Lecture 9: Computational Learning Theory

Winter 2018

Kai-Wei Chang

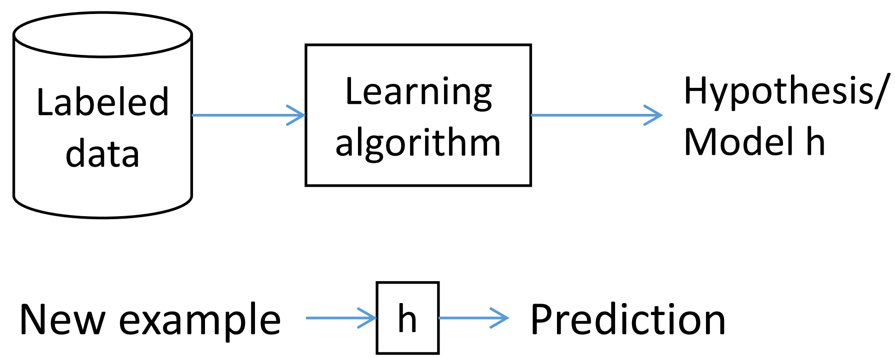
CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikuar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Checkpoint: The bigger picture

- Supervised learning: instances, concepts, and hypotheses
- Specific learners
  - Decision trees
  - K-NN
  - Perceptron
  - Logistic regression
- General ML ideas
  - Features as high dimensional vectors
  - Overfitting
  - Probabilistic model



This lecture: Computational Learning Theory

- The Theory of Generalization
- Probably Approximately Correct (PAC) learning
- Shattering and the VC dimension

The Theory of Generalization

- The Theory of Generalization
  - When can be trust the learning algorithm?
  - What functions can be learned?
  - What is the meaning of learning

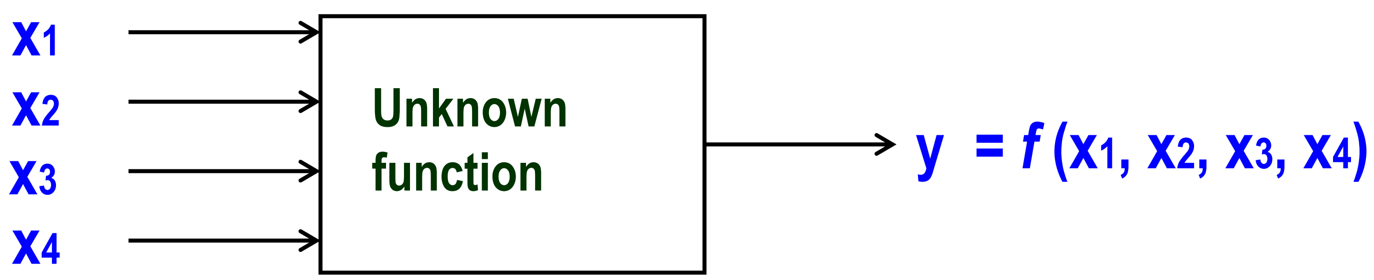
Computational Learning Theory

Are there general "laws of nature" related to learnability?

We want theory that can relate

- Probability of successful Learning
- Number of training examples
- Complexity of hypothesis space
- Accuracy to which target concept is approximated
- Manner in which training examples are presented

A Learning Problem



Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Can you learn this function?

What is it?

A function g is consistent to a dataset $\mathrm{D} = \{(x_i, y_i)\}$ if $g(x_i) = y_i, \forall\ i$

How many possible functions over four features?

How many function is consistent to D on the left

Hypothesis Space

Complete Ignorance:
There are $2^{16} = 65536$ possible functions over four input features.

We can't figure out which one is correct until we've seen every possible input-output pair.

After observing seven examples we still have 2^9 possibilities for f

Is Learning Possible?

Example	x_1	x_2	x_3	x_4	y
	0	0	0	0	?
	0	0	0	1	?
	0	0	1	0	0
	0	0	1	1	1
	0	1	0	0	0
	0	1	0	1	0
	0	1	1	0	0
	0	1	1	1	?
	1	0	0	0	?
	1	0	0	1	0
	1	0	1	0	?
	1	0	1	1	?
	1	1	0	0	0
	1	1	0	1	?
	1	1	1	0	?
	1	1	1	1	?

Hypothesis Space

Complete Ignorance:
There are 2^{16} = 65536 possible functions over four input features.

We c
corre
poss

After
have 2^9 possibilities for f

- There are $|Y|^{|X|}$ possible functions f(**x**) from the instance space **X** to the label space **Y.**
- Learners typically consider only a *subset* of the functions from **X** to **Y**, called the hypothesis space **H** . **H** $\subseteq |Y|^{|X|}$

Is Learning Possible?

Example	x_1	x_2	x_3	x_4	y
	0	0	0	0	?
	0	0	0	1	?
	0	0	1	0	0
					1
					0
					0
					0
					?
					?
					0
					?
					?
	1	1	0	0	0
	1	1	0	1	?
	1	1	1	0	?
	1	1	1	1	?

Hypothesis Space

1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	0
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Simple Rules: There are only 16 simple **conjunctive rules**

of the form $y=x_i \wedge x_j \wedge x_k$

Rule	Counterexample	Rule	Counterexample
$y=c$		$x_2 \wedge x_3$	
x_1		$x_2 \wedge x_4$	
x_2		$x_3 \wedge x_4$	
x_3		$x_1 \wedge x_2 \wedge x_3$	
x_4		$x_1 \wedge x_2 \wedge x_4$	
$x_1 \wedge x_2$		$x_1 \wedge x_3 \wedge x_4$	
$x_1 \wedge x_3$		$x_2 \wedge x_3 \wedge x_4$	
$x_1 \wedge x_4$		$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	

No simple rule explains the data. The same is true for **simple clauses**.

Hypothesis Space

1	0	0	1	0	**0**
2	0	1	0	0	**0**
3	0	0	1	1	**1**
4	1	0	0	1	**0**
5	0	1	1	0	**0**
6	1	1	0	0	**0**
7	0	1	0	1	**0**

Simple Rules: There are only 16 simple **conjunctive rules**

of the form $y=x_i \wedge x_j \wedge x_k \wedge x_p$

Rule	Counterexample	Rule	Counterexample
$y=c$	1100 0	$x_2 \wedge x_3$	0011 1
x_1	1100 0	$x_2 \wedge x_4$	0011 1
x_2	0100 0	$x_3 \wedge x_4$	OK
x_3	0110 0	$x_1 \wedge x_2 \wedge x_3$	0011 1
x_4	0101 1	$x_1 \wedge x_2 \wedge x_4$	0011 1
$x_1 \wedge x_2$	1100 0	$x_1 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_3$	0011 1	$x_2 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_4$	0011 1	$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	0011 1

No simple rule explains the data. The same is true for **simple clauses**.

Hypothesis Space

1	0	0	1	0	**0**
2	0	1	0	0	**0**
3	0	0	1	1	**1**
4	1	0	0	1	**0**
5	0	1	1	0	**0**
6	1	1	0	0	**0**
7	0	1	0	1	**0**

Simple Rules: There are only 16 simple **conjunctive rules**

of the form $y = x_i \wedge x_j \wedge x_k \wedge x_p$

Rule	Counterexample	Rule	Counterexample
$y = c$	1100 0	$x_2 \wedge x_3$	0011 1
x_1	1100 0	$x_2 \wedge x_4$	0011 1
$x_1 \wedge x_3$	0011 1	$x_2 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_4$	0011 1	$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	0011 1

How many examples we need to figure out the right function?

No simple rule explains the data. The same is true for **simple clauses**.

Learning protocol

Provide the learning examples

Learn the model

$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

Learning Monotone Conjunctions

- Protocol 1:
  Teacher provides a set of example (x, f(x))
  - <(1,1,1,1,1,1,…,1,1), 1>
  - <(1,1,1,0,0,0,…,0,0), 0>
  - <(1,1,1,1,1,0,...0,1,1), 1>
  - <(1,0,1,1,1,0,...0,1,1), 0>
  - <(1,1,1,1,1,0,...0,0,1), 1>
  - <(1,0,1,0,0,0,...0,1,1), 0>
  - <(1,1,1,1,1,1,…,0,1), 1>
  - <(0,1,0,1,0,0,...0,1,1), 0>

$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

Learning Monotone Conjunctions

- Student: Elimination algorithm
  - Start with the set of all literals as candidates
  - Eliminate a literal that is not active (0) in a positive example

$f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge \ldots \wedge x_{100}$

- <(1,1,1,1,1,1,…,1,1), 1> Learn nothing
- <(1,1,1,0,0,0,…,0,0), 0> Learn nothing
- <(1,1,1,1,1,0,...0,1,1), 1> $f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{99} \wedge x_{100}$
- <(1,0,1,1,1,0,...0,1,1), 0> Learn nothing
- <(1,1,1,1,1,0,...0,0,1), 1> $f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$
- <(1,0,1,0,0,0,...0,1,1), 0>
- <(1,1,1,1,1,1,…,0,1), 1>
- <(0,1,0,1,0,0,...0,1,1), 0>

We can determine the # of mistakes we'll make before reaching the exact target function, but not how many examples are need to guarantee good performance.

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning Monotone Conjunctions

- Protocol 2: Alternatively, we can have the learner to propose instances as queries to the teacher
- Since we know we have a Monotone conjunction:
- Is x_{100} in? <(1,1,1…,1,0), ?> f(x)=0 (conclusion: Yes)
- Is x_{99} in? <(1,1,…1,0,1), ?> f(x)=1 (conclusion: No)
- Is x_1 in ? <(0,1,…1,1,1), ?> f(x)=1 (conclusion: No)

- A straight forward algorithm requires n=100 queries, and will produce as a result the hidden conjunction (exactly).

Two Models for How good is our learning algorithm?

- Analyze the probabilistic intuition
  - Never saw a feature in positive examples, maybe we'll never see it
  - And if we do, it will be with small probability, so the concepts we learn may be *pretty good*
    - *Pretty good:* In terms of performance on future data
  - ***PAC framework***
- *Mistake Driven* Learning algorithms
  - Update your hypothesis only when you make mistakes
  - Define *good* in terms of how many mistakes you make before you stop
  - ***Online learning***

The mistake-bound approach

- The mistake bound model is a theoretical approach
  - We can determine the number of mistakes the learning algorithm can make before converging
- But no answer to "*How many examples do you need before converging to a good hypothesis?*"
- Because the mistake-bound model makes no assumptions about the order or distribution of training examples
  - Both a strength and a weakness of the mistake bound model

PAC learning

- A model for *batch learning*
  - Train on a fixed training set
  - Then deploy it in the wild

- How well will your learning algorithm do on *future* instances?

The setup

- Instance Space: X, the set of examples
- Concept Space: C, the set of possible target functions:
  $f \in C$ is the hidden target function
  - Eg: all n-conjunctions; all n-dimensional linear functions, …
- Hypothesis Space: H, the set of possible hypotheses
  - This is the set that the learning algorithm explores
- Training instances: S x {-1,1}: positive and negative examples of the target concept. (S is a finite subset of X)

$$< x_1, f(x_1) >, < x_2, f(x_2) >, \ldots < x_n, f(x_n) >$$

- What we want: A hypothesis h $\in$ H such that h(x) = f(x)
  - A hypothesis h $\in$ H such that h(x) = f(x) for all x $\in$ S ?
  - A hypothesis h $\in$ H such that h(x) = f(x) for all x $\in$ X ?

