Machine Learning

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

Recall - *Quantifying Performance*

- We want to be able to say something rigorous about the performance of our learning algorithm

- We will concentrate on discussing the number of examples one needs to **see** before we can say that our learned hypothesis is good.

Learning Conjunctions

- There is a hidden conjunction the learner is to learn

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

- *How many examples are needed to learn it ? How ?*
  - **Protocol I**: The learner proposes instances as queries to the teacher ("active learner")
  - **Protocol II:** The teacher (who knows f) provides training examples ("learning with teacher")
  - **Protocol III:** Some random source (e.g., Nature) provides training examples; the Teacher (Nature) provides the labels (f(x))

Learning Conjunctions

- Protocol III: *Some random source provides examples*
  - Teacher (Nature) provides the labels (f(x))
- Algorithm: ***Elimination***
  - *Start with the set of all literals as candidates*
  - *Eliminate a literal that is not active (0) in a positive example*

- <(1,1,1,1,1,1,…,1,1), 1>
- <(1,1,1,0,0,0,…,0,0), 0> learned nothing
- <(1,1,1,1,1,0,…0,1,1), 1>
- <(1,0,1,1,0,0,…0,0,1), 0> learned nothing
- <(1,1,1,1,1,0,…0,0,1), 1>
- <(1,0,1,0,0,0,…0,1,1), 0> Final hypothesis:
- <(1,1,1,1,1,1,…,0,1), 1>
- <(0,1,0,1,0,0,…0,1,1), 0>

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

$$f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{99} \wedge x_{100}$$

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

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

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

Is that *good* ? ***Performance*** ? ***# examples*** ?

Learning Coniunctions

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

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

Is that *good* ?
Performance ? *# of* ***examples*** ?

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

To understand this better, let's consider the types of mistakes our learning algorithm will make. *Can you characterize the type of mistakes?*

Can we make mistakes on negative examples?
Can we make mistakes on positive examples?
When will we make mistakes on a positive example?

We will only make positive mistakes in cases x_1 is not active

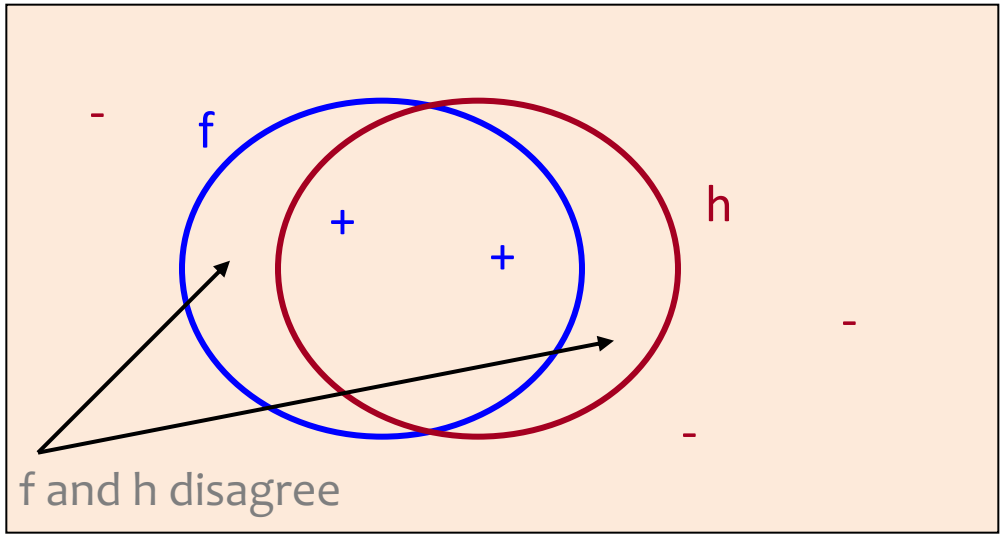
What is the probability of a mistake? (aka *error rate of the learned function)*

PAC Learning – Intuition

Can we bound the Error

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

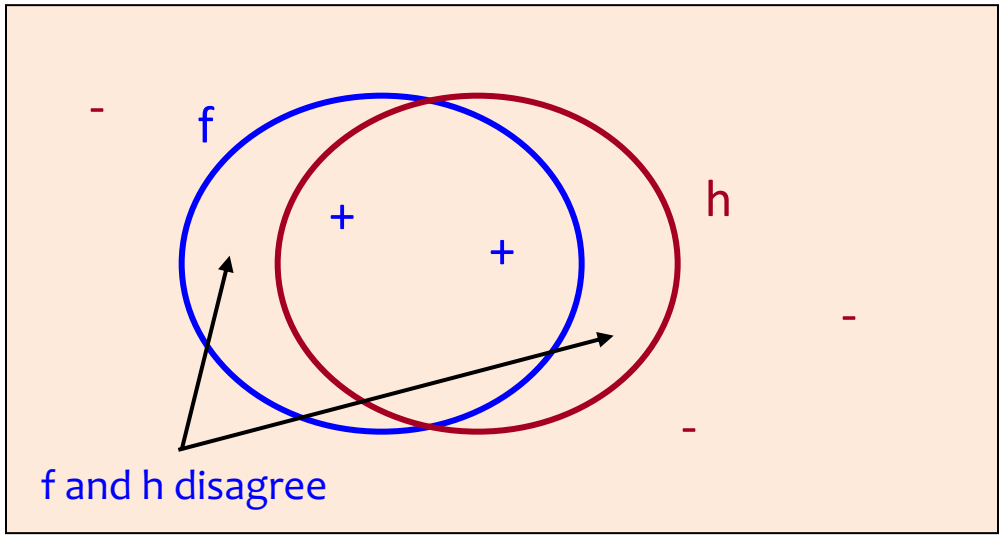
given what we know about the training instances ?



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

PAC Learning – Intuition

- We have seen many examples (drawn according to 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 the chance of "keeping" x_1 is small.

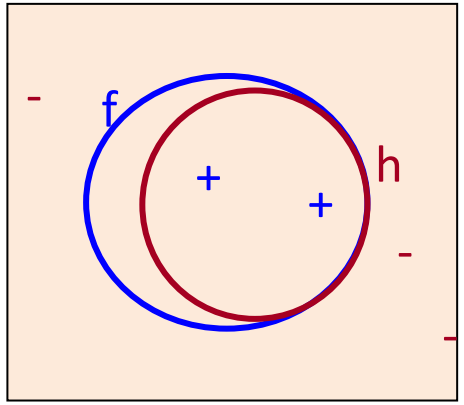


$$\text{Error}_D = \Pr_{x \in D}[f(x) \neq h(x)]$$

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

Learning Conjunctions– Analysis (1)

- Let z be a *literal*.
- Let $p(z)$ be the probability that z *is false in positive example* sampled from *D.*
  - $p(z)$ is also the probability *that z is deleted from h* in a randomly chosen positive example (*during training*)
  - If z *is in the target concept*, than $p(z) = 0$.

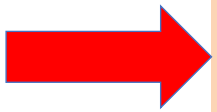


$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning Conjunctions– Analysis (1)

- **Claim**: *h will make mistakes only on positive examples*.
  - A mistake is made only if a literal z, that is in h but not in f, is false in a positive example. In this case, h will say **NEG**, but the example is *positive*.
  - Thus, $p(z)$ is also the probability that z causes h to make a mistake on a randomly drawn example from *D* .
  - There may be overlapping reasons for mistakes, **but the sum clearly bounds it.**

Can we bound this error? How many examples do we need to see?



$$\text{Error}(h) \leq \sum_{z \in h} P(z)$$

Learning Conjunctions– Analysis (2)

- Call a literal z **bad** if $p(z) > \varepsilon/n$.
  - A bad literal is a literal that has a significant probability to appear with a positive example but, nevertheless, it has not appeared with one in the training data.
- *Claim*: If there are no bad literals, than $error(h) < \varepsilon$.

Flows directly from the definition of a bad variable and the bound the error of learned classifier

$$\mathbf{Error(h) \leq \sum_{z \in h} P(z)}$$

Learning Conjunctions– Analysis (2)

- Call a literal z **bad** if $p(z) > \varepsilon/n$.

What if there are bad literals ?

- Let z be a bad literal.
- *The probability that it will not be eliminated by a given example:*

$$\Pr(z \text{ survives one example}) = 1 - \Pr(z \text{ is eliminated by one example}) = 1 - p(z) < 1 - \varepsilon/n$$

The probability that z will not be eliminated by m examples is:

$$\Pr(z \text{ survives m independent examples}) = (1 - p(z))^m < (1 - \varepsilon/n)^m$$

- There are at most n bad literals, so the probability that some bad literal survives m examples is bounded by $n(1 - \varepsilon/n)^m$

How can we decrease the probability of a bad literal?

