

# Advanced Machine Learning



Bogdan Alexe,

[bogdan.alexe@fmi.unibuc.ro](mailto:bogdan.alexe@fmi.unibuc.ro)

University of Bucharest, 2<sup>nd</sup> semester, 2019-2020

# Assignment 1 updated

## Assignment 1

Deadline: Friday, 17<sup>th</sup> of April

1. (0.5 points) Consider  $\mathcal{H} = \{h_{\theta_1}: \mathbb{R} \rightarrow \{0,1\}, h_{\theta_1}(x) = \mathbf{1}_{[x \geq \theta_1]}(x) = \mathbf{1}_{[\theta_1, +\infty)}(x), \theta_1 \in \mathbb{R}\} \cup \{h_{\theta_2}(x) = \mathbf{1}_{[x < \theta_2]}(x) = \mathbf{1}_{(-\infty, \theta_2)}(x), \theta_2 \in \mathbb{R}\}$ . Compute  $\text{VCdim}(\mathcal{H})$ .
2. (0.75 points) Consider  $\mathcal{H}$  to be the class of all centered in origin sphere classifiers in the 3D space. A centered in origin sphere classifier in the 3D space is a classifier  $h_r$  that assigns the value 1 to a point if and only if it is inside the sphere with radius  $r > 0$  and center given by the origin  $O(0,0,0)$ . Consider the realizability assumption.
  - a. show that the class  $\mathcal{H}$  can be  $(\epsilon, \delta)$  – PAC learned by giving the algorithm A and determining the sample complexity  $m_{\mathcal{H}}(\epsilon, \delta)$  such that the definition of PAC-learnability is satisfied. (0.5 points)
  - b. compute  $\text{VCdim}(\mathcal{H})$ . (0.25 points)
3. (1 point) Let  $\mathcal{H} = \{h_{\theta}: \mathbb{R} \rightarrow \{0,1\}, h_{\theta}(x) = \mathbf{1}_{[\theta, \theta+1] \cup [\theta+2, +\infty)}(x), \theta \in \mathbb{R}\}$ . Compute  $\text{VCdim}(\mathcal{H})$ .

# Materials uploaded on Moodle



# Materials uploaded on Moodle

- seminar 1 solutions available
- solutions written in Latex for seminars 2 and 3 provided by your colleague Micluța-Câmpeanu Marius available here:  
[https://drive.google.com/drive/folders/1vPcjr3cye0Rg6IXsN\\_gMTFY\\_Q2uNJLh](https://drive.google.com/drive/folders/1vPcjr3cye0Rg6IXsN_gMTFY_Q2uNJLh)
- seminar 2 in Latex uploaded also by me on Moodle
- I didn't have time to check the Latex sources for seminar 3
- today we have seminar 4 – VCdim stuff

# Recap: The fundamental theorem of statistical learning

**Theorem** (The Fundamental Theorem of Statistical Learning).

Let  $\mathcal{H}$  be a hypothesis class of functions from a domain  $\mathcal{X}$  to  $\{0,1\}$  and let the loss function be the 0–1 loss. Then, the following are equivalent:

1.  $\mathcal{H}$  has the uniform convergence property.
2. Any ERM rule is a successful agnostic PAC learner for  $\mathcal{H}$ .
3.  $\mathcal{H}$  is agnostic PAC learnable.
4.  $\mathcal{H}$  is PAC learnable.
5. Any ERM rule is a successful PAC learner for  $\mathcal{H}$ .
6.  $\mathcal{H}$  has a finite VC-dimension.

*A finite VC- dimension guarantees learnability. Hence, the VC-dimension characterizes PAC learnability.*

# The Growth function

## Definition

Let  $\mathcal{H}$  be a hypothesis class. Then the growth function of  $\mathcal{H}$ , denoted by  $\tau_{\mathcal{H}}$ , where  $\tau_{\mathcal{H}}: \mathbf{N} \rightarrow \mathbf{N}$ , is defined as:

$$\tau_{\mathcal{H}}(m) = \max_{C \subseteq X: |C|=m} |H_C|$$

*In other words,  $\tau_{\mathcal{H}}(m)$  is the maximum number of different functions from a set  $C$  of size  $m$  to  $\{0,1\}$  that can be obtained by restricting  $\mathcal{H}$  to  $C$ .*

**Observation:** if  $\text{VCdim}(\mathcal{H}) = d$  then for any  $m \leq d$  we have  $\tau_{\mathcal{H}}(m) = 2^m$ . In such cases,  $\mathcal{H}$  induces all possible functions from  $C$  to  $\{0,1\}$ .

*What happens when  $m$  becomes larger than the VC-dimension?*

Answer given by the Sauer's lemma: the growth function  $\tau_{\mathcal{H}}$  increases polynomially rather than exponentially with  $m$ .

