Machine Learning

COLT+ Ensembles

Dan Goldwasser

dgoldwas@purdue.edu

Computational Learning Theory

- *What general laws constrain inductive learning ?*
  - *What learning problems can be solved? (what is learnable?)*
  - *When can we trust the output of a learning algorithm ?*
- We seek theory to relate
  - Probability of successful Learning
  - Number of training examples
  - Relation to the *complexity of hypothesis space*
  - *Accuracy* to which target concept is approximated

Requirements of Learning

- **Cannot** expect a learner to learn a concept **exactly**, since
  - *Typically multiple concepts will be consistent with available data* (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*. (agree = good approximation of f)
- **Cannot** always expect to learn a **close approximation** to the target concept since
  - 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: *with high probability learn a close approximation to the target concept*

Probably Approximately Correct

- Realistically - successful learning has high probability to learn a close approximation of the target concept.

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

The reason we can hope for that 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.*

C is **PAC learnable** by L using H if for all $f \in C$, for all distribution D over X, and fixed $0 < \varepsilon, \delta < 1$, Given a collection of m examples sampled independently according to the distribution D, L produces with probability at least $(1 - \delta)$ a hypothesis $h \in H$ with error at most ε, where m is polynomial in $1/\varepsilon$, $1/\delta$, n and size(C)

C is **efficiently learnable** if *L can produce the hypothesis in time polynomial in* $1/\varepsilon$, $1/\delta$, *n and size(C)*

PAC Learnability

- 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 an *arbitrary* small error for every $f \in C$. We generally assume $H \supseteq C$

- (*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

Occam's Razor

"plurality should not be posited without necessity"
William of Ockham (ca. 1285-1349)

Claim

The probability that there exists a hypothesis h∈H that

(1) is consistent with m examples and
(2) satisfies error(h) > ε ("bad hypothesis")

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

($Error_D(h) = Pr_{x \in D}[f(x) \neq h(x)]$)

Proof

Let h be such a bad hypothesis.

The probability that h is consistent with one example of f is

$$\Pr_{x \in D}[f(x) = h(x)] < 1 - \varepsilon$$

- Since the m examples are drawn independently of each other.
- The prob. that h is consistent with m examples of f is < $(1-\varepsilon)^m$
- The probability that some hypothesis in H is consistent with m examples is less than $|H|(1-\varepsilon)^m$

Occam's Razor

- We want the probability of finding a **bad *consistent*** hypothesis to be bounded by **δ**

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

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

Using the inequality:

$$\mathbf{e^{-x} > 1-x;}$$

Note: $\ln (1-\varepsilon) < -\varepsilon$; *gives a safer* δ

$$m > \frac{1}{\varepsilon}\{\ln(|H|) + \ln(1/\delta)\}$$

What do we know now about the Consistent Learner scheme?

We showed that a m-consistent hypothesis generalizes well (err< ε) (Appropriate m is a function of |H|, ε, δ)

It is called Occam's razor, because it indicates a preference towards small hypothesis spaces

Consistent Learners

- From the definition, we get the following general scheme for 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 need not be too large:
    we showed that the "closeness" guarantee requires that
    $$m > 1/\varepsilon \, (\ln |H| + \ln 1/\delta)$$
  - Show that the consistent hypothesis $h \in H$ can be computed efficiently

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

PAC Learning: Extensions

- So far we assumed:
  - Consistent learner (no mistakes on training data)
  - Finite size hypothesis space
- **Restrictive Assumptions!**

In general, we cannot assume:

- (1) The learner will be consistent with the examples
  - Relax consistency assumption.
- (2) Restrict ourselves to finite hypothesis spaces
  - Bound depends on |H| : *a way to measure H's complexity*
  - VC dimension: measure for the complexity of the hypothesis space

Agnostic Learning

- Assume we are trying to learn a concept f using hypotheses in H, **but** f $\notin$ H

- In this case, our goal should be to find a hypothesis h $\in$ H, with a small training error:

$$Err_{TR}(h) = \frac{1}{m} |\{x \in training\_examples; f(x) \neq h(x)\}|$$

Agnostic Learning

- We want a guarantee that a hypothesis with a small training error will have a good accuracy on unseen examples

$$Err_D(h) = \Pr_{x \in D}[f(x) \neq h(x)]$$

- **Hoeffding** bounds characterize the deviation between the *true probability* of some event and its *observed frequency* over m independent trials.

e.g., *p is the underlying probability of a binary variable (e.g., toss=Head) being 1*

$$\mathbf{Pr}[p > \hat{p} + \varepsilon] < e^{-2m\varepsilon^2}$$

Agnostic Learning

- Therefore, the probability that an element in H will have training error which is off by more than ε can be bounded as follows:

$$\mathbf{Pr}[Err_D(h) > Err_{TR}(h) + \varepsilon] < e^{-2m\varepsilon^2}$$

- Applying the union bound principle as before, with

$$\delta = |H| \exp\{-2m\varepsilon^2\}$$

- We get a generalization bound – a bound on how much will the true error deviate from the observed error.

For any distribution D generating training and test instance, with probability at least 1-δ over the choice of the training set of size m, (drawn IID), for all h∈H :

$$Error_D(h) < Error_{TR}(h) + \sqrt{\frac{\log|H| + \log(1/\delta)}{2m}}$$

Agnostic Learning

An ***agnostic*** learner which makes *no commitment to whether f is in H* and returns the hypothesis with least training error over at least the following number of examples can guarantee with probability at least (1-**δ**) that its training error is not off by more than **ε** from the true error.

$$m > \frac{1}{2\varepsilon^2}\{\ln(|H|) + \ln(1/\delta)\}$$

Key Idea: Learnability depends on the log of the size of the hypothesis space

Infinite Hypothesis Space

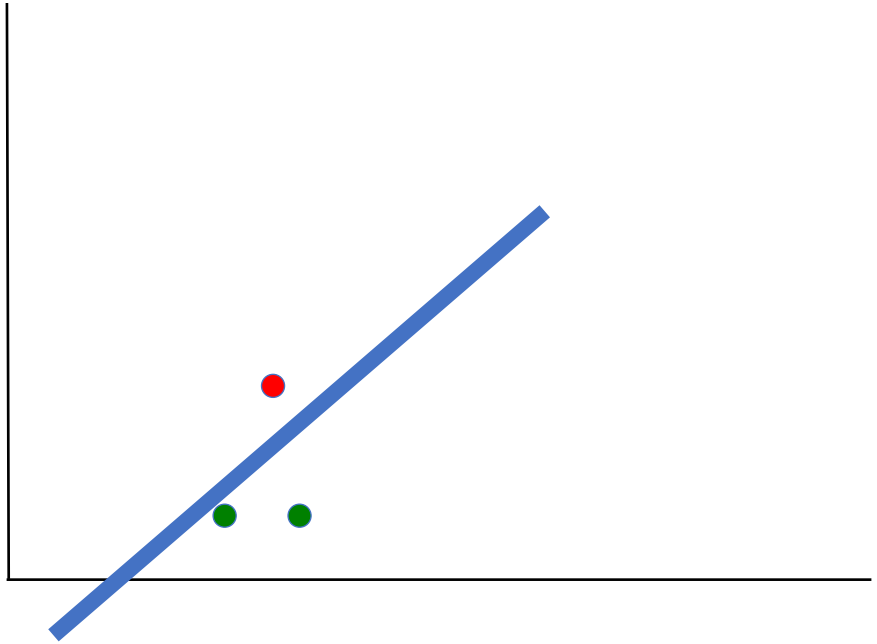
- Our analysis so far focused on *finite hypothesis spaces*
  - The size of H was a way to capture H‘s expressiveness
- Some infinite hypothesis spaces are more expressive than others
  - E.g., Rectangles, vs. 17- sides convex, Linear threshold function vs. a *conjunction* of linear threshold functions

VC Dimension: Intuition

Consider two hypothesis spaces defined over 2 variables:

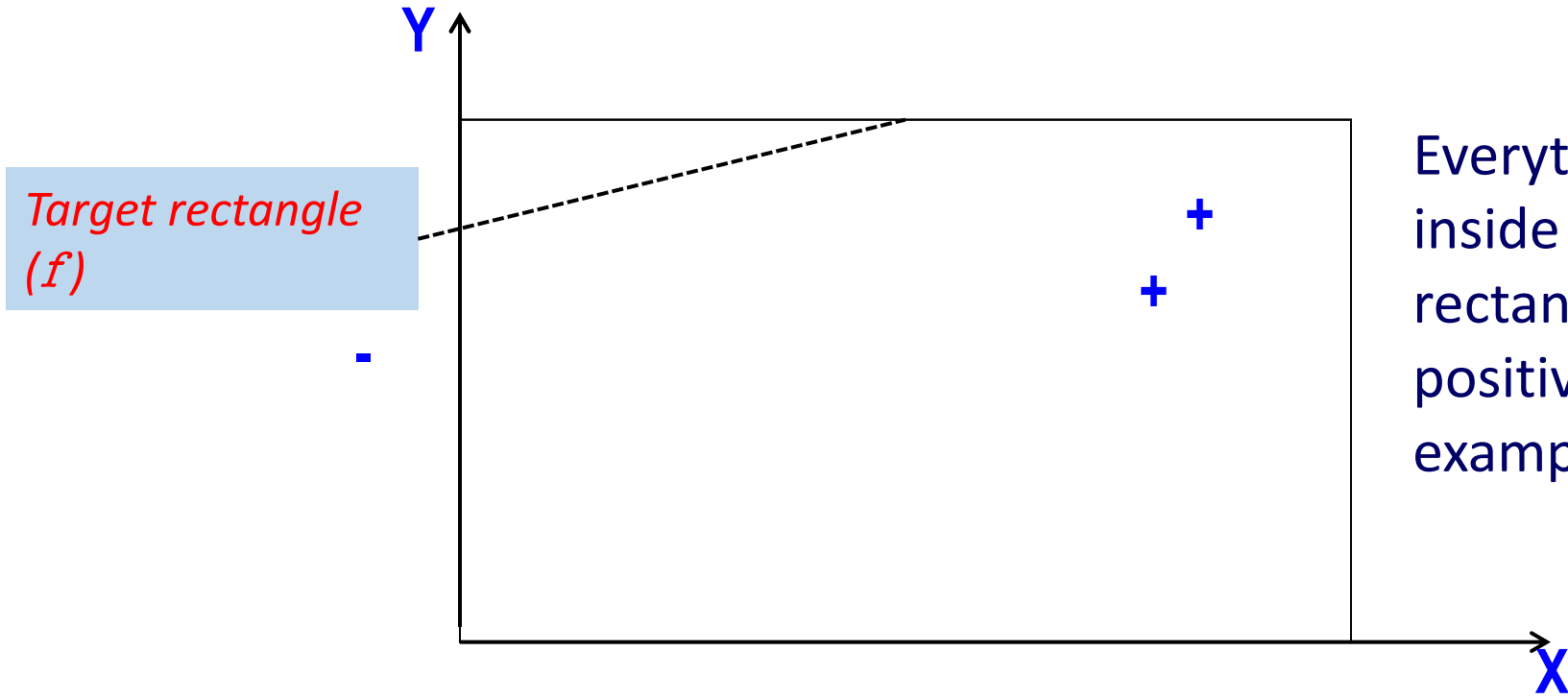
1. Y-Axis parallel lines
2. Lines

- Both spaces are infinite
- Which one is more complex? Why?



Learning Rectangles

- Assume the target concept is an axis parallel rectangle



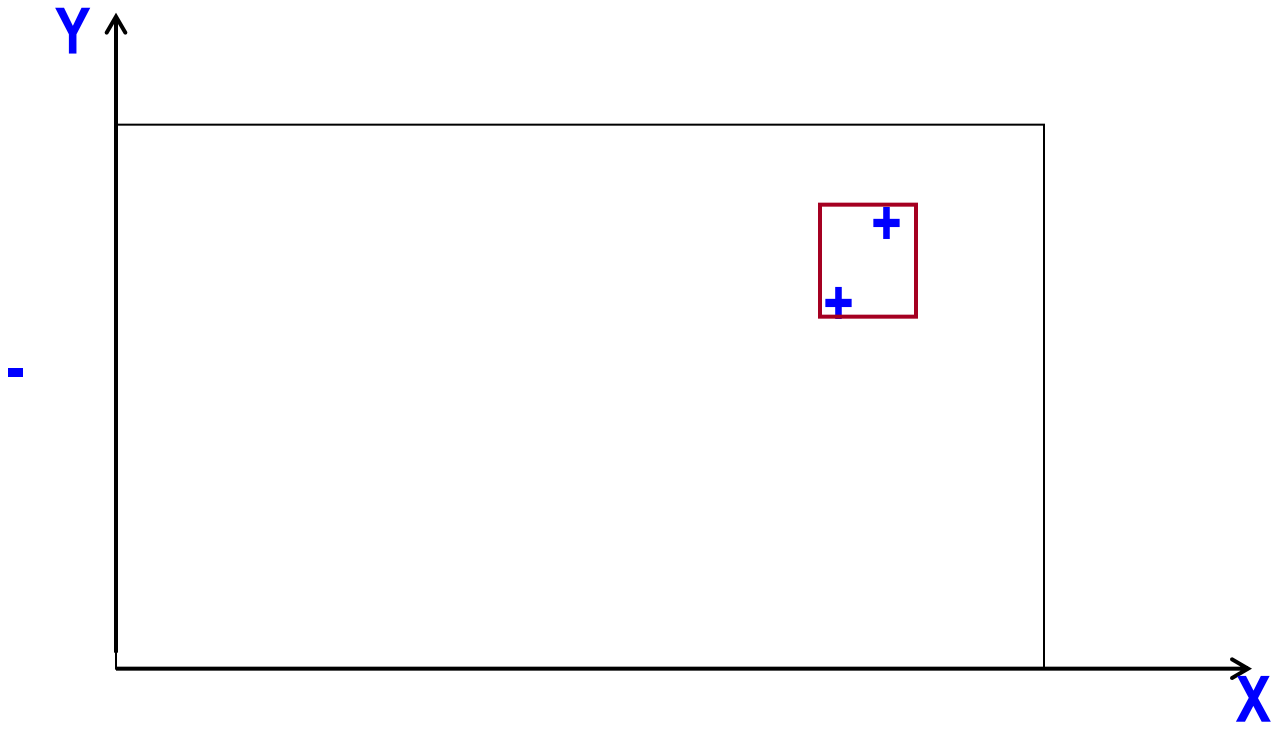
Everything inside the rectangle is a positive example

How can you learn using this hypothesis class?