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning Conjunctions

- Protocol 1:
  Teacher provides a set of example (x, f(x))
  - <(1,1,1,1,1,1,…,1,1), 1>
  - <(1,1,1,0,0,0,…,0,0), 0>
  - <(1,1,1,1,1,0,...0,1,1), 1>
  - <(1,0,1,1,1,0,...0,1,1), 0>
  - <(1,1,1,1,1,0,...0,0,1), 1>
  - <(1,0,1,0,0,0,...0,1,1), 0>
  - <(1,1,1,1,1,1,…,0,1), 1>
  - <(0,1,0,1,0,0,...0,1,1), 0>

What would f look like?

PAC Learning – Intuition

- The assumption of fixed distribution is important:
  1. What we learn on the training data will be meaningful on future examples
  2. Also gives a well-defined notion of the error of a hypothesis according to the target function

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning Conjunctions

- Protocol 1:
  Teacher provides a set of example (x, f(x))
  - <(1,1,1,1,1,1,…,1,1), 1>
  - <(1,1,1,0,0,0,…,0,0), 0>
  - <(1,1,1,1,1,0,...0,1,1), 1>
  - <(1,0,1,1,1,0,...0,1,1), 0>
  - <(1,1,1,1,1,0,...0,0,1), 1>
  - <(1,0,1,0,0,0,...0,1,1), 0>
  - <(1,1,1,1,1,1,…,0,1), 1>
  - <(0,1,0,1,0,0,...0,1,1), 0>

What would f look like?

Whenever the output is 1, x_1 is present

With the given data, we only learned an *approximation* to the true concept. Is it good enough?

"The future will be like the past":

- We have seen many examples (drawn according to the distribution D)
  - Since in all the positive examples x_1 was active, it is very likely that it will be active in future positive examples
  - If not, in any case, x_1 is active only in a small percentage of the examples so our error will be small

Error of a hypothesis

Definition

Given a distribution $\mathcal{D}$ over examples, the *error* of a hypothesis h with respect to a target concept f is

$$\mathrm{err}_{\mathcal{D}}(h) = \Pr_{x \sim \mathcal{D}}[h(x) \neq f(x)]$$

Error of a hypothesis

Definition

Given a distribution $\mathcal{D}$ over examples, the *error* of a hypothesis h with respect to a target concept f is

$$\mathrm{err}_{\mathcal{D}}(h) = \Pr_{x \sim \mathcal{D}}[h(x) \neq f(x)]$$

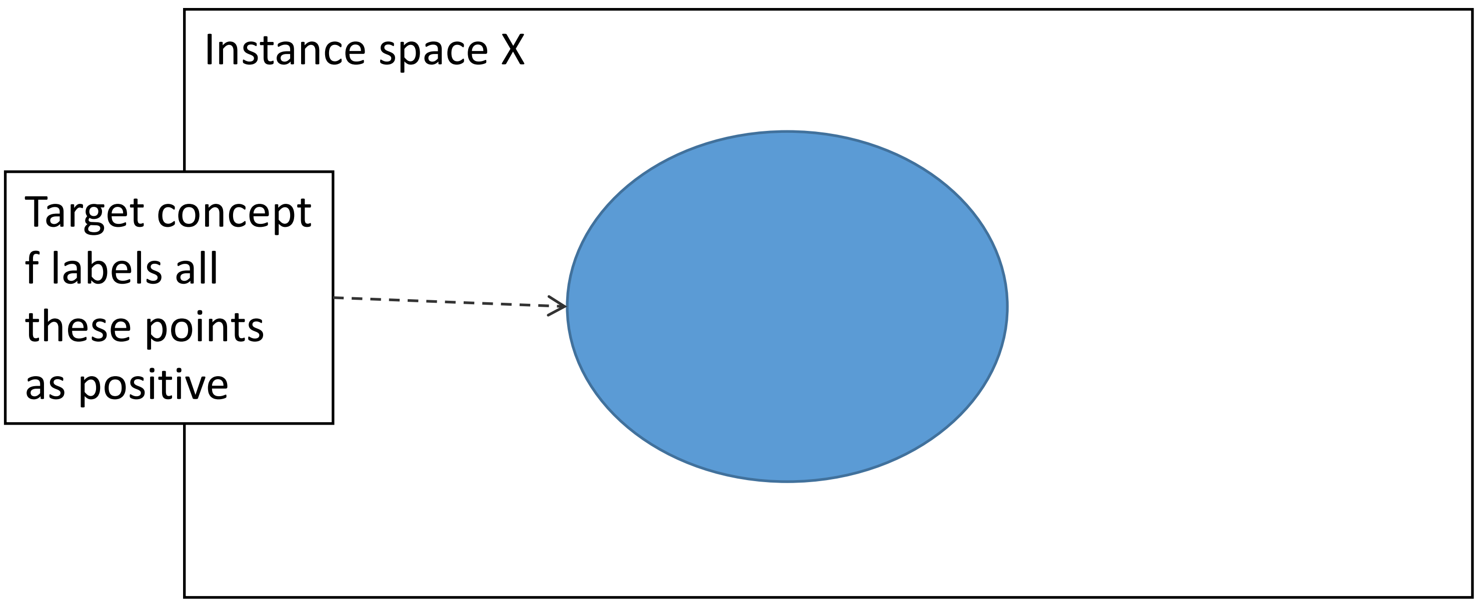
Instance space X

Error of a hypothesis

Definition

Given a distribution D over examples, the *error* of a hypothesis h with respect to a target concept f is

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

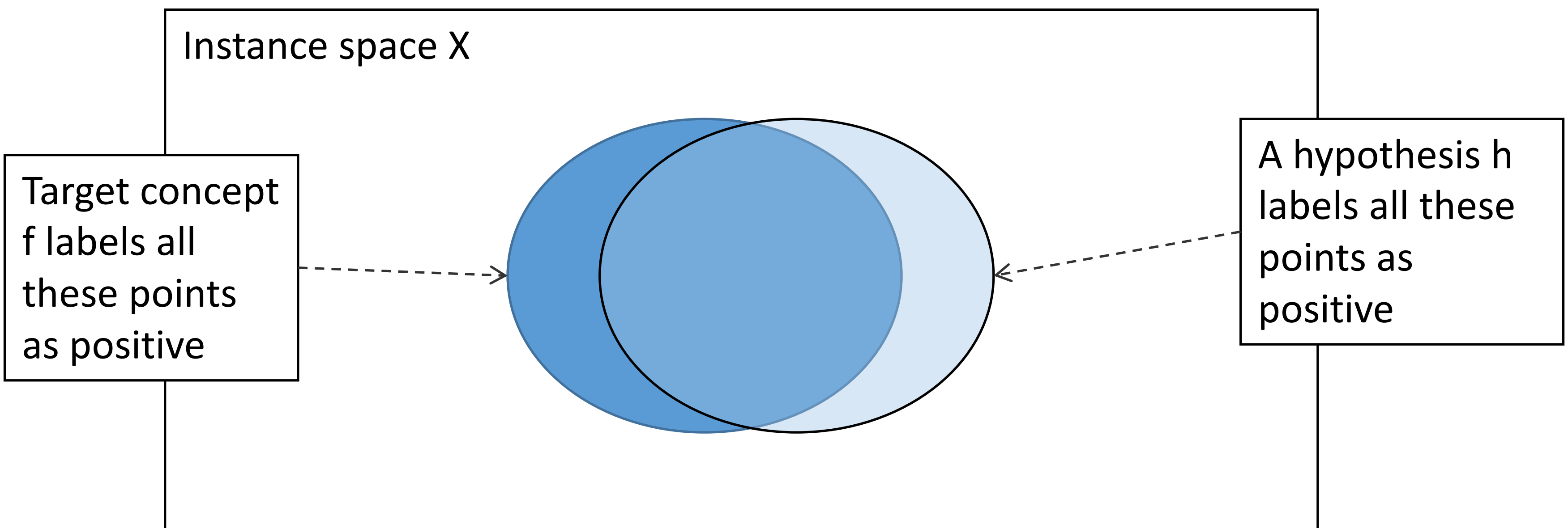


Error of a hypothesis

Definition

Given a distribution $\mathcal{D}$ over examples, the *error* of a hypothesis h with respect to a target concept f is

$$err_{\mathcal{D}}(h) = Pr_{x \sim \mathcal{D}}[h(x) \neq f(x)]$$

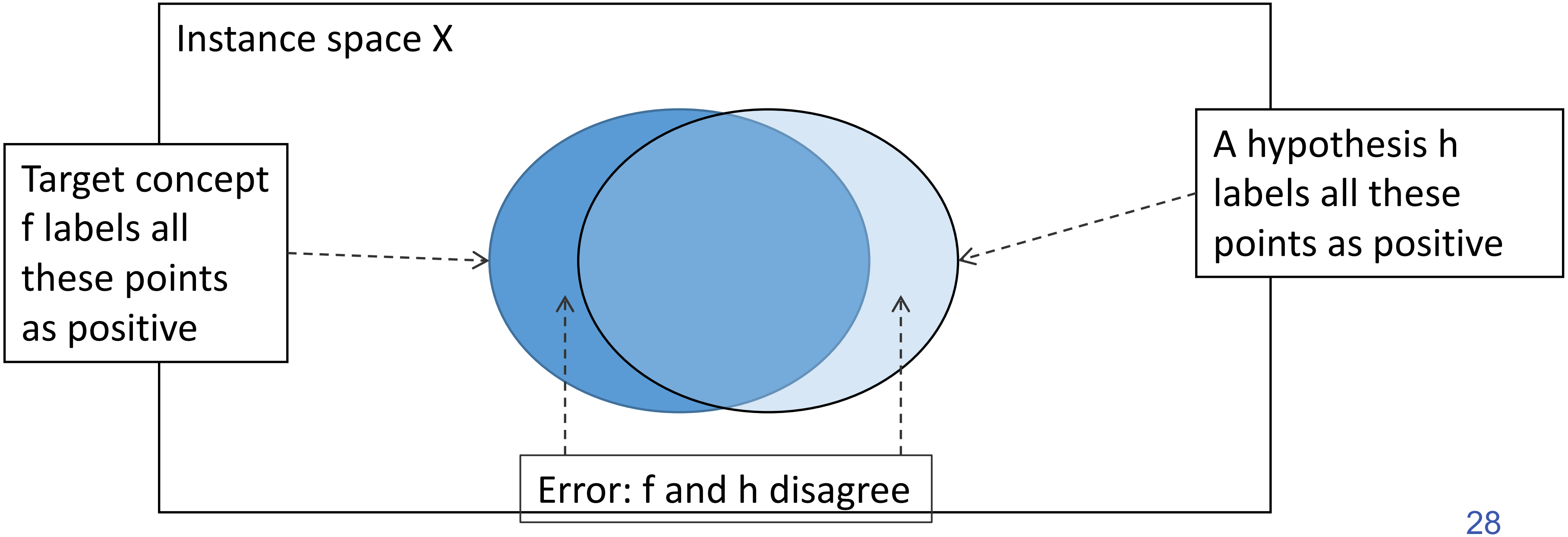


Error of a hypothesis

Definition

Given a distribution $\mathcal{D}$ over examples, the *error* of a hypothesis h with respect to a target concept f is

$$\mathrm{err}_{\mathcal{D}}(h) = \Pr_{x \sim \mathcal{D}}[h(x) \neq f(x)]$$



Empirical error

Contrast true error against the *empirical error*

For a target concept f, the empirical error of a hypothesis h is defined for a training set S as the fraction of examples x in S for which the functions f and h disagree.

