

Computational Learning Theory

Lecture 8

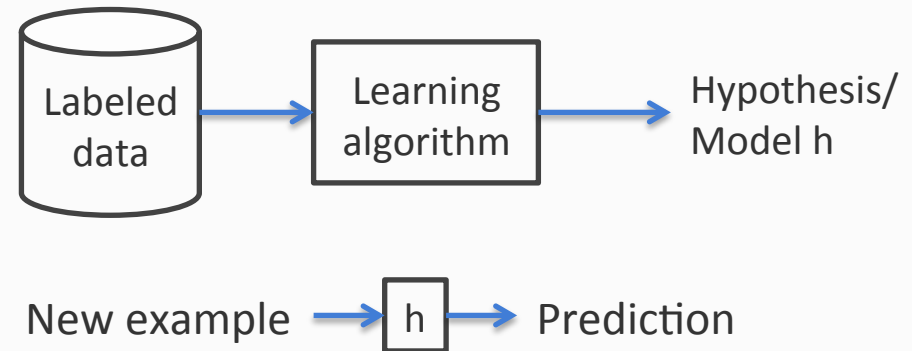
Machine Learning
Fall 2015



Checkpoint: The bigger picture

- Supervised learning: instances, concepts, and hypotheses

- Specific learners
 - Decision trees
 - Nearest neighbors
 - Perceptron
 - Winnow



- General ML ideas
 - Features as high dimensional vectors
 - Overfitting
 - The curse of dimensionality
 - Mistake-bound: One way of asking “Can my problem be learned?”

Questions?

This lecture: Computational Learning Theory

- The Theory of Generalization
- Probably Approximately Correct (PAC) learning
- Positive and negative learnability results
- Agnostic Learning
- Shattering and the VC dimension

This lecture: Computational Learning Theory

- The Theory of Generalization
 - When can we trust the learning algorithm?
 - What functions can be learned?
 - Batch Learning
- Probably Approximately Correct (PAC) learning
- Positive and negative learnability results
- Agnostic Learning
- Shattering and the VC dimension

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

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

Learning Conjunctions

Some random source (nature) provides training examples

- Teacher (Nature) provides the labels ($f(x)$)
 - $\langle (1,1,1,1,1,1,\dots,1,1), 1 \rangle$
 - $\langle (1,1,1,0,0,0,\dots,0,0), 0 \rangle$
 - $\langle (1,1,1,1,1,0,\dots,0,1,1), 1 \rangle$
 - $\langle (1,0,1,1,1,0,\dots,0,1,1), 0 \rangle$
 - $\langle (1,1,1,1,1,0,\dots,0,0,1), 1 \rangle$
 - $\langle (1,0,1,0,0,0,\dots,0,1,1), 0 \rangle$
 - $\langle (1,1,1,1,1,1,\dots,0,1), 1 \rangle$
 - $\langle (0,1,0,1,0,0,\dots,0,1,1), 0 \rangle$

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

Learning Conjunctions

Some random source (nature) provides training examples

- Teacher (Nature) provides the labels ($f(x)$)

- $\langle (1,1,1,1,1,1,\dots,1,1), 1 \rangle$
- $\langle (1,1,1,0,0,0,\dots,0,0), 0 \rangle$
- $\langle (1,1,1,1,1,0,\dots,0,1,1), 1 \rangle$
- $\langle (1,0,1,1,1,0,\dots,0,1,1), 0 \rangle$
- $\langle (1,1,1,1,1,0,\dots,0,0,1), 1 \rangle$
- $\langle (1,0,1,0,0,0,\dots,0,1,1), 0 \rangle$
- $\langle (1,1,1,1,1,1,\dots,0,1), 1 \rangle$
- $\langle (0,1,0,1,0,0,\dots,0,1,1), 0 \rangle$

For a reasonable learning algorithm (by *elimination*), the final hypothesis will be

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

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

Learning Conjunctions

Some random source (nature) provides training examples

- Teacher (Nature) provides the labels ($f(x)$)

- $\langle (1, 1, 1, 1, 1, 1, \dots, 1, 1), 1 \rangle$
- $\langle (1, 1, 1, 0, 0, 0, \dots, 0, 0), 0 \rangle$
- $\langle (1, 1, 1, 1, 1, 0, \dots, 0, 1, 1), 1 \rangle$
- $\langle (1, 0, 1, 1, 1, 0, \dots, 0, 1, 1), 0 \rangle$
- $\langle (1, 1, 1, 1, 1, 0, \dots, 0, 0, 1), 1 \rangle$
- $\langle (1, 0, 1, 0, 0, 0, \dots, 0, 1, 1), 0 \rangle$
- $\langle (1, 1, 1, 1, 1, 1, \dots, 0, 1), 1 \rangle$
- $\langle (0, 1, 0, 1, 0, 0, \dots, 0, 1, 1), 0 \rangle$

For a reasonable learning algorithm (by *elimination*), the final hypothesis will be

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

Whenever the output is 1, x_1 is present

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

Learning Conjunctions

Some random source (nature) provides training examples

- Teacher (Nature) provides the labels ($f(x)$)

- $\langle (1, 1, 1, 1, 1, 1, \dots, 1, 1), 1 \rangle$
- $\langle (1, 1, 1, 0, 0, 0, \dots, 0, 0), 0 \rangle$
- $\langle (1, 1, 1, 1, 1, 0, \dots, 0, 1, 1), 1 \rangle$
- $\langle (1, 0, 1, 1, 1, 0, \dots, 0, 1, 1), 0 \rangle$
- $\langle (1, 1, 1, 1, 1, 0, \dots, 0, 0, 1), 1 \rangle$
- $\langle (1, 0, 1, 0, 0, 0, \dots, 0, 1, 1), 0 \rangle$
- $\langle (1, 1, 1, 1, 1, 1, \dots, 0, 1), 1 \rangle$
- $\langle (0, 1, 0, 1, 0, 0, \dots, 0, 1, 1), 0 \rangle$

For a reasonable learning algorithm (by *elimination*), the final hypothesis will be

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

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?

Two Models for How good is our learning algorithm?

1. 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***

2. 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:** $S \times \{-1,1\}$: positive and negative examples of the target concept. (S is a finite subset of X)

$$\langle x_1, f(x_1) \rangle, \langle x_2, f(x_2) \rangle, \dots, \langle x_n, f(x_n) \rangle$$

- **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 $x \in X$?

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:** $S \times \{-1,1\}$: positive and negative examples of the target concept. (S is a finite subset of X)
 - *Training instances are generated by a fixed unknown probability distribution D over X*
- **What we want:** A hypothesis $h \in H$ such that $h(x) = f(x)$
 - Evaluate h on subsequent examples $x \in X$ drawn according to D

PAC Learning – Intuition

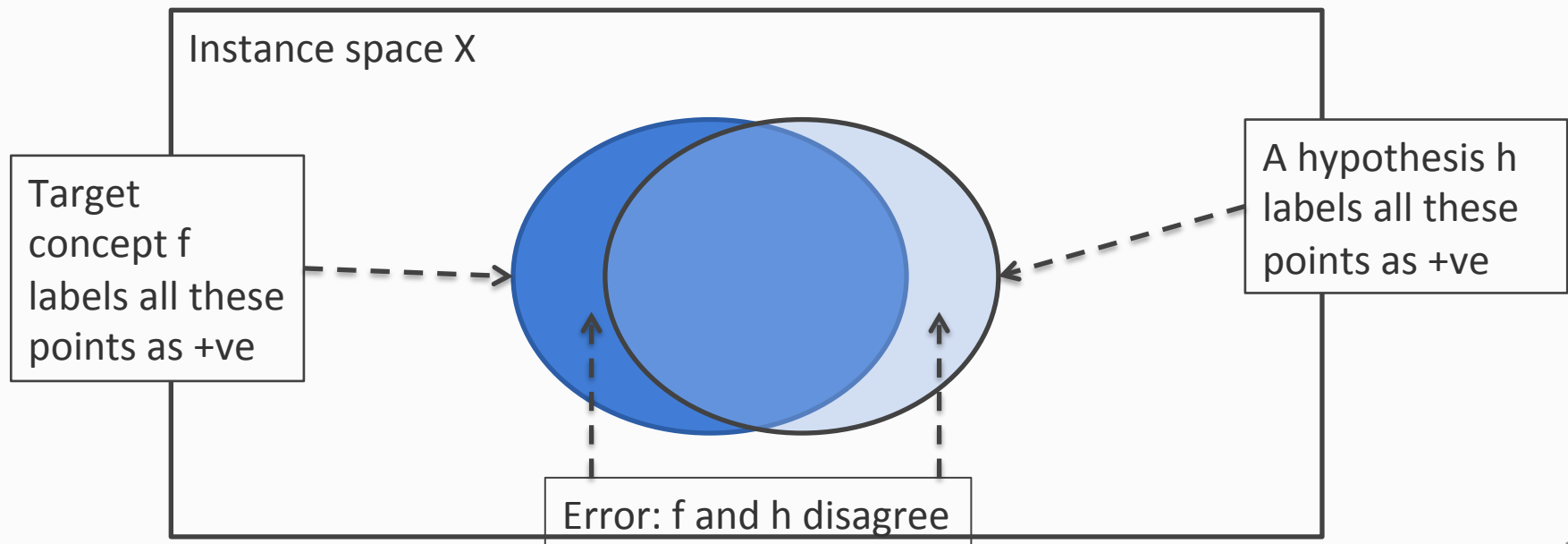
- The assumption of fixed distribution is important for two reasons:
 1. Gives us hope that what we learn on the training data will be meaningful on future examples
 2. Also gives a well-defined notion of the error of a hypothesis according to the target function
- “*The future will be like the past*”: 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 D over examples, the *error* of a hypothesis h with respect to a target concept f is

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



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. That is, $h(x) \neq f(x)$

Denoted by $\text{err}_S(h)$

Overfitting: When the empirical error on the training set $\text{err}_S(h)$ is substantially lower than $\text{err}_D(h)$

Batch learning

The goal of batch learning

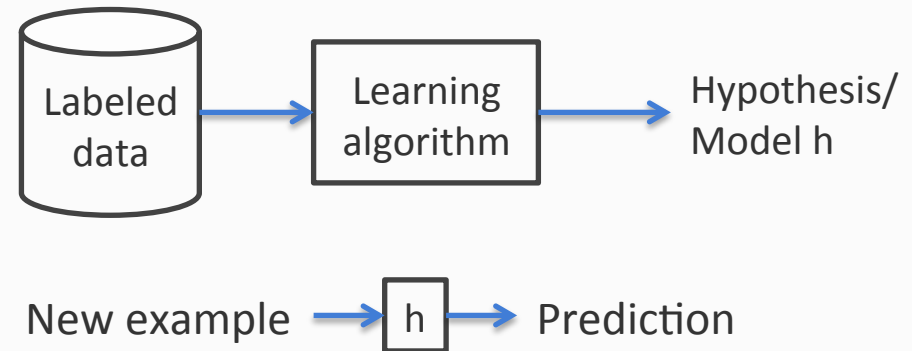
To devise good learning algorithms that avoid overfitting

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

Checkpoint: The bigger picture

- Supervised learning: instances, concepts, and hypotheses

- Specific learners
 - Decision trees
 - Nearest neighbors
 - Perceptron
 - Winnow



- General ML ideas
 - Features as high dimensional vectors
 - Overfitting
 - The curse of dimensionality
 - Mistake-bound: One way of asking “Can my problem be learned?”