➔ What are the parameters we need to learn?

Learning Rectangles

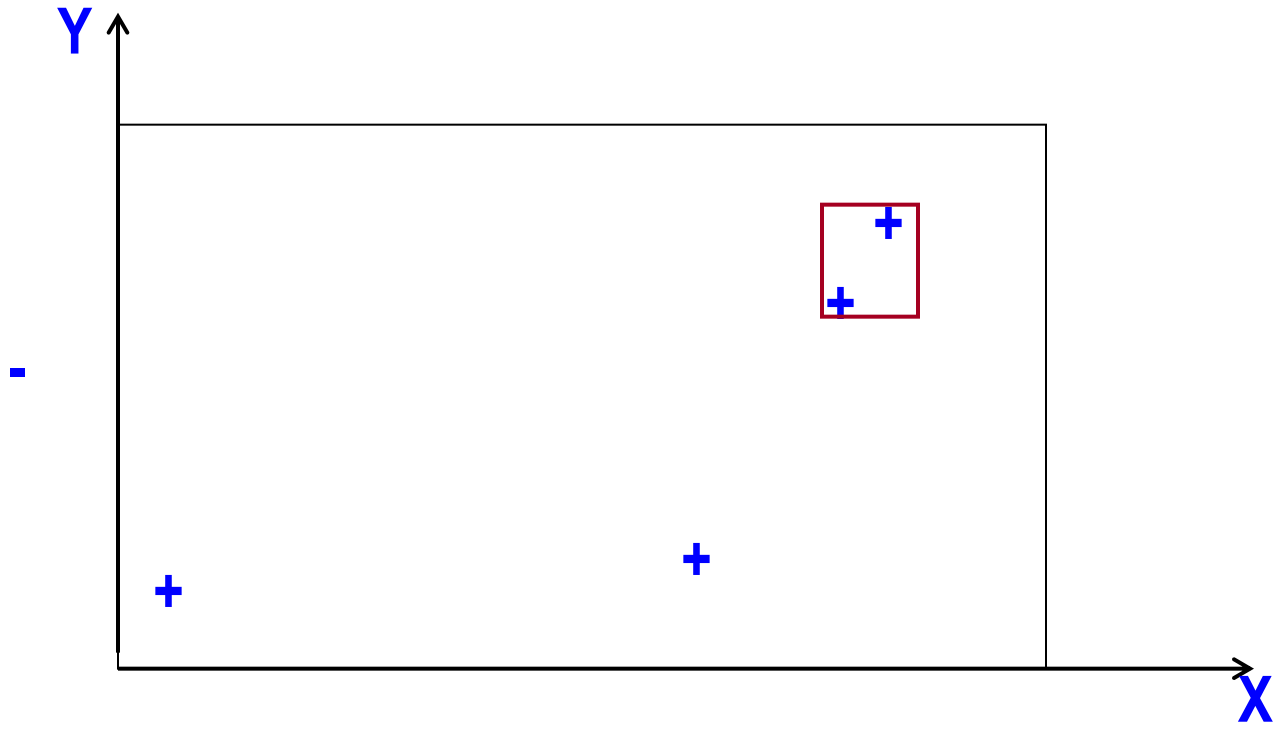
- Assume the target concept is an axis parallel rectangle



Four parameters determining an interval on each one of the axis (min x, max x, min y, max y)

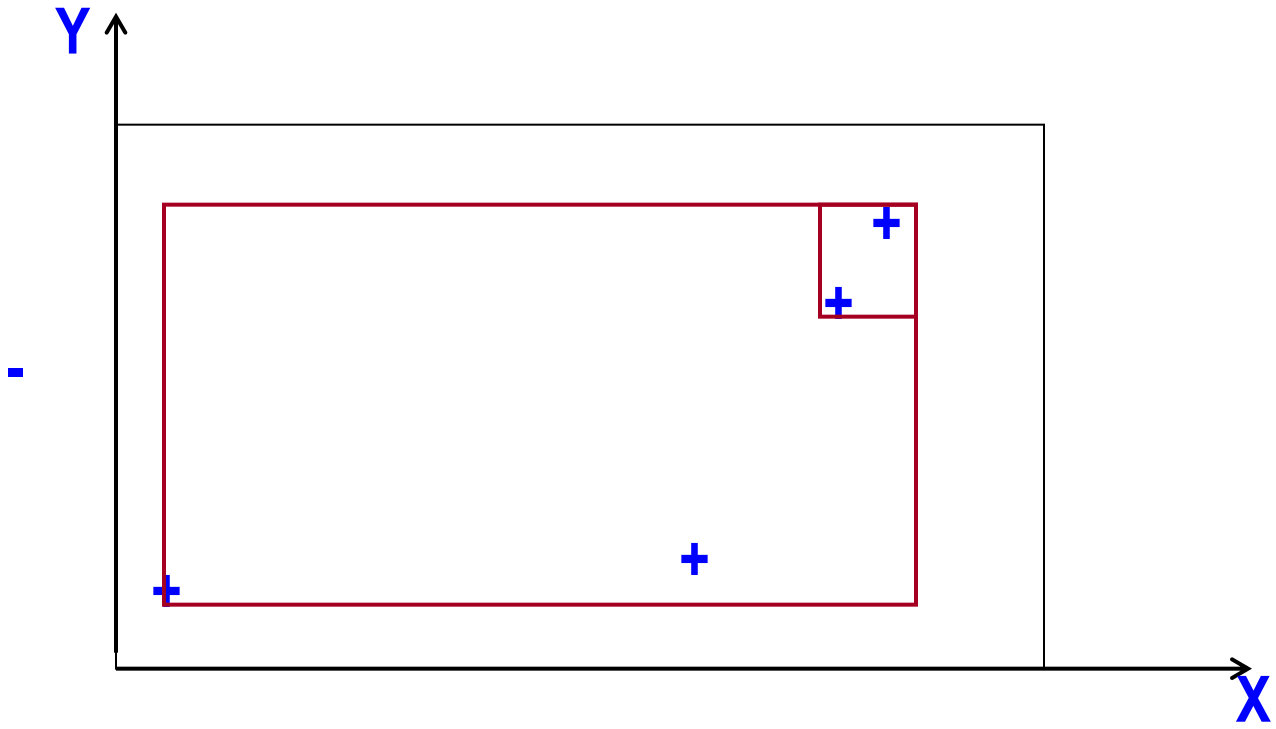
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



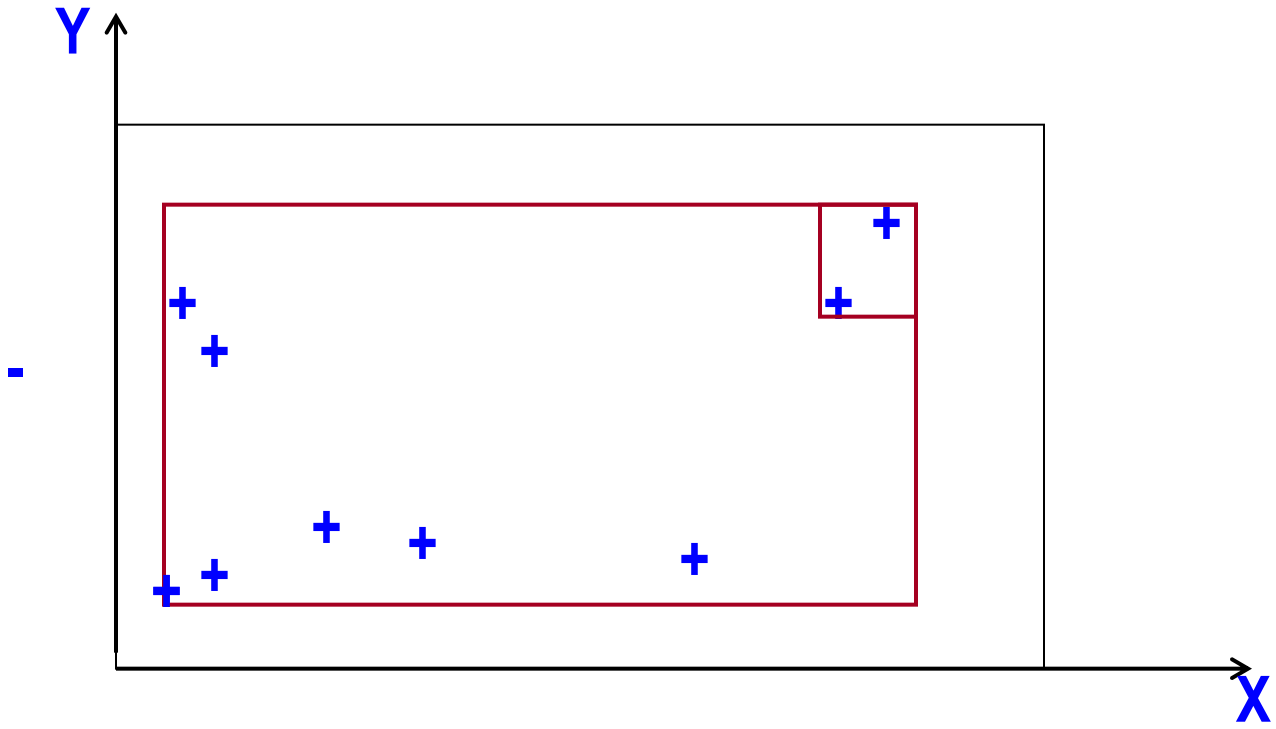
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



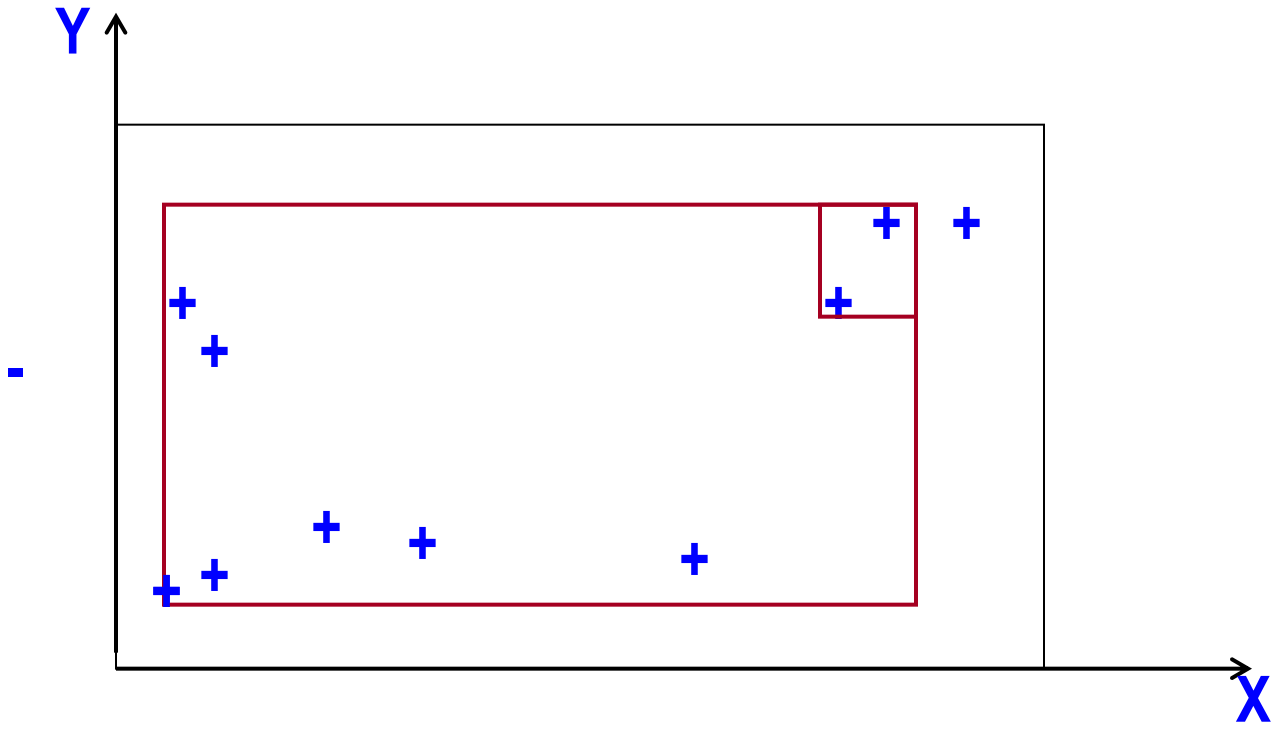
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