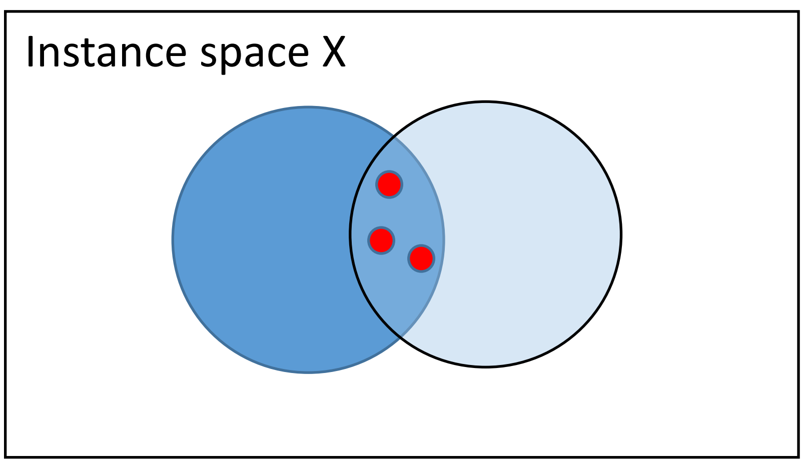
Denoted by $err_S(h)$

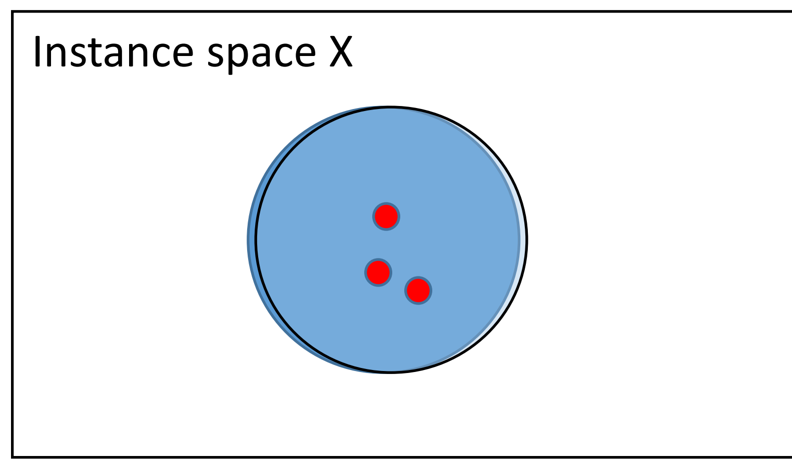
Overfitting: When the empirical error on the training set $err_S(h)$ is substantially lower than $err_D(h)$

The goal of learning

To devise good learning algorithms that avoid overfitting

- Not fooled by functions that only appear to be good because they explain the training set very well



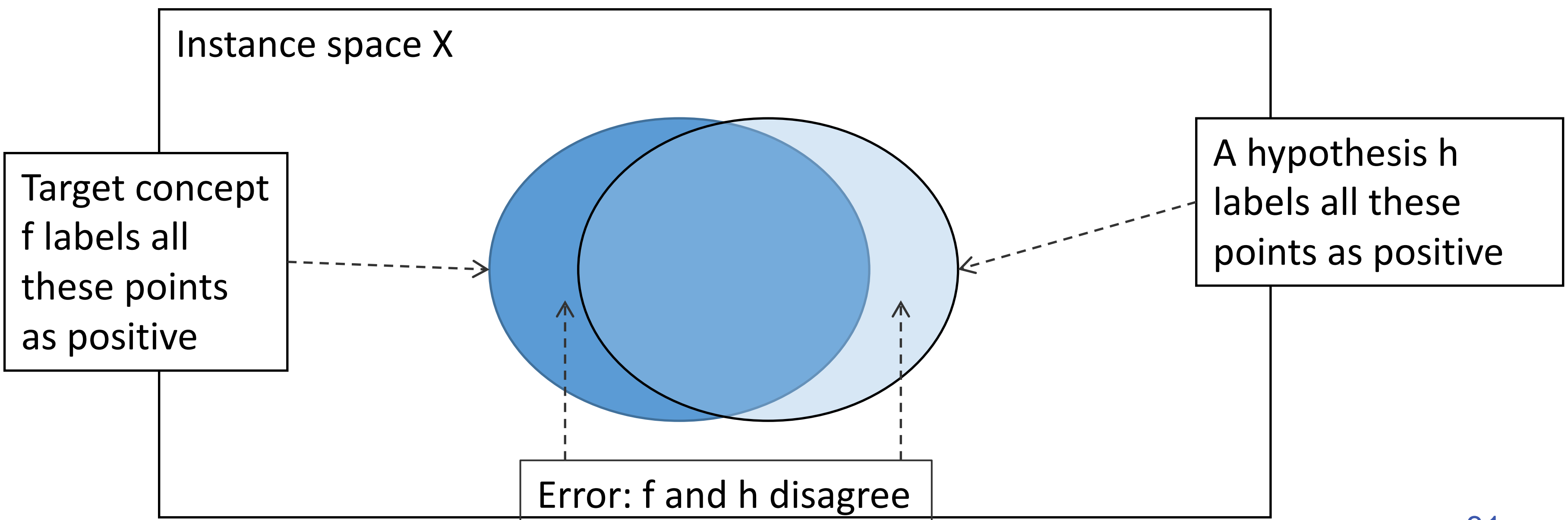


Error of a hypothesis

Definition

Given a distribution $\mathcal{D}$ over examples, the *error* of a hypothesis h with respect to a target concept f is

$$err_{\mathcal{D}}(h) = Pr_{x \sim \mathcal{D}}[h(x) \neq f(x)]$$



This lecture: Computational Learning Theory

- The Theory of Generalization
- Probably Approximately Correct (PAC) learning
- Shattering and the VC dimension

Probably Approximately Correct (PAC) learning

1. Analyze a simple algorithm for learning conjunctions

2. Define the PAC model of learning

Example: Learning Conjunctions

The true function $f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

Training data

- <(1,1,1,1,1,1,…,1,1), 1>
- <(1,1,1,0,0,0,…,0,0), 0>
- <(1,1,1,1,1,0,...0,1,1), 1>
- <(1,0,1,1,1,0,...0,1,1), 0>
- <(1,1,1,1,1,0,...0,0,1), 1>
- <(1,0,1,0,0,0,...0,1,1), 0>
- <(1,1,1,1,1,1,…,0,1), 1>
- <(0,1,0,1,0,0,...0,1,1), 0>

Learning Conjunctions

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Training data

- <(1,1,1,1,1,1,…,1,1), 1>
- ~~<(1,1,1,0,0,0,…,0,0), 0>~~
- <(*1*,*1*,*1*,*1*,*1*,0,...0,1,*1*), 1>
- ~~<(1,0,1,1,1,0,...0,1,1), 0>~~
- <(*1*,*1*,*1*,*1*,*1*,0,...0,0,*1*), 1>
- ~~<(1,0,1,0,0,0,...0,1,1), 0>~~
- <(*1*,*1*,*1*,*1*,*1*,1,…,0,*1*), 1>
- ~~<(0,1,0,1,0,0,...0,1,1), 0>~~

A simple learning algorithm (*Elimination*)

- Discard all negative examples

Learning Conjunctions

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Training data

- <(1,1,1,1,1,1,…,1,1), 1>
- ~~<(1,1,1,0,0,0,…,0,0), 0>~~
- <(1,1,1,1,1,0,...0,1,1), 1>
- ~~<(1,0,1,1,1,0,...0,1,1), 0>~~
- <(1,1,1,1,1,0,...0,0,1), 1>
- ~~<(1,0,1,0,0,0,...0,1,1), 0>~~
- <(1,1,1,1,1,1,…,0,1), 1>
- ~~<(0,1,0,1,0,0,...0,1,1), 0>~~

A simple learning algorithm (*Elimination*)

- Discard all negative examples
- Build a conjunction using the features that are common to all positive conjunctions

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning Conjunctions

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Training data

- <(*1*,*1*,*1*,*1*,*1*,1,…,1,*1*), 1>
- ~~<(1,1,1,0,0,0,…,0,0), 0>~~
- <(*1*,*1*,*1*,*1*,*1*,0,...0,1,*1*), 1>
- ~~<(1,0,1,1,1,0,...0,1,1), 0>~~
- <(*1*,*1*,*1*,*1*,*1*,$\boxed{0}$,...0,$\boxed{0}$,*1*), 1>
- ~~<(1,0,1,0,0,0,...0,1,1), 0>~~
- <(*1*,*1*,*1*,*1*,*1*,1,…,$\boxed{0}$,*1*), 1>
- ~~<(0,1,0,1,0,0,...0,1,1), 0>~~

A simple learning algorithm (*Elimination*)

- Discard all negative examples
- Build a conjunction using the features that are common to all positive conjunctions

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Positive examples ***eliminate*** irrelevant features

Learning Conjunctions

$$f = x_2 \land x_3 \land x_4 \land x_5 \land x_{100}$$

Training data

- <(1,1,1,1,1,1,…,1,1), 1>
- <(1,1,1,0,0,0,…,0,0), 0>
- <(1,1,1,1,1,0,...0,1,1), 1>
- <(1,0,1,1,1,0,...0,1,1), 0>
- <(1,1,1,1,1,0,...0,0,1), 1>
- <(1,0,1,0,0,0,...0,1,1), 0>
- <(1,1,1,1,1,1,…,0,1), 1>
- <(0,1,0,1,0,0,...0,1,1), 0>

A simple learning algorithm:

- Discard all negative examples
- Build a conjunction using the features that are common to all positive conjunctions

$$h = x_1 \land x_2 \land x_3 \land x_4 \land x_5 \land x_{100}$$

Clearly this algorithm produces a conjunction that is consistent with the data, that is $err_S(h) = 0$, if the target function is a monotone conjunction

Learning Conjunctions

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Training data

- <(1,1,1,1,1,1,…,1,1), 1>
- <(1,1,1,0,0,0,…,0,0), 0>
- <(1,1,1,1,1,0,...0,1,1), 1>
- <(1,0,1,1,1,0,...0,1,1), 0>
- <(1,1,
- <(1,0,
- <(1,1,
- <(0,1,0,1,0,0,...0,1,1), 0>

A simple learning algorithm:

- Discard all negative examples
- Build a conjunction using the features
- ositive

$x_5 \wedge x_{100}$

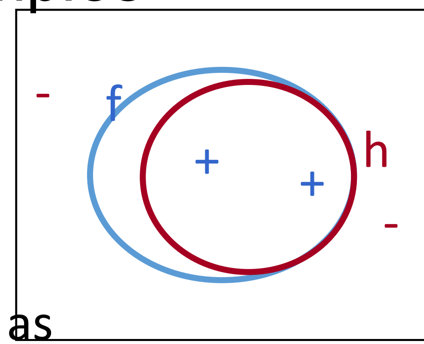
Does the true error $err_D(h)$ also 0?