Questions?

Online learning vs. Batch learning

Online learning

- No assumptions about the distribution of examples
- Learning is a sequence of trials
 - Learner sees a single example, makes a prediction
 - If mistake, update hypothesis
- **Goal**: To bound the total number of mistakes over time

Batch learning

- Examples are drawn from a fixed (perhaps unknown) probability distribution D over the instance space
- Learning uses a training set S , drawn i.i.d from the distribution D
- **Goal**: To find a hypothesis that has low chance of making a mistake on a new example from D

Where are we?

- The Theory of Generalization
 - When can we trust the learning algorithm?
 - What functions can be learned?
 - Batch Learning
- Probably Approximately Correct (PAC) learning
- Positive and negative learnability results
- Agnostic Learning
- Shattering and the VC dimension

This section

- Analyze a simple algorithm for learning conjunctions
- Define the PAC model of learning
- Make formal connections to the principle of Occam's razor

Learning Conjunctions

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

Training data

- $\langle (1,1,1,1,1,1,\dots,1,1), 1 \rangle$
- $\langle (1,1,1,0,0,0,\dots,0,0), 0 \rangle$
- $\langle (1,1,1,1,1,0,\dots,0,1,1), 1 \rangle$
- $\langle (1,0,1,1,1,0,\dots,0,1,1), 0 \rangle$
- $\langle (1,1,1,1,1,0,\dots,0,0,1), 1 \rangle$
- $\langle (1,0,1,0,0,0,\dots,0,1,1), 0 \rangle$
- $\langle (1,1,1,1,1,1,\dots,0,1), 1 \rangle$
- $\langle (0,1,0,1,0,0,\dots,0,1,1), 0 \rangle$

Learning Conjunctions

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

Training data

- $\langle (1,1,1,1,1,1,\dots,1,1), 1 \rangle$
- ~~$\langle (1,1,1,0,0,0,\dots,0,0), 0 \rangle$~~
- $\langle (1,1,1,1,1,0,\dots,0,1,1), 1 \rangle$
- ~~$\langle (1,0,1,1,1,1,0,\dots,0,1,1), 0 \rangle$~~
- $\langle (1,1,1,1,1,0,\dots,0,0,1), 1 \rangle$
- ~~$\langle (1,0,1,0,0,0,\dots,0,1,1), 0 \rangle$~~
- $\langle (1,1,1,1,1,1,\dots,0,1), 1 \rangle$
- ~~$\langle (0,1,0,1,0,0,\dots,0,1,1), 0 \rangle$~~

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

- $\langle (1, 1, 1, 1, 1, 1, \dots, 1, 1), 1 \rangle$
- $\langle (1, 1, 1, 0, 0, 0, \dots, 0, 0), 0 \rangle$
- $\langle (1, 1, 1, 1, 1, 0, \dots, 0, 1, 1), 1 \rangle$
- $\langle (1, 0, 1, 1, 1, 0, \dots, 0, 1, 1), 0 \rangle$
- $\langle (1, 1, 1, 1, 1, 0, \dots, 0, 0, 1), 1 \rangle$
- $\langle (1, 0, 1, 0, 0, 0, \dots, 0, 1, 1), 0 \rangle$
- $\langle (1, 1, 1, 1, 1, 1, \dots, 0, 1), 1 \rangle$
- $\langle (0, 1, 0, 1, 0, 0, \dots, 0, 1, 1), 0 \rangle$

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

- $\langle (1, 1, 1, 1, 1, 1, \dots, 1, 1), 1 \rangle$
- $\langle (1, 1, 1, 0, 0, 0, \dots, 0, 0), 0 \rangle$
- $\langle (1, 1, 1, 1, 1, 0, \dots, 0, 1, 1), 1 \rangle$
- $\langle (1, 0, 1, 1, 1, 0, \dots, 0, 1, 1), 0 \rangle$
- $\langle (1, 1, 1, 1, 1, \boxed{0}, \dots, 0, \boxed{0}, 1), 1 \rangle$
- $\langle (1, 0, 1, 0, 0, 0, \dots, 0, 1, 1), 0 \rangle$
- $\langle (1, 1, 1, 1, 1, 1, \dots, \boxed{0}, 1), 1 \rangle$
- $\langle (0, 1, 0, 1, 0, 0, \dots, 0, 1, 1), 0 \rangle$

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

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

Training data

- $\langle (1,1,1,1,1,1,\dots,1,1), 1 \rangle$
- $\langle (1,1,1,0,0,0,\dots,0,0), 0 \rangle$
- $\langle (1,1,1,1,1,0,\dots,0,1,1), 1 \rangle$
- $\langle (1,0,1,1,1,0,\dots,0,1,1), 0 \rangle$
- $\langle (1,1,1,1,1,0,\dots,0,0,1), 1 \rangle$
- $\langle (1,0,1,0,0,0,\dots,0,1,1), 0 \rangle$
- $\langle (1,1,1,1,1,1,\dots,0,1), 1 \rangle$
- $\langle (0,1,0,1,0,0,\dots,0,1,1), 0 \rangle$

A simple learning algorithm:

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

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

Clearly this algorithm produces a conjunction that is consistent with the data, that is $\text{err}_S(h) = 0$, if the target function is a monotone conjunction

Exercise: Why?

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

Why?

Learning Conjunctions: Analysis

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

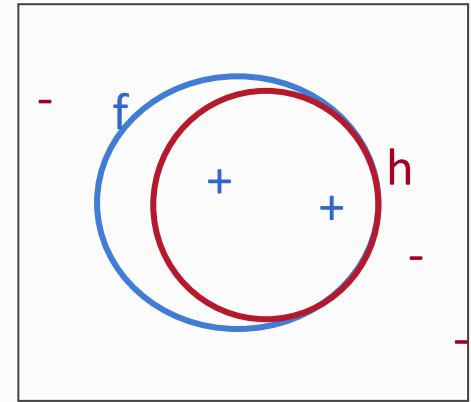
$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

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

Why?

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 $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ 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

$$m > \frac{n}{\epsilon} \left(\log(n) + \log \left(\frac{1}{\delta} \right) \right) \quad \text{Poly in } n, 1/\delta, 1/\epsilon$$

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

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

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

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

Let's prove this assertion

Learning Conjunctions: Analysis

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

- That is, 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, $p(z) = 0$

Remember that there will only be mistakes on positive examples for this toy problem

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

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

$\langle (0, 1, 1, 1, 1, 0, \dots, 0, 1, 1), 1 \rangle$

$p(x_1)$: Probability that this situation occurs

Learning Conjunctions: Analysis

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

- That is, 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, $p(z) = 0$

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

Via direct application of the union bound

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)

If there are no bad literals, then $\text{err}_D(h) < \epsilon$

- Why? Because $\text{err}_D(h) \leq \sum_{z \in h} p(z)$

Let us try to see when this will not happen

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)

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?

$$\begin{aligned} \Pr(z \text{ survives one example}) &= 1 - \Pr(z \text{ is eliminated by one example}) \\ &\leq 1 - p(z) \\ &< 1 - \frac{\epsilon}{n} \end{aligned}$$

There was one example of this kind

$\langle (1, 1, 1, 1, 1, 0, \dots, 0, 1, 1), 1 \rangle$

Learning Conjunctions: Analysis

n = dimensionality

What we know so far:

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

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

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

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

Learning Conjunctions: Analysis

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

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 δ

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

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

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

Learning Conjunctions: Analysis

To guarantee a probability of failure (i.e, error $> \epsilon$) that is less than δ , the number of examples we need is

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

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

That is, if m has this property, then

- With probability $1 - \delta$, there will be no bad literals,
- Or equivalently, with probability $1 - \delta$, we will have $\text{err}_D(h) < \epsilon$

How to use this:

- If $\epsilon = 0.1$ and $\delta = 0.1$, then for $n = 100$, we need 6908 training examples
- If $\epsilon = 0.1$ and $\delta = 0.1$, then for $n = 10$, we need only 461 examples
- If $\epsilon = 0.1$ and $\delta = 0.01$, then for $n = 10$, we need 691 examples

Learning Conjunctions: Analysis

To guarantee a probability of failure (i.e, error $> \epsilon$) that is less than δ , the number of examples we need is

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

That is, if m has this property, then

- With probability $1 - \delta$, there will be no bad literals,
- Or equivalently, with probability $1 - \delta$, we will have $\text{err}_D(h) < \epsilon$

What we have here is a PAC guarantee

*Our algorithm is **Probably Approximately Correct***

This section

- ✓ Analyze a simple algorithm for learning conjunctions
- Define the PAC model of learning
- Make formal connections to the principle of Occam's razor

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 $X \rightarrow Y$, taken from a concept class C
- h : a hypothesis function $X \rightarrow Y$ that the learning algorithm selects from a hypothesis class H
- S : a set of m training examples drawn from D , labeled with f
- $\text{err}_D(h)$: The true error of any hypothesis h
- $\text{err}_S(h)$: 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 (which represent a small fraction of the available instance space)
 - 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 (only in rare learning situations, we hope) the training set will not be representative (will contain uncommon examples)
- The only realistic expectation of a good learner is that **with high probability** it will learn a **close approximation** to the target concept

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 $1 - \delta$, a learner produces a hypothesis with error at most ϵ
- The only 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 $0 < \epsilon, \delta < 1$,

given m examples sampled independently according to D , the algorithm L produces, with probability at least $(1 - \delta)$, a hypothesis $h \in H$ that has error at most ϵ ,

where m is **polynomial** in $1/\epsilon$, $1/\delta$, n and $\text{size}(H)$

recall that $\text{Err}_D(h) = \Pr_D[f(x) \neq h(x)]$

The concept class C is **efficiently learnable** if L can produce the hypothesis in time that is polynomial in $1/\epsilon$, $1/\delta$, n and $\text{size}(H)$

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 ?