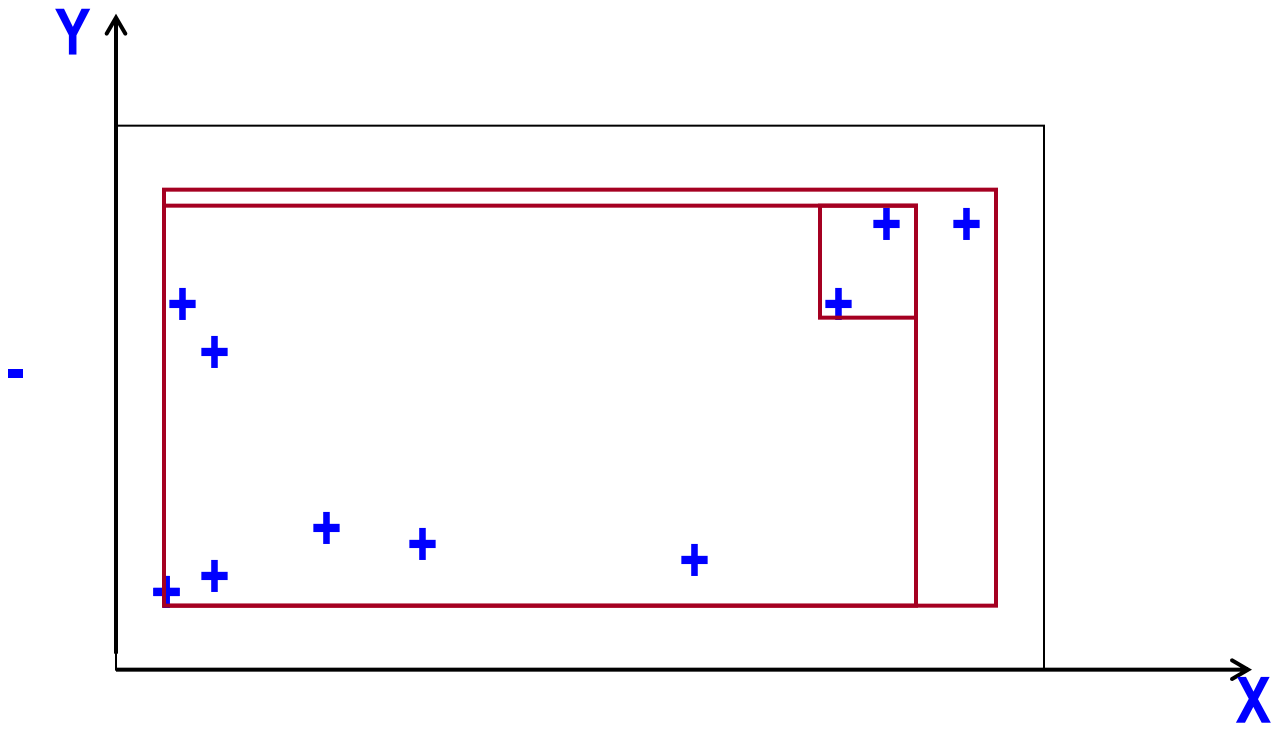
Learning Rectangles

- 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 ?

Learning Rectangles

- Assume the target concept is an axis parallel rectangle

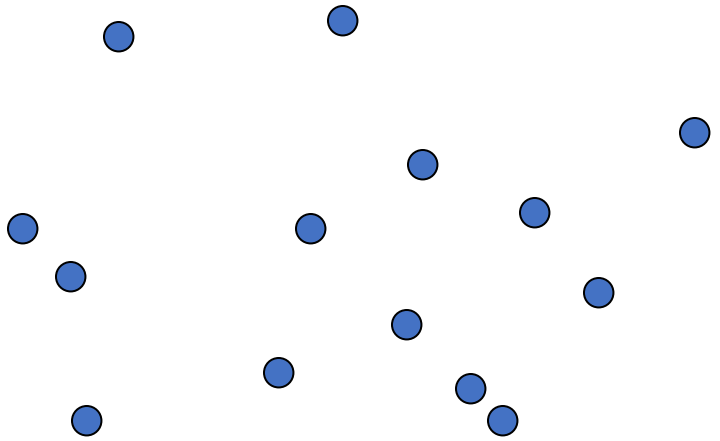
This is an infinite hypothesis space. Can we still make any claims?

- Will we be able to learn the target rectangle ?

Infinite Hypothesis Space

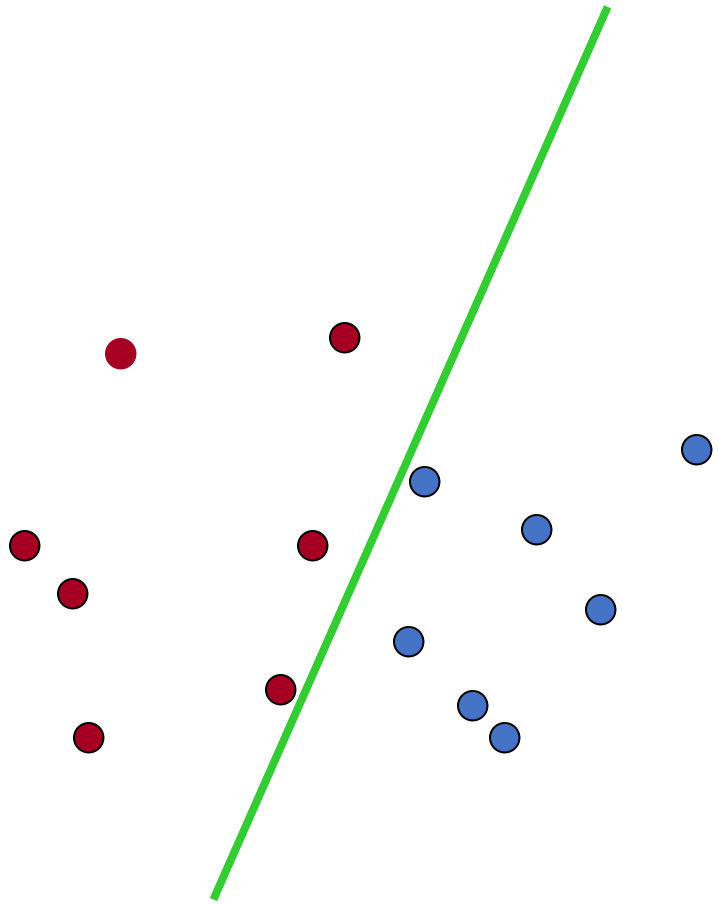
- Our analysis so far focused on *finite hypothesis spaces*
  - The size of H was a way to capture H's expressiveness
- Some infinite hypothesis spaces are more expressive than others
  - E.g., Rectangles, vs. 17- sides convex, Linear threshold function vs. a *conjunction* of linear threshold functions
- Need a measure of the *expressiveness* of an infinite hypothesis space other than its *size*
- The Vapnik-Chervonenkis (VC) dimension
  - Analogous to |H| bounds for sample complexity use VC(H)

Shattering



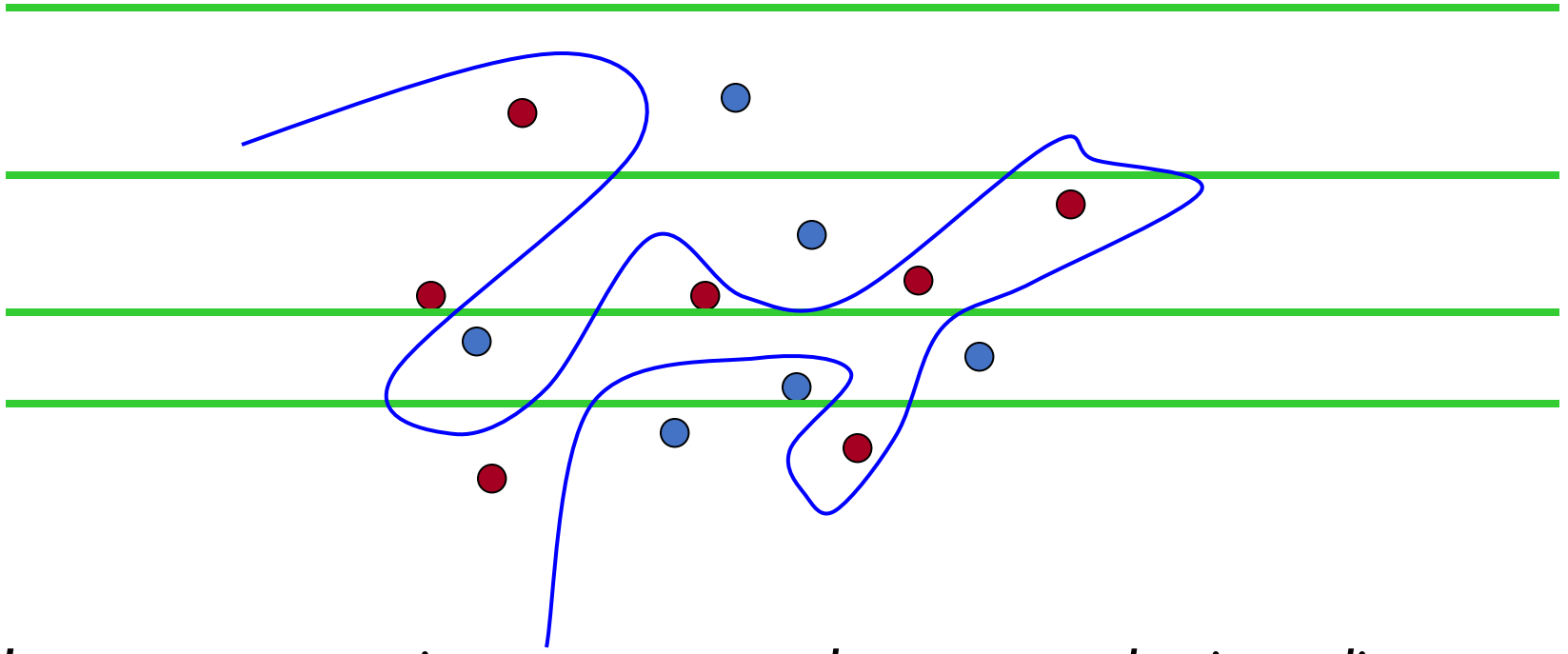
Given a set of examples (points), there are many ways to assign labels