Learning Conjunctions– Analysis (3)

- **We want this probability to be small.**
  - We want to choose m large enough such that the probability that some z survives m examples is less than δ.
- *I.e., that z remains in h, and makes it different from the target function*

$$\Pr(z \text{ survives } m \text{ example}) = n(1-\varepsilon/n)^m < \delta$$

- Using $1-x < e^{-x}$ (x>0) it is sufficient to require that $n\, e^{-m\varepsilon/n} < \delta$
- Therefore, we need m examples to guarantee a probability of failure (error > ε) of less than δ, where m is:

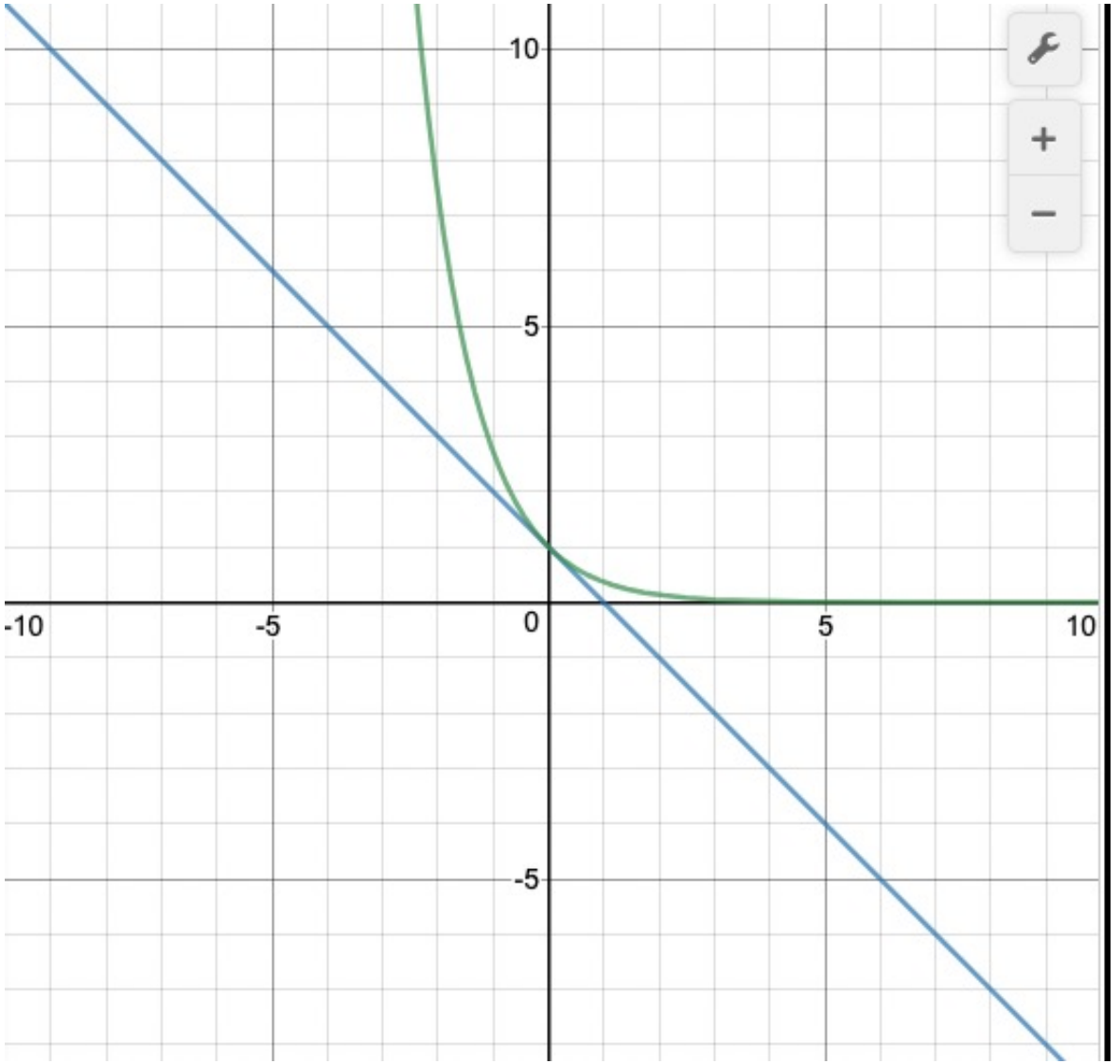
$$m > \frac{n}{\varepsilon}\{\ln(n) + \ln(1/\delta)\}$$

Sidenote

$1-x \leq e^{-x}$ (for all x)

but for x>0

$1-x < e^{-x}$ (x>0)



Learning Conjunctions– Analysis (3)

- Therefore, we need m examples to guarantee a probability of failure (error >ε) of less than δ, where m is $m > \frac{n}{\varepsilon}\{\ln(n) + \ln(1/\delta)\}$

Theorem: If m is as above, then:

- With **probability > 1-δ**, there are **no bad literals**; equivalently, **with probability > 1-δ**, **Err(h) < ε**

- **Examples:**
  - With **δ=0.**1, **ε=0.1**, and n=100, we need 6907 examples.
  - With **δ=0.1**, **ε=0.1**, and n=10, we need only 460 example, only 690 for **δ=0.01**

The change in δ resulted in a smaller increase in examples. ***Why***?

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.

Examples: Conjunctions

- Conjunction (general): *The size of the hypothesis space is* 3^n

$$m > \frac{1}{\varepsilon}\{\ln(3^n) + \ln(1/\delta)\} = \frac{1}{\varepsilon}\{n \ln 3 + \ln(1/\delta)\}$$

(slightly different than previous bound)

- 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 we go to n=100, this goes just to 1130, (linear with n) but changing the confidence to 1% it goes just to 1145 (logarithmic with **δ**)

- These results hold for any ***consistent learner***.

Examples: K-CNF

CNF: Conjunctive Normal Form

- Very expressive class, consists of conjunctions of disjunctions of literals: $(l_1 \vee l_2 \vee ... \vee l) \wedge (l_1 \vee l_2 \vee ... \vee l_{30})$

K- CNF: Each disjunction clause consists of k-terms

$$f = \wedge_{i=1}^{m} (l_{i_1} \vee l_{i_2} \vee ... \vee l_{i_k})$$

K-term CNF: Formula contains k disjunctions

$$f = \wedge_{i=1}^{k} (l_{i_1} \vee l_{i_2} \vee ... \vee l_{i_m})$$

Examples: K-CNF

$$f = \wedge_{i=1}^{m}(l_{i_1} \vee l_{i_2} \vee \ldots \vee l_{i_k})$$

How can we prove learnability?

(1)Sample complexity
(2)Time complexity

Occam Algorithm for f ∈ k-CNF

- Draw a sample D of size m
- Find a hypothesis h that is consistent with all the examples in D
- Determine sample complexity:

Sample complexity: correlates with the size of the hypothesis class. ***Count***!

Examples: K-CNF

Occam Algorithm for f ∈ k-CNF

- Draw a sample D of size m
- Find a hypothesis h that is consistent with all the examples in D
- Determine sample complexity:

Sample complexity**: correlates with the size of the hypothesis class. **Count!

$$f = C_1 \wedge C_2 \wedge .. \wedge .C_m ; ..... ; C_i = l_1 \vee l_2 \vee ... \vee l_k$$

(1) How many disjunctions, over n variables, of size k are there?

Each clause has to "pick" k literals out of 2n possibilities: $\binom{2n}{k} \approx 2n^k$

(2) How many possibilities for combinations of disjunctive clauses?

The are n^k possible disjunctions, each can appear or not $2^{(2n)^k}$

This is a HUGE number! ***Is K-CNF learnable?***

Examples: K-CNF

Occam Algorithm for f ∈ k-CNF

- Draw a sample D of size m
- Find a hypothesis h that is consistent with all the examples in D
- Determine sample complexity:

$$\ln(|\ k-CNF\ |) = O(n^k) \qquad \ldots\ldots\ldots 2^{(2n)^k} \ldots\ldots\ldots (2n)^k$$

- Due to the sample complexity result h is guaranteed to be a PAC hypothesis; but we need to learn a consistent hypothesis.

How do we find the consistent hypothesis h ?

Examples: *Elimination* K-CNF

How do we find the consistent hypothesis h ?

- Define a new set of features (literals), one for each clause of size k

$$y_j = l_{i_1} \vee l_{i_2} \vee \ldots \vee l_{i_k}\ ; j = 1,2,\ldots,n^k$$

- Use the algorithm for learning monotone conjunctions, over the new set of literals

Example: n=4, k=2; monotone k-CNF

$$y_1 = x_1 \vee x_2 \qquad y_2 = x_1 \vee x_3 \qquad y_3 = x_1 \vee x_4$$

$$y_4 = x_2 \vee x_3 \qquad y_5 = x_2 \vee x_4 \qquad y_6 = x_3 \vee x_4$$

Original examples: (0000,l) (1010,l) (1110,l) (1111,l)

New examples: (000000,l) (111101,l) (111111,l) (111111,l)

Computational Complexity

$$f = \vee_{i=1}^{k} (l_{i_1} \wedge l_{i_2} \vee \ldots \vee l_{i_m})$$

Can we learn k-term DNF?