To be PAC learnable, there must be a hypothesis $h \in H$ with arbitrary small error for every $f \in C$. We assume $H \subseteq C$. (*Properly* PAC learnable if $H=C$)

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

This section

- ✓ Analyze a simple algorithm for learning conjunctions
- ✓ Define the PAC model of learning
- Make formal connections to the principle of Occam's razor

Occam's Razor

Named after William of Occam

– AD 1300s

Prefer simpler explanations over more complex ones

“Numquam ponenda est pluralitas sine necessitate”

Historically, a widely prevalent idea across different schools of philosophy



Towards formalizing Occam's Razor

(Assuming consistency)

Claim: The probability that there is a hypothesis $h \in H$ that:

1. Is **Consistent** with m examples, and
 2. Has $\text{err}_D(h) > \epsilon$
- is less than $|H| (1 - \epsilon)^m$
- } That is, **consistent** yet **bad**

Proof: Let h be such a bad hypothesis that has an error $> \epsilon$

Probability that h is consistent with one example is $\Pr[f(x) = h(x)] < 1 - \epsilon$

The training set consists of m examples drawn independently

So, probability that h is consistent with m examples $< (1 - \epsilon)^m$

Probability that *some bad hypothesis* in H is consistent with m examples is less than $|H| (1 - \epsilon)^m$

Occam's Razor

The probability that there is a hypothesis $h \in H$ that is

1. Consistent with m examples, and

2. Has $\text{err}_D(h) > \epsilon$

is less than $|H| (1 - \epsilon)^m$

Just like before, we want to make this probability small, say smaller than δ

$$|H| (1 - \epsilon)^m < \delta$$

$$\ln(|H|) + m \ln(1 - \epsilon) < \ln \delta$$

We know that $e^x = 1 - x + \frac{x^2}{2} - \frac{x^3}{6} \cdots > 1 - x$

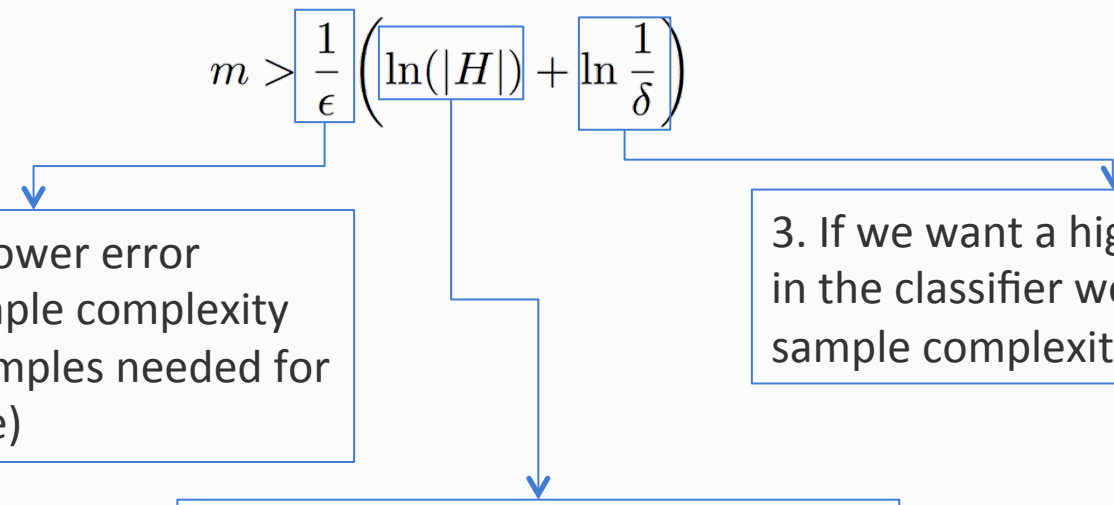
Let's use $\ln(1 - \epsilon) < -\epsilon$ to get a safer δ

That is, if $m > \frac{1}{\epsilon} \left(\ln(|H|) + \ln \frac{1}{\delta} \right)$ then, the probability of getting a bad hypothesis is small

Occam's Razor

Let H be any hypothesis space.

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

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


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.

Occam's Razor

Let H be any hypothesis space.

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

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

This is called the **Occam's Razor** because it expresses a preference towards smaller hypothesis spaces.

Shows when a m -consistent hypothesis generalizes well (i.e error $< \epsilon$).

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

Consistent Learners and Occam's Razor

From the definition, we get the following general scheme for PAC learning

Given a sample D of m examples

- Find some $h \in H$ that is consistent with *all* m examples
 - If m is large enough, a consistent hypothesis must be *close enough* to f
 - Check that m does not have to be too large (i.e polynomial in the relevant parameters): we showed that the “closeness” guarantee requires that
$$m > 1/\epsilon (\ln |H| + \ln 1/\delta)$$
- Show that the consistent hypothesis $h \in H$ can be computed efficiently

We worked out the details for conjunctions

- The Elimination algorithm to find a hypothesis h that is consistent with the training set (easy to compute)
- We showed directly that if we have sufficiently many examples (polynomial in the parameters), then h is close to the target function.

Exercises

We have seen the decision tree learning algorithm. Suppose our problem has n binary features. What is the size of the hypothesis space?

Are decision trees efficiently PAC learnable?

This lecture: Computational Learning Theory

- The Theory of Generalization
- Probably Approximately Correct (PAC) learning
- Positive and negative learnability results
- Agnostic Learning
- Shattering and the VC dimension

What can be learned

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

General conjunctions are PAC learnable

– $|H|$ = Number of conjunctions of n variables = 3^n

$$\ln(|H|) = n \ln(3)$$

– Number of examples needed $m > \frac{1}{\epsilon} \left(n \ln(3) + \ln \frac{1}{\delta} \right)$

- If we want to guarantee a 95% chance of learning a hypothesis of at least 90% accuracy, with $n=10$ Boolean variable, $m > (\ln(1/0.05) + 10\ln(3))/0.1 = 140$
- If $n=100$, this goes just to 1129. (linearly increases with n)
- Increasing the confidence to 1% will cost 1145 examples (logarithmic with δ)

These results hold for **any** consistent learner

What can be learned

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

3-CNF

$$(l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23}) \vee \dots$$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

What can be learned

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

3-CNF $(l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23}) \vee \dots$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

What is the **sample complexity**?

That is, if we had a consistent learner, how many examples will it need to guarantee PAC learnability?

We need the size of the hypothesis space. How many 3CNFs are there?

- Number of conjuncts = $O((2n)^3)$
- A 3-CNF is a conjunction with these many variables.
- $|H|$ = Number of 3-CNFs = $O(2^{(2n)^3})$
- $\log(|H|) = O(n^3)$

$\log(|H|)$ is polynomial in n
 \Rightarrow the sample complexity is also polynomial in n

What can be learned

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

3-CNF $(l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23}) \vee \dots$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

What is the **sample complexity**?

That is, if we had a consistent learner, how many examples will it need to guarantee PAC learnability?

We need the size of the hypothesis space. How many 3CNFs are there?

- Number of conjuncts = $O((2n)^3)$
- A 3-CNF is a conjunction with these many variables.
- $|H| = \text{Number of 3-CNFs} = O(2^{(2n)^3})$
- $\log(|H|) = O(n^3)$

$\log(|H|)$ is polynomial in n
 \Rightarrow the sample complexity is also polynomial in n

For PAC learnability, we still need an efficient algorithm that will find a consistent hypothesis.

Exercise: Find one

What can be learned

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

General Boolean functions

- How many Boolean functions exist with n variables? 2^{2^n}
So $\log(|H|)$ is exponential.
- General Boolean functions are **not** PAC learnable

Sample Complexity

- **k-CNF:** Conjunctions of any number of clauses where each disjunctive clause has at most k literals.
- **k-clause-CNF:** Conjunctions of at most k disjunctive clauses.

$$f = C_1 \wedge C_2 \wedge \dots \wedge C_k; \quad \ln(|k\text{-clause-CNF}|) = O(kn)$$
$$C_i = l_1 \vee l_2 \vee \dots \vee l_m$$

- **k-DNF:** Disjunctions of any number of terms where each conjunctive term has at most k literals. $f = T_1 \vee T_2 \vee \dots \vee T_m$
 $T_i = l_1 \wedge l_2 \wedge \dots \wedge l_k$
- **k-term-DNF:** Disjunctions of at most k conjunctive terms.

All these classes can be learned using a polynomial size sample

Suppose we want to learn a 2-term-DNF
What should our hypothesis class be?

Computational Complexity

Suppose we want to learn a 2-term-DNF

- Though sample complexity is polynomial, the computational complexity is prohibitive in this case
- Determining whether there is a 2-term DNF consistent with a set of training data is NP-hard
- That is, the class of k -term-DNF is **not** efficiently (properly) PAC learnable due to computational complexity
- But, we have seen an algorithm for learning k -CNF
- And, k -CNF is a superset of k -term-DNF

(That is, every k -term-DNF can be written as a k -CNF)

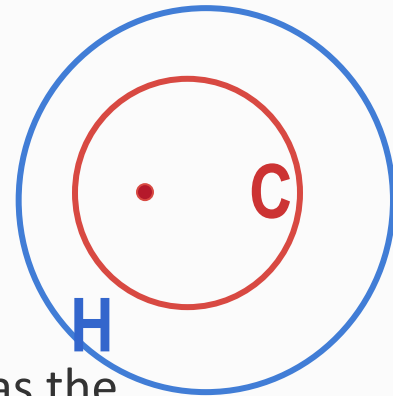
$$T_1 \vee T_2 \vee T_3 = \prod_{x \in T_1, y \in T_2, z \in T_3} x \vee y \vee z$$

$$(a \wedge b \wedge c) \vee (b \wedge d \wedge e) =$$

Example: $(a \vee b) \wedge (a \vee d) \wedge (a \vee e) \wedge b \wedge (b \vee d) \wedge (b \vee e) \wedge (c \vee b) \wedge (c \vee d) \wedge (c \vee e)$

Computational Complexity

- Determining whether there is a 2-term DNF consistent with a training set is NP-Hard
- That is, the class of k -term-DNF is **not** efficiently (*properly*) PAC learnable due to computational complexity
- But, we have seen an algorithm for learning k -CNF
 - Actually, left as an exercise
- And, k -CNF is a superset of k -term-DNF
 - (That is, every k -term-DNF can be written as a k -CNF)
- Therefore, $C = k$ -term-DNF can be learned as using $H = k$ -CNF as the hypothesis space



Computational Complexity

- Determining whether there is a 2-term DNF consistent with a training set is NP-Hard
- That is, the class of k -term-DNF is **not** efficiently (*properly*) PAC learnable due to computational complexity
- But, we have seen an algorithm for learning k -CNF
 - Actually, left as an exercise
- And, k -CNF is a superset of k -term-DNF
 - (That is, every k -term-DNF can be written as a k -CNF)
- Therefore, $C = k$ -term-DNF can be learned as using $H = k$ -CNF as the hypothesis space

The lesson: Importance of representation Concepts that cannot be learned using one representation can *sometimes* be learned using a different, more expressive, representation

Computational Complexity

- Determining whether there is a 2-term DNF consistent with a training set is NP-Hard
- That is, the class of k -term-DNF is **not** efficiently (*properly*) PAC learnable due to computational complexity
- But, we have seen an algorithm for learning k -CNF
 - Actually, left as an exercise
- And, k -CNF is a superset of k -term-DNF
 - (That is, every k -term-DNF can be written as a k -CNF)
- Therefore, $C = k$ -term-DNF can be learned as using $H = k$ -CNF as the hypothesis space

The lesson: Importance of representation Concepts that cannot be learned using one representation can *sometimes* be learned using a different, more expressive, representation

We have seen this idea before: Linear classifiers for conjunctions

Negative Results – Examples

- Two types of non-learnability results

1. Complexity Theoretic (computational complexity bad)

- Showing that various concepts classes cannot be learned, based on well-accepted assumptions from computational complexity theory
- Takes the form “A concept class C cannot be learned unless $P=NP$ ”

2. Information Theoretic (sample complexity bad)

- The concept class is sufficiently rich that a polynomial number of examples may not be sufficient to distinguish a particular target concept
- Both type involve “representation dependent” arguments
- The proof shows that a given class cannot be learned by algorithms using hypotheses from the same class. (So?)

