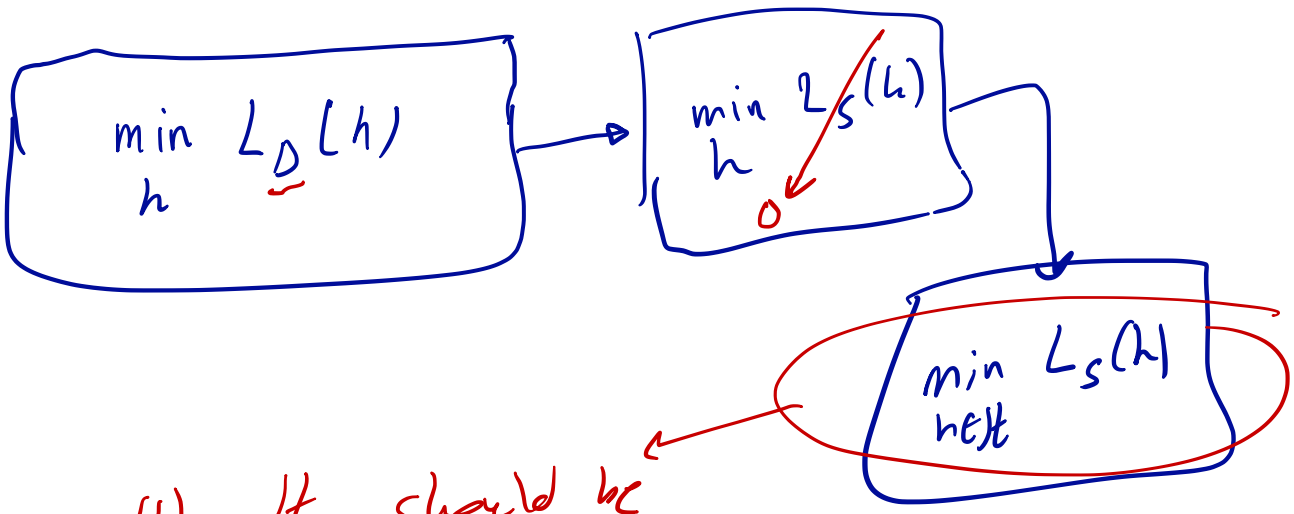
$$H = \{ h_a(x) \;,\; a \in (-\infty, \infty) \}$$

infinitely many functions

One important consequece of PAC-learnebility is that it's sufficient to work with a finite data set and we don't lose any thing in terms of generalization

Lecture 26:

- $(x, y) \sim D$ →



(1) H should be sufficiently complex

(2) Avoid overfitting: have sufficiently many data points

PAC: H is PAC learnable:

(1) $n_0(\epsilon, \delta) \in \mathbb{N}$

(2) learning alg. $S \longrightarrow h_S^* \in H$ (S: training data)

If $|S| \geq n_0(\epsilon, \delta)$:

w.p. $1-\delta$:

$$L_D(h_S^*) \leq \min_{h \in H} L_D(h) + \epsilon$$

$H = \{h_1, \dots, h_m\}$

- $n_0(\epsilon, \delta) = \frac{1}{\epsilon^2} \log \frac{2m}{\delta} \approx$ (up to constants) $\frac{\log \frac{1}{\delta} + \log m}{\epsilon^2}$

- learning algorithm: $h_S^* = \underset{i=1,\dots,m}{\operatorname{argmin}} L_S(h_i)$

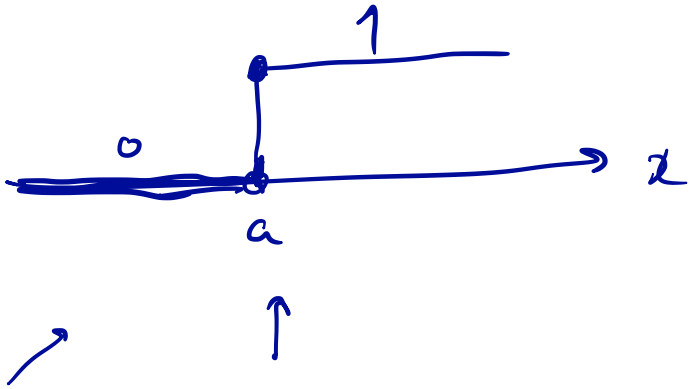
H is PAC-learnable with $n_0(\epsilon, \delta)$ samples and ERM_H as the learning algorithm.