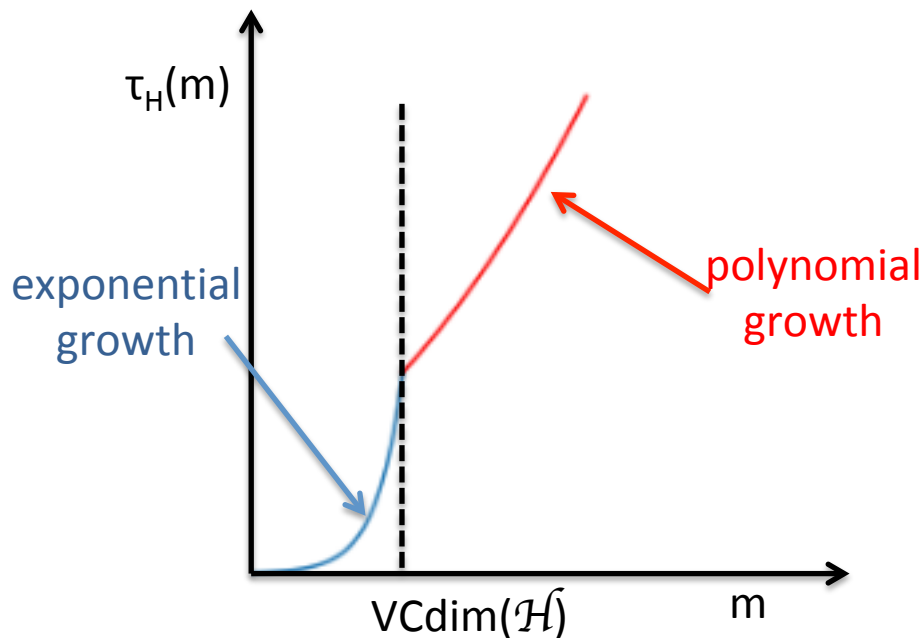
# The Sauer's lemma

## Lemma (Sauer – Shelah – Perles)

Let  $\mathcal{H}$  be a hypothesis class with  $\text{VCdim}(\mathcal{H}) \leq d < \infty$ . Then, for all  $m$ , we have that:

$$\tau_{\mathcal{H}}(m) \leq \sum_{i=0}^d C_m^i$$

In particular, if  $m > d + 1$  then  $\tau_{\mathcal{H}}(m) \leq (em/d)^d = O(m^d)$



# Proof for $6 \rightarrow 1$

We want to prove that finite VC-dimension  $\rightarrow$  *uniform convergence property*

## Two steps:

1. (Sauer's lemma) If  $\text{VCdim}(\mathcal{H}) \leq d < \infty$ , then even though  $\mathcal{H}$  might be infinite, when restricting it to a finite set  $C \subseteq \mathcal{X}$ , its “effective” size,  $|\mathcal{H}_C|$ , is only  $O(|C|^d)$ . That is, the size of  $\mathcal{H}_C$  grows polynomially rather than exponentially with  $|C|$ .
2. we have shown in lecture 4 that finite hypothesis classes enjoy the uniform convergence property. We generalize this result and show that uniform convergence holds whenever the hypothesis class has a “small effective size.” By “small effective size” we mean classes for which  $|\mathcal{H}_C|$  grows polynomially with  $|C|$ .



# The fundamental theorem of statistical learning – quantitative version

## Theorem

Let  $\mathcal{H}$  be a hypothesis class of functions from a domain  $\mathcal{X}$  to  $\{0,1\}$  and let the loss function be the 0–1 loss. Assume that  $\text{VCdim}(\mathcal{H}) = d < \infty$ . Then, there are absolute constants  $C_1, C_2$  such that:

1.  $\mathcal{H}$  has the uniform convergence property with sample complexity:

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}^{\text{UC}}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}$$

2.  $\mathcal{H}$  is agnostic PAC learnable with sample complexity:

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}$$

3.  $\mathcal{H}$  is PAC learnable with sample complexity:

$$C_1 \frac{d + \log(1/\delta)}{\epsilon} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon}$$

The VC dimension determines (along with  $\epsilon, \delta$ ) the samples complexities of learning a class. It gives us a lower and an upper bound.

# Computational complexity of learning

# Computational resources of learning

For learning we need 2 type of resources:

## 1. *Information = training data*

- so far we analyzed how much training data (sample size) we need in order to learn
- *sample complexity*

## 2. *Computation = runtime*

- for how much time an algorithm (that implements learning) will run, once we have sufficiently many training examples
- *computational complexity*
- crucial when we need fast ML applications (driver surveillance, stock exchange trading, etc)
- runtime = number of elementary instructions executed - arithmetic operations over real numbers - in an asymptotic sense (with respect to input size) of the algorithm, e.g.  $O(n)$  – where  $n$  is the size of the input size

# Input size parameter of learning

What should play the role of the input size parameter in learning?