Shattering



For example, this label assignment can be separated using a linear function

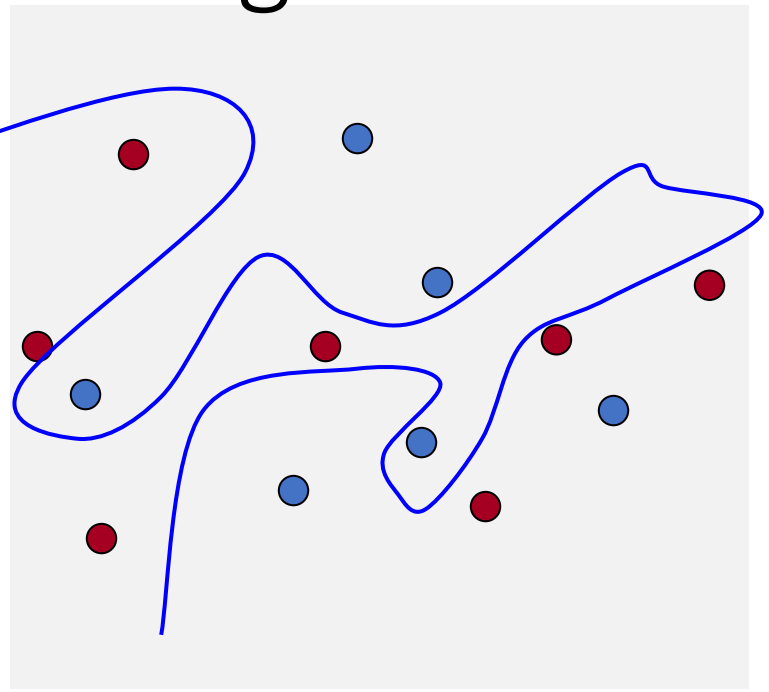
Shattering

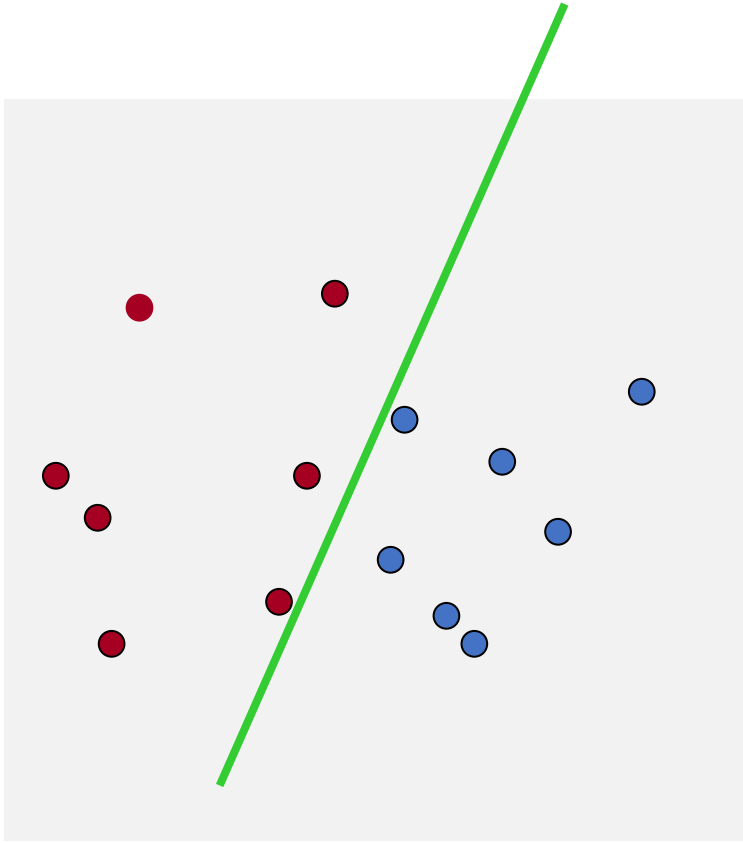


However some assignments cannot be separated using a linear function.

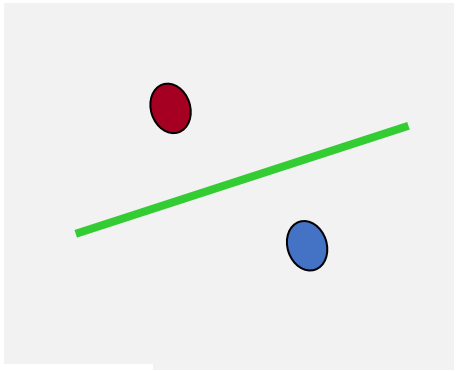
➔*You will need a more complex function to separate this assignment*

Shattering





Let's consider two points – can we separate them with a linear function? For any label assignment?



Shattering

We say that 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 richer set of functions can shatter larger sets of points*

Determining the complexity of H:

Show that there exists a set of points S of size K that functions in H are able to shatter

OR

Show that NO such set of points exists – all set of points of size K cannot be shattered by functions in H

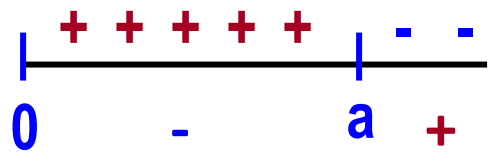
Shattering

Left bounded intervals on the real axis: [0,a), for a real number a>0

Everything inside the interval is positive

Shattering

Left bounded intervals on the real axis: [0,a), for a real number a>0

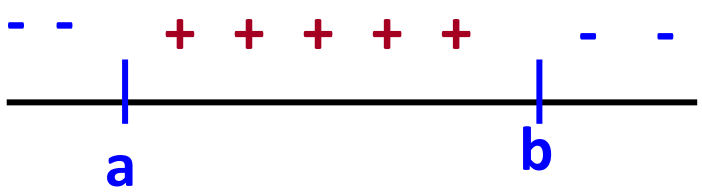


Sets of two points cannot be shattered.

I.e., *given two points, you can label them in such a way that no concept in this class will be consistent with their labeling*

Shattering

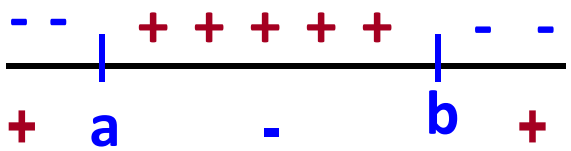
Intervals on the real axis: [a,b], for some real numbers b>a



Everything inside the interval is positive

Shattering

Intervals on the real axis: [a,b], for some real numbers b>a

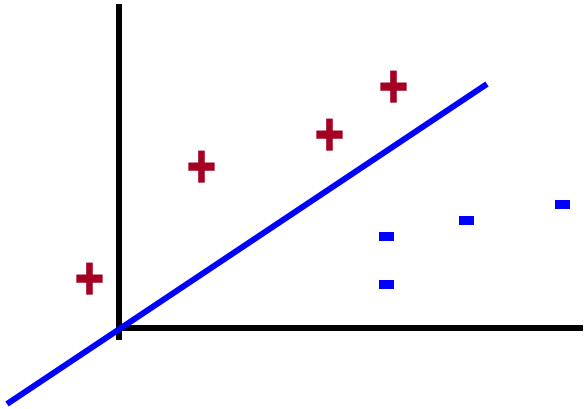


All sets of one or two points can be shattered
but sets of **three** points cannot be shattered

Shattering

We say that 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

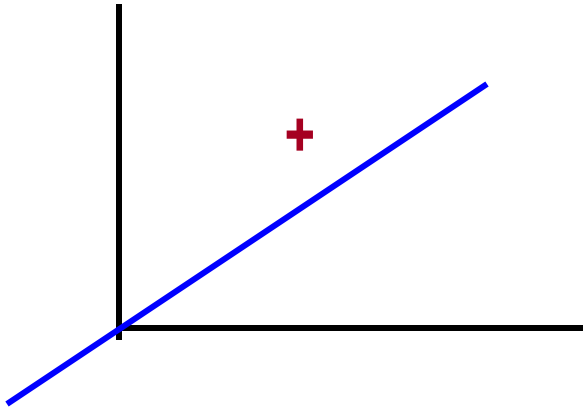
Half-spaces in the plane:



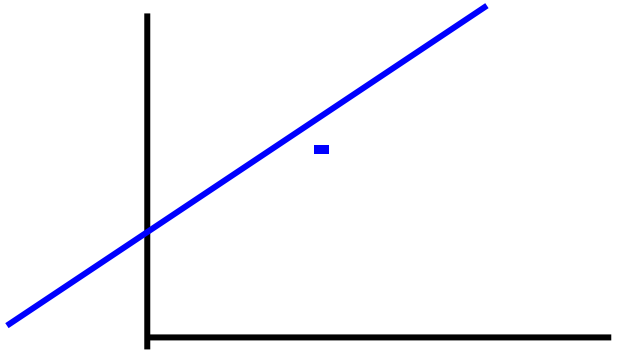
Shattering

We say that 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

Half-spaces in the plane:



Can a set of one point be shattered?

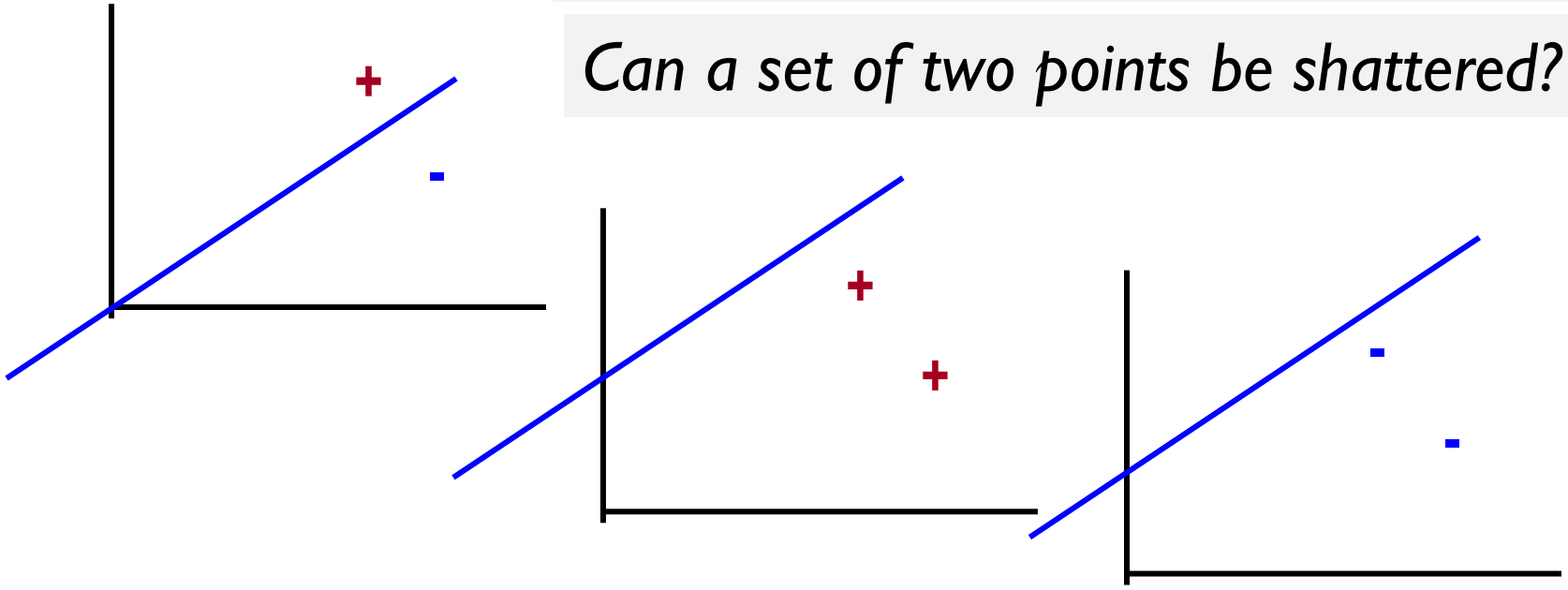


Shattering

Half-spaces in the plane:

Can a set of one point be shattered?

Can a set of two points be shattered?



Shattering

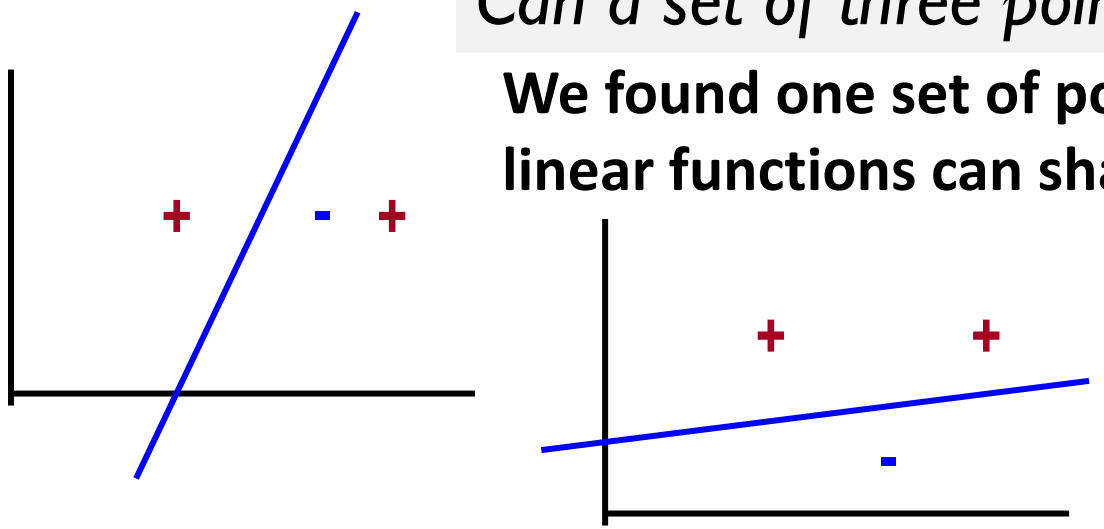
Half-spaces in the plane:

Can a set of one point be shattered?

Can a set of two points be shattered?

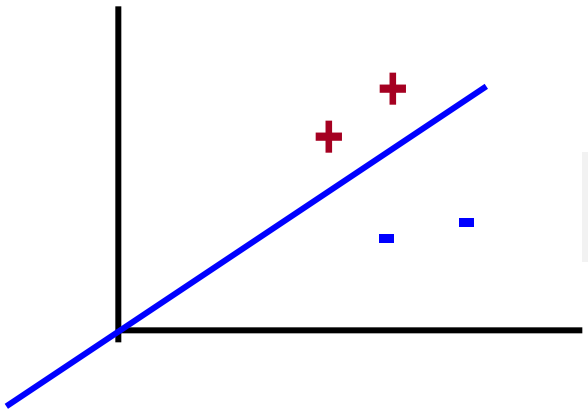
Can a set of three points be shattered?