- *Although from a sample complexity perspective things are good, they are not good from a computational complexity perspective*
  - Determining whether there is a k-term DNF consistent with a set of training data is NP-Hard. Therefore 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)

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

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

$$\wedge \{a \vee b;\ a \vee d;\ a \vee e;\quad b;\ b \vee d;\ b \vee e;\ c \vee b;\ c \vee d;\ c \vee e;\}$$

Computational Complexity

- Determining if there is a 2-term DNF consistent with the training data is NP-Hard
  - Therefore the class of k-term-DNF is **not** efficiently (***properly***) PAC learnable due to computational complexity
- 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)
- Therefore, C=k-term-DNF can be learned as using H=k-CNF as the hypothesis Space
- ***Importance of representation:*** concepts that cannot be learned using one representation can be learned using another (more expressive) representation.

This result is analogous to an earlier observation that it's better to learn **linear separators** than **conjunctions**.

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

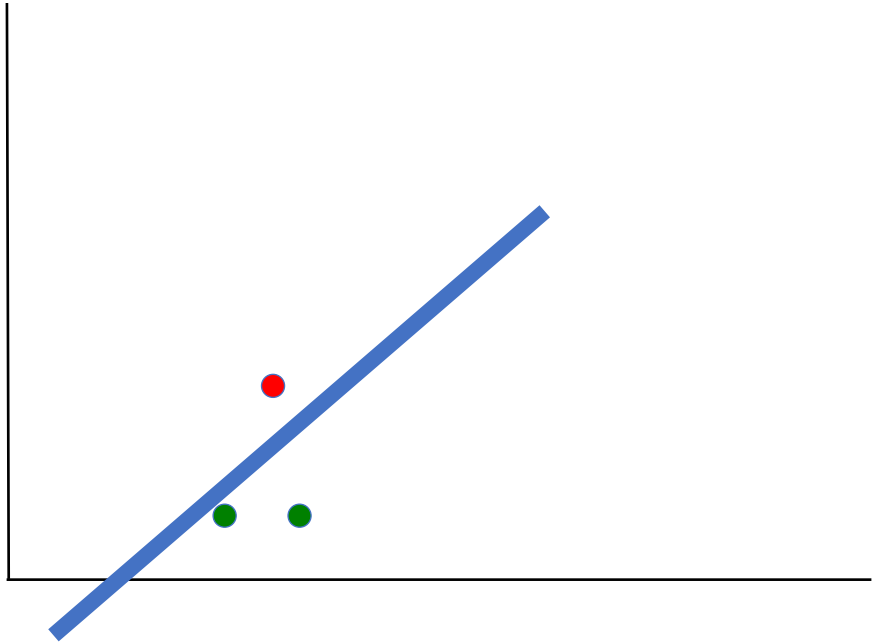
- Our analysis so far focused on *finite hypothesis spaces*
  - The size of H was a way to capture H‘s expressiveness
- Some infinite hypothesis spaces are more expressive than others
  - E.g., Rectangles, vs. 17- sides convex, Linear threshold function vs. a *conjunction* of linear threshold functions

VC Dimension: Intuition

Consider two hypothesis spaces defined over 2 variables:

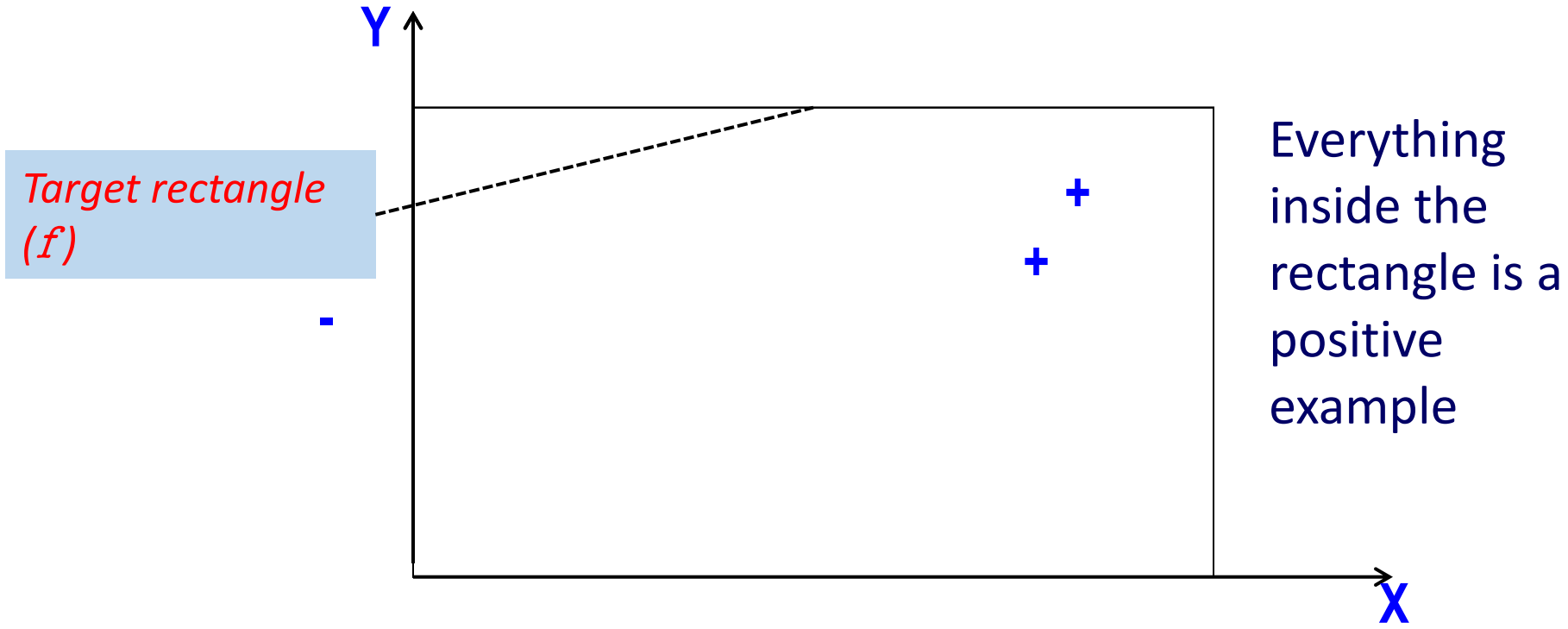
1. Y-Axis parallel lines
2. Lines

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



Learning Rectangles

- Assume the target concept is an axis parallel rectangle



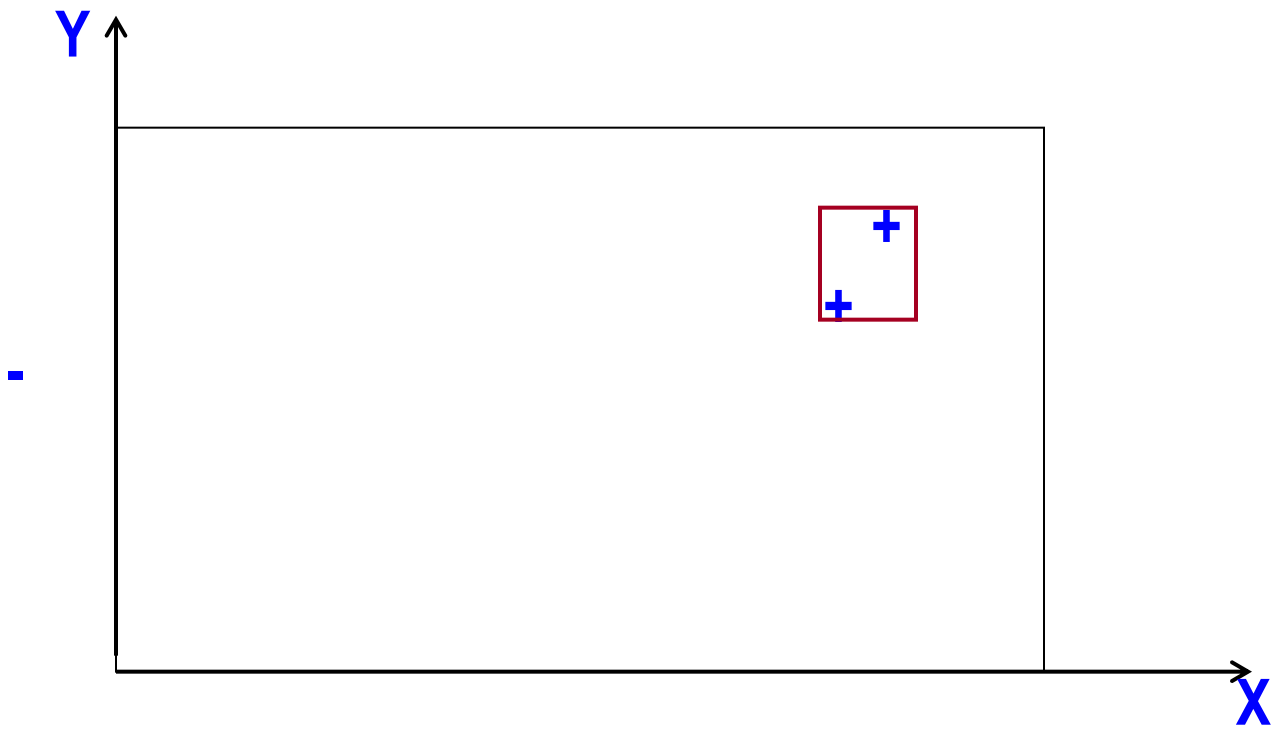
Everything inside the rectangle is a positive example

How can you learn using this hypothesis class?