- size of the training set that the algorithm receives?
  - for a very large number of examples, much larger than the sample complexity of learning, the algorithm can ignore the extra samples
  - a larger training set does not make the problem more difficult
- size of the hypothesis class?
  - might be infinite:  $|\mathcal{H}_{\text{thresholds}}| = \infty$
- accuracy  $\varepsilon$ , confidence  $\delta$  and another parameter  $n$  related to the size/complexity of  $\mathcal{X}$ ,  $\mathcal{H}$ 
  - how much computation we need in order to get accuracy  $\varepsilon$  with confidence  $\delta$
  - want to have *efficient learning* (give a formal definition later): polynomial in  $1/\varepsilon$ ,  $1/\delta$  and  $n$  (some parameter related to the size/complexity of domain/hypothesis class: more complex hypothesis needs more computation time)

# Input size parameter of learning

What should play the role of the input size parameter in learning?

- accuracy  $\varepsilon$ , confidence  $\delta$  and another parameter  $n$  related to the size/complexity of  $\mathcal{X}$ ,  $\mathcal{H}$ 
  - how much computation we need in order to get accuracy  $\varepsilon$  with confidence  $\delta$
  - want to have *efficient learning* (give a formal definition later): polynomial in  $1/\varepsilon$ ,  $1/\delta$  and  $n$  (some parameter related to the size/complexity of domain/hypothesis class: more complex hypothesis needs more computation time)
- parameter  $n$  can be the embedding dimension
  - if we decide to use  $n$  features to describe objects, how will that increase runtime?
- we study the runtime in an asymptotic sense by defining a sequence of pairs  $(\mathcal{X}_n, \mathcal{H}_n)_{n=1,2,\dots}$  and studying asymptotic complexity of learning  $\mathcal{X}_n$ ,  $\mathcal{H}_n$  as  $n$  grows to  $\infty$

# Prevent “cheating”

The output of the learning algorithm  $L$  is a hypothesis  $h$  from  $\mathcal{H}$ .

- a learning algorithm  $L$  can “cheat” by transferring the computational burden to the output hypothesis
  - define the output hypothesis to be the function that stores the training set in memory and computes the ERM hypothesis on the training set and applies it to a test example  $x$
- the runtime of a learning algorithm  $A$  defined as the maximum of:
  - the time it takes  $A$  to output some  $h$
  - the time it takes  $h$  to output a label on any given  $x$  from  $\mathcal{X}$

# Example 1: Conjunctions of Boolean literals

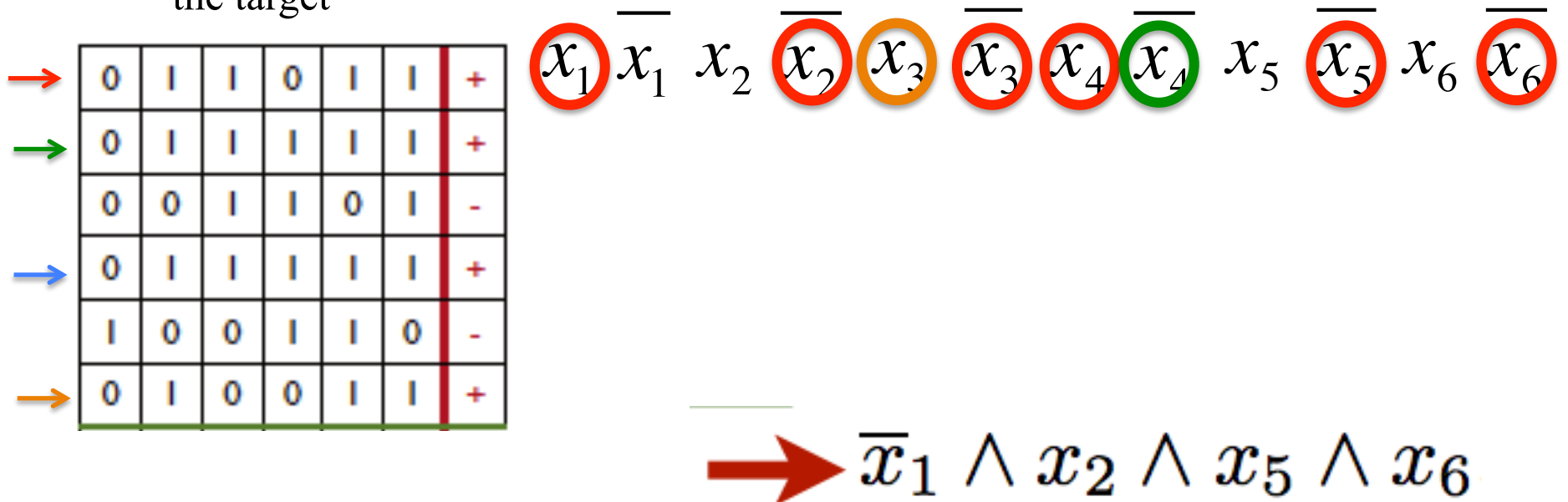
- $\mathcal{H}_{\text{conj}}^d$  = class of conjunctions of at most  $d$  Boolean literals  $x_1, \dots, x_d$ 
  - a Boolean literal is either  $x_i$  or its negation  $\overline{x_i}$
  - can interpret  $x_i$  as feature  $i$
  - example:  $h = x_1 \wedge \overline{x_2} \wedge x_4$  where  $\overline{x_2}$  denotes the negation of the Boolean literal  $x_2$
  - $\mathcal{X} = \{0,1\}^d$
- consider the realizable case
  - there is a conjunction  $h^*$  in  $\mathcal{H}_{\text{conj}}^d$  that labels the examples
- $|\mathcal{H}_{\text{conj}}^d| = 3^d + 1 < \infty$  so it has finite VC dimension (less than  $\log_2(3^d + 1)$ ), so it's PAC learnable. In seminar class 3 we shown that  $\text{VCdim}(\mathcal{H}_{\text{conj}}^d) = d$  so the sample complexity  $m_{\mathcal{H}}(\epsilon, \delta)$  is bounded by:

$$C_1 \frac{d + \log(1/\delta)}{\epsilon} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon}$$

- So,  $m_{\mathcal{H}}(\epsilon, \delta)$  is polynomial in  $1/\epsilon$ ,  $1/\delta$ ,  $d$  (measures the complexity of the hypothesis class  $\mathcal{H}_{\text{conj}}^d$ )

# Example 1: Conjunctions of Boolean literals

- $\mathcal{H}_{\text{conj}}^d$  = class of conjunctions of at most  $d$  Boolean literals  $x_1, \dots, x_d$
- a simple algorithm for finding an ERM hypothesis is based on positive examples and consists of the following:
  - for each positive example  $(b_1, \dots, b_d)$ ,
    - if  $b_i = 1$  then  $\overline{x_i}$  is ruled out as a possible literal in the concept class
    - if  $b_i = 0$  then  $x_i$  is ruled out.
  - the conjunction of all the literals not ruled out is thus a hypothesis consistent with the target





# Example 1: Conjunctions of Boolean literals

- $\mathcal{H}_{\text{conj}}^d$  = class of conjunctions of at most  $d$  Boolean literals  $x_1, \dots, x_d$
- a simple algorithm for finding an ERM hypothesis is based on positive examples and consists of the following:
  - for each positive example  $(b_1, \dots, b_n)$ ,
    - if  $b_i = 1$  then  $\overline{x_i}$  is ruled out as a possible literal in the concept class
    - if  $b_i = 0$  then  $x_i$  is ruled out.
  - the conjunction of all the literals not ruled out is thus a hypothesis consistent with the target
- runtime of the algorithm is  $O(m_{\mathcal{H}}(\varepsilon, \delta) * d)$ , so is polynomial in  $1/\varepsilon, 1/\delta, d$
- in the agnostic (unrealizable) case: unless  $P = NP$ , there is no algorithm whose running time is polynomial in  $m_{\mathcal{H}}(\varepsilon, \delta)$  and  $d$  that is guaranteed to find an ERM hypothesis for the class of Boolean conjunctions.

# Example 2: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{\text{rec}}^d$  = the class of axis aligned rectangles in  $\mathbf{R}^d$

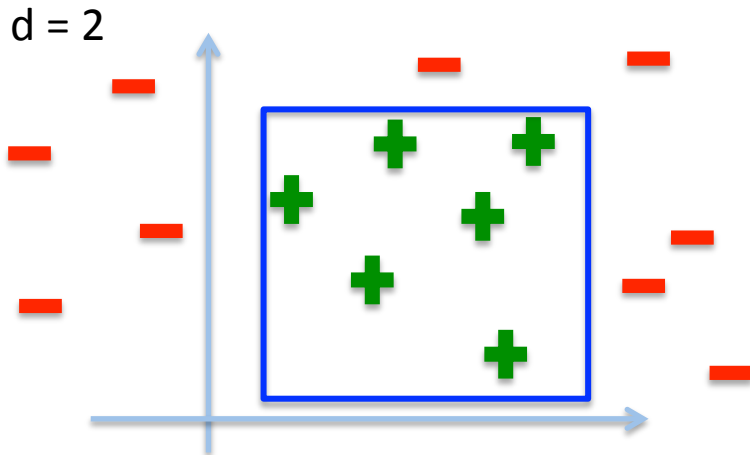
$$H_{\text{rec}}^d = \{h_{a_1, b_1, a_2, b_2, \dots, a_d, b_d} : \mathbf{R}^d \rightarrow \{0, 1\} \mid a_1 \leq b_1, a_2 \leq b_2, \dots, a_d \leq b_d, a_i \in \mathbf{R}, b_i \in \mathbf{R}\}$$

$$h_{a_1, b_1, a_2, b_2, \dots, a_d, b_d}(x_1, x_2, \dots, x_d) = \begin{cases} 1, & \text{if } a_1 \leq x_1 \leq b_1, a_2 \leq x_2 \leq b_2, \dots, a_d \leq x_d \leq b_d \\ 0, & \text{otherwise} \end{cases}$$

- consider the realizable case:
  - there exists a rectangle  $h^*$  in  $\mathcal{H}_{\text{rec}}^d$  with real risk = 0

We have shown in the seminar class that:

- the algorithm that returns the rectangle enclosing all positive examples is ERM
- $\mathcal{H}_{\text{rec}}^d$  is PAC learnable with sample size



$$m_{H_{\text{rec}}^d}(\epsilon, \delta) \leq \left\lceil \frac{2d \log\left(\frac{2d}{\delta}\right)}{\epsilon} \right\rceil$$

- the runtime is  $O(m_H d)$  as for each dimension, the algorithm has to find the minimal and the maximal values among the positive instances in the training sequence. So it is polynomial in  $1/\epsilon, 1/\delta, d$ .