Clearly this algorithm produces a conjunction that is consistent with the data, that is $err_S(h) = 0$, if the target function is a monotone conjunction

Learning Conjunctions: Analysis

$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$ $h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

Claim 1: Any hypothesis consistent with the training data will only make mistakes on positive future examples



A mistake will occur only if some literal z (in our example x_1) is present in h but not in f

This mistake can cause a positive example to be predicted as negative by h

Specifically: $x_1 = 0$, $x_2 = 1$, $x_3=1$, $x_4=1$, $x_5=1$, $x_{100}=1$

The reverse situation can never happen

For an example to be predicted as positive in the training set, every relevant literal must have been present

Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon}\left(\log(n) + \log\left(\frac{1}{\delta}\right)\right)$$

then, with probability > 1 - δ, the error of the learned hypothesis $err_D(h)$ will be less than ϵ.

Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon}\left(\log(n) + \log\left(\frac{1}{\delta}\right)\right)$$

Poly in n, $1/\delta$, $1/\epsilon$

then, with probability > 1 - δ, the error of the learned hypothesis $err_D(h)$ will be less than ϵ.

n: # literals

If we see these many training examples, then the algorithm will produce a conjunction that, with high probability, will make few errors

Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon}\left(\log(n) + \log\left(\frac{1}{\delta}\right)\right)$$

Poly in n, $1/\delta$, $1/\epsilon$

then, with probability > 1 - δ, the error of the learned hypothesis $err_D(h)$ will be less than ϵ.

Let's prove this assertion

Proof Intuition

$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

$h = \boxed{x_1 \wedge} x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

What kinds of examples would drive a hypothesis to make a mistake and update?

Positive examples, where $x_1 = 0$

h would say true and f would say false

None of these examples appeared during training

Otherwise x_1 would have been eliminated

If they never appeared during training, maybe their appearance in the future would also be rare!

Let's quantify our surprise at seeing such examples

Proof Intuition

$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

$h = \boxed{x_1 \wedge} x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

What kinds of examples would drive a hypothesis to make a mistake and update?

Positive

h wo

None of

Otherwise x_1 would have been eliminated

Key idea: If they never appeared during training, they are not likely to appear in test as well

If they never appeared during training, maybe their appearance in the future would also be rare!

Let's quantify our surprise at seeing such examples

Learning Conjunctions: Analysis

Let p(z) be the probability that, in an example drawn from D, the feature z = 0 but the example has a positive label

- In the training – this is an example that can help we learn the right h
- In the test – this is an example that make an error

<(1,1,1,1,1,1,…,1,1), 1>
<(1,1,1,1,1,0,…0,1,1), 1>
<(1,1,1,1,1,0,…0,0,1), 1>
<(1,1,1,1,1,1,…,0,1), 1>

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning Conjunctions: Analysis

Let p(z) be the probability that, in an example drawn from D, the feature z=0 but the example has a positive label

- i.e., after training is done, p(z) is the probability that in a randomly drawn example, the literal z causes a mistake
- For any z in the target function, p(z) = 0

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

<($\boxed{0}$,*1*,*1*,*1*,*1*,0,...0,1,*1*), $\boxed{1}$>

p(x_1): Probability that this situation occurs

How likely we find h is wrong

Let p(z) be the probability that, in an example drawn from D, the feature z is absent but the example has a positive label

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

We know that $err_D(h) \leq \sum_{z \in h} p(z)$

This is a loose bound

Via direct application of the union bound

How likely we find h is wrong

Let p(z) be the probability that, in an example drawn from D, the feature z is absent but the example has a positive label

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

We know that $err_D(h) \leq \sum_{z \in h} p(z)$

This is a loose bound

Via direct application of the union bou

Union bound
For a set of events, probability that at least one of them happens < the sum of the probabilities of the individual events

Learning Conjunctions: Analysis

n = dimensionality

- Call a literal z *bad* if $p(z) > \frac{\epsilon}{n}$
- Intuitively, a bad literal is one that has a significant probability of not appearing with a positive example
  - (And, if it appears in all positive training examples, it can cause errors)

$$err_D(h) \leq \sum_{z \in h} p(z)$$

If there are no bad literals, then $err_D(h) \leq \epsilon$

- Because $p(z) \leq \frac{\epsilon}{n}$ and $err_D(h) \leq \sum_{z \in h} p(z)$

Learning Conjunctions: Analysis

n = dimensionality

- Call a literal z *bad* if $p(z) > \frac{\epsilon}{n}$
- Intuitively, a bad literal is one that has a significant probability of not appearing with a positive example
  - (And, if it appears in all positive training examples, it can cause errors)

$$err_D(h) \leq \sum_{z \in h} p(z)$$

What if there are bad literals?

Let z be a bad literal

What is the probability that it will not be eliminated by one training example?

There was one example of this kind

<(*1*,*1*,*1*,*1*,*1*,0,...0,1,*1*), 1>

Learning Conjunctions: Analysis

What we know so far:

n = dimensionality

$$Pr(\text{A } \textcolor{red}{\text{bad literal}} \text{ is not eliminated by one example}) < 1 - \frac{\epsilon}{n}$$

Learning Conjunctions: Analysis

What we know so far:

n = dimensionality

$$Pr(\text{A bad literal is not eliminated by one example}) < 1 - \frac{\epsilon}{n}$$

But say we have m training examples. Then

$$Pr(\text{A bad literal survives } m \text{ examples}) < \left(1 - \frac{\epsilon}{n}\right)^m$$

Learning Conjunctions: Analysis

What we know so far:

n = dimensionality

$$Pr(\text{A bad literal is not eliminated by one example}) < 1 - \frac{\epsilon}{n}$$

But say we have m training examples. Then

$$Pr(\text{A bad literal survives } m \text{ examples}) < \left(1 - \frac{\epsilon}{n}\right)^m$$

There are at most n bad literals. So

$$Pr(\text{Any bad literal survives } m \text{ examples}) < n\left(1 - \frac{\epsilon}{n}\right)^m$$

Learning Conjunctions: Analysis

$$Pr(\text{Any bad literal survives } m \text{ examples}) < n\left(1 - \frac{\epsilon}{n}\right)^m$$

We want this probability to be small

Why? So that we can choose enough training examples so that the probability that any z survives all of them is less than some δ

Learning Conjunctions: Analysis

$$Pr(\text{Any bad literal survives } m \text{ examples}) < n\left(1 - \frac{\epsilon}{n}\right)^m$$

We want this probability to be small

Why? So that we can choose enough training examples so that the probability that any z survives all of them is less than some δ

That is, we want

$$n\left(1 - \frac{\epsilon}{n}\right)^m < \delta$$

We know that $1 - x < e^{-x}$. So it is sufficient to require

$$ne^{-\frac{m\epsilon}{n}} < \delta$$

Learning Conjunctions: Analysis

$$Pr(\text{Any bad literal survives } m \text{ examples}) < n\left(1-\frac{\epsilon}{n}\right)^m$$

We want this probability to be small

Why? So that we can choose enough training examples so that the probability that any z survives all of them is less than some ±

That is, we want $n\left(1-\frac{\epsilon}{n}\right)^m < \delta$

We know that 1 – $x < e^{-x}$. So it is sufficient to require $ne^{-\frac{m\epsilon}{n}} < \delta$

Or equivalently,

$$m > \frac{n}{\epsilon}\left(\log(n) + \log\left(\frac{1}{\delta}\right)\right)$$

Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon}\left(\log(n) + \log\left(\frac{1}{\delta}\right)\right)$$

then, with probability > 1 - δ, the error of the learned hypothesis $err_D(h)$ will be less than ϵ.

Probably Approximately Correct (PAC) learning

1. Analyze a simple algorithm for learning conjunctions

2. Define the PAC model of learning

Formulating the theory of prediction

All the notation we have so far on one slide

In the general case, we have

- X: instance space, Y: output space = {+1, -1}
- D: an unknown distribution over X
- f: an unknown target function X → Y, taken from a concept class C
- h: a hypothesis function X → Y that the learning algorithm selects from a hypothesis class H
- S: a set of m training examples drawn from D, labeled with f
- $err_D(h)$: The true error of any hypothesis h
- $err_S(h)$: The empirical error or training error or observed error of h

Theoretical questions

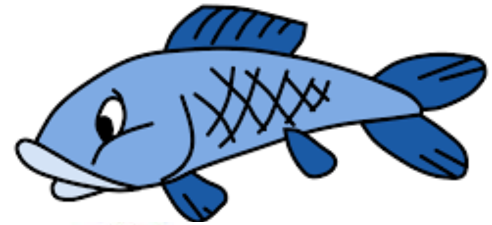
- Can we describe or bound the true error (err_D) given the empirical error (err_S)?
- Is a concept class C learnable?
- Is it possible to learn C using only the functions in H using the supervised protocol?
- How many examples does an algorithm need to guarantee good performance?

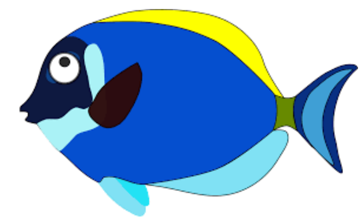
Requirements of Learning

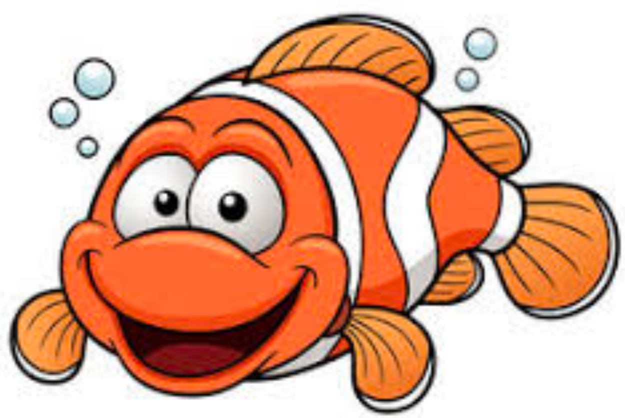
- Cannot expect a learner to learn a concept exactly
  - There will generally be multiple concepts consistent with the available data
  - Unseen examples could *potentially* have any label
  - We "agree" to misclassify *uncommon* examples that do not show up in the training set

Example

Example 2







Requirements of Learning

- Cannot expect a learner to learn a concept exactly
  - There will generally be multiple concepts consistent with the available data
  - Unseen examples could *potentially* have any label
  - We “agree” to misclassify *uncommon* examples that do not show up in the training set

- Cannot always expect to learn a close approximation to the target concept
  - Sometimes the training set will not be representative

Probably approximately correctness

- The only realistic expectation of a good learner is that with high probability it will learn a close approximation to the target concept

- In Probably Approximately Correct (PAC) learning, one requires that
  - given small parameters ϵ and δ,
  - With probability at least 1 - ϵ, a learner produces a hypothesis with error at most δ

- The reason we can hope for this is the *consistent distribution assumption*

PAC Learnability

Consider a concept class C defined over an instance space X (containing instances of length n), and a learner L using a hypothesis space H

The concept class C is PAC learnable by L using H if for all $f \in C$, for all distribution D over X, and fixed $\epsilon > 0$, $\delta < 1$, given m examples sampled i.i.d. according to D, the algorithm L produces, with probability at least (1- δ), a hypothesis h ∈ H that has error at most ε, where m is *polynomial* in 1/ ε, 1/ δ, n and size(H)

efficiently learnability

- The concept class C is *efficiently learnable* if L can produce the hypothesis in time that is polynomial in 1/ ε, 1/ δ, n and size(H)

PAC Learnability

- We impose two limitations
- Polynomial *sample complexity* (information theoretic constraint)
  - Is there enough information in the sample to distinguish a hypothesis h that approximate f ?
- Polynomial *time complexity* (computational complexity)
  - Is there an efficient algorithm that can process the sample and produce a good hypothesis h ?