➔ What are the parameters we need to learn?

Learning Rectangles

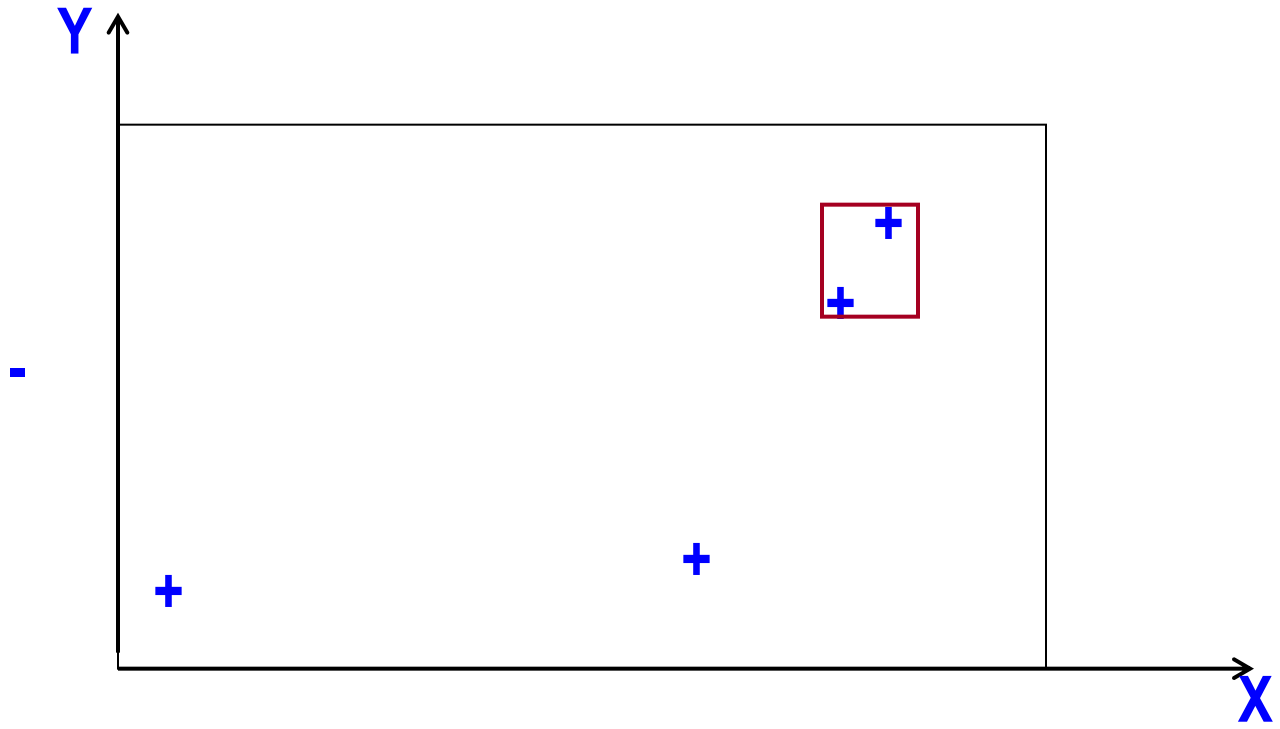
- Assume the target concept is an axis parallel rectangle



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

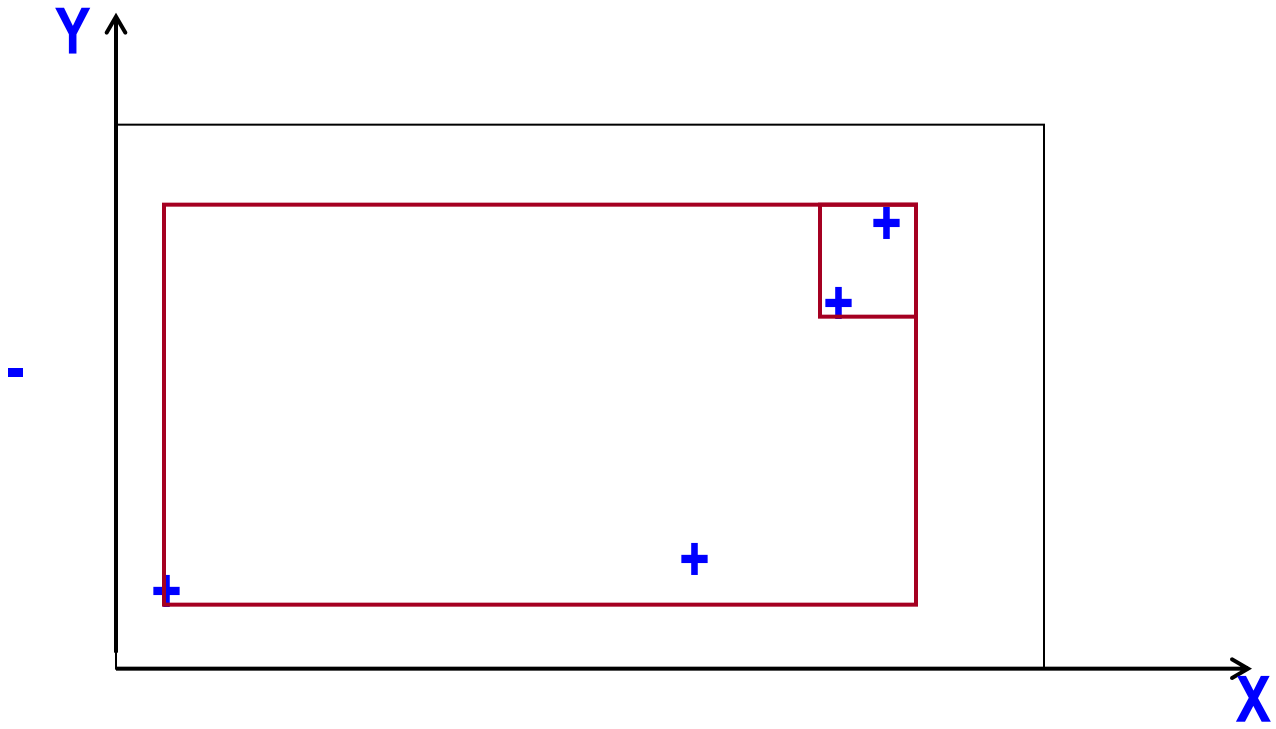
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



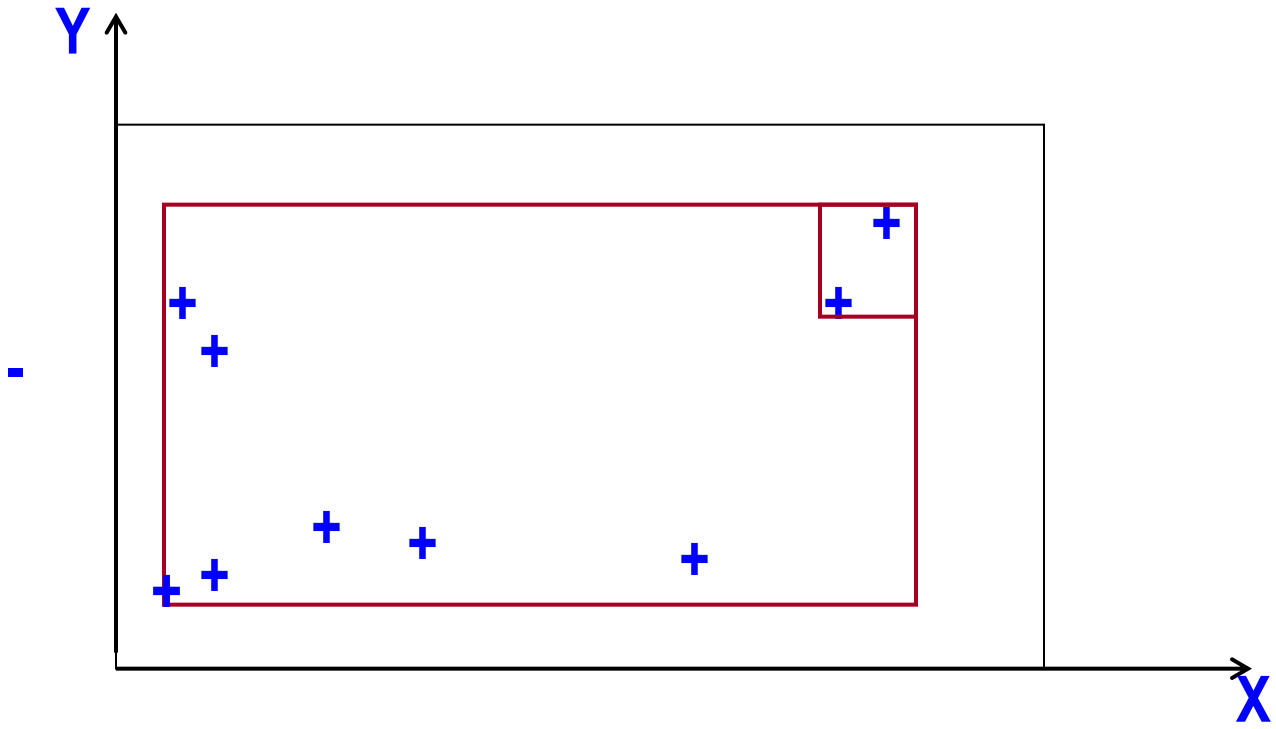
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