Negative Results for Learning

- Complexity Theoretic

- k -term DNF, for $k > 1$ (k -clause CNF, $k > 1$)
- Neural Networks of fixed architecture (3 nodes; n inputs)
- “read-once” Boolean formulas
- Quantified conjunctive concepts

- Information Theoretic

- Arbitrary Boolean functions (DNF Formulas or CNF Formulas)
- Deterministic Finite Automata
- Context Free Grammars

This lecture: Computational Learning Theory

- The Theory of Generalization
- Probably Approximately Correct (PAC) learning
- Positive and negative learnability results
- Agnostic Learning
- Shattering and the VC dimension

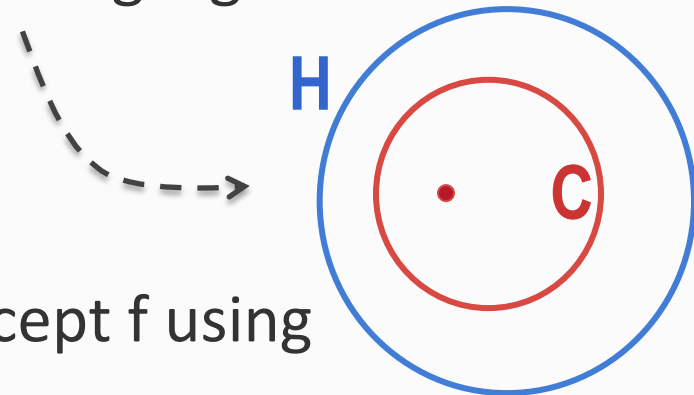
So far we have seen...

- The general setting for batch learning
- PAC learning and Occam's Razor
 - How good will a classifier that is *consistent* on a training set be?
- Two assumptions so far:
 1. Training and test examples come from the same distribution
 2. For any concept, there is some function in the hypothesis space that is consistent with the training set

Is the second assumption reasonable?

What is agnostic learning?

- So far, we have assumed that the learning algorithm could find the true concept



- **What if:** We are trying to learn a concept f using hypotheses in H , but $f \notin H$
 - That is C is not a subset of H
 - This setting is called *agnostic learning*
 - Can we say something about sample complexity?

More realistic setting than before

Agnostic Learning

Learn a concept f using
hypotheses in H , but $f \notin H$

- Are we guaranteed that training error will be zero?
 - No. There may be no consistent hypothesis in the hypothesis space!
- Our goal should be to find a classifier $h \in H$ that has low training error

$$err_S(h) = \frac{|\{f(x) \neq h(x), x \in S\}|}{m}$$

- This is the fraction of training examples that are misclassified

Agnostic Learning

Learn a concept f using
hypotheses in H , but $f \notin H$

- Our goal should be to find a classifier $h \in H$ that has low training error

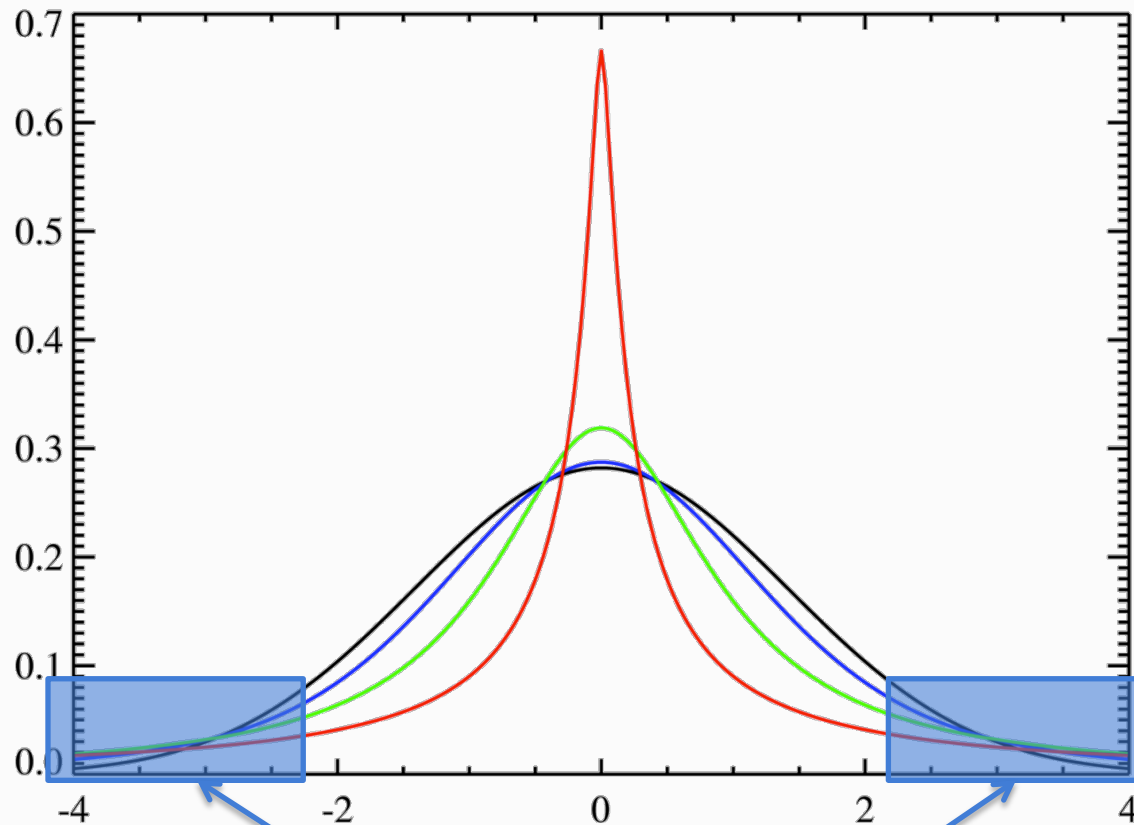
$$err_S(h) = \frac{|\{f(x) \neq h(x), x \in S\}|}{m}$$

- What we want: A guarantee that a hypothesis with small training error will have a good accuracy on unseen examples

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

Detour: Tail bounds

How far can a random variable get from its mean?



Tails of these distributions

Bounding probabilities

- **Markov's inequality**: Bounds the probability that a non-negative random variable exceeds a fixed value

$$P[X \geq a] \leq \frac{E[X]}{a}$$

- **Chebyshev's inequality**: Bounds the probability that a random variable differs from its expected value by more than a fixed number of standard deviations

$$P[|X - \mu| \geq k\sigma] \leq \frac{1}{k^2}$$

- What we want: To bound sums of random variables
 - Why? Because the training error depends on the number of errors on the training set

Hoeffding's inequality

Upper bounds on how much the sum of a set of random variables differs from its expected value

$$P[p > \bar{p} + \epsilon] \leq e^{-2m\epsilon^2}$$

Expected mean
(Eg. For a coin toss, the
probability of seeing heads)

Empirical mean, computed
over m independent trials

What this tells us: The empirical mean will not be too far from the expected mean if there are many samples.

And, it quantifies the convergence rate as well.


Back to agnostic learning

Suppose we consider the true error (a.k.a generalization error) $err_D(h)$ to be a random variable

The training error over m examples $err_S(h)$ is the empirical estimate of this true error

Let's apply Hoeffding's inequality

$$P[Err_D(h) > Err_S(h) + \epsilon] \leq e^{-2m\epsilon^2}$$


$$err_D(h) = Pr_{x \sim D}[f(x) \neq h(x)] \quad err_S(h) = \frac{|\{f(x) \neq h(x), x \in S\}|}{m}$$

Agnostic learning

The probability that a single hypothesis h has a training error that is more than ϵ away from the true error is bounded above

$$P [Err_D(h) > Err_S(h) + \epsilon] \leq e^{-2m\epsilon^2}$$

The learning algorithm looks for the best one of the $|H|$ possible hypotheses

The probability that there is a hypothesis in $|H|$ whose training error is ϵ away from the true error is bounded above

$$P [\exists h; Err_D(h) > Err_S(h) + \epsilon] \leq |H|e^{-2m\epsilon^2}$$

Agnostic learning

The probability that there is a hypothesis in $|H|$ whose training error is ϵ away from the true error is bounded above

$$P [\exists h; Err_D(h) > Err_S(h) + \epsilon] \leq |H|e^{-2m\epsilon^2}$$

Same game as before: We want this probability to be smaller than δ

$$|H|e^{-2m\epsilon^2} \leq \delta$$

Rearranging this gives us

$$m \geq \frac{1}{2\epsilon^2} \left[\ln |H| + \ln \left(\frac{1}{\delta} \right) \right]$$

Agnostic learning: Interpretations

1. An agnostic learner makes no commitment to whether f is in H and returns the hypothesis with least training error over at least m examples.

It can guarantee with probability $1 - \delta$ that the training error is not off by more than ϵ from the training error if

$$m \geq \frac{1}{2\epsilon^2} \left[\ln |H| + \ln \left(\frac{1}{\delta} \right) \right]$$

Difference between generalization and training errors: How much worse will the classifier be in the future than it is at training time?

Size of the hypothesis class:
Again an Occam's razor argument – prefer smaller sets of functions

Agnostic learning: Interpretations

1. An agnostic learner makes no commitment to whether f is in H and returns the hypothesis with least training error over at least m examples.

It can guarantee with probability $1 - \delta$ that the training error is not off by more than ϵ from the training error if

$$m \geq \frac{1}{2\epsilon^2} \left[\ln |H| + \ln \left(\frac{1}{\delta} \right) \right]$$

2. We have a *generalization bound*: A bound on how much the true error will deviate from the training error

$$err_D(h) - err_S(h) \leq \sqrt{\frac{\ln |H| + \ln(1/\delta)}{2m}}$$

Expected error

Training error

What we have seen so far

Occam's razor: When the hypothesis space contains the true concept

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

Agnostic learning: When the hypothesis space may not contain the true concept

$$m \geq \frac{1}{2\epsilon^2} \left[\ln |H| + \ln \left(\frac{1}{\delta} \right) \right]$$

Learnability depends on the log of the size of the hypothesis space

Have we solved everything? Eg: What about linear classifiers?