Worst Case definition: the algorithm must meet its accuracy

- for every distribution (The distribution free assumption)
- for every target function f in the class C

Example: Learning Conjunctions

Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon}\left(\log(n) + \log\left(\frac{1}{\delta}\right)\right)$$

This term is often related to log (size(H)) if the learner is consistent

then, with probability > 1 - δ, the error of the learned hypothesis $err_D(h)$ will be less than ϵ.

m is *polynomial* in 1/ ε, 1/ δ, n and size(H)

A general result

Let H be any hypothesis space.

With probability 1 -δ a hypothesis h → H that is consistent with a training set of size m will have an error < ϵ on future examples if

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

1. Expecting lower error increases sample complexity (i.e more examples needed for the guarantee)

2. If we have a larger hypothesis space, then we will make learning harder (i.e higher sample complexity)

3. If we want a higher confidence in the classifier we will produce, sample complexity will be higher.

A general result

Let H be any hypothesis space.

With probability 1 -δ a hypothesis h → H that is consistent with a training set of size m will have an error < ϵ on future examples if

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

It expresses a preference towards smaller hypothesis spaces.

Complicated/larger hypothesis spaces are not necessarily bad. But simpler ones are unlikely to fool us by being consistent with many examples!

A general result

Let H be any hypothesis space.

With probability $1 - \delta$ a hypothesis $h \rightarrow H$ that is consistent with a training set of size m will have an error $< \epsilon$ on future examples if

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

It expresses a preference towards smaller hypothesis spaces.

Next question: What if size(H) is infinity?

Complicated/larger hypothesis spaces are not necessarily bad. But simpler ones are unlikely to fool us by being consistent with many examples!

This lecture: Computational Learning Theory

- The Theory of Generalization

- Probably Approximately Correct (PAC) learning

- Shattering and the VC dimension

Infinite Hypothesis Space

- The previous analysis was restricted to finite hypothesis spaces
- Some infinite hypothesis spaces are more expressive than others
  - Linear threshold function vs. a combination of LTUs
- Need a measure of the expressiveness of an infinite hypothesis space other than its size

Vapnik-Chervonenkis dimension

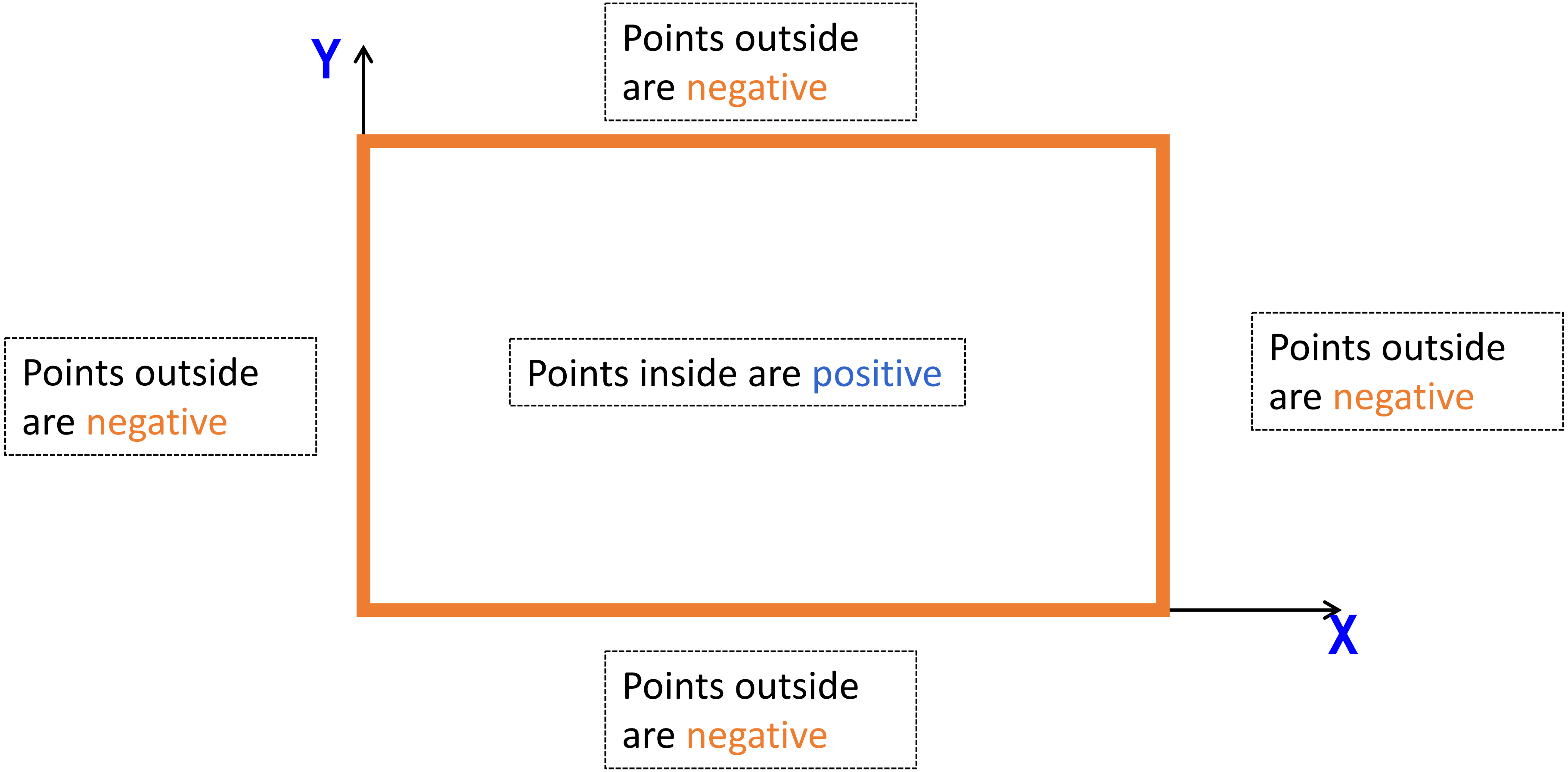
- The Vapnik-Chervonenkis dimension (VC dimension) provides such a measure
  - "What is the expressive *capacity* of a set of functions?"
- Analogous to |H|, there are bounds for sample complexity using VC(H)