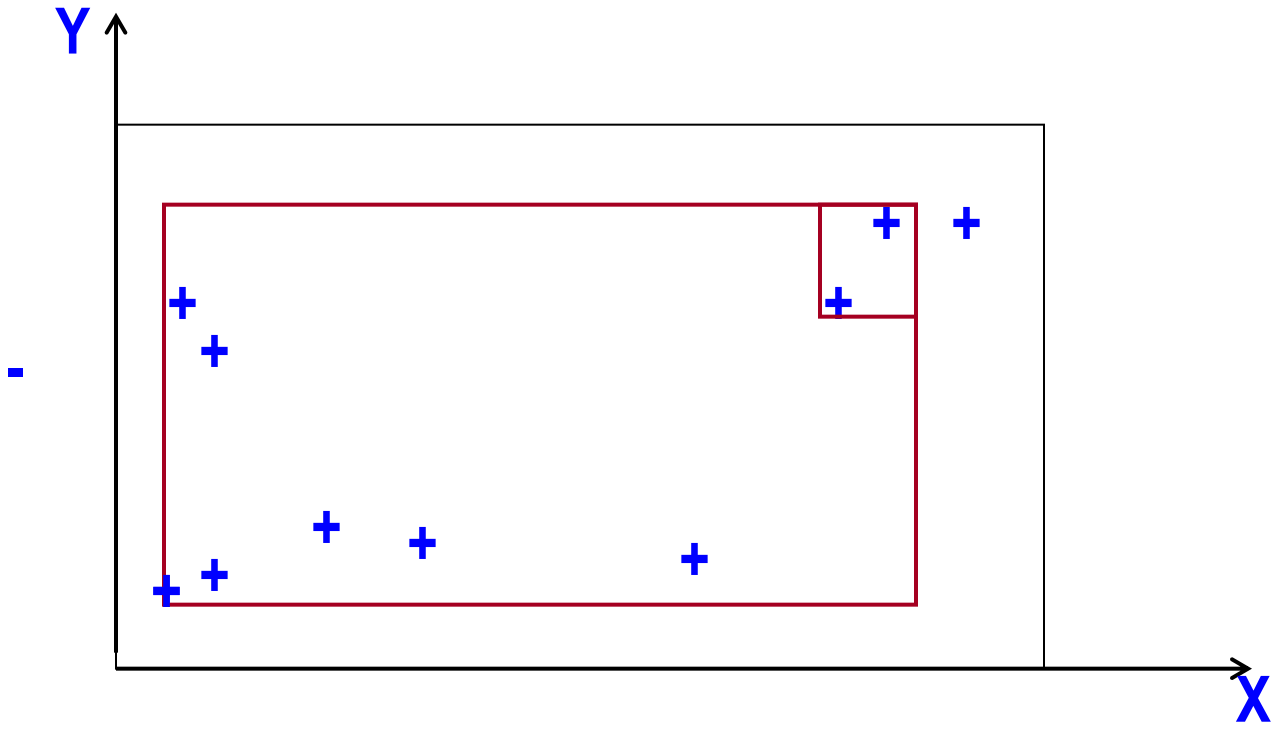
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



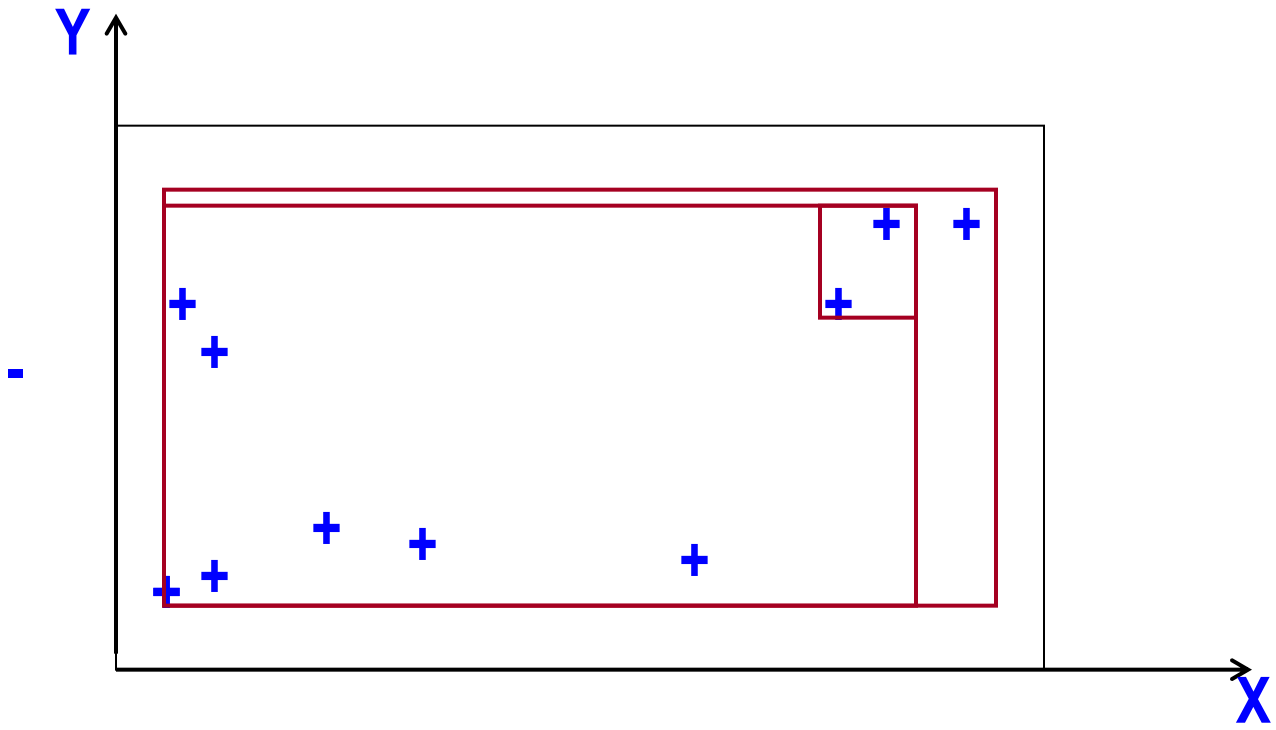
Learning Rectangles

- 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

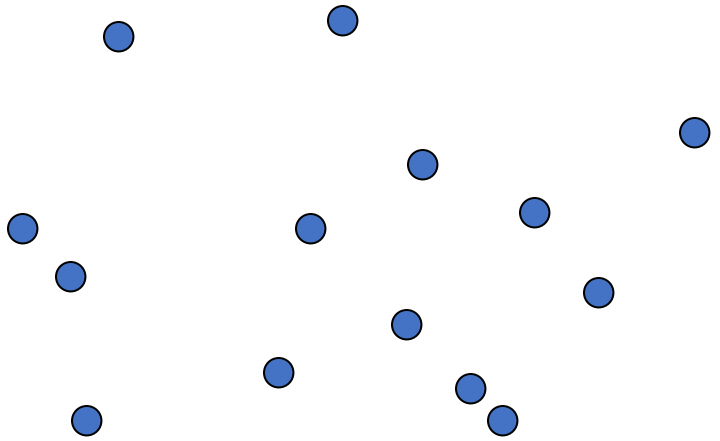
This is an infinite hypothesis space. Can we still make any claims?

- Will we be able to learn the target rectangle ?