---

what about infinite H?

$H = \{ h_a(x) : \mathbb{R} \to \{0,1\}, a \in \mathbb{R} \}$

$h_a(x) \to$



The fundamental theorem of learning thorg:

For any function class $\mathcal{H}$, if the number of training data point is larger than $n_0(\epsilon, \delta) = \frac{\log 1/\delta + \overbrace{VC\text{-}dim(\mathcal{H})}}{\epsilon^2}$, then

w.p. $1-\delta$:

$$L_D(h_S^*) \leq \min_{h \in \mathcal{H}} L_D(h) + \epsilon.$$

where $h_S^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}}\ L_S(h)$.

→ $VC\text{-}dim(\mathcal{H})$ = Complexity of the class

→ if $VC\text{-}dim(\mathcal{H})$ is finite, then

$\mathcal{H}$ is PAC-learnable with

$n_0(\epsilon, \delta) = \frac{\log 1/\delta + VC\text{-}dim(\mathcal{H})}{\epsilon^2}$ and

ERM as the learning Algorithm.

Important implication:

We can learn from finite data sets.

$$L_D(h_S^*) \leq \underbrace{\min_{h \in \mathcal{H}} L_D(h)}_{\text{the best that we could do}} + \epsilon$$

h_S^*: tractable

$\min_{h \in \mathcal{H}} L_D(h)$: intractable

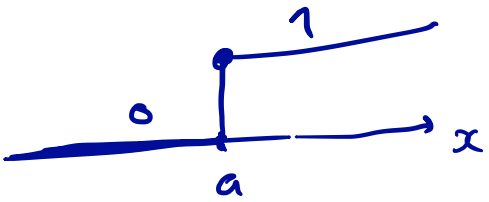
As long as we have sufficiently many data points we'll be fine with solving the "tractable" ERM-H problem and we lose at most a small value ϵ with respect to the optimal accuracy.

VC dimension:

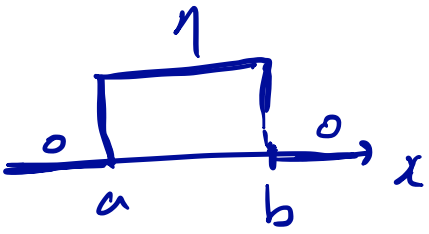
Vapnik-Chernovskii

Consider the following 4 function classes:

$$H_1 = \{ h_a(x),\ a \in \mathbb{R} \} \rightarrow$$

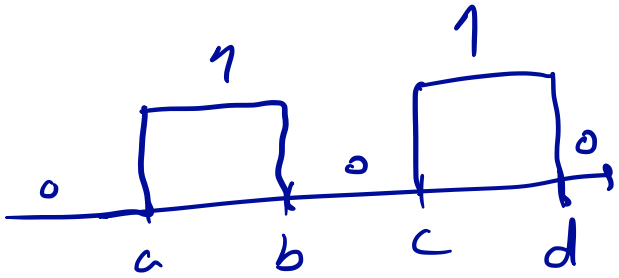


$$H_2 = \{ h_{ab}(x),\ a,b \in \mathbb{R} \} \rightarrow$$



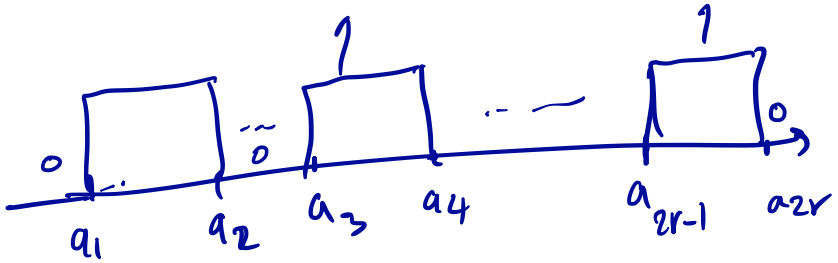
$$H_3 = \{ h_{abcd}(x),\ a,b,c,d \in \mathbb{R} \}$$