This lecture: Computational Learning Theory

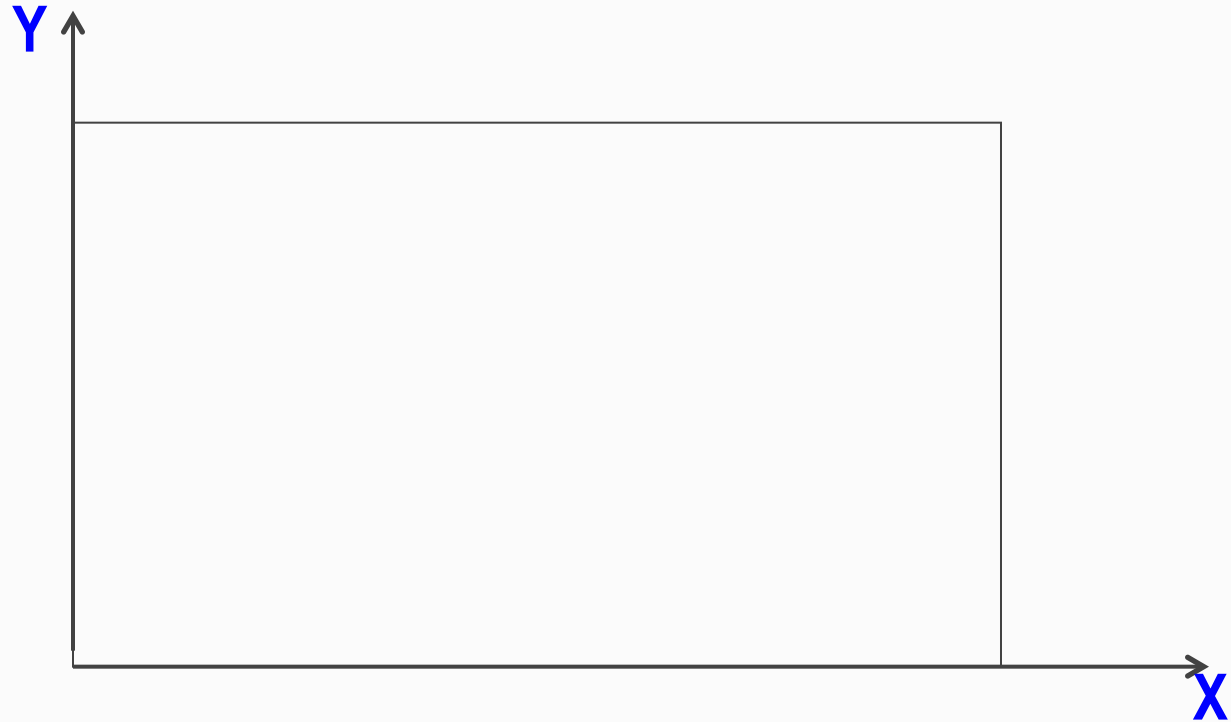
- The Theory of Generalization
- Probably Approximately Correct (PAC) learning
- Positive and negative learnability results
- Agnostic Learning
- Shattering and the VC dimension

Infinite Hypothesis Space

- The previous analysis was restricted to finite hypothesis spaces
- Some infinite hypothesis spaces are more expressive than others
 - E.g., Rectangles, vs. 17- sides convex polygons vs. general convex polygons
 - Linear threshold function vs. a combination of LTUs
- Need a measure of the expressiveness of an infinite hypothesis space other than its size
- 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)**