Infinite Hypothesis Space

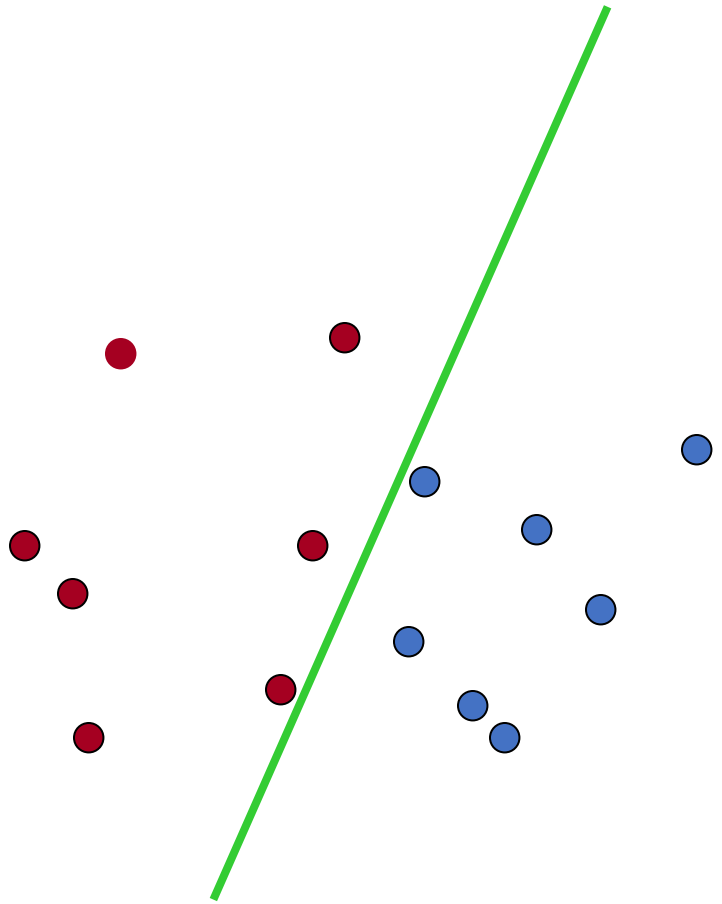
- Our analysis so far focused on *finite hypothesis spaces*
  - The size of H was a way to capture H's expressiveness
- Some infinite hypothesis spaces are more expressive than others
  - E.g., Rectangles, vs. 17- sides convex, Linear threshold function vs. a *conjunction* of linear threshold functions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



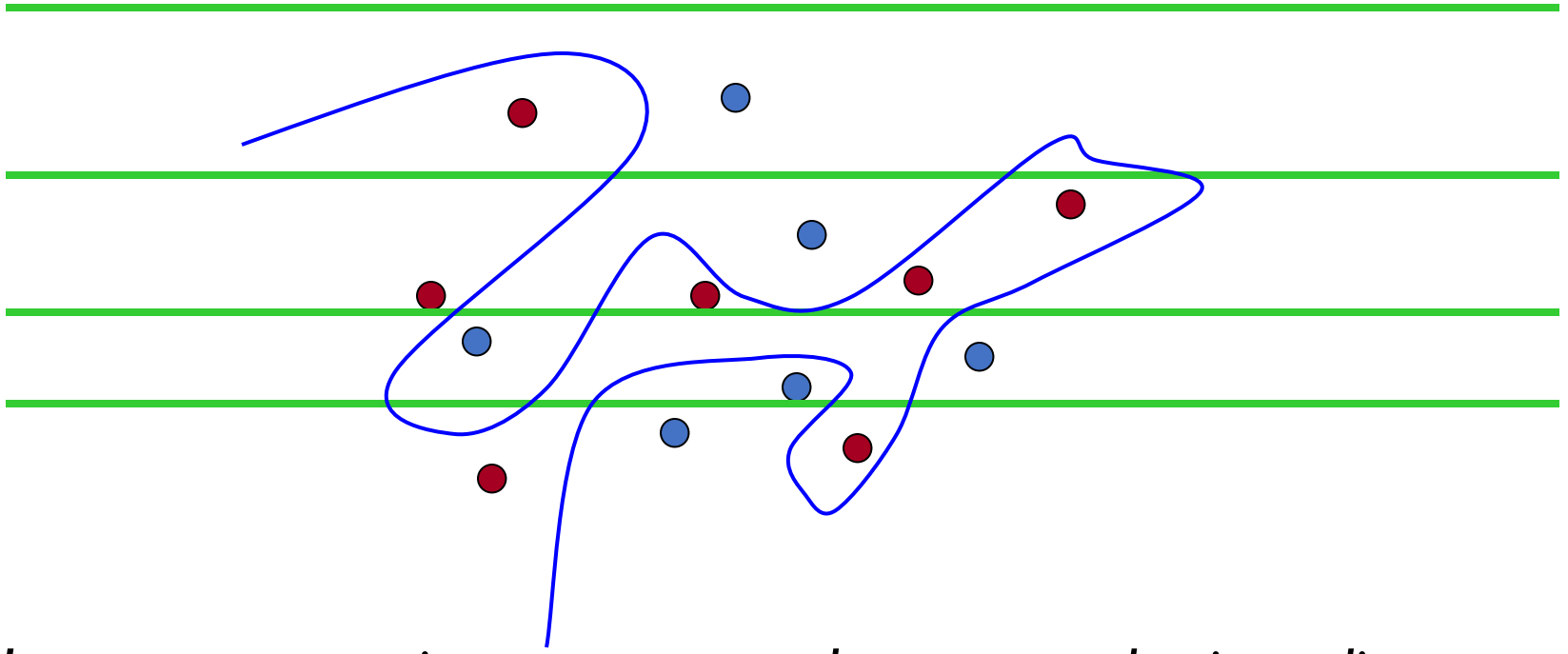
Given a set of examples (points), there are many ways to assign labels

Shattering



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

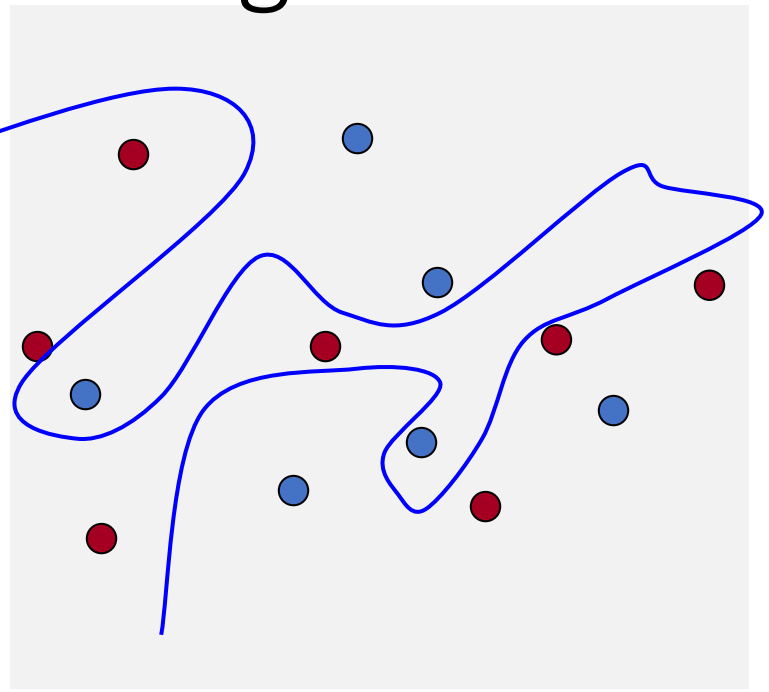
Shattering

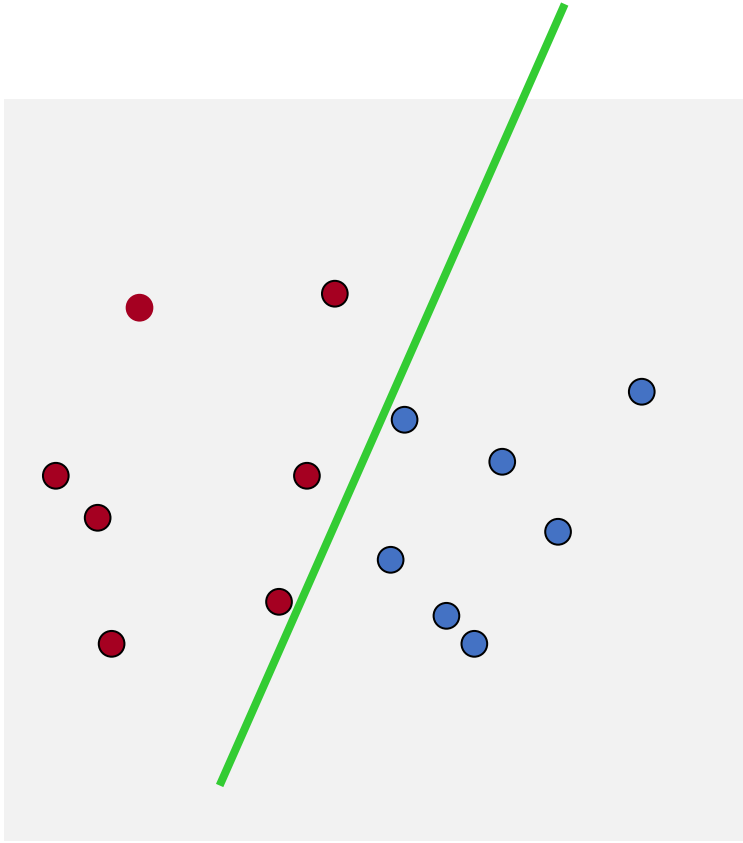


However some assignments cannot be separated using a linear function.