We found one set of points of size 3 that linear functions can shatter



Shattering

Half-spaces in the plane:



Can a set of one point be shattered?

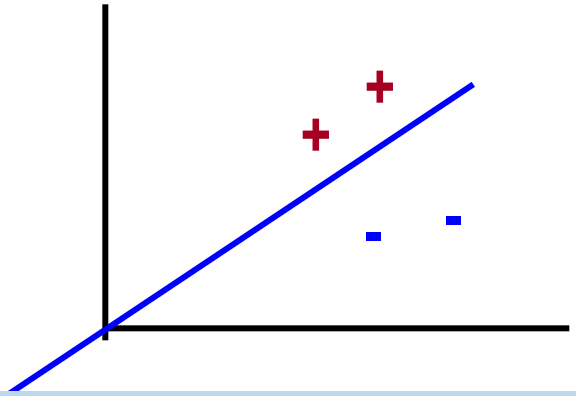
Can a set of two points be shattered?

Can a set of three points be shattered?

Can a set of four points be shattered?

Shattering

Half-spaces in the plane:



Can a set of one point be shattered?

Can a set of two points be shattered?

Can a set of three points be shattered?

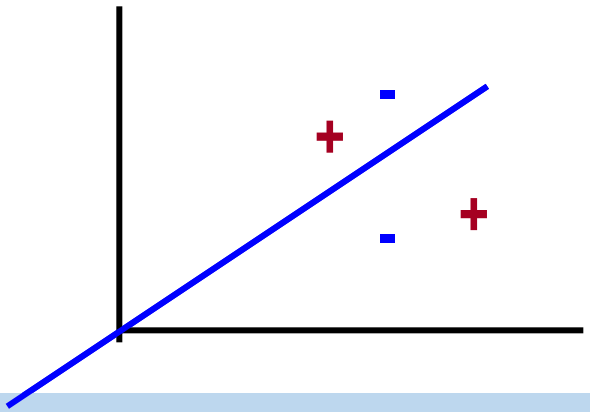
Can a set of four points be shattered?

How can you show that a set of 4 points cannot be shattered?

1. If the 4 points form a convex polygon
2. If one point is in the convex hull defined by the other three

Shattering

Half-spaces in the plane:



Can a set of one point be shattered?

Can a set of two points be shattered?

Can a set of three points be shattered?

Can a set of four points be shattered?

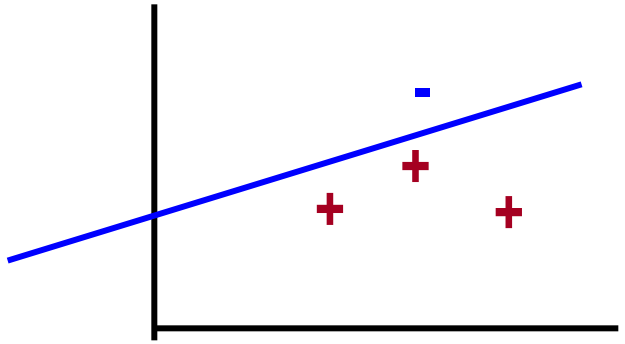
How can you show that a set of 4 points cannot be shattered?

1. If the 4 points form a convex polygon

2. If one point is in the convex hull defined by the other three

Shattering

Half-spaces in the plane:



Can a set of one point be shattered?

Can a set of two points be shattered?

Can a set of three points be shattered?

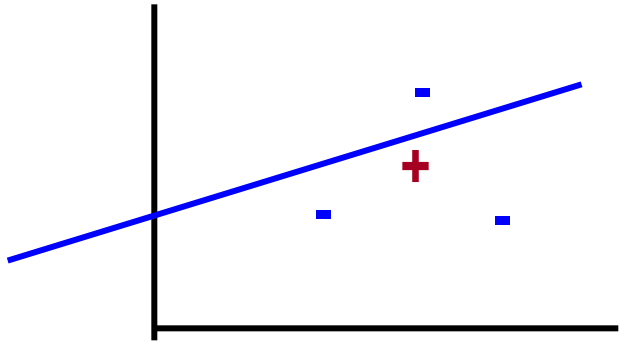
Can a set of four points be shattered?

How can you show that a set of 4 points cannot be shattered?

2. If one point is in the convex hull defined by the other three

Shattering

Half-spaces in the plane:



Can a set of one point be shattered?

Can a set of two points be shattered?

Can a set of three points be shattered?

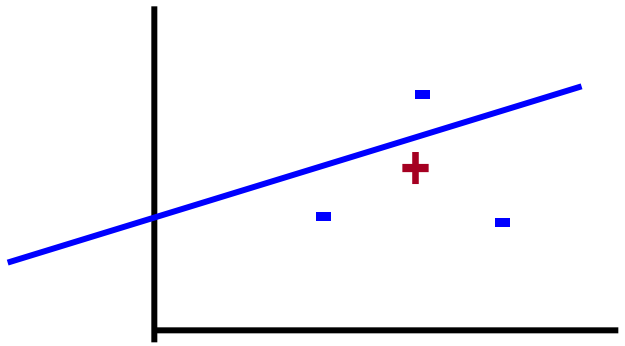
Can a set of four points be shattered?

How can you show that a set of 4 points cannot be shattered?

2. If one point is in the convex hull defined by the other three

Shattering

Half-spaces in the plane:



Conclusion for Half-spaces:

sets of one, two or three points can be shattered **but** there is no set of *four* points that can be shattered

VC 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 assigns similar labels.

- 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.
  - ***Even if only one subset of this size does it***

VC Dimension

- *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.*
  - ***Even if only one subset of this size does it***

- If there exists a subset of size d that can be shattered, then VC(H) >=d, If **no** subset of size d can be shattered, then VC(H) < d

- VC(Half intervals) = 1 (*no subset of size 2 can be shattered*)
- VC(Intervals) = 2 (*no subset of size 3 can be shattered*)
- VC(Half-spaces in the plane) = 3 (*no subset of size 4 can be shattered*)

Sample Complexity & VC Dimension

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

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

Sample Complexity & VC Dimension

- Given a sample D of m examples, find some h∈H that is consistent with all m examples

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

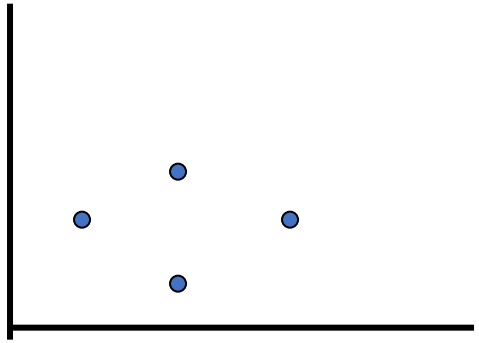
- Then, with prob. at least (1-δ), h has error less than ε.
- if m is polynomial we have a PAC learning algorithm
- We also need to produce the hypothesis h *efficiently*.
- What if |H| is finite?

➔**Note that to shatter m examples-**

$|H|>2^m$, so $\log(|H|)>=VC(H)$

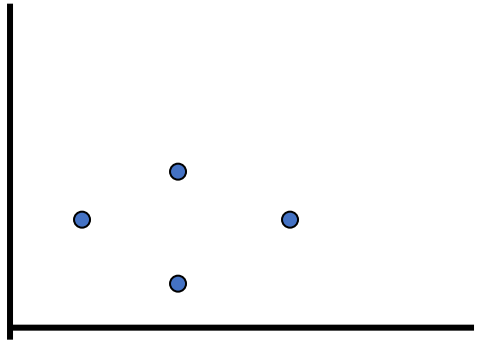
Learning Axis Parallel Rectangles

- Consider axis parallel rectangles
- Can we PAC learn it ?
  1. **What is the VC dimension ?**



Learning Axis Parallel Rectangles

- Consider axis parallel rectangles
- Can we PAC learn it ?
  1. **What is the VC dimension ?**



Can a set of ***one*** point be shattered?

Can a set of ***two*** points be shattered?

Can a set of ***three*** points be shattered?

Can a set of ***four*** points be shattered?

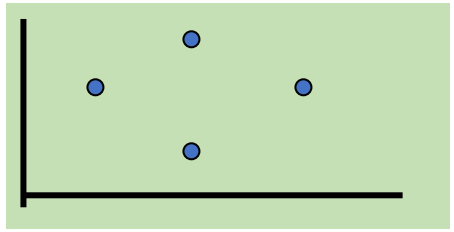
Can a set of ***five*** points be shattered?

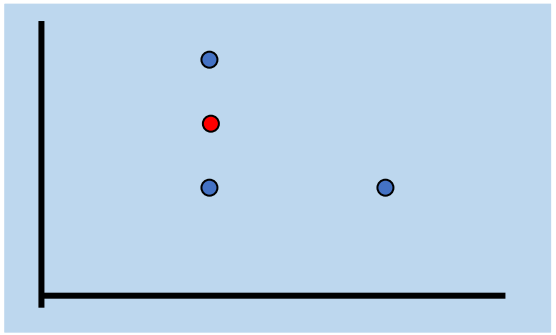
Can a set of ***six*** points be shattered?

...

Learning Axis Parallel Rectangles

- Consider axis parallel rectangles
- Can we PAC learn it ?
  1. **What is the VC dimension ?**





Can a set of ***one*** point be shattered?

Can a set of ***two*** points be shattered?

Can a set of ***three*** points be shattered?

Can a set of ***four*** points be shattered?

We found one set of points that can be shattered! VC(H) >= 4

Learning Axis Parallel Rectangles

- Consider axis parallel rectangles
- Can we PAC learn it ?
  1. **What is the VC dimension ?**

Can a set of ***one*** point be shattered?

Can a set of ***two*** points be shattered?

Can a set of ***three*** points be shattered?

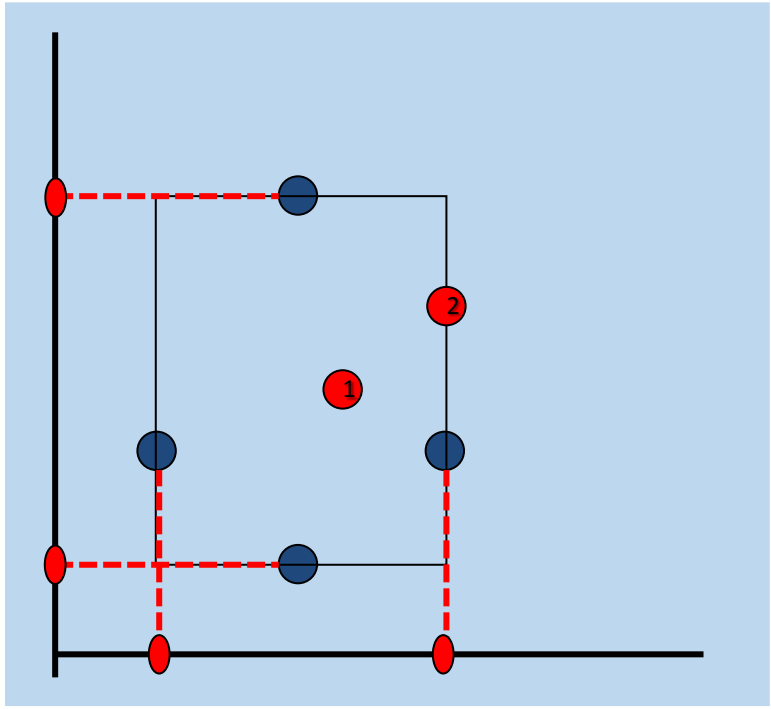
Can a set of ***four*** points be shattered?

Can a set of ***five*** points be shattered?

Learning Axis Parallel Rectangles

- Consider axis parallel rectangles
- Can we PAC learn it ?
  1. **What is the VC dimension ?**

Can a set of ***five*** points be shattered?

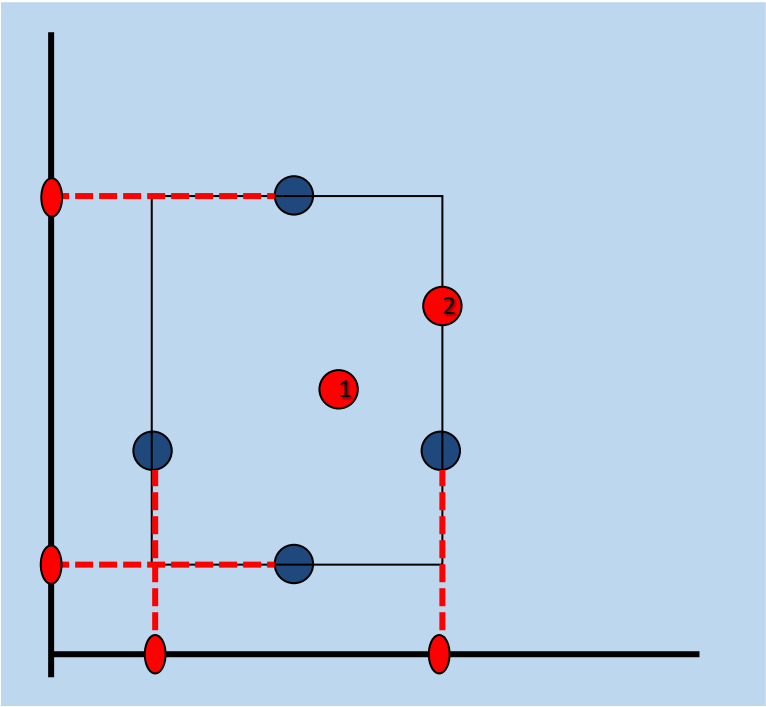


Consider the rectangle defined by *the min-x,max-x, min-y, max-y* values of the four extreme points. The fifth point must lie either inside the rectangle or on the edges

➔ For either case it is possible to come up with a label assignment that cannot be represented using an axis parallel rectangle

Learning Axis Parallel Rectangles

- Consider axis parallel rectangles
- Can we PAC learn it ?
  1. **What is the VC dimension ?**



Can a set of ***five*** points be shattered?