## Example 2: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{\text{rec}}^d$  = the class of axis aligned rectangles in  $\mathbf{R}^d$

$$H_{\text{rec}}^d = \{h_{a_1, b_1, a_2, b_2, \dots, a_d, b_d} : \mathbf{R}^d \rightarrow \{0, 1\} \mid a_1 \leq b_1, a_2 \leq b_2, \dots, a_d \leq b_d, a_i \in \mathbf{R}, b_i \in \mathbf{R}\}$$

$$h_{a_1, b_1, a_2, b_2, \dots, a_d, b_d}(x_1, x_2, \dots, x_d) = \begin{cases} 1, & \text{if } a_1 \leq x_1 \leq b_1, a_2 \leq x_2 \leq b_2, \dots, a_d \leq x_d \leq b_d \\ 0, & \text{otherwise} \end{cases}$$

- consider the agnostic case:
  - distribution  $\mathcal{D}$  over  $\mathcal{Z} = \mathbf{R}^d \times \{0, 1\}$  (a sample could get both labels)
  - if there exist a labeling function  $f$  this might not be in  $\mathcal{H}_{\text{rec}}^d$
- $\text{VCdim}(\mathcal{H}_{\text{rec}}^d) = 2d$  (see seminar class), so we have that:

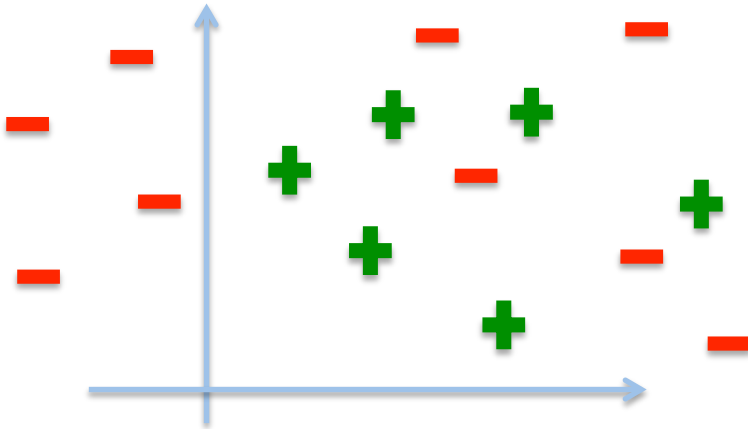
$$C_1 \frac{2d + \log(\frac{1}{\delta})}{\varepsilon^2} \leq m_{H_{\text{rec}}^d}(\varepsilon, \delta) \leq C_2 \frac{2d + \log(\frac{1}{\delta})}{\varepsilon^2}$$

- $m_{H_{\text{rec}}^d}(\varepsilon, \delta)$  is polynomial in  $1/\varepsilon$ ,  $1/\delta$ ,  $d$  (measures the complexity of the  $\mathcal{H}_{\text{rec}}^d$ )

## Example 2: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{\text{rec}}^d$  = the class of axis aligned rectangles in  $\mathbf{R}^d$
- consider that we have a sample  $S$  of size:  $m_{H_{\text{rec}}^d}(\epsilon, \delta) \approx C \frac{2d + \log(\frac{1}{\delta})}{\epsilon^2}$
- what is the runtime of the ERM algorithm?
  - how long it will take to find the best rectangle in  $\mathbf{R}^d$ ?
  - go over all axis aligned rectangles in  $\mathbf{R}^d$  and choose the best one (based on minimizing the error on the training data)