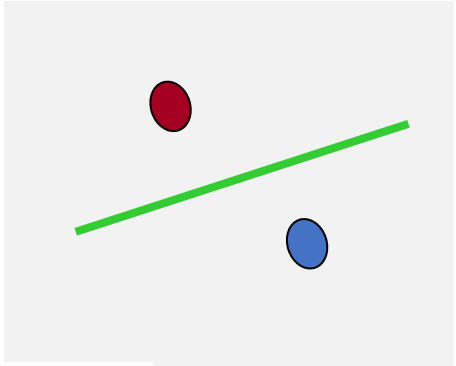
➔*You will need a more complex function to separate this assignment*

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

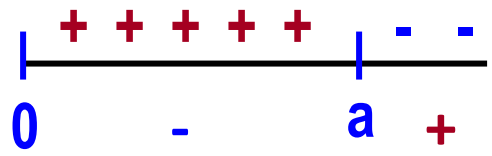
Shattering

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

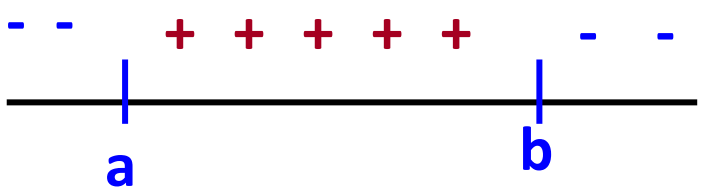


Sets of two points cannot be shattered.

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

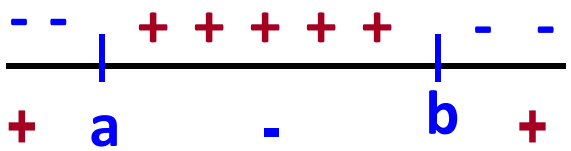
Intervals on the real axis: [a,b], for some real numbers b>a



Everything inside the interval is positive

Shattering

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

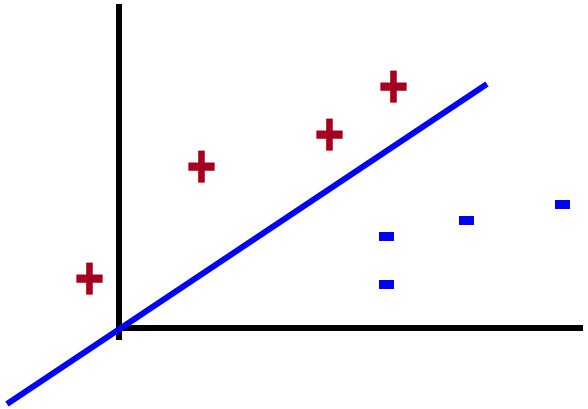


All sets of one or two points can be shattered
but sets of ***three*** points cannot be shattered

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

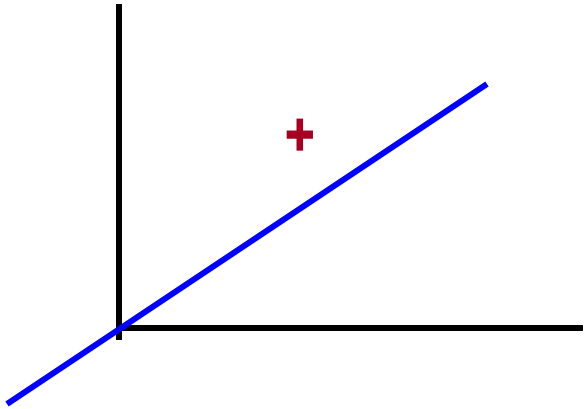
***Half-spaces in the plane*:**



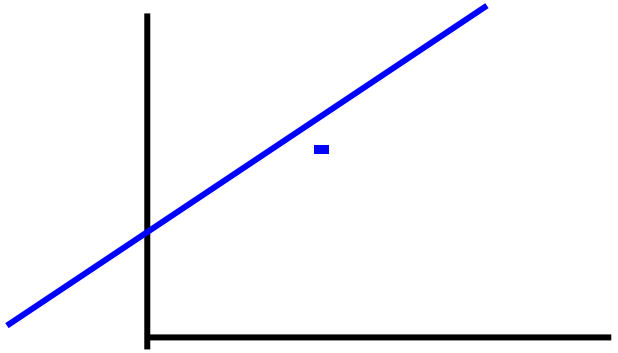
Shattering

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?

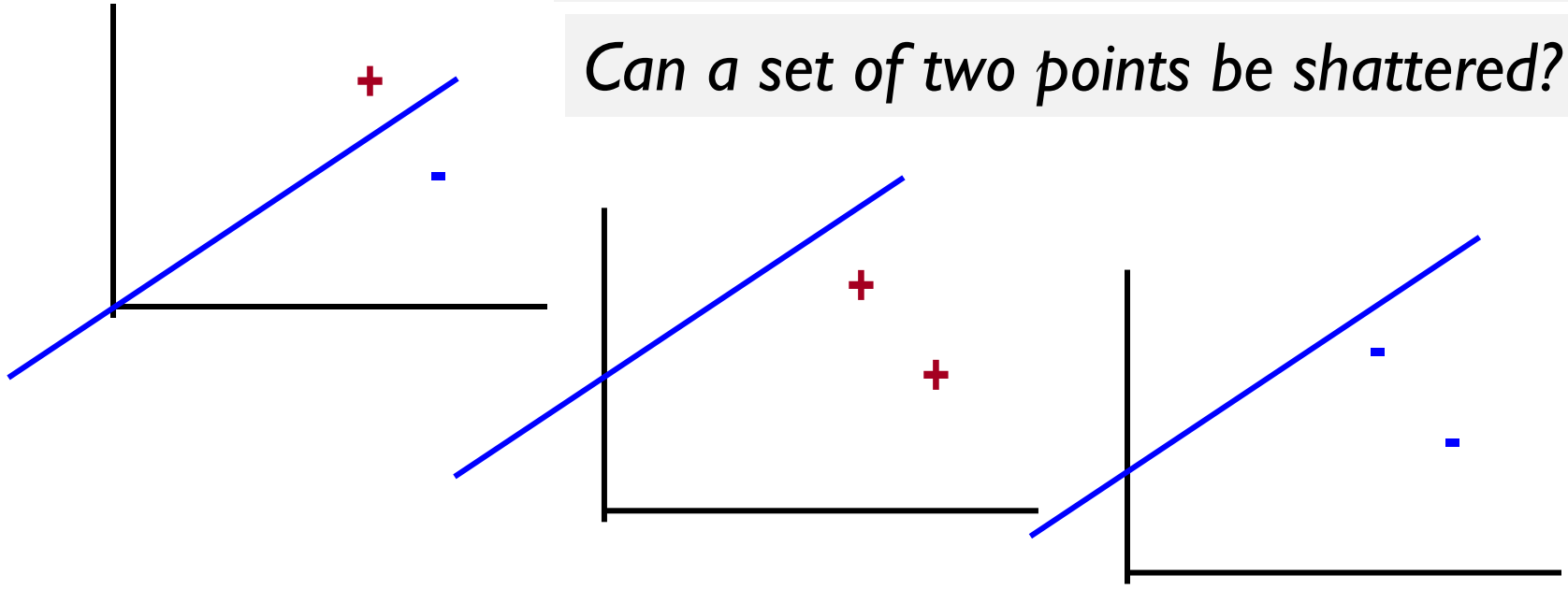


Shattering

***Half-spaces in the plane*:**

Can a set of one point be shattered?

Can a set of two points be shattered?



Shattering

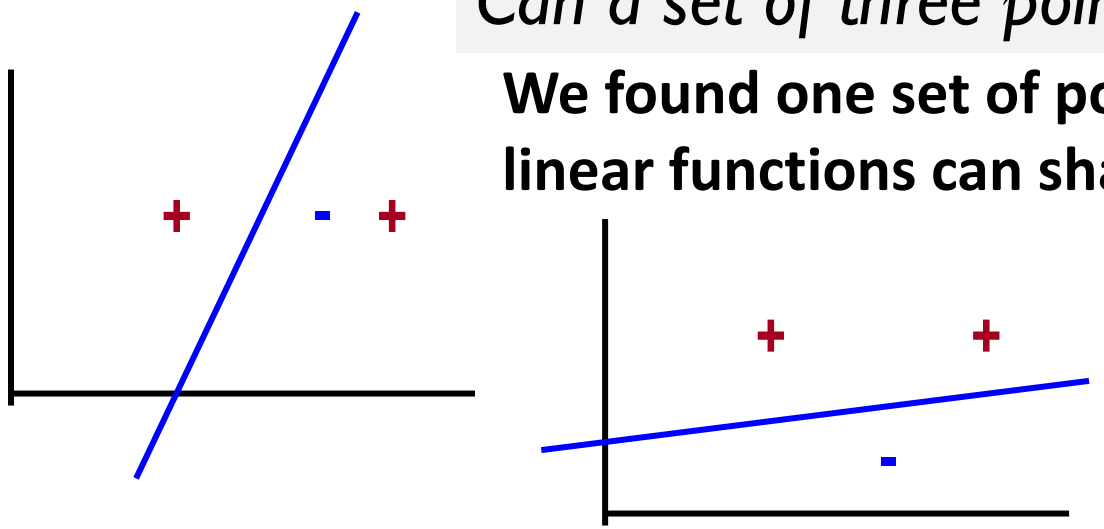
***Half-spaces in the plane*:**

Can a set of one point be shattered?

Can a set of two points be shattered?