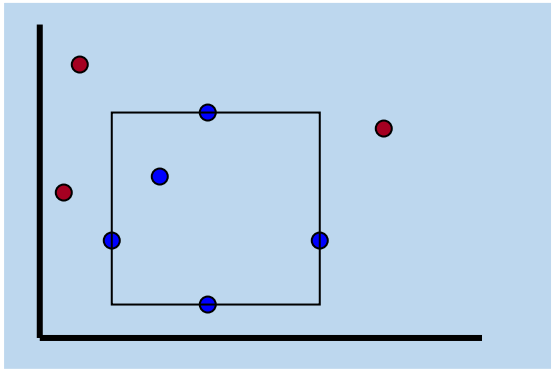
No set of five points can be shattered!

➔ VC(H) = 4

As far as sample complexity, *PAC learnability* is guaranteed.

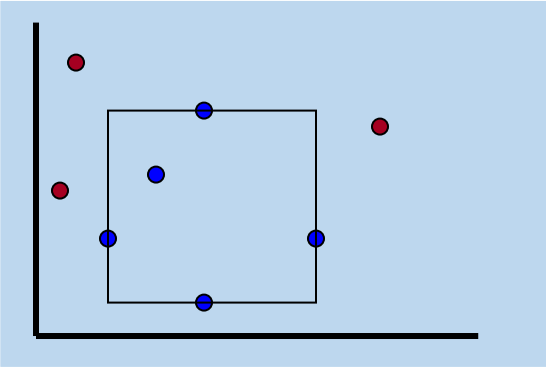
Learning Rectangles

- Consider axis parallel rectangles
- Can we PAC learn it ?
  1. What is the VC dimension ?
  2. **Can we give an efficient algorithm?**



Learning Rectangles

- Consider axis parallel rectangles
- Can we PAC learn it ?
  1. What is the VC dimension ?
  2. **Can we give an efficient algorithm?**



Find the smallest rectangle that contains the positive examples (necessarily, it will not contain any negative example, if the hypothesis is consistent.

➔Axis parallel rectangles are efficiently PAC learnable.

COLT conclusions

- The PAC framework provides a reasonable model for theoretically analyzing the effectiveness of learning algorithms.
  - No Distributional Assumption
  - Training Distribution is the same as the Test Distribution
  - **IID Assumption is the basis for theoretical support**

- The sample complexity for a consistent learner over H, can be determined from a measure of H's expressiveness (|H|, VC(H))
- Sample complexity bounds given here are far from being tight, but separates learnable classes from non-learnable classes

And now to something completely different

…. Ensemble methods

Ensemble Learning Algorithms

- **Simple idea**: *combine the predictions of multiple classifiers*
- *Don't repeat the same classification multiple times*
  - *combine different classifiers, train over different datasets, use different feature representation.*
- **Intuition**: *"wisdom of the crowds" principle*
  - *Many non-experts can perform better*

Ensemble Learning Algorithms

- **Simple idea**: *combine the predictions of multiple classifiers*
- *Many ways to combine the classifiers:*
  - *Majority vote, weighted vote, confidence-based,..*
- *For example:*

$$y_1 = f_1(x, w)$$
$$y_2 = f_1(x, w) \quad \Rightarrow \quad y_{final} = Sign\left(\sum_i \alpha_i y_i\right)$$
$$y_3 = f_1(x, w)$$
$$\ldots$$

Where have we seen it before?

Ensemble Learning Algorithms

- In this lecture we will talk about two ensembles
  - Boosting
  - Bagging

- Both combine classifiers of the **same type**, multiple times by modifying the training set

- **Big idea**: *The methods will be used to help control the **bias** and **variance** in a different way*

http://cseweb.ucsd.edu/~yfreund/adaboost/

Boosting

- Boosting is a general learning paradigm for putting together a **Strong Learner**, given a collection of Weak Learners.
- The original Boosting Algorithm was proposed as an answer to a theoretical question in PAC learning.
  - [The Strength of Weak Learnability; Schapire, 89]
- Importance: Theoretical and Practical
  - Strong and weak learners in the **PAC framework**
- **Intuition:** reducing bias by increasing the complexity of simple classifiers

Boosting Motivation

Example: "How May I Help You?"

[Gorin et al.]

- goal: automatically categorize type of call requested by phone customer (Collect, CallingCard, PersonToPerson, etc.)
  - `yes I'd like to place a collect call long distance please` (Collect)
  - `operator I need to make a call but I need to bill it to my office` (ThirdNumber)
  - `yes I'd like to place a call on my master card please` (CallingCard)
  - `I just called a number in sioux city and I musta rang the wrong number because I got the wrong party and I would like to have that taken off of my bill` (BillingCredit)
- observation:
  - easy to find "rules of thumb" that are "often" correct
    - e.g.: "IF '`card`' occurs in utterance THEN predict 'CallingCard' "
  - hard to find single highly accurate prediction rule

Rules-of-thumb are accurate but have a **low** recall

The Boosting Approach

- **Algorithm**
  - Select a small subset of examples
  - Derive a rough rule of thumb
  - Examine 2nd set of examples
  - Derive 2nd rule of thumb
  - Repeat T times
  - Combine the learned rules into a single hypothesis

Note the design decisions made:
- Rules-of-thumb
- Small sub-set of examples

- **Questions**:
  - *How to choose subsets of examples to examine on each round?*
    - ***Emphasize*** *subsets instead of selecting subsets*
  - How to combine all the rules into single prediction rule?

- **Boosting**
  - General method of converting rough rules of thumb into highly accurate prediction rule

Theoretical Motivation

- **"Strong" PAC algorithm *(for class H):***
  - for *any distribution*
  - $\forall\ \varepsilon, \delta > 0$
  - Given polynomially many random examples
  - Finds hypothesis with error $\leq \varepsilon$ with probability $\geq (1-\delta)$

- **"Weak" PAC algorithm**
  - Same, but only for $\varepsilon \geq \frac{1}{2} - \gamma$

Not trivial: for any distribution you can find a hypothesis that performs better than chance (for any training set)

- [Kearns & Valiant '88]:
  - *Does weak learnability imply strong learnability?*
    - *I.e., "can we boost a better-than-chance learner to be as good as we want it to be?"*

History

- [Schapire '89]:
  - **First provable boosting algorithm**
    - Call weak learner three times on three modified distributions
    - Get slight boost in accuracy
    - apply recursively
- [Freund '90]:
  - "Optimal" algorithm that "**boosts by majority**"
- [Drucker, Schapire & Simard '92]:
  - **First practical experiments** using boosting
  - Limited by *practical drawbacks*
- [Freund & Schapire '95]:
  - Introduced "AdaBoost" algorithm
  - Strong practical advantages over previous algorithms
  - AdaBoost was followed by a huge number of papers and practical applications

A Formal View of Boosting

- Given training set $(x_1, x_1), \ldots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ is the correct label of instance $x_i \in X$
- For t = 1, …, T
  - Construct a distribution D_t on {1,…m}
  - Find weak hypothesis ("*rule of thumb*") $h_t : X \to \{-1, +1\}$

with small error ϵ_t on D_t: $\epsilon_t = P_{D_t}[h_t(x_i) \neq y_i]$

Error is measured according to the distribution at step t!

- **Output**: final hypothesis H_{final}

Source: A Tutorial on Boosting Yoav Freund, Rob Schapire

How can we construct the distribution?

Constructing D_t on {1,..,m}:

$$D_1(i) = \frac{1}{m}$$

Initialization : Construct the **uniform distribution**

Given D_t and h_t :

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if} \quad y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if} \quad y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \cdot e^{(-\alpha_t y_i h_t(x_i))}$$

Next : Given the old distribution (D_t), construct the new distribution (D_{t+1}), by **assigning weight to each example**

How can we construct the distribution?

Constructing D_t on {1,..,m}:

$$D_1(i) = \frac{1}{m}$$

The **old distribution** divided by a **normalization factor** (constant)

Given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if} \quad y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if} \quad y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \cdot e^{(-\alpha_t y_i h_t(x_i))}$$

Multiplied by **penalty** term for **correct** classification, and a **promotion** term for **mistakes**

Where:

$Z_t = \text{Normalization constant}$

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

Positive due to the weak learning assumption

Examples that we predicted correctly are demoted, others promoted

How can we construct the distribution?

Constructing D_t on {1,..,m}:

$$D_1(i) = \frac{1}{m}$$

Given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if} \quad y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if} \quad y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \cdot e^{(-\alpha_t y_i h_t(x_i))}$$

Where:

Z_t = Normalization constant

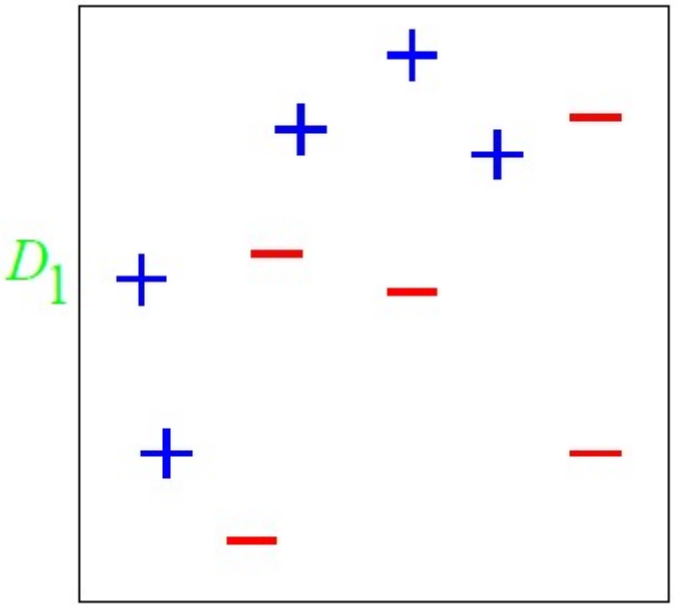
$$\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

Note that the weight of each hypothesis correlates with its error

Final Hypothesis:

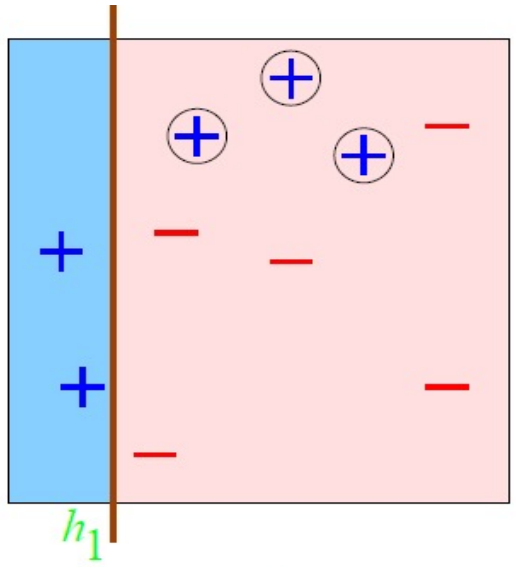
$$H_{\text{final}}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$$

A Toy Example



Weak learner: axis parallel lines

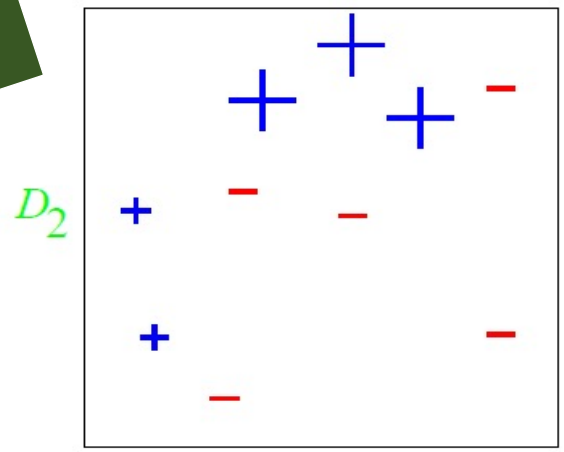
A Toy Example: **Round 1**



Weak learner h_1
Three mistakes under the **uniform** distribution

$$\epsilon_1 = 0.3$$
$$\alpha_1 = 0.42$$

Generate a **new probability distribution** over the data

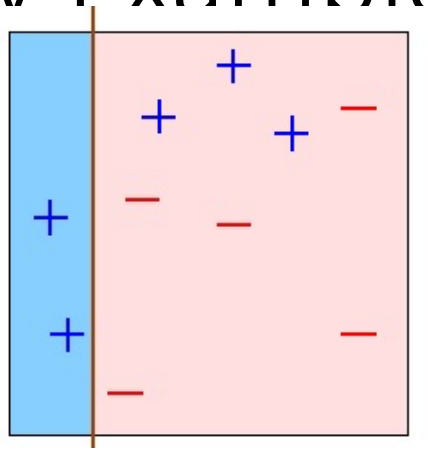


What is the effect of increasing the probability of examples with wrong classification?