$h_{abcd}(x)$:



$$H_4 = \{ h_{a_1 a_2 - a_{2r}}(x),\ a_1, \dots, a_{2r} \in \mathbb{R} \}$$

$h_{a_1, a_2, \dots, a_{2r}}(x)$:



$H_1 < H_2 < H_3 < H_4 \rightarrow$ complexity

Definition: (restriction):

Let $\mathcal{H}$ be a class of functions from X to $\{0,1\}$ $(\forall h \in \mathcal{H},\ h: X \to \{0,1\})$.

Let $C = \{x_1, x_2, \ldots, x_k\} \subseteq X$.

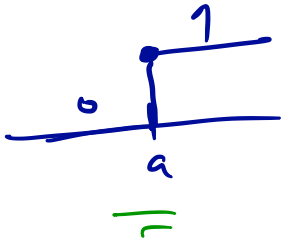
The restriction of $\mathcal{H}$ to C is the set of all the k-tuples

$$\mathcal{H}_C = \{ (h(x_1), h(x_2), \ldots, h(x_k)) ,\ h \in \mathcal{H} \}.$$

Example.

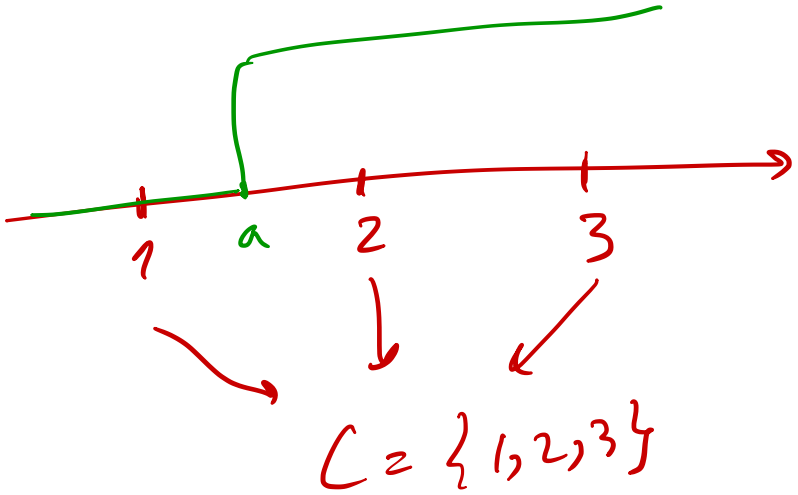
e.g. if $\mathcal{H} = \{ h_a(x) ,\ a \in \mathbb{R} \}$

$h_a(x) =$



Let $C = \{1, 2, 3\}$.

$$\mathcal{H}_C = \{ (h_a(1), h_a(2), h_a(3)) ,\ h_a \in \mathcal{H},\ a \in \mathbb{R} \}$$



$C = \{1, 2, 3\}$

$a < 1 : (h_a(1), h_a(2), h_a(3)) = (1, 1, 1)$

$a \in [1, 2) :$ " $= (0, 1, 1)$

$a \in [2, 3)$ " $= (0, 0, 1)$

$a \geq 3$ " $= (0, 0, 0)$

$$H_C = \{(1,1,1), (0,1,1), (0,0,1), (0,0,0)\}$$

$$|H_C| = 4 < 2^{|C|} = 2^3 = 8$$

↳ It is not sufficiently complex to produce all the possible binary 3-tuples on C.

Note:

If $C = \{x_1, \ldots, x_k\}$

$$|H_C| \leq 2^{|C|}$$

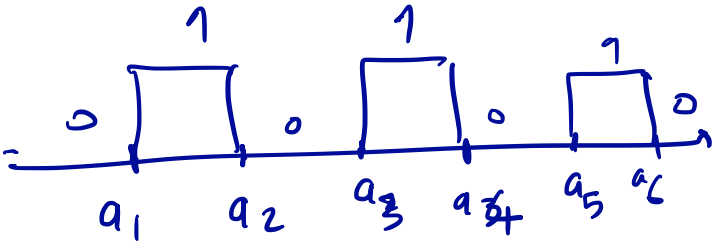
$$H_C = \underbrace{\{(0,0,\ldots,0), (1,0,\ldots,0), (0,1,0,\ldots,0), \ldots, (1,1,\ldots,1)\}}_{2^k}$$