Learning Rectangles

Assume the target concept is an axis parallel rectangle

Learning Rectangles

Assume the target concept is an axis parallel rectangle

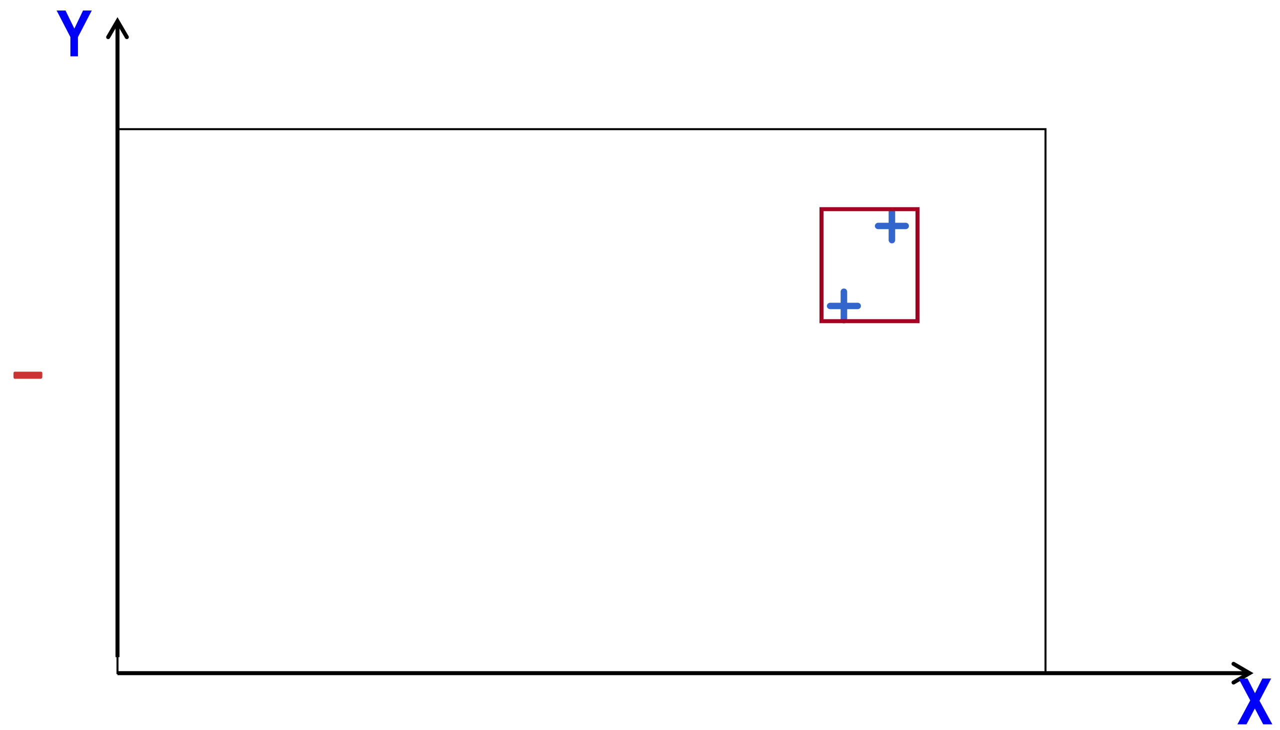


Learning Rectangles

Assume the target concept is an axis parallel rectangle

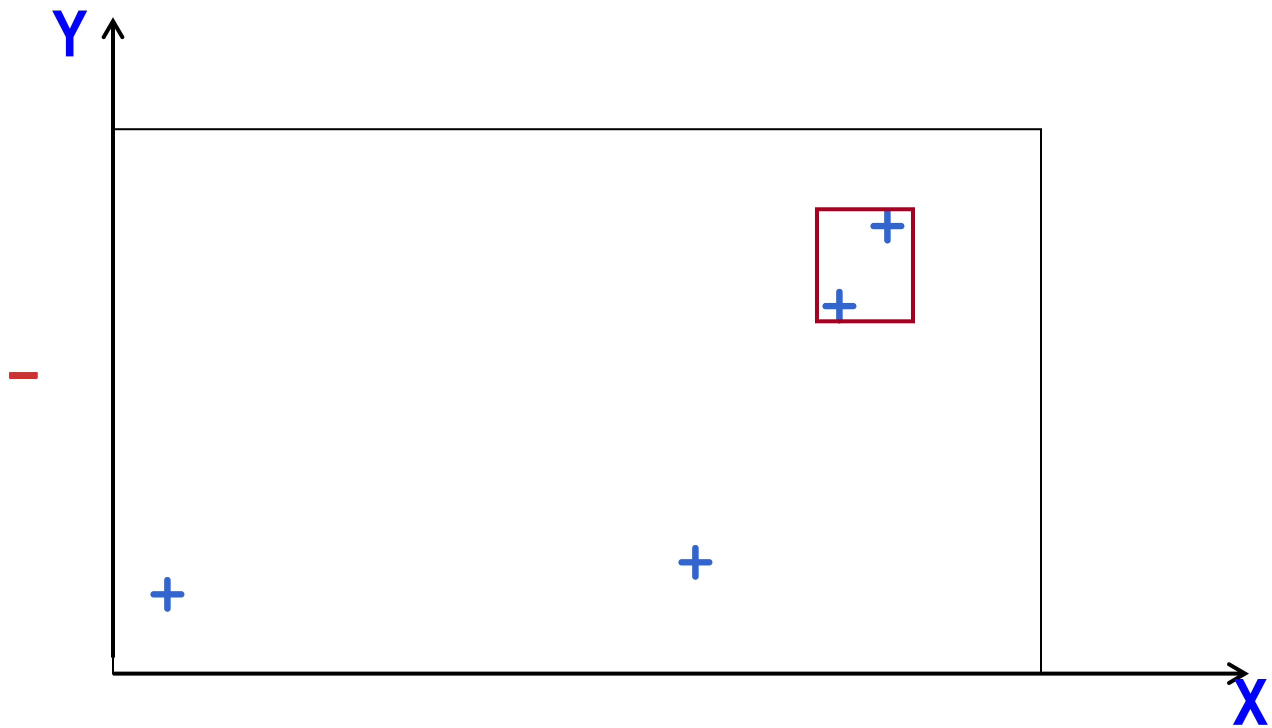
Learning Rectangles

Assume the target concept is an axis parallel rectangle



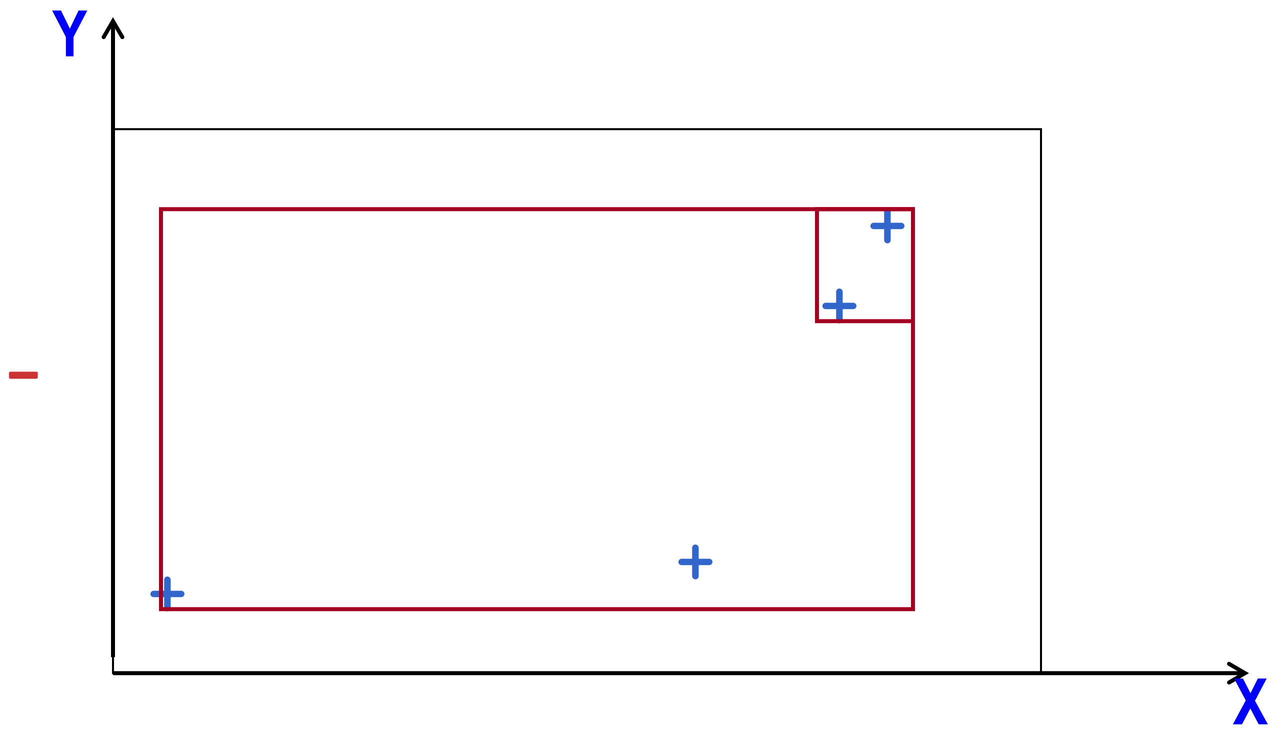
Learning Rectangles

Assume the target concept is an axis parallel rectangle



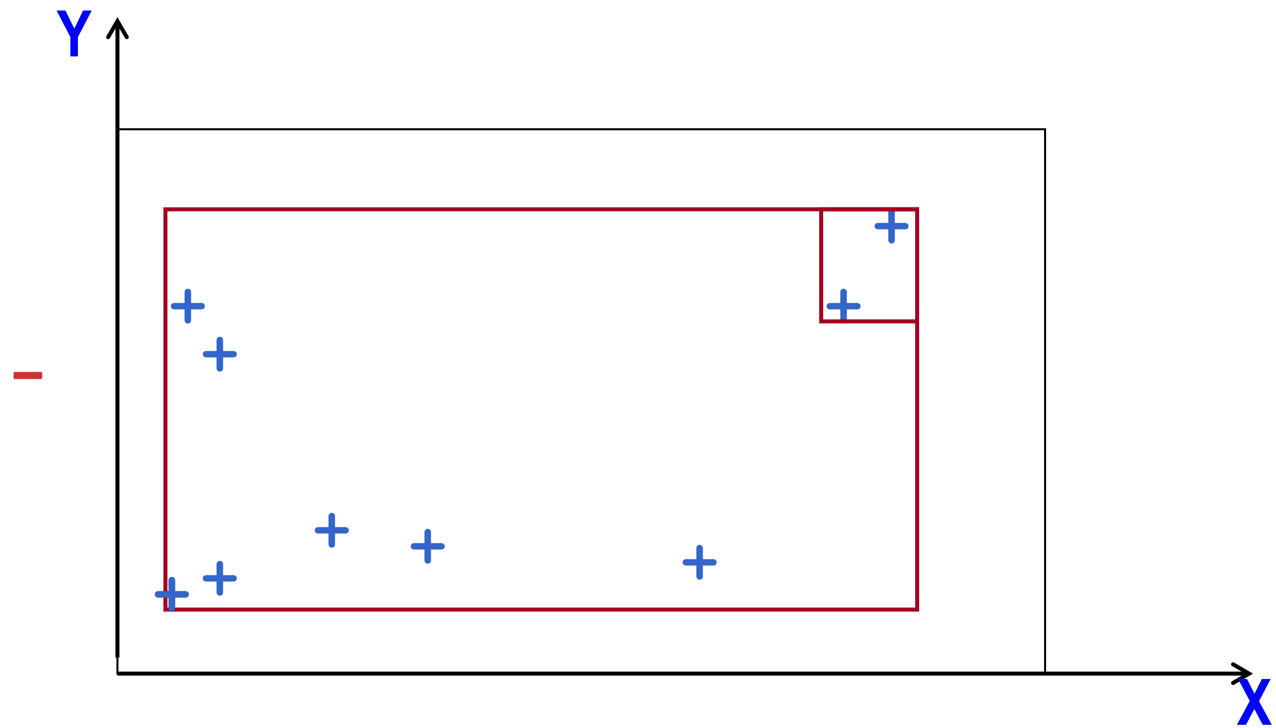
Learning Rectangles

Assume the target concept is an axis parallel rectangle



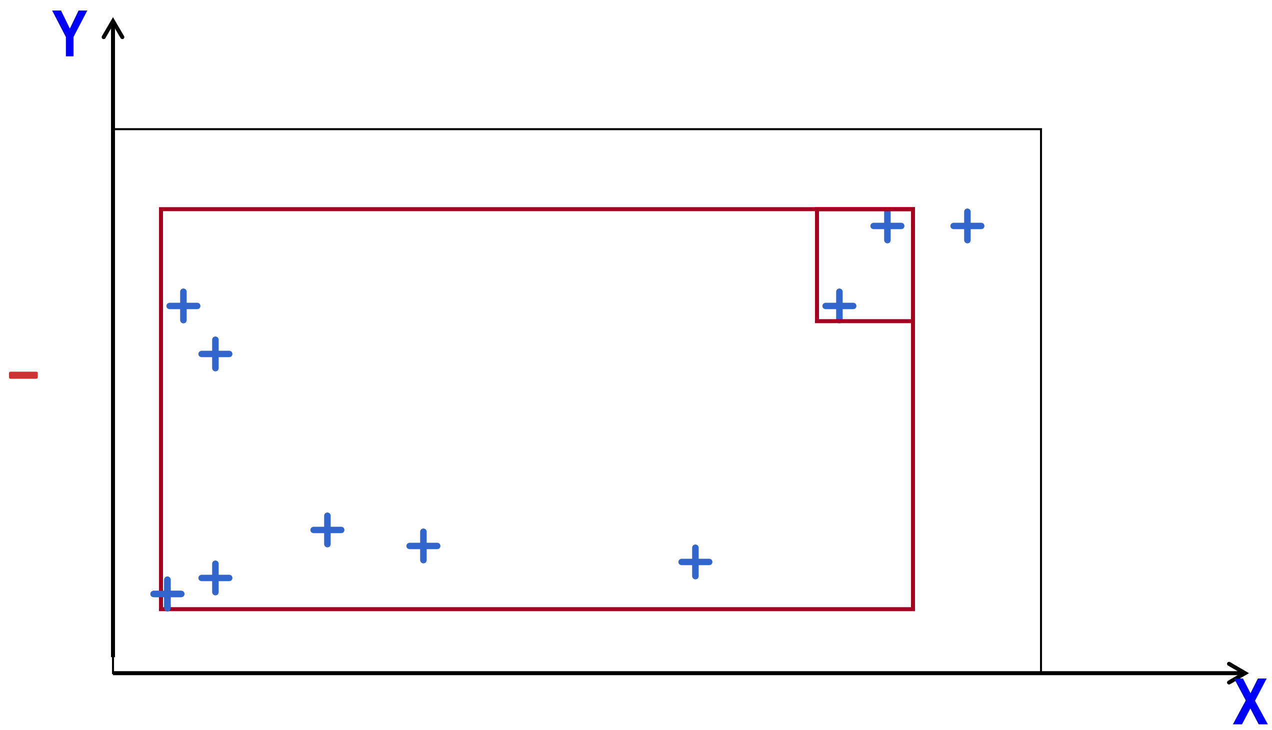
Learning Rectangles

Assume the target concept is an axis parallel rectangle



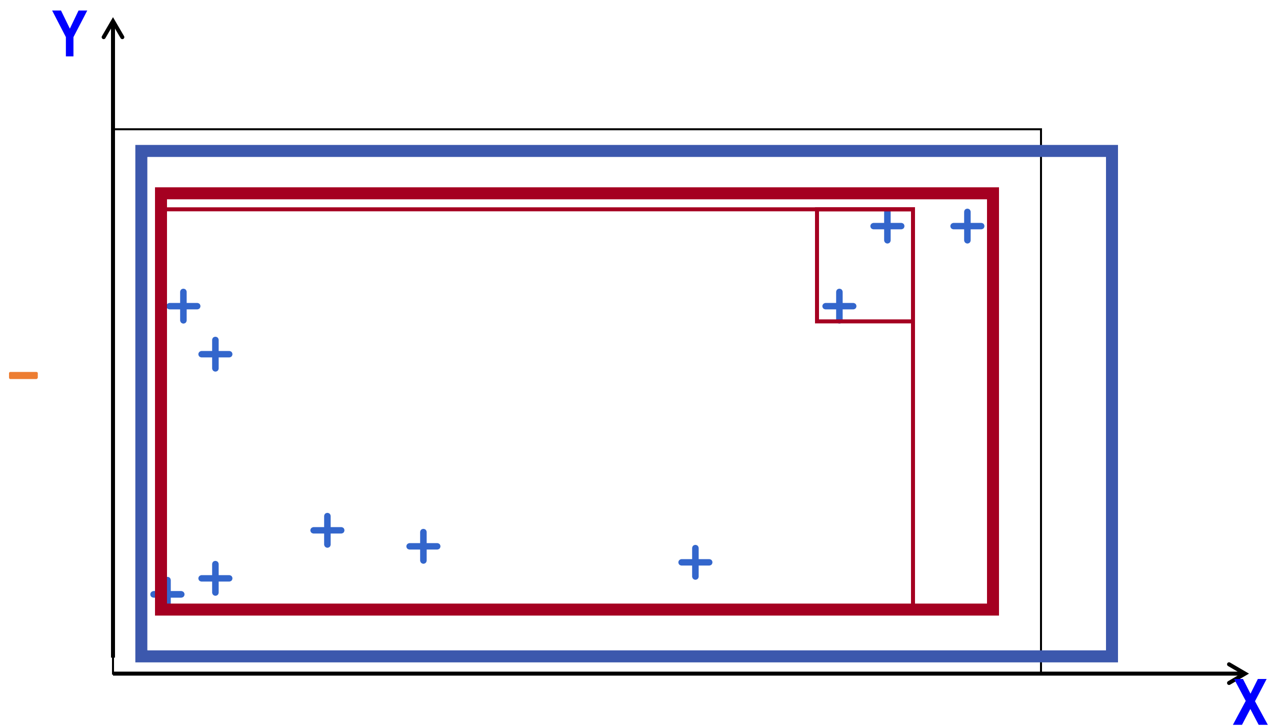