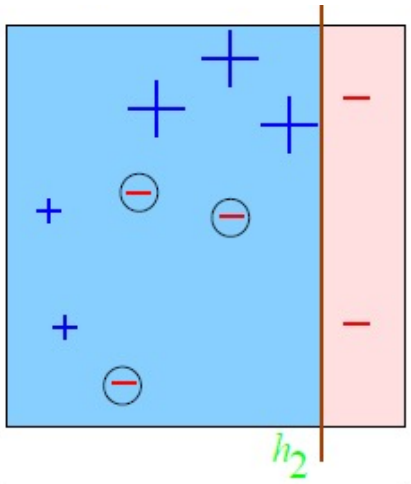
The error of the current classifier will rise over the new distribution

(So what?)

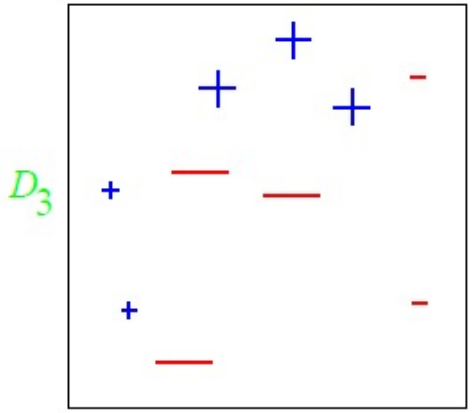
A Toy Example: Round 2



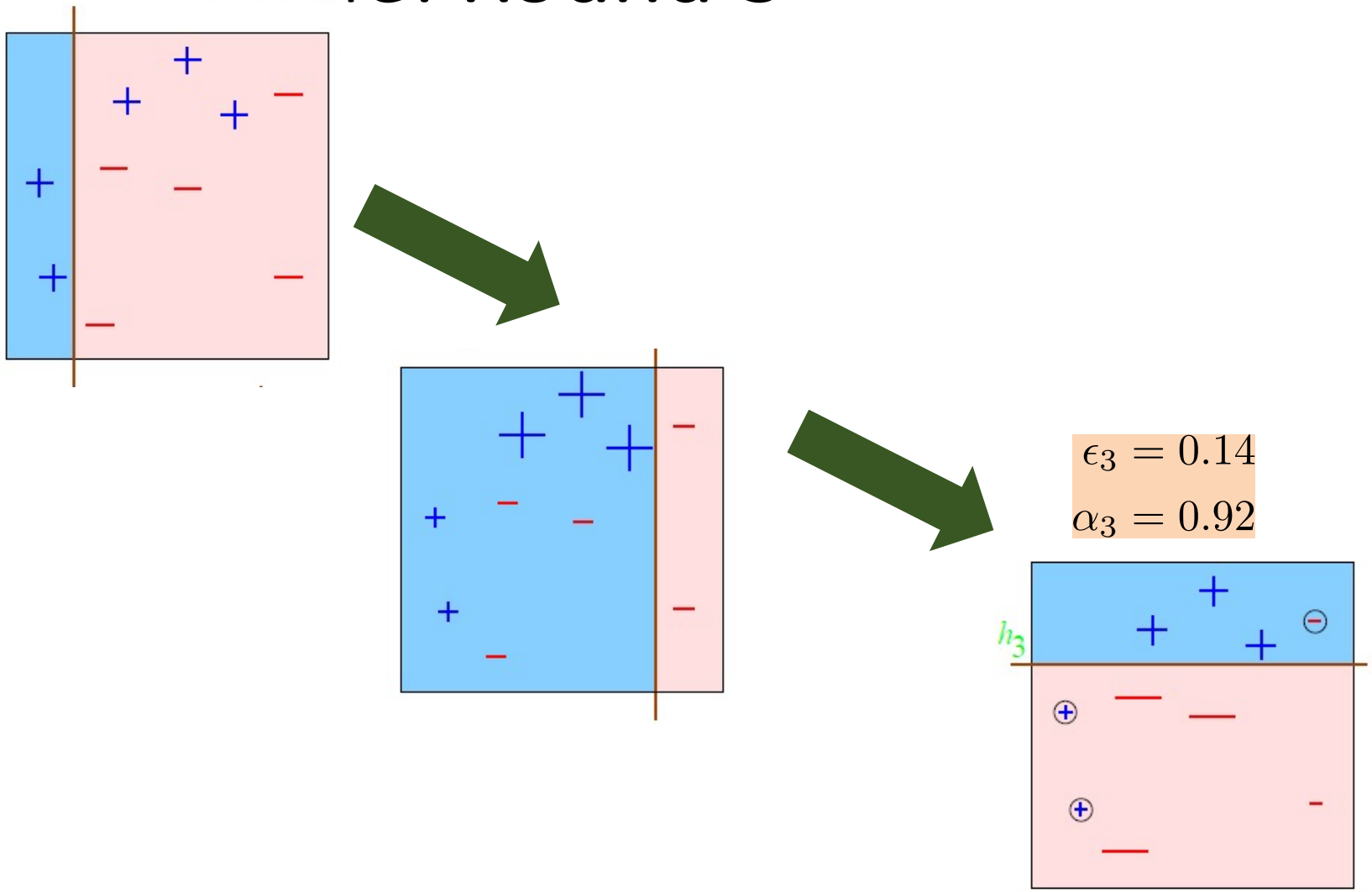
Weak learner h_1
Three mistakes under
the **uniform** distribution



$$\epsilon_2 = 0.21$$
$$\alpha_2 = 0.65$$

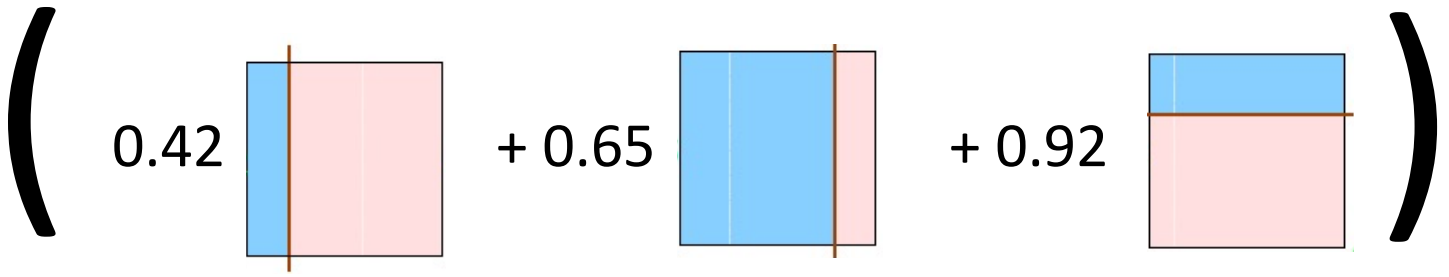


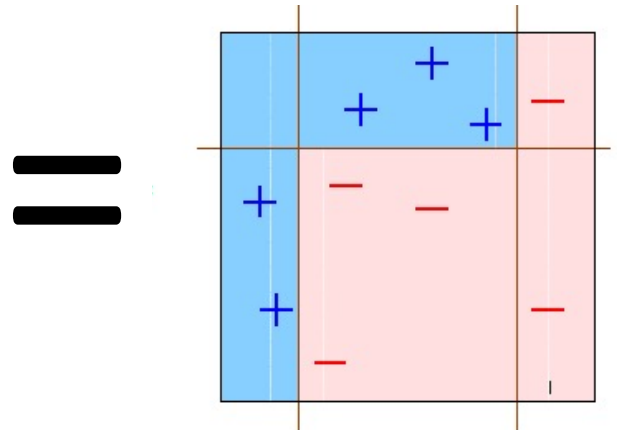
A Toy Example: Round 3



A Toy Example: Final Hypothesis

$$H_{Final} =$$





Note: It is possible that the combined hypothesis makes no mistakes on the training data, but boosting can still learn, by adding more weak hypotheses.

Analyzing Adaboost

Theorem:

- Run AdaBoost for T rounds
- Let $\epsilon_t = ½ - \gamma_t$
- Let $0 < \gamma \leq \gamma_t$ for all t
- Then: *Training error* (H_{final}) $\leq \prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right]$

$$= \prod_t \sqrt{1-4\gamma_t^2}$$

$$\leq e^{(-2\sum_t \gamma^2)}$$

Do we care about the training error?

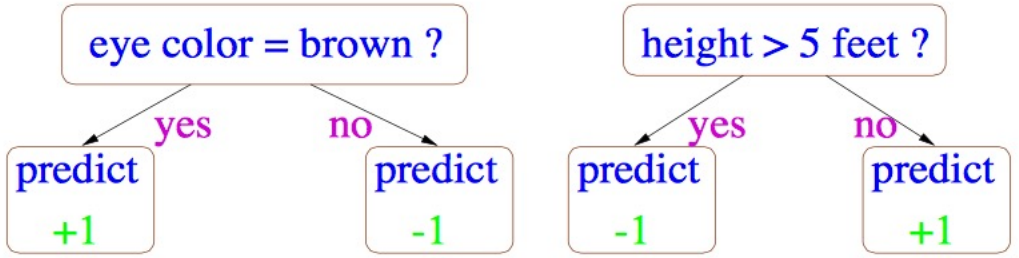
As T (number of iterations) increases, the training error drop exponentially!

AdaBoost in practice

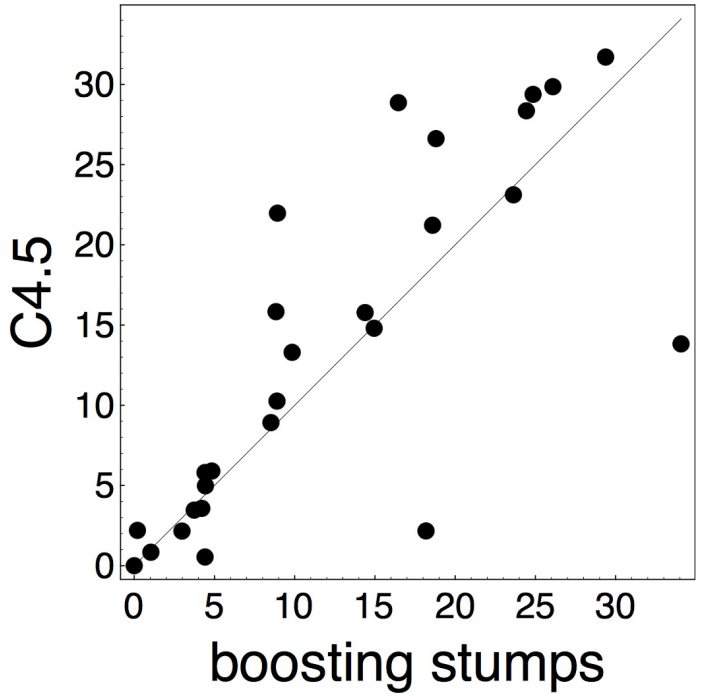
- **Initialization**:
  - *Weigh all training samples equally*
- **Iteration Step:**
  - Train model on (weighted) train set
  - Compute error of model on train set
  - Increase weights on training cases model gets wrong
    - Focus the next classifier on "difficult" examples
- *Slow convergence*
  - *Typically requires 100's to 1000's of iterations*
- **Return final model:**
  - Carefully weighted prediction of each model