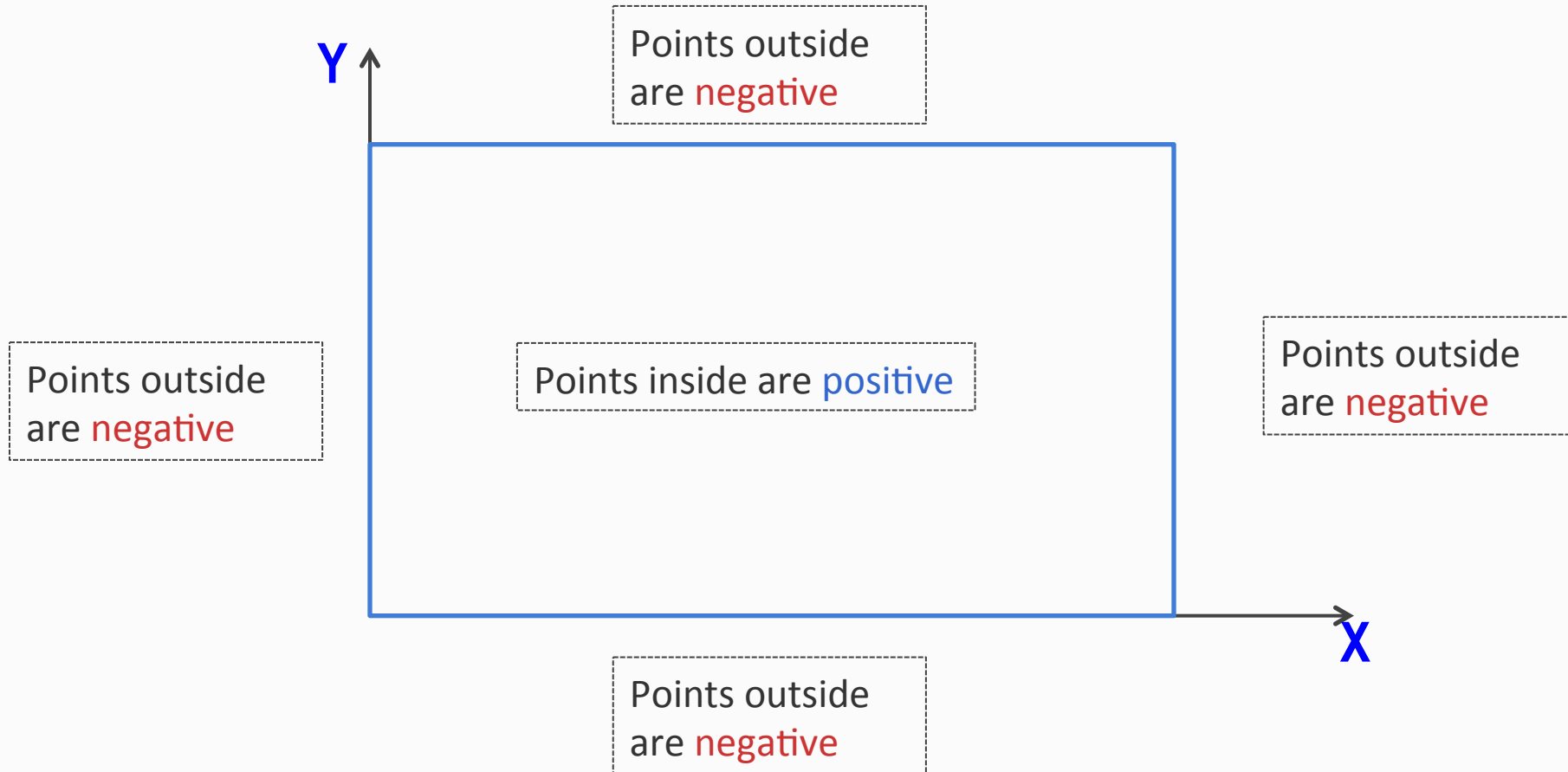
Learning Rectangles

Assume the target concept is an axis parallel rectangle



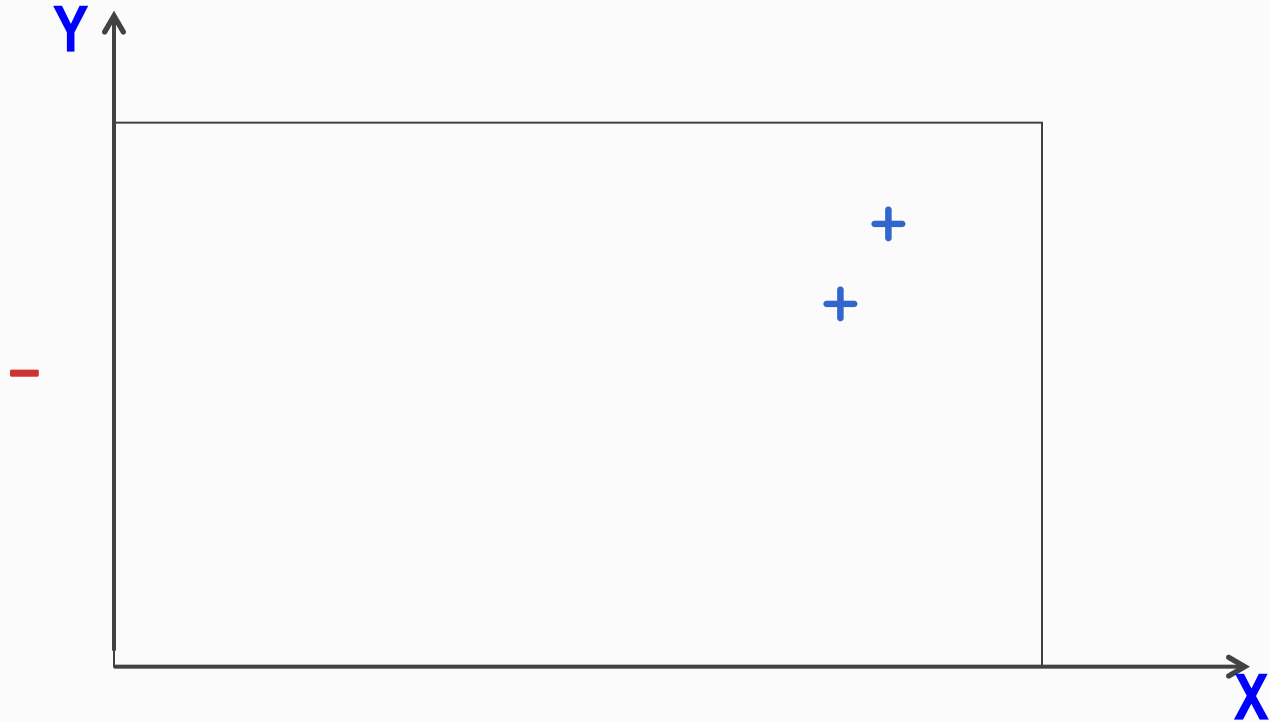
Learning Rectangles

Assume the target concept is an axis parallel rectangle



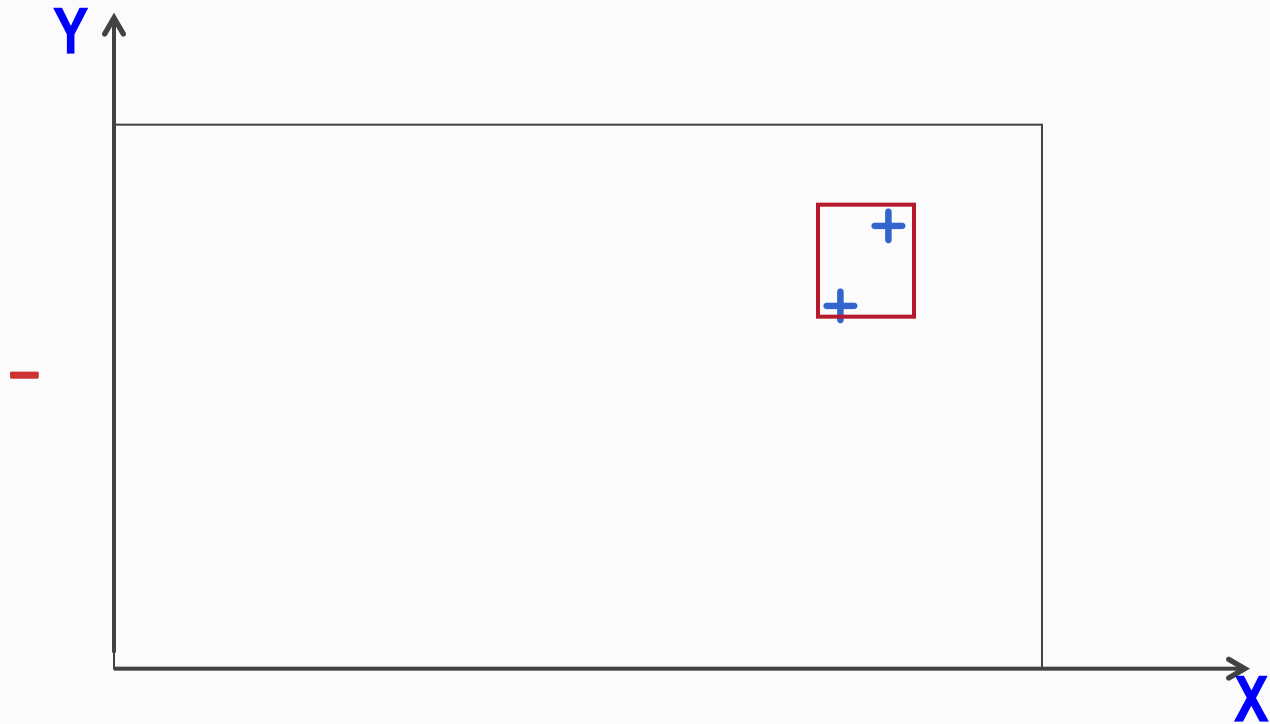
Learning Rectangles

Assume the target concept is an axis parallel rectangle



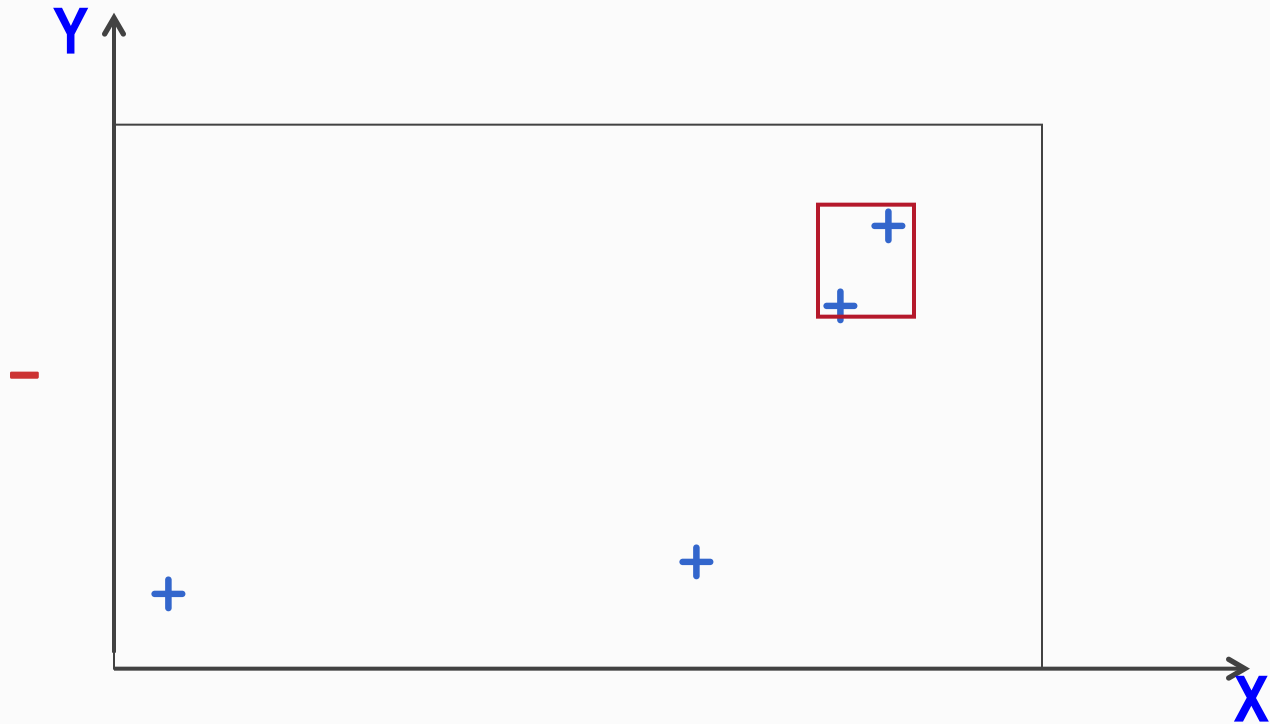
Learning Rectangles

Assume the target concept is an axis parallel rectangle



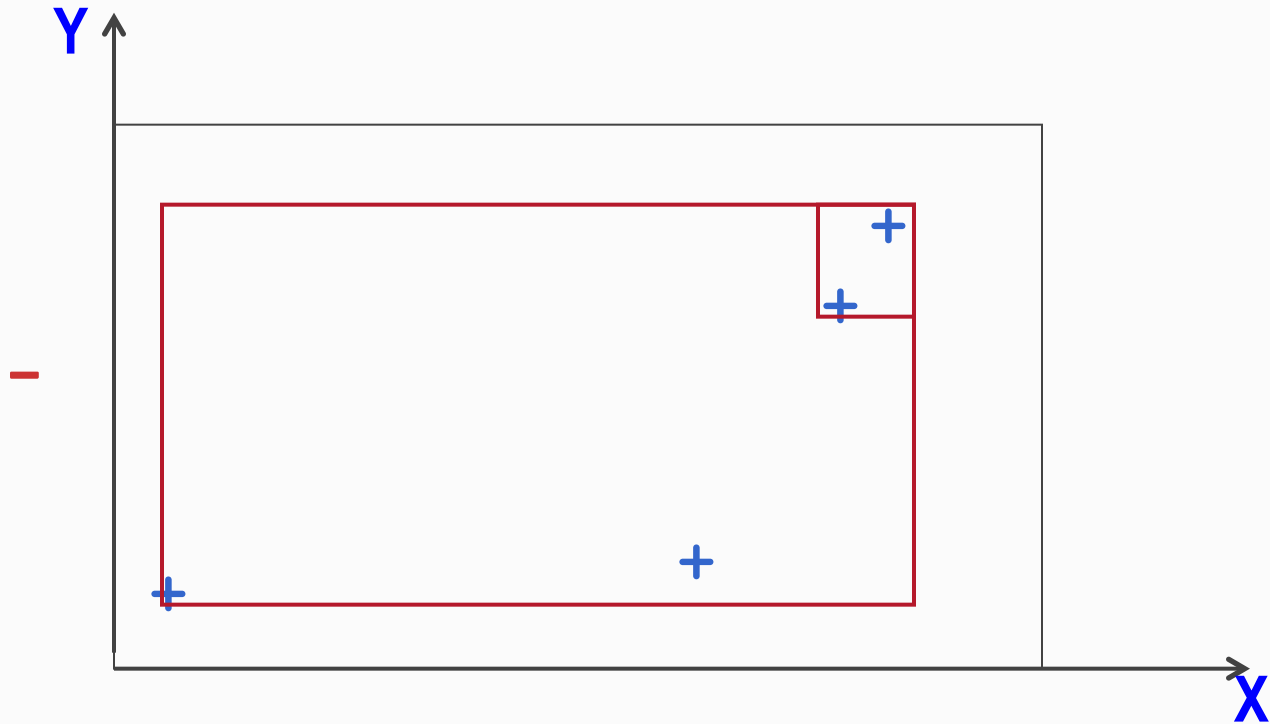
Learning Rectangles

Assume the target concept is an axis parallel rectangle



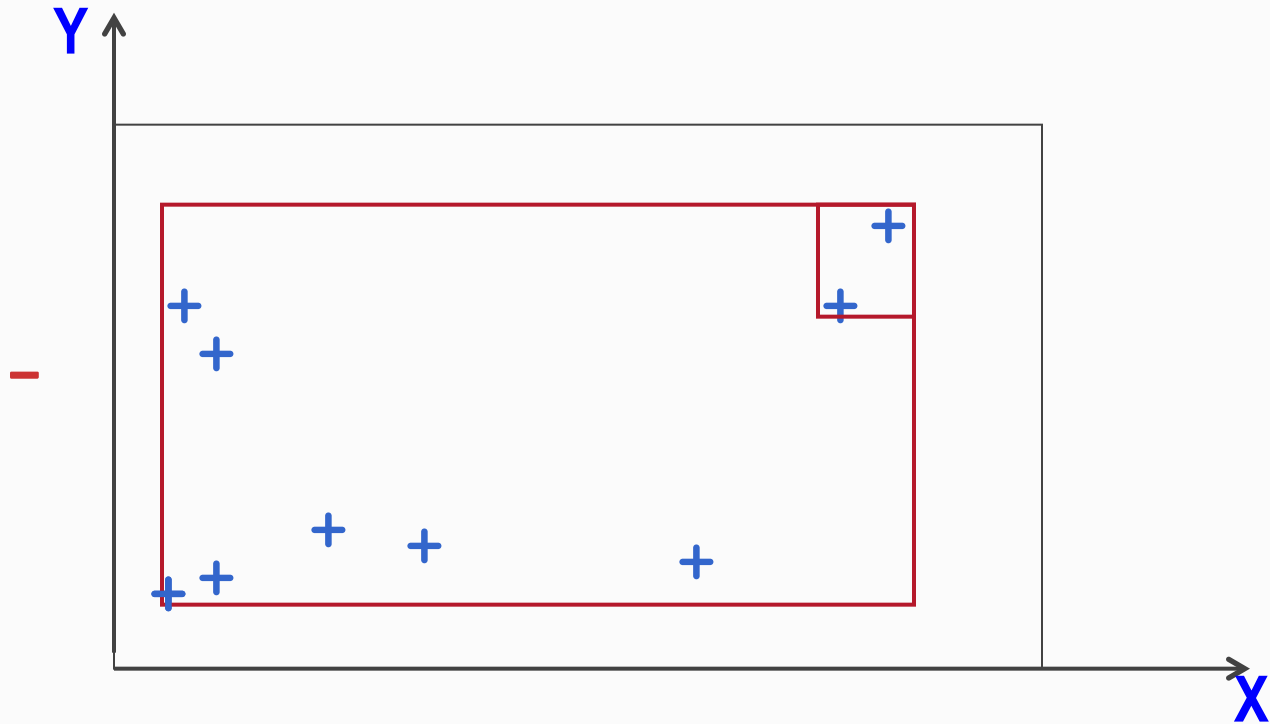
Learning Rectangles

Assume the target concept is an axis parallel rectangle



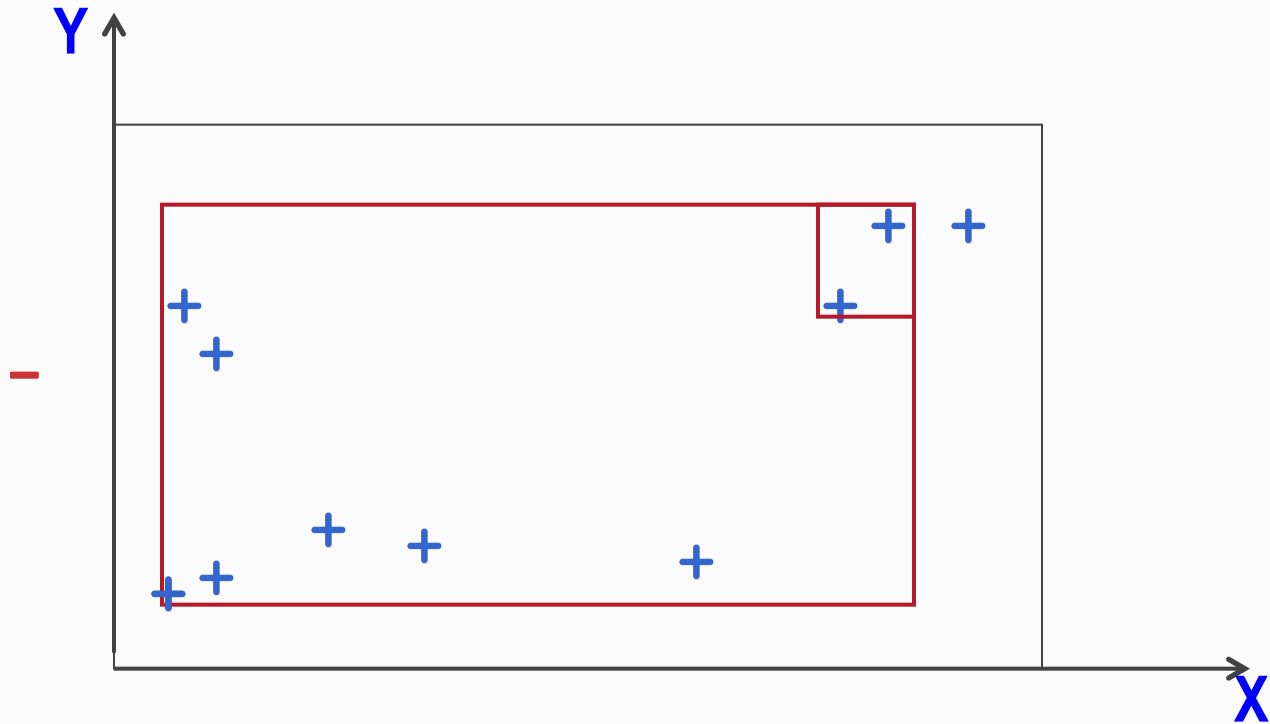