Learning Rectangles

Assume the target concept is an axis parallel rectangle



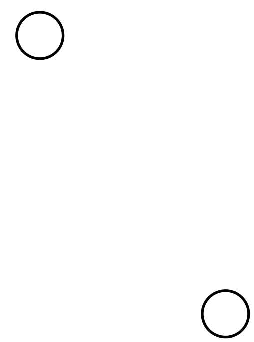
Learning Rectangles

Assume the target concept is an axis parallel rectangle



Key observation: Despite there are infinite # hypothesis
The blue & red rectangles have the same predictions

Let’s think about expressivity of functions

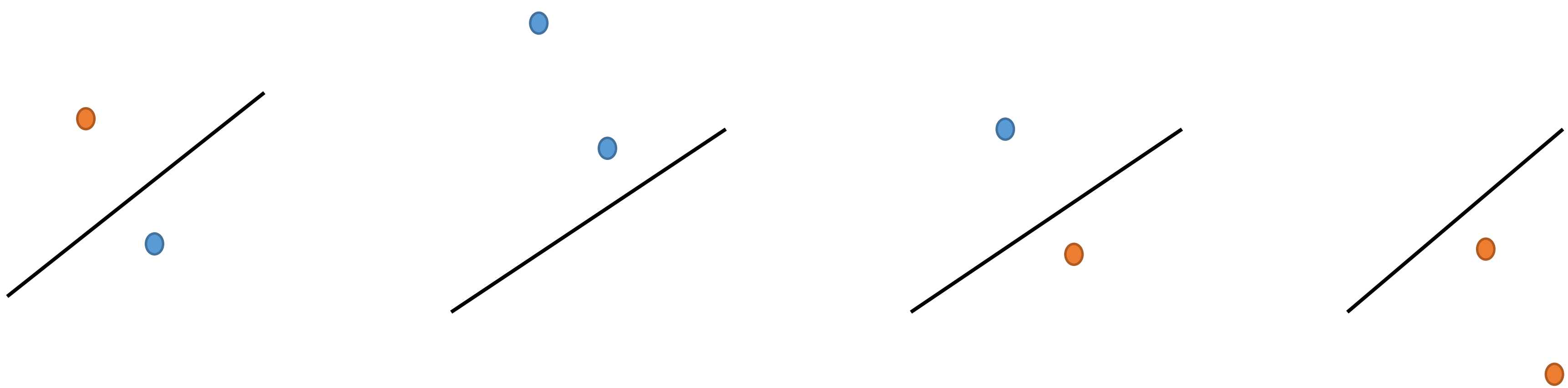


Suppose we have two points.

Can linear classifiers correctly classify any labeling of these points?

Linear functions are expressive enough to ***shatter*** 2 points

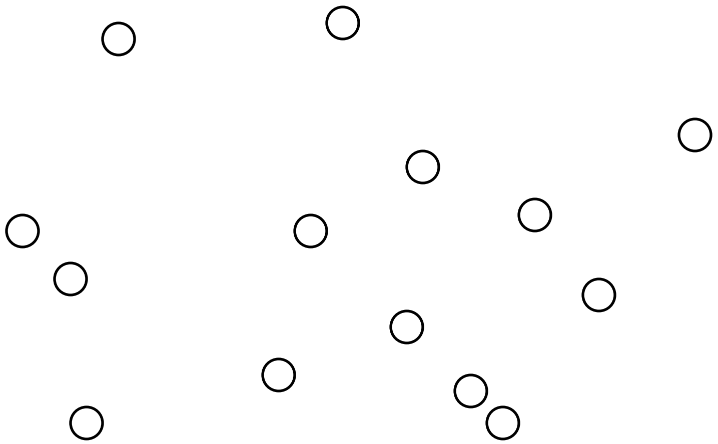
Let’s think about expressivity of functions



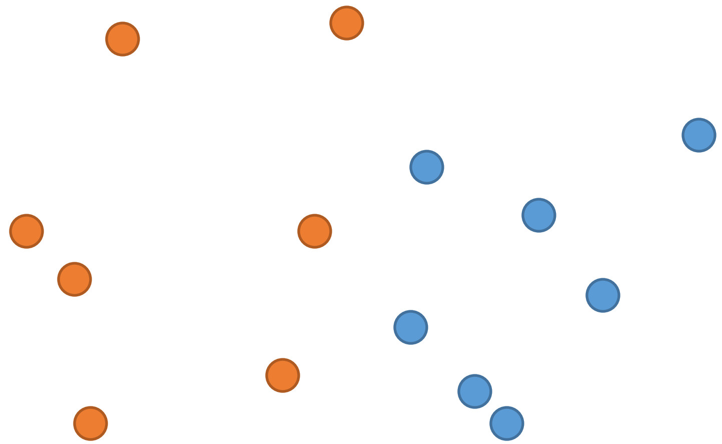
Suppose we have two points.
Can linear classifiers correctly classify any labeling of these points?