Key idea: change the distribution during training

AdaBoost in practice: Boosting Decision Stumps



Decision Stumps:
decision rule splitting on one attribute

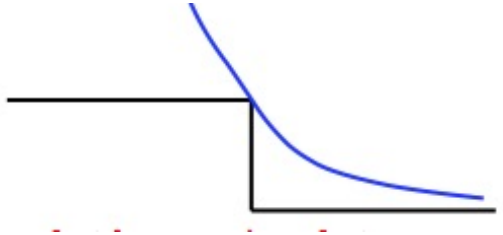


AdaBoost in practice: Loss minimization

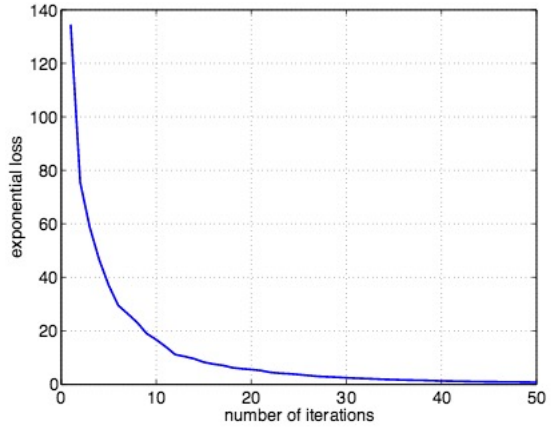
- *AdaBoost corresponds to minimizing the Exponential loss function*

$$C_{ada} = \sum_i \exp[-y^{(i)} f(x^i)]$$

- *Convex and smooth surrogate loss function*

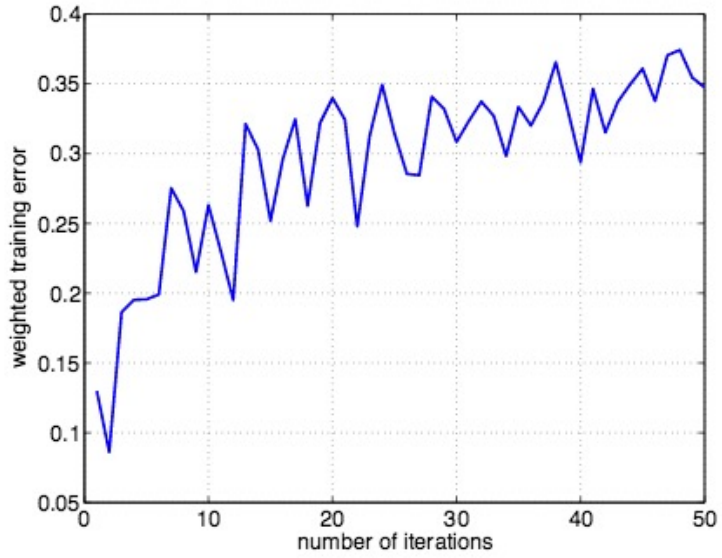


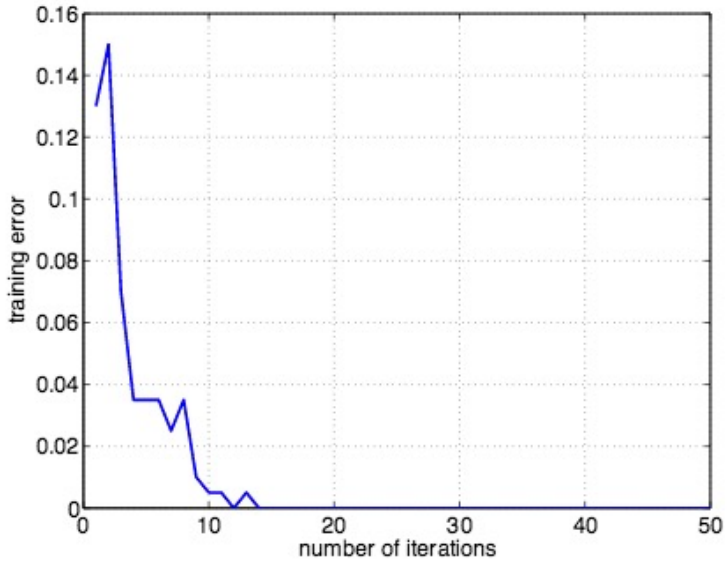
- At each iteration, it will have a lower exponential loss



AdaBoost in practice

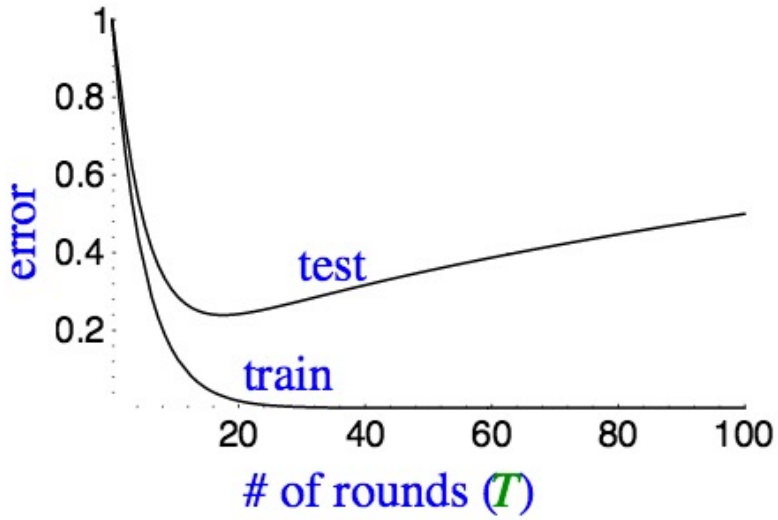
- *Note the difference between the **individual classifiers** (that tend to get worse over time), and the **combined hypothesis** (that improves over time)*





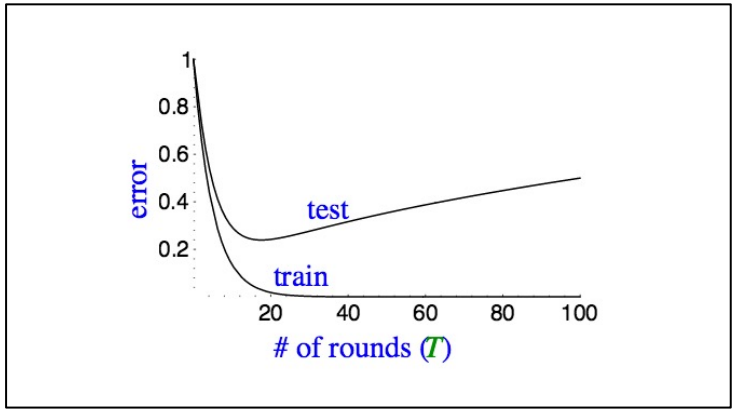
AdaBoost at Test time

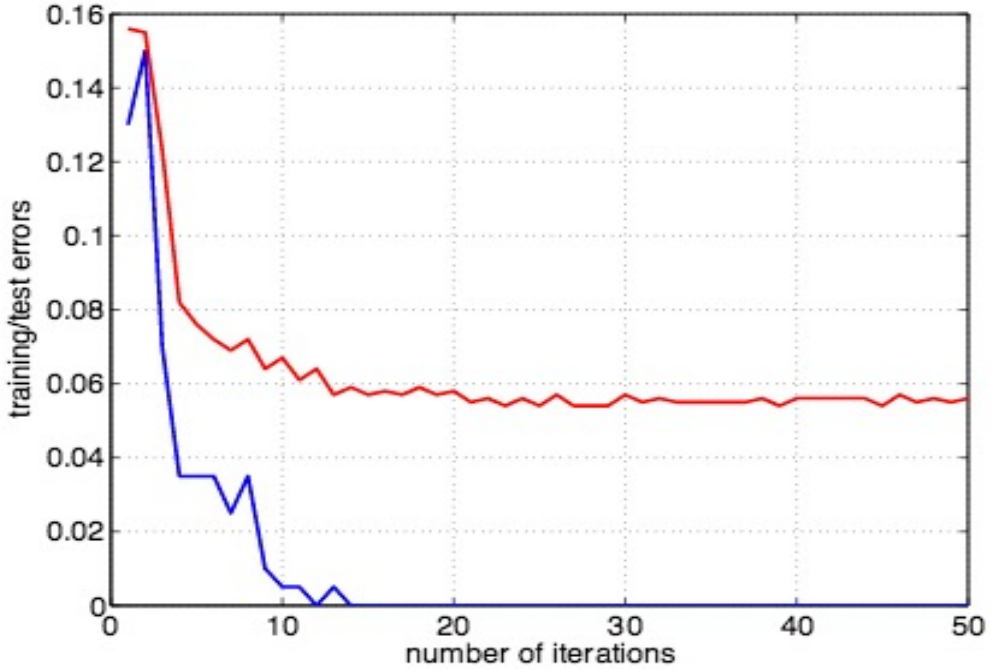
- The training error will reach zero (or keep decreasing) as we make the hypothesis more complex
- As a result, the test error will start increasing as the final hypothesis becomes overly complex



AdaBoost in practice

Interestingly, it does not happen!





Two observations:

(1) Adaboost does not over fit easily

(2)The *test error* can continue to decrease even when we already have *zero training error*

Boosting Summary

- Combine weak learners into a powerful learner
- Reduce bias without increasing variance!
- **Strong points**:
  - Effective and easy to implement
  - You can plug in your favorite learner
  - Only hyperparameter is T
- **Caveats**
  - Will overfit if the weak learners are too complex
  - Will underfit if the weak learners are too weak
    - $\gamma_t \rightarrow 0$ too quickly

Bagging

- **Bagging Intuition:**
  - Overfitting occurs when the model start memorizing the data (no generalization)
  - Variations in the data will results in different models
  - Bagging: **bootstrapped Aggregation**
    - Sample smaller sets of the data, each specific learner cannot memorize the entire dataset
  - Appropriate for: *Low bias-High Variance learners* (e.g., expressive learners)
    - Helps reduce variance!

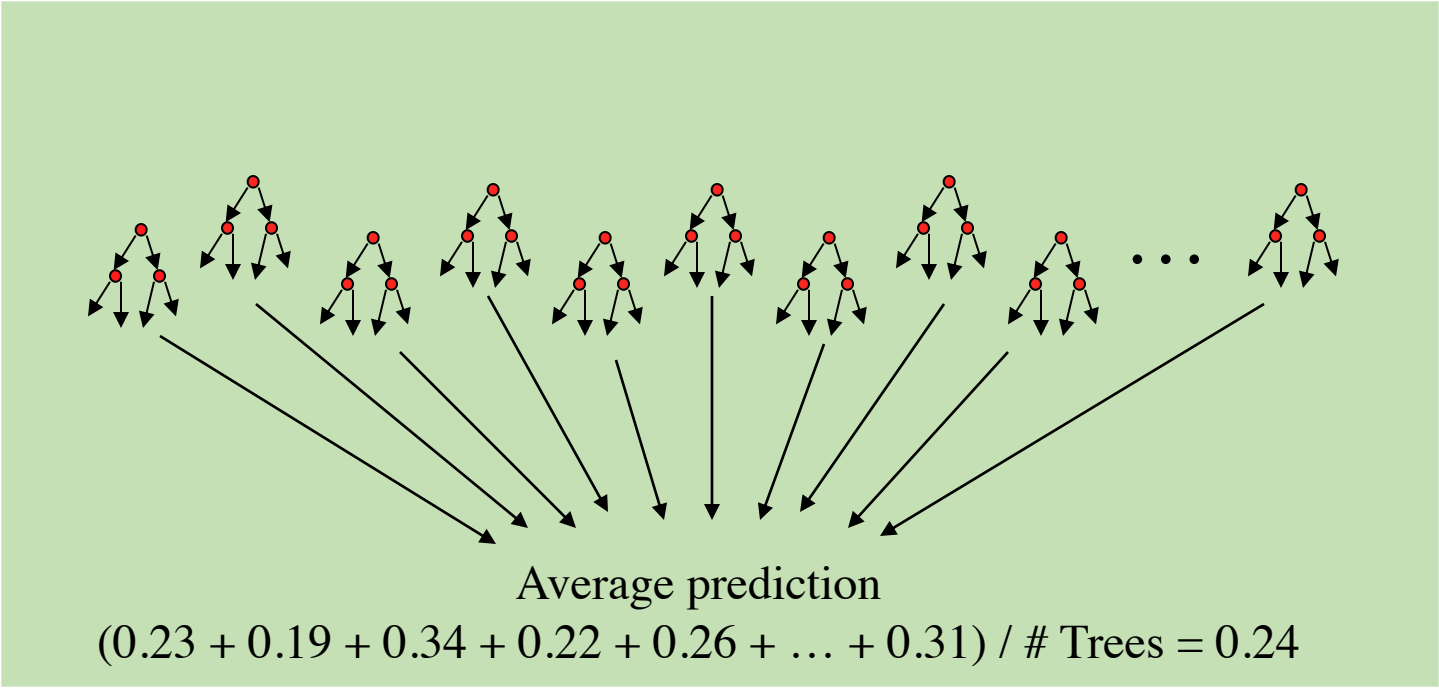
Bagging

- Bagging predictors generates multiple versions of a predictor and uses these to get an **aggregated predictor**
- The aggregation **averages over the versions** when predicting a numerical value and a **majority vote** when predicting a class.
- The multiple versions are formed by making bootstrap replicates of the learning set and using these as new learning sets.
  - *That is, use samples of the data, with repetition*

- The vital element is the **instability of the prediction** method. If perturbing the learning set can cause significant changes in the predictor constructed then bagging can improve accuracy.

Example: Bagged Decision Trees

- Draw 100 bootstrap samples of data
- Train trees on each sample → 100 trees
- Average prediction of trees on test examples (or majority vote)



Random Forests (*Bagged Trees*++)

Key idea: sample data (bagging) + sample features

- Draw ***1000+*** bootstrap samples of data
- ***Draw sample of available attributes at each split***
- Train trees on each sample/attribute set → ***1000+*** trees
- Average prediction of trees on out-of-bag samples