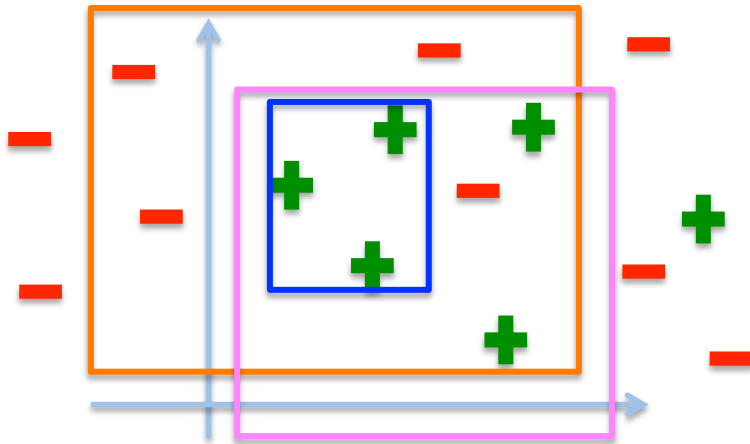
$d = 2$



## Example 2: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{\text{rec}}^d$  = the class of axis aligned rectangles in  $\mathbf{R}^d$
- consider that we have a sample  $S$  of size:  $m_{H_{\text{rec}}^d}(\varepsilon, \delta) \approx C \frac{2d + \log(\frac{1}{\delta})}{\varepsilon^2}$
- what is the runtime of the ERM algorithm?
  - how long it will take to find the best rectangle in  $\mathbf{R}^d$ ?
  - go over all axis aligned rectangles in  $\mathbf{R}^d$  and choose the best one (based on minimizing the error on the training data)

$d = 2$



error:  $(1 + 4) / (6 + 9) = 5/15$

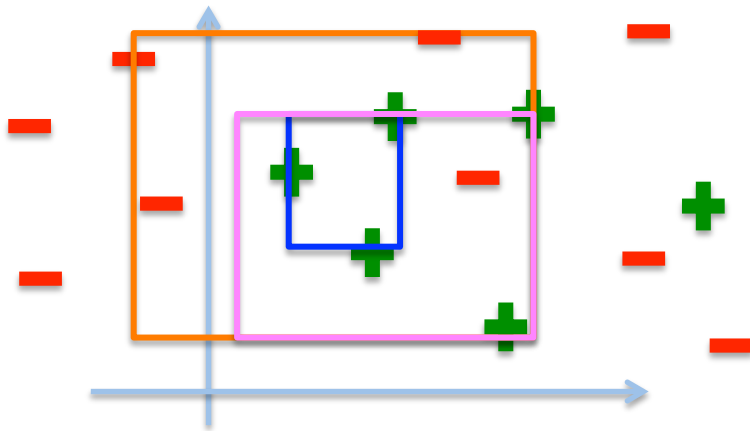
error:  $(3 + 0) / (6 + 9) = 3/15$

error:  $(1 + 1) / (6 + 9) = 2/15$

## Example 2: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{\text{rec}}^d$  = the class of axis aligned rectangles in  $\mathbf{R}^d$
- consider that we have a sample  $S$  of size:  $m_{H_{\text{rec}}^d}(\epsilon, \delta) \approx C \frac{2d + \log(\frac{1}{\delta})}{\epsilon^2}$
- what is the runtime of the ERM algorithm?
  - how long it will take to find the best rectangle in  $\mathbf{R}^d$ ?
  - go over all axis aligned rectangles in  $\mathbf{R}^d$  and choose the best one (based on minimizing the error on the training data)

$d = 2$



error:  $(1 + 4) / (6 + 9) = 5/15$

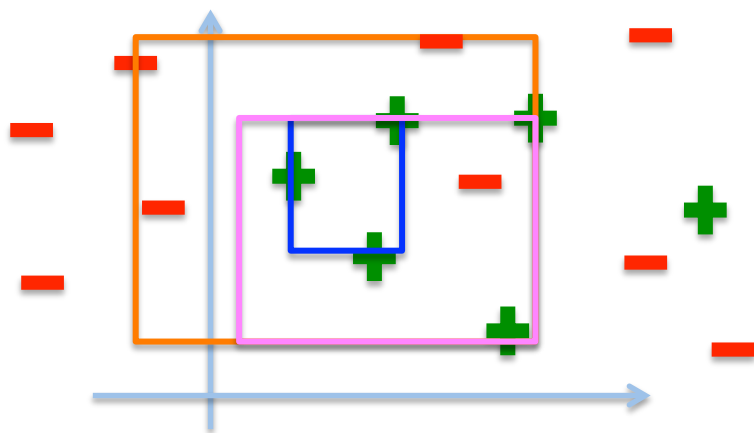
error:  $(3 + 0) / (6 + 9) = 3/15$

error:  $(1 + 1) / (6 + 9) = 2/15$

- the number of all possible rectangles can be reduced to all possible rectangles that have points of  $S$  on every boundary edge (very efficient algorithm)

## Example 2: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{\text{rec}}^d$  = the class of axis aligned rectangles in  $\mathbf{R}^d$
- consider that we have a sample  $S$  of size:  $m_{H_{\text{rec}}^d}(\varepsilon, \delta) \approx C \frac{2d + \log(\frac{1}{\delta})}{\varepsilon^2}$
- what is the runtime of the ERM algorithm?
  - how long it will take to find the best rectangle in  $\mathbf{R}^d$ ?
  - Step 1: generate all the rectangles based on the sample points in  $\mathbf{R}^d$
  - Step 2: for each such rectangle compute the training error
  - Step 3: choose the rectangle with the smallest training error



error:  $(1 + 4) / (6 + 9) = 5/15$

error:  $(3 + 0) / (6 + 9) = 3/15$

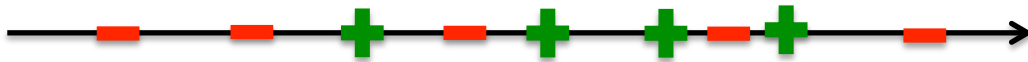
error:  $(1 + 1) / (6 + 9) = 2/15$

- how many possible rectangles can we construct based on the points in the sample  $S$ ?