Definition (Shattering): We say that a function class H shatters a set C if $|H_C| = 2^{|C|}$

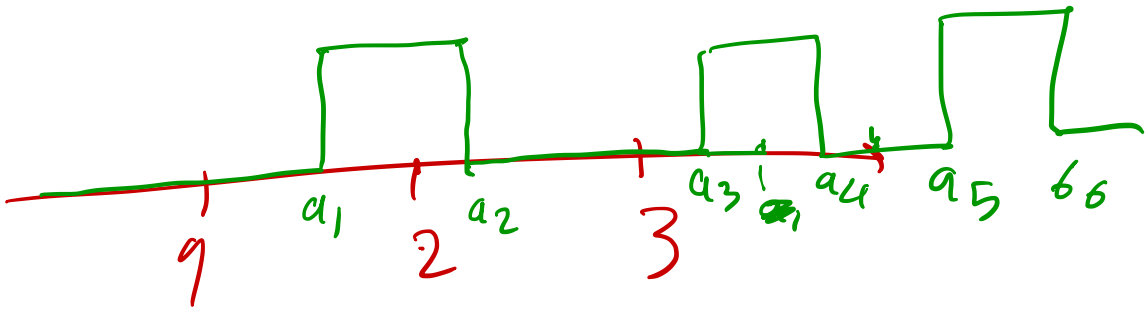
Example:

$$\mathcal{H} = \{ h_{a_1 a_2 a_3 a_4 a_5 a_6}(x) \;,\; a_1 \to a_6 \in \mathbb{R} \}$$

$h_{a_1 \to a_6}(x)$ =



$$C = \{1, 2, 3\}$$



$$\mathcal{H}_C = \{ (0,0,0), (1,0,0), (0,1,0), (0,0,1), \ldots, (1,1,1) \}$$

↳ $\mathcal{H}_C$ is all the 8 possible binary 3-tuples on C.

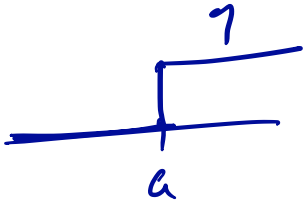
↳ It shatters C.

Definition (VC-dimension):

The VC-dimension of a function class $\mathcal{H}$ is the largest number k such that exists a set C of size k which is shattered by $\mathcal{H}$.

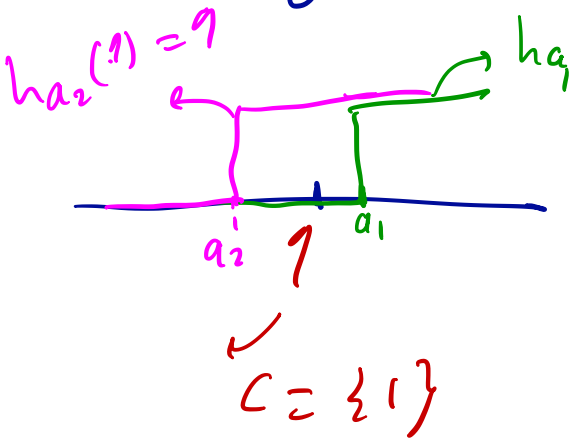
Examples:

$\mathcal{H} = \{ h_a(x), \; a \in \mathbb{R} \}$ →



VC-dim($\mathcal{H}$) =

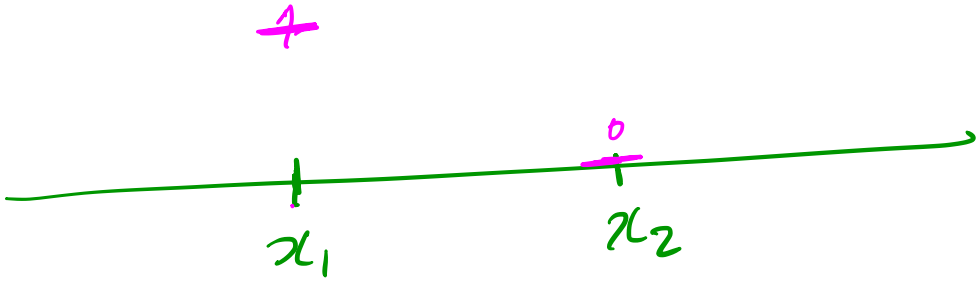
Is there a set of size 1 that's shattered by $\mathcal{H}$? YES.