Learning Rectangles

Assume the target concept is an axis parallel rectangle



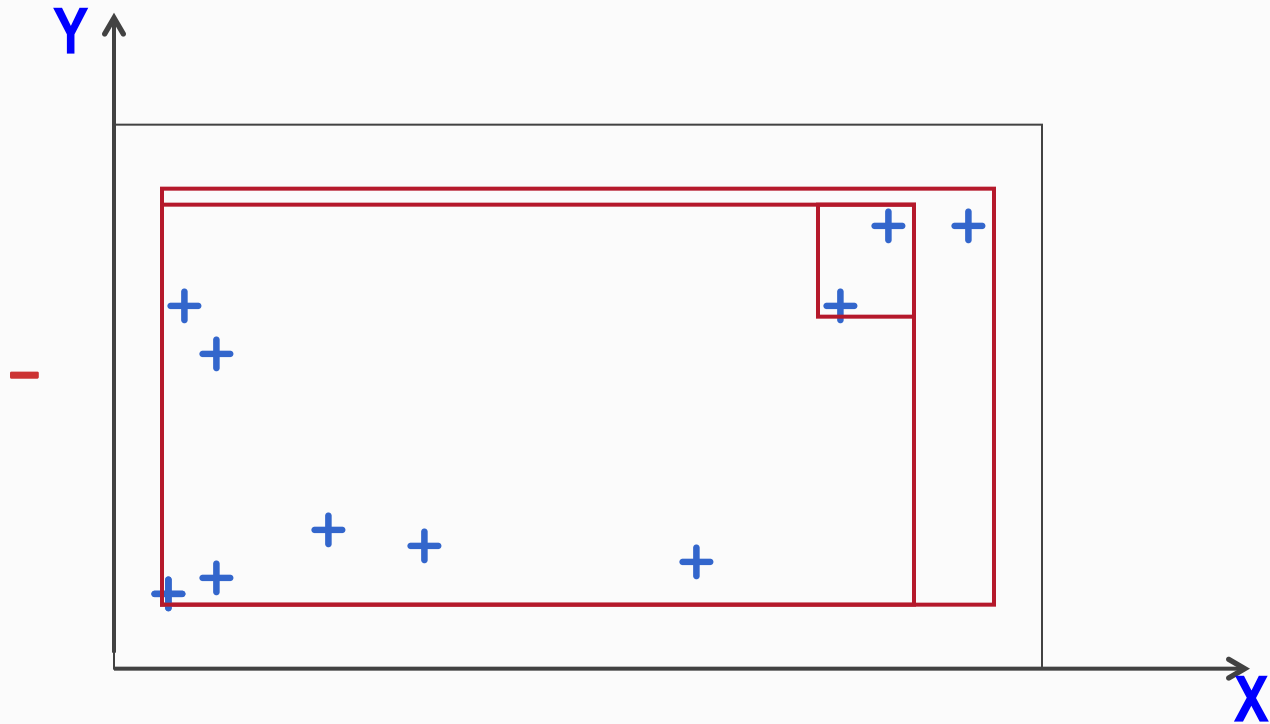
Learning Rectangles

Assume the target concept is an axis parallel rectangle



Learning Rectangles

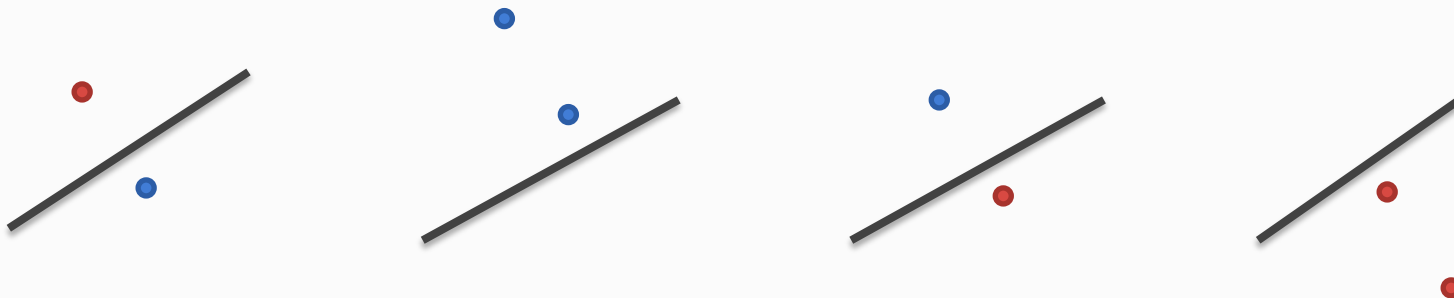
Assume the target concept is an axis parallel rectangle



Will we be able to learn the target rectangle?

Can we come close?

Let's think about expressivity of functions



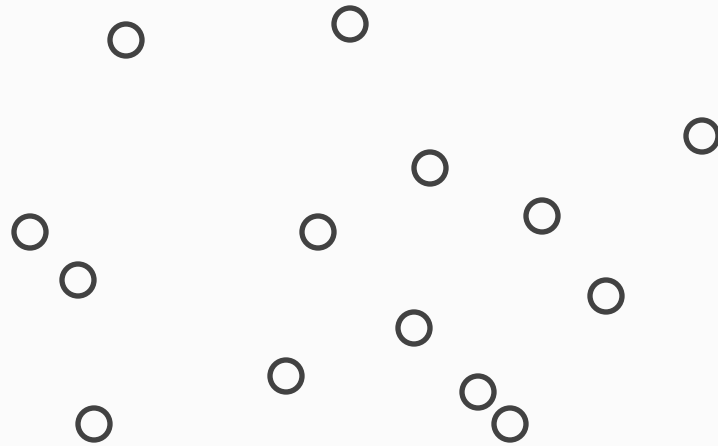
There are four ways to label two points

And it is possible to draw a line that separates positive and negative points in all four cases

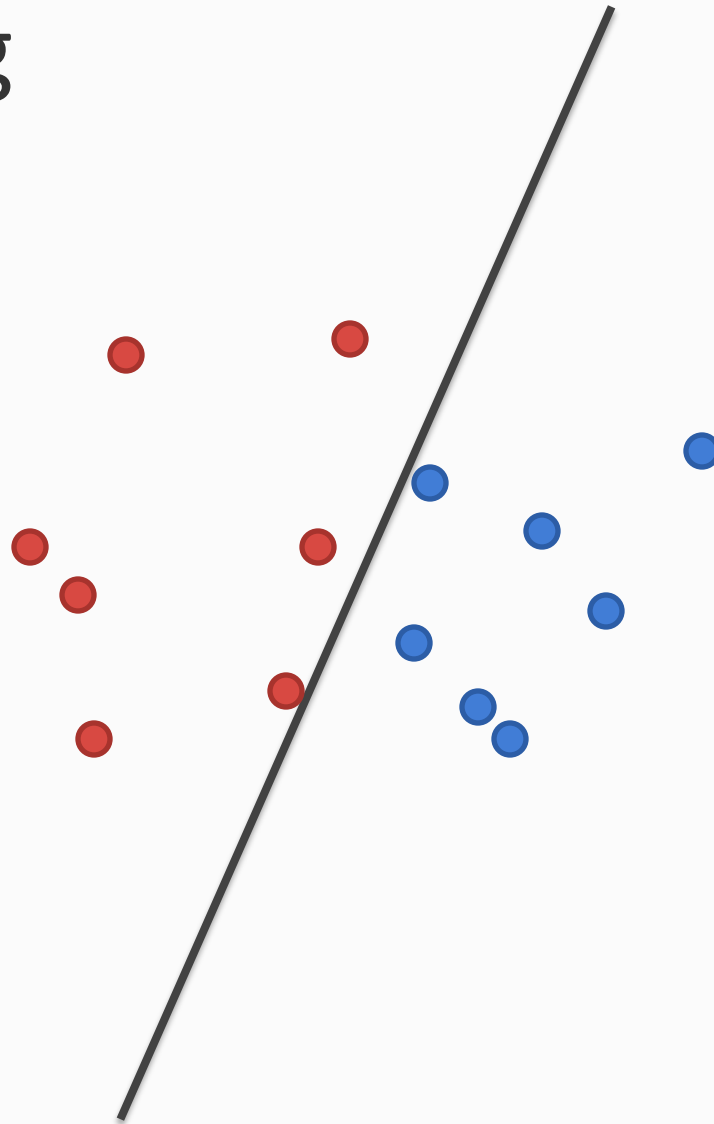
Linear functions are expressive enough to *shatter* 2 points

What about fourteen points?