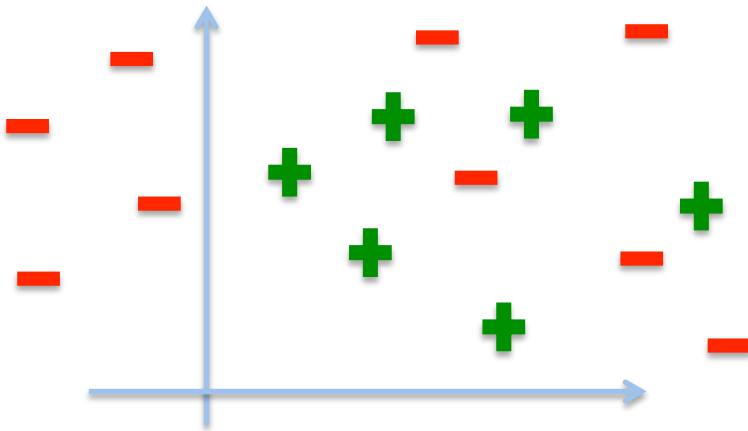
## Example 2: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{\text{rec}}^d$  = the class of axis aligned rectangles in  $\mathbf{R}^d$
- how many possible rectangles can we construct based on the points in the sample  $S$ ?

$d = 1$



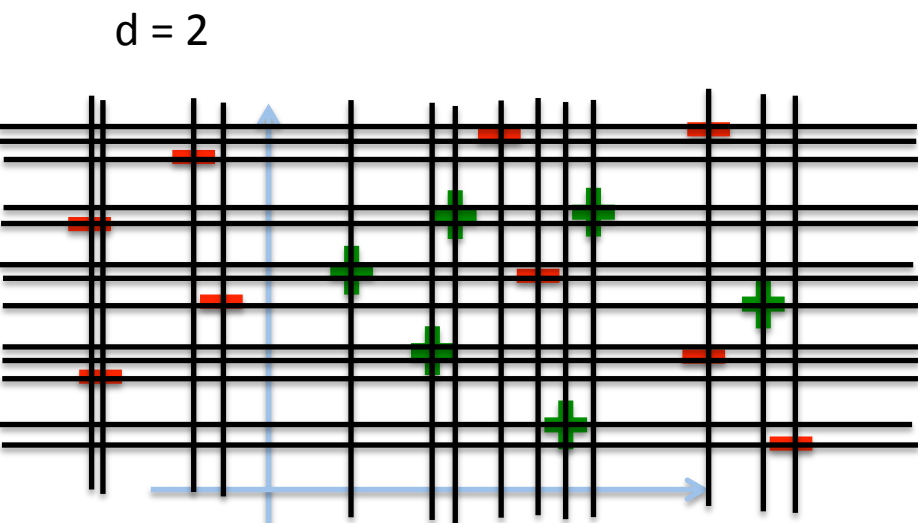
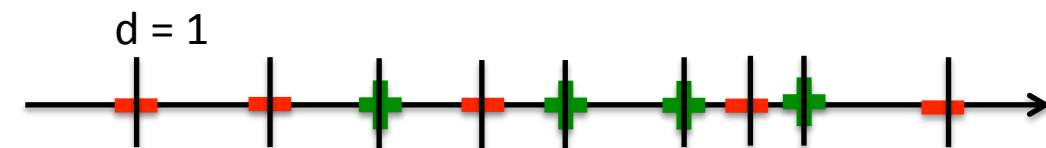
$d = 2$





## Example 2: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{\text{rec}}^d$  = the class of axis aligned rectangles in  $\mathbf{R}^d$
- how many possible rectangles can we construct based on the points in the sample  $S$ ?



Every such rectangle is determined by at most  $2d$  points from  $S$

So there are at most  $|S|^{2d}$  such rectangles.

For each rectangle we need to iterate over all examples to compute the training error.

So, the runtime is:  $O\left(\left[C \frac{2d + \log(\frac{1}{\delta})}{\varepsilon^2}\right]^{2d+1}\right)$

## Example 2: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{\text{rec}}^d$  = the class of axis aligned rectangles in  $\mathbf{R}^d$

- the runtime of the  $\text{ERM}_H$  is:  $O\left(\left[C \frac{2d + \log(\frac{1}{\delta})}{\epsilon^2}\right]^{2d+1}\right)$

- for every fixed dimension  $d$ ,  $\text{ERM}_H$  can be implemented in time which is polynomial in  $1/\epsilon$ ,  $1/\delta$ ,  $d$  (measures the complexity of the  $\mathcal{H}_{\text{rec}}^d$ ) therefore we have efficient learning (see the formal definition later)

- however, as a function of  $d$  the runtime of the algorithm implementing the  $\text{ERM}_H$  presented is exponential in  $d$ . It can be proved that there is no better algorithm (unless  $P = NP$ ) than the one proposed.