$\rightsquigarrow H_C = \{0, 1\}$

↳ $\mathcal{H}$ shatters C.

Is there a set of size 2 which is shattered by $\mathcal{H}$? No



$C = \{x_1, x_2\}$ (assume $x_1 < x_2$)

↳ $\mathcal{H}_C = \{(-,-), (+,-), (-,+), (+,+)\}$

↳ there is no $h_a(x)$ that produces this binary 2-tuple

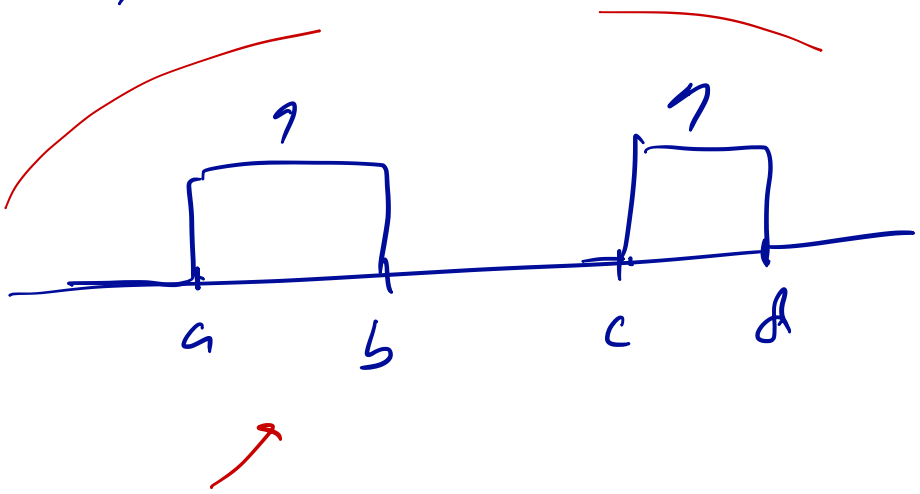
$VC\text{-}dim(\mathcal{H}) = 1$

⇓

there exists no set C of size 2 which is shattered by $\mathcal{H}$

Example:

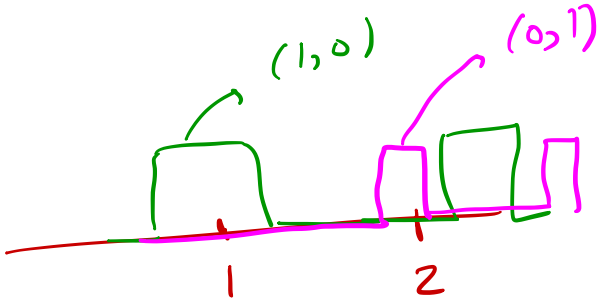
$$\mathcal{H} = \{ h_{a,b,c,d}(x) : a,b,c,d \in \mathbb{R} \}$$



is there a set C of size k which is shattered by $\mathcal{H}$?

$k \geq 1 \longrightarrow$ YES

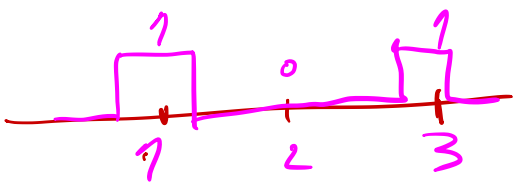
$k = 2 \longrightarrow$ YES



$k = 3 \longrightarrow$ YES

$k = 4 \longrightarrow$ YES

$k = 5 \longrightarrow$ NO



$$\mathcal{H}_C = \{ (0,0,0), (1,0,0), (0,1,0), (0,0,1), (1,0,1), (1,1,0), (0,1,1), (1,1,1) \}$$

$K = 5$

h should have 3 jumps from 0 to 1.