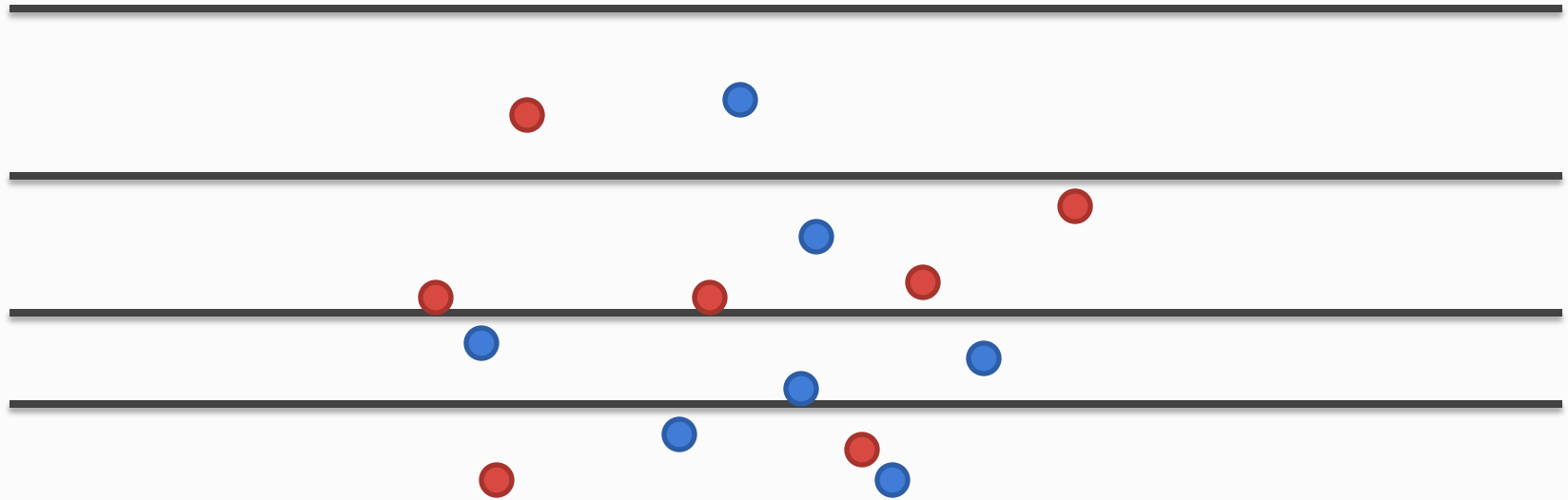
Shattering



Shattering

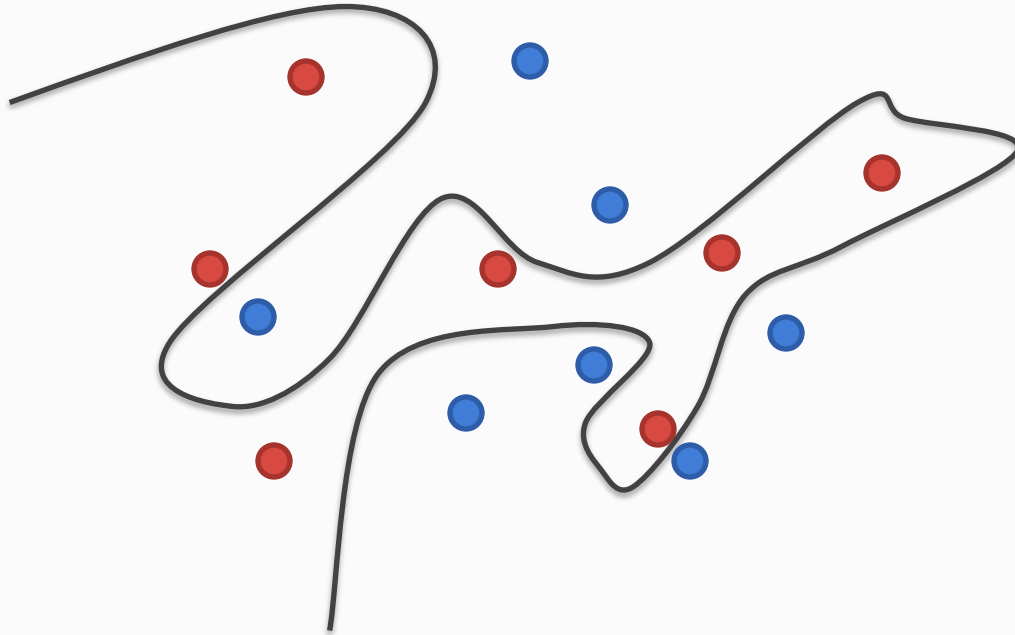


Shattering



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

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

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



Left bounded intervals

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

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

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

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

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

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

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



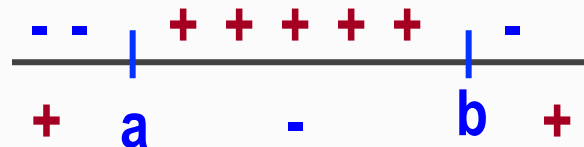
Real intervals

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

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

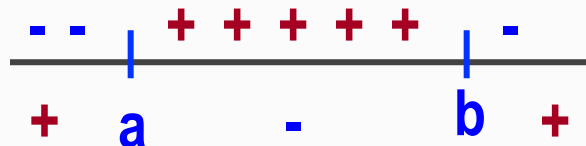
Example 2: Hypothesis class is the set of intervals on the real axis: $[a, b]$, for some real numbers $b > a$



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

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



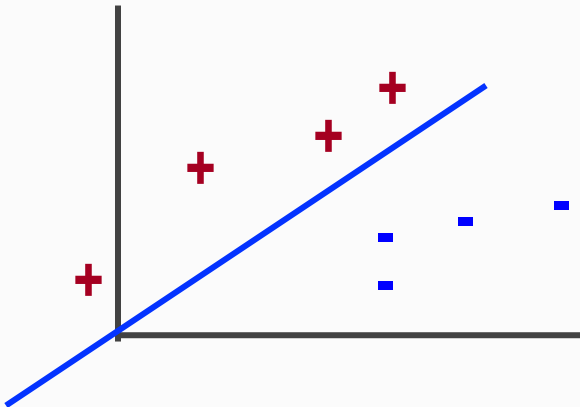
All sets of one or two points can be shattered
But sets of **three** points **cannot** be shattered

Proof? Enumerate all possible three points

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

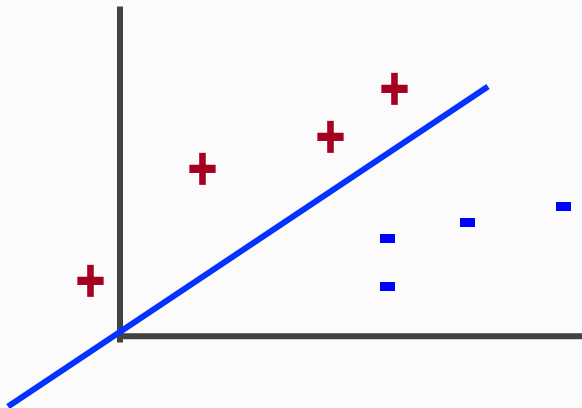
Example 3: Half spaces in a plane



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

Example 3: Half spaces in a plane



Can one point be shattered?

Two points?

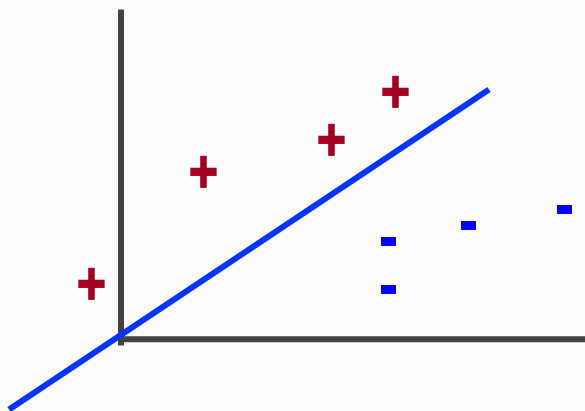
Three points? Can any three points be shattered?

Half spaces on a plane: 3 points

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

Example 3: Half spaces in a plane



Can four points be shattered?

Suppose three of them lie on the same line, label the outside points + and the inner one –

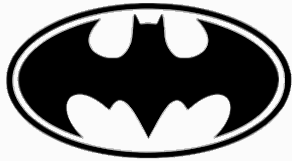
Other wise, make a convex hull. Label points outside + and the inner one –

Four points **cannot** be shattered!

Half spaces on a plane: 4 points

Shattering: The adversarial game

You



An adversary



You: Hypothesis class H can shatter these d points

Adversary: That's what you think!
Here is a labeling that will defeat you.

You: Aha! There is a function $h \in H$ that correctly predicts your evil labeling

Adversary: Argh! You win this round.
But I'll be back.....

Some functions can shatter infinite points!

If arbitrarily large finite subsets of the instance space X can be shattered by a hypothesis space H , then its VC dimension is infinite

An unbiased hypothesis space H shatters the entire instance space X , i.e., it can induce every possible partition on the set of all possible instances

The larger the subset X that can be shattered, the more expressive a hypothesis space is, i.e., the less biased

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

Vapnik-Chervonenkis Dimension

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

Definition: The **VC dimension** of hypothesis space H over instance space X is the size of the largest finite subset of X that is shattered by H

- If there **exists** any subset of size d that can be shattered, $VC(H) \geq d$
 - Even one subset will do
- If **no subset** of size d can be shattered, then $VC(H) < d$

What we have managed to prove

Concept class	VC Dimension	Why?
Half intervals	1	There is a dataset of size 1 that can be shattered No dataset of size 2 can be shattered
Intervals	2	There is a dataset of size 2 that can be shattered No dataset of size 3 can be shattered
Half-spaces in the plane	3	There is a dataset of size 3 that can be shattered No dataset of size 4 can be shattered

More VC dimensions

Concept class	VC Dimension
Linear threshold unit in d dimensions	$d + 1$
Neural networks	Number of parameters
1 nearest neighbors	infinite

What is the number of parameters needed to specify a linear threshold unit in d dimensions? $d + 1$

Local minima in learning means neural networks may not find the best parameters

Exercise: Try to prove this

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

Why VC dimension?

- Remember sample complexity
 - Occam's razor
 - Agnostic learning
- Sample complexity in both cases depends on the log of the size of the hypothesis space
- For infinite hypothesis spaces, its VC dimension behaves like $\log(|H|)$

VC dimension and Occam's razor

- Using $VC(H)$ as a measure of expressiveness we have an **Occam algorithm** for infinite hypothesis spaces
- Given a sample D with m examples, find some $h \in H$ is consistent with all m examples. If

$$m > \frac{1}{\epsilon} \left(8VC(H) \log \frac{13}{\epsilon} + 4 \log \frac{2}{\delta} \right)$$

Then with probability at least $(1-\delta)$, h has error less than ϵ .

That is, if m is polynomial we have a PAC learning algorithm;

To be efficient, we need to produce the hypothesis h efficiently

VC dimension and Agnostic Learning

- Similar statement can be made for the agnostic setting as well

If we have m examples, then with probability $1 - \delta$, the true error of a hypothesis h with training error $err_S(h)$ is bounded by

$$err_D(h) \leq err_S(h) + \sqrt{\frac{VC(H) \left(\ln \frac{2m}{VC(H)} + 1 \right) + \ln \frac{4}{\delta}}{m}}$$

Exercises

- What is the VC dimension of a finite hypothesis class?
- Your homework 3 asks you to compute the VC dimension of different classes of functions

PAC learning: What you need to know

- What is PAC learning?
 - Remember: We care about generalization error, not training error
- Finite hypothesis spaces
 - Connection between size of hypothesis space and sample complexity
 - Derive and understand the sample complexity bounds
 - Count number of hypotheses in a hypothesis class
- Infinite hypotheses classes
 - What is shattering and VC dimension?
 - How to find VC dimension of simple concept classes?
 - Higher VC dimensions \Rightarrow more sample complexity

Computational Learning Theory

- The Theory of Generalization
 - When can we trust the learning algorithm?
 - What functions can be learned?
 - Batch Learning
- Probably Approximately Correct (PAC) learning
- Positive and negative learnability results
- Agnostic Learning
- Shattering and the VC dimension

Why computational learning theory

- Answers questions such as
 - What is learnability? How good is my class of functions?
 - Is a concept learnable? How many examples do I need?
- Mistake bounds imply PAC-learnability
- Raises interesting theoretical questions
 - If a concept class is weakly learnable (i.e there is a learning algorithm that can produce a classifier that does slightly better than chance), does this mean that the concept class is strongly learnable?

Boosting

- We have seen bounds of the form
$$\text{true error} < \text{training error} + (\text{a term with } \delta \text{ and VC dimension})$$

Can we use this to define a learning algorithm?

Structural Risk Minimization principle

Support Vector Machine