# Formal definition of efficient learning

## Definition 1

Given a function  $f : (0,1)^2 \rightarrow \mathbb{N}$ , a learning task  $(\mathcal{Z}, \mathcal{H}, \ell)$ , and a learning algorithm  $A$ , we say that  $A$  solves the learning task in time  $O(f)$  if there exists some constant number  $c$ , such that for every probability distribution  $\mathcal{D}$  over  $\mathcal{Z}$ , and input  $\varepsilon, \delta \in (0,1)$ , when  $A$  has access to samples generated i.i.d by  $\mathcal{D}$ , we have that:

- $A$  terminates after performing at most  $c * f(\varepsilon, \delta)$  operations;
- the output of  $A$ , denoted  $h_A$ , can be applied to predict the label of a new example while performing at most  $c * f(\varepsilon, \delta)$  operations;
- the output of  $A$  is probably approximately correct; namely, with probability of at least  $1 - \delta$  (over the random samples  $A$  receives):

$$L_{\mathcal{D}}(h_A) \leq \min_h L_{\mathcal{D}}(h) + \varepsilon$$

# Formal definition of efficient learning

## Definition (for graded hypothesis spaces)

Consider a sequence of learning problems,  $(Z_n, \mathcal{H}_n, \ell_n)_{n=1,2,\dots}$  where problem  $n$  is defined by a domain  $Z_n$ , a hypothesis class  $\mathcal{H}_n$ , and a loss function  $\ell_n$ . Let  $A$  be a learning algorithm designed for solving learning problems of this form. Given a function  $g: \mathbb{N} \times (0,1)^2 \rightarrow \mathbb{N}$ , we say that the runtime of  $A$  with respect to the preceding sequence is  $O(g)$ , if for all  $n$ ,  $A$  solves the problem  $(Z_n, \mathcal{H}_n, \ell_n)$  in time  $O(f_n)$ , where  $f_n: (0,1)^2 \rightarrow \mathbb{N}$  is defined by  $f_n(\epsilon, \delta) = g(n, \epsilon, \delta)$ .

We say that  $A$  is an *efficient* PAC algorithm with respect to a sequence  $(Z_n, \mathcal{H}_n, \ell_n)$  if its runtime is  $O(p(n, 1/\epsilon, 1/\delta))$  for some polynomial  $p$ .

# Formal definition of efficient PAC learning (Valiant 1984)

In 1984, Leslie Valiant defined efficient PAC learning: PAC learnability + require the number of examples and the runtime of the algorithm  $A$  (training + testing) to be polynomial in  $1/\epsilon$ ,  $1/\delta$ ,  $n$ .

## Example:

- $\mathcal{U}_n = \{h: B^n \rightarrow \{0,1\}\}$  - the concept class formed by all subsets of  $B^n$
- $|\mathcal{U}_n| = 2^{2^n}$  - finite, so is PAC learnable with  $m_{\mathcal{H}}(\epsilon, \delta)$  in the order of  $m$ :

$$m \geq \left\lceil \frac{1}{\epsilon} \left( 2^n \log(2) + \log\left(\frac{1}{\delta}\right) \right) \right\rceil$$

- sample complexity exponential in  $n$ , number of variables
- it is not efficient PAC-learnable in any practical sense (need polynomial sample complexity)

# Relation between consistency and PAC learning

- a learning rule is consistent if:
  - input:  $\mathcal{H}$  and  $S = (x_1, y_1), \dots, (x_m, y_m)$
  - output:  $h \in \mathcal{H}$ ,  $h$  is an ERM hypothesis, i.e.  $h(x_i) = y_i$
- if  $\text{VCdim}(\mathcal{H}) \leq d$  then  $\mathcal{H}$  is PAC learnable (in the realizable case) by the consistent rule with sample complexity:

$$C_1 \frac{d + \log(1/\delta)}{\epsilon} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon}$$

- if  $d$  is polynomially in  $n$  and the consistent rule has runtime polynomial in the sample size  $m_{\mathcal{H}}(\epsilon, \delta)$  then we have efficient PAC learning
- so, efficient ‘consistent-hypothesis-finder’  $\rightarrow$  efficient PAC learning
- does the converse implication holds?
  - yes, based on randomised algorithms

# Relation between consistency and PAC learning

## **Theorem**

Let  $\mathcal{H}$  be a hypothesis class that is efficient PAC learnable. Then, there exists a randomised algorithm which solves the problem of finding a hypothesis in  $\mathcal{H}$  consistent with a given training sample, and which has runtime polynomial in  $m$  (the length of the training sample).