Linear functions are expressive enough to ***shatter*** 2 points

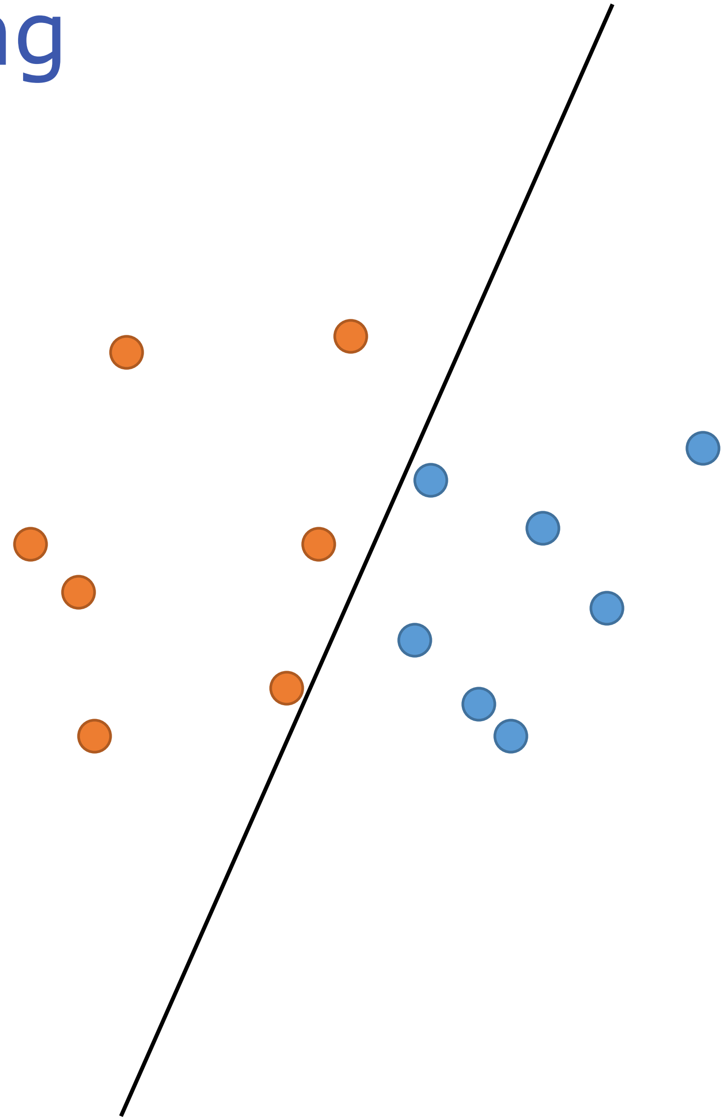
Shattering



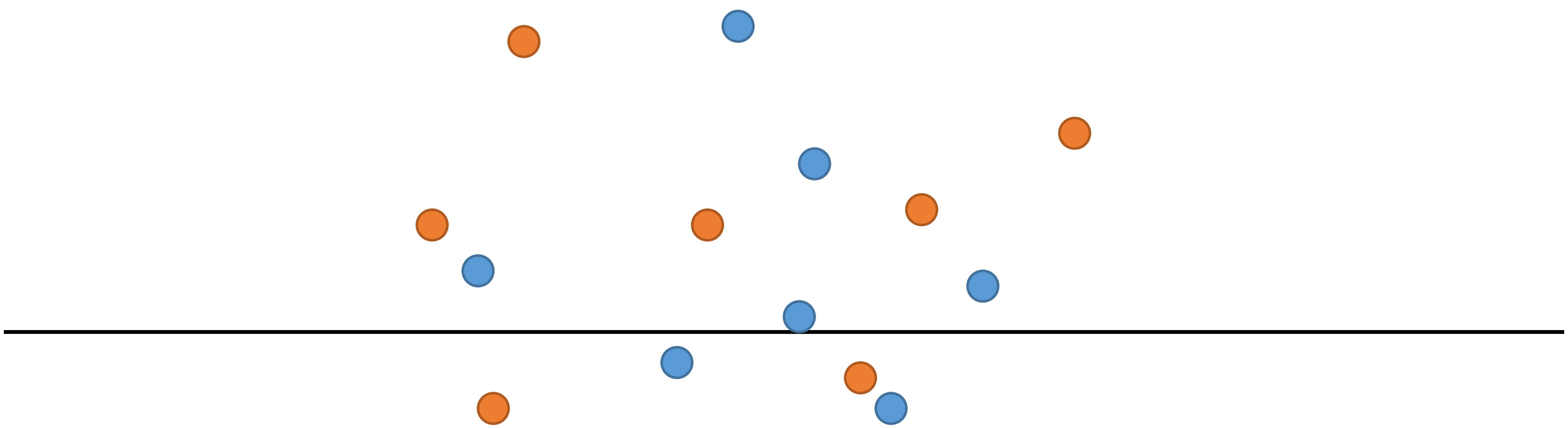
Shattering



Shattering

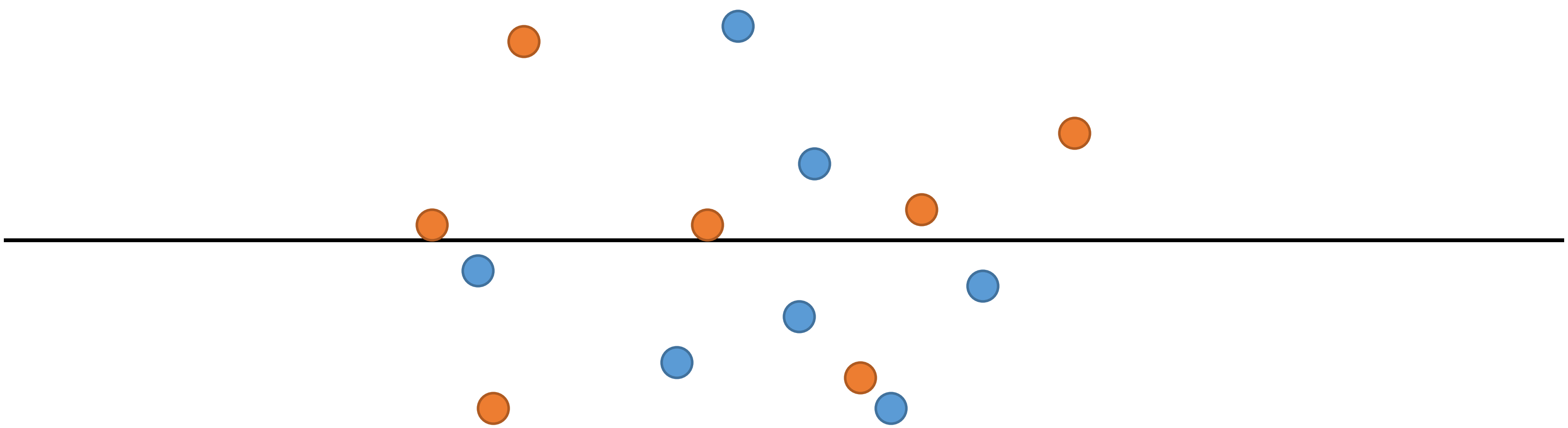


Shattering



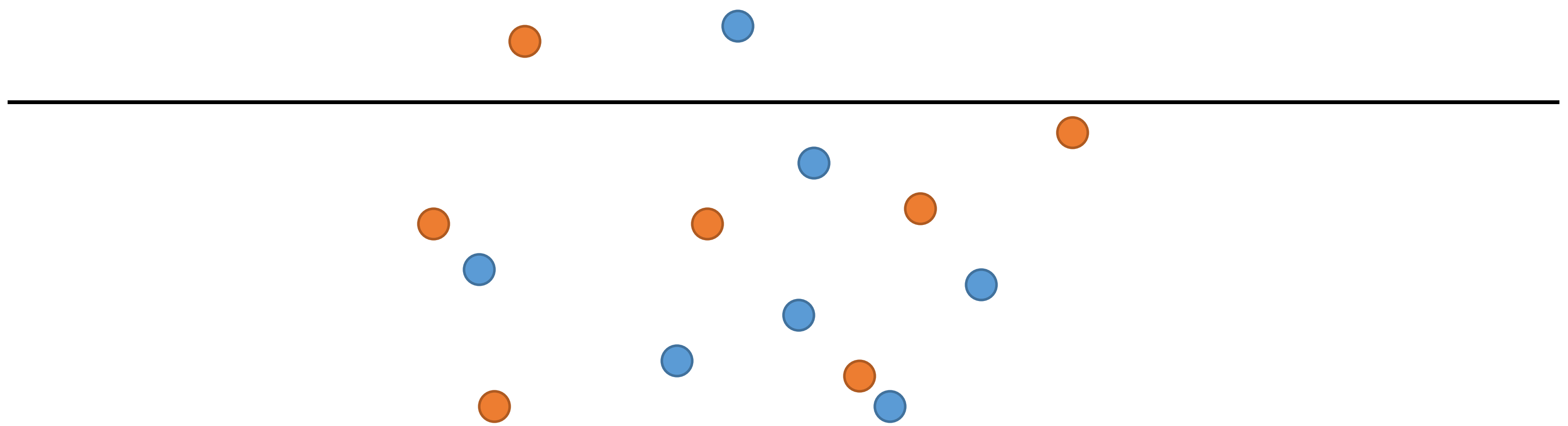
This particular labeling of the points can not be separated by *any* line

Shattering



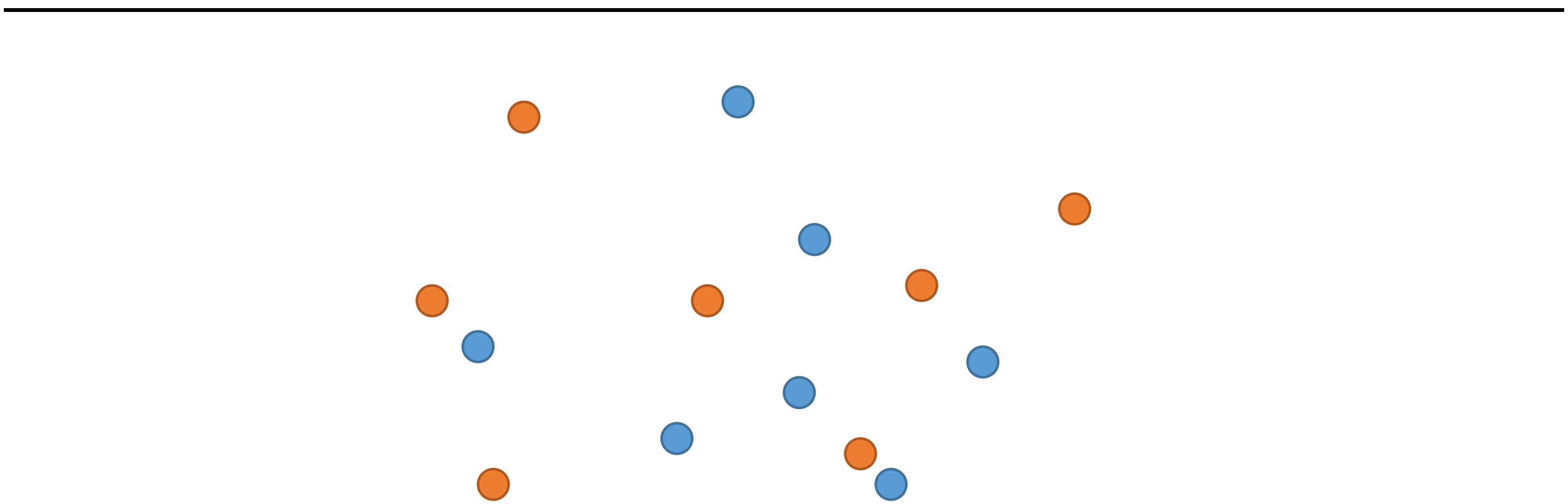
This particular labeling of the points can not be separated by *any* line

Shattering



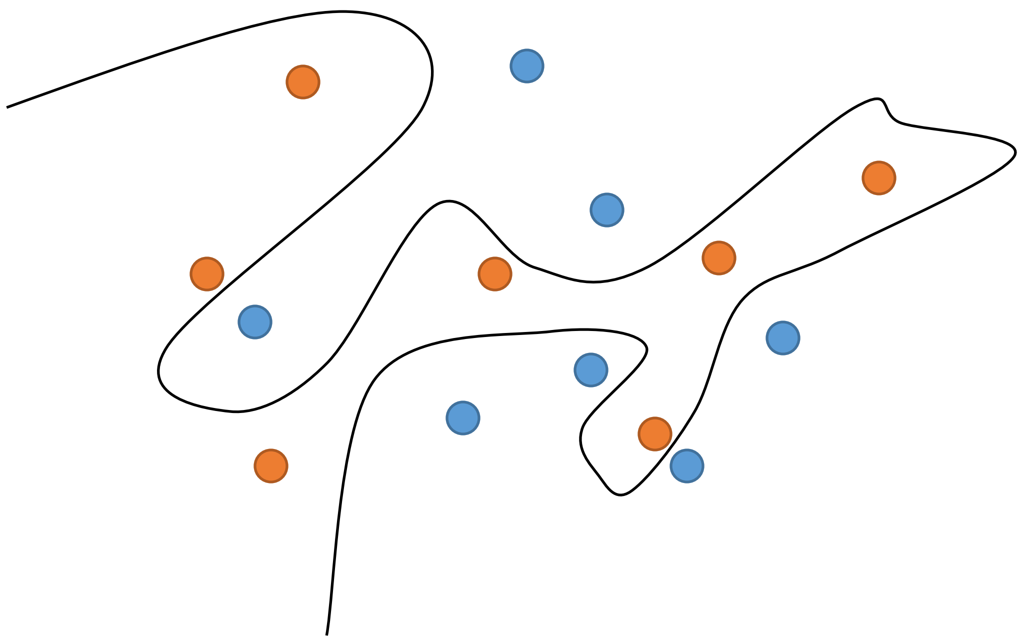
This particular labeling of the points can not be separated by *any* line

Shattering



This particular labeling of the points can not be separated by *any* line

Shattering



Linear functions are not expressive to shatter fourteen points

Because there is a labeling that can not be separated by them

Of course, a more complex function could separate them

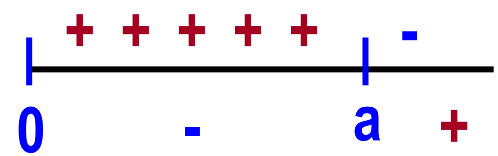
Shattering

Definition: A *set S of examples* is shattered by a *set of functions* H if for *every* partition of the examples in S into positive and negative examples there is a function in H that gives exactly these labels to the examples

Intuition: A rich set of functions shatters large sets of points

Left bounded intervals

Example 1: Hypothesis class of left bounded intervals on the real axis: [0,a) for some real number a>0

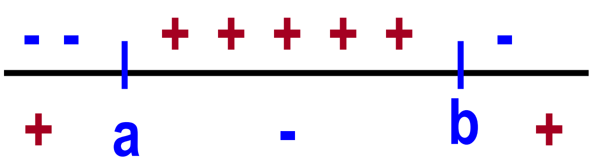


Sets of **two** points cannot be shattered

That is: given two points, you can label them in such a way that no concept in this class will be consistent with their labeling

Real intervals

Example 2: Hypothesis class is the set of intervals on the real axis: [a,b],for some real numbers b>a



All sets of one or two points can be shattered
But some sets of three points cannot be shattered