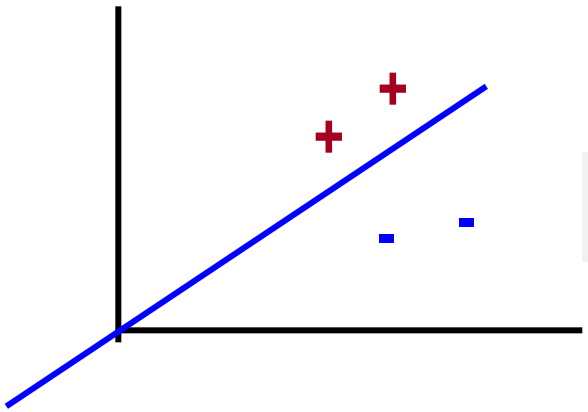
Can a set of three points be shattered?

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?

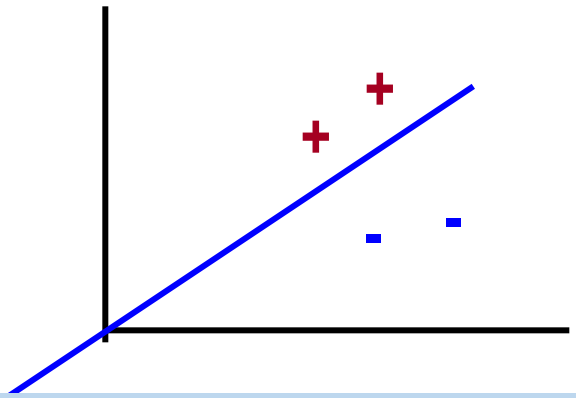
Can a set of two points be shattered?

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?

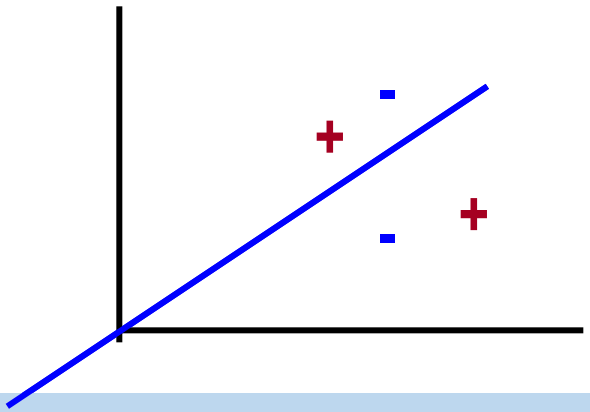
Can a set of four points be shattered?

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?

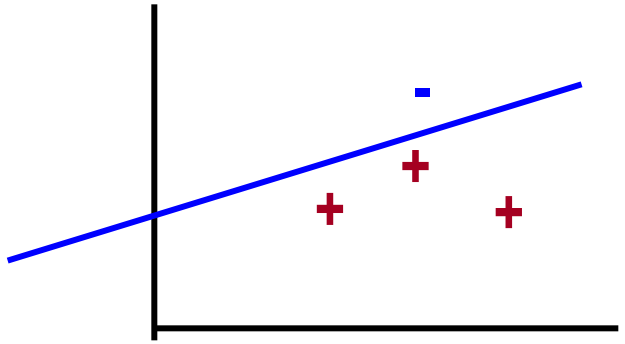
How can you show that a set of 4 points cannot be shattered?

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?

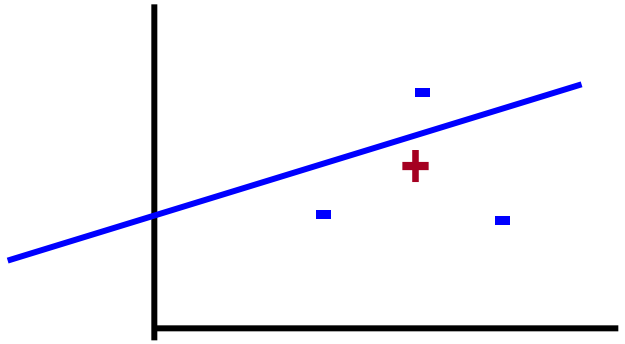
Can a set of four points be shattered?

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?

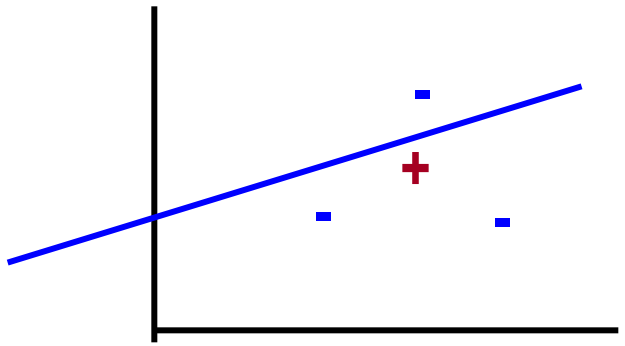
Can a set of four points be shattered?

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 ∈ 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

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

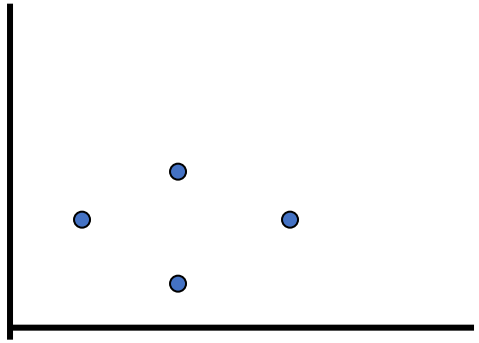
- Then, with prob. at least (1-δ), h has error less than ε.
- if m is polynomial we have a PAC learning algorithm
- We also need to produce the hypothesis h *efficiently*.
- What if |H| is finite?

➔**Note that to shatter m examples-**

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

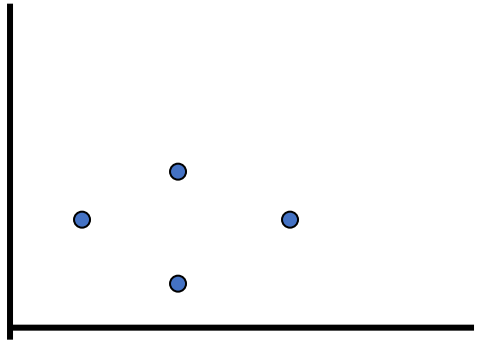
Learning Axis Parallel Rectangles

- 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?

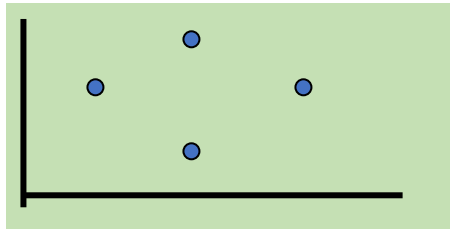
Can a set of ***five*** points be shattered?

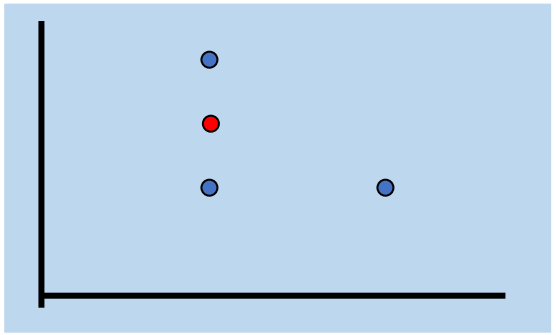
Can a set of ***six*** points be shattered?

...

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?

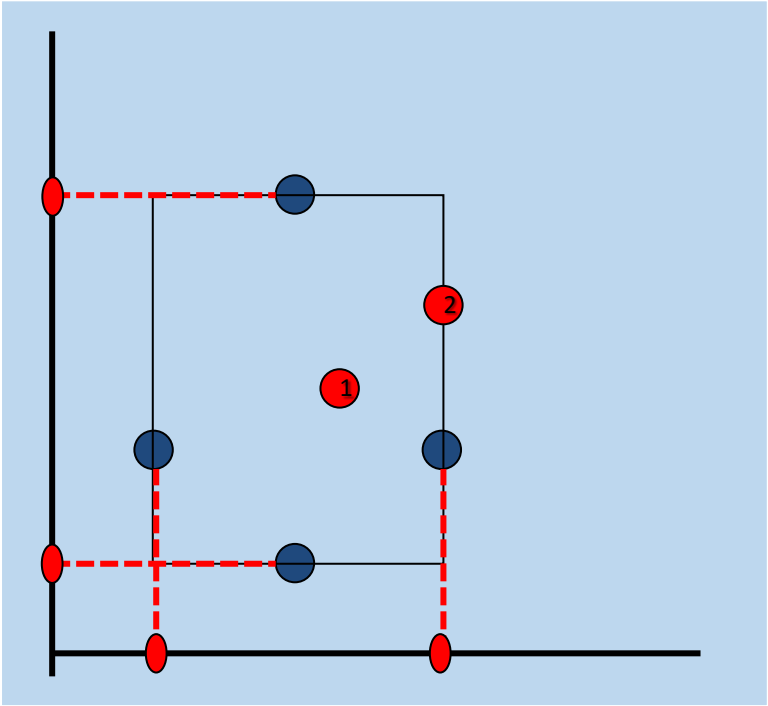
Can a set of ***four*** points be shattered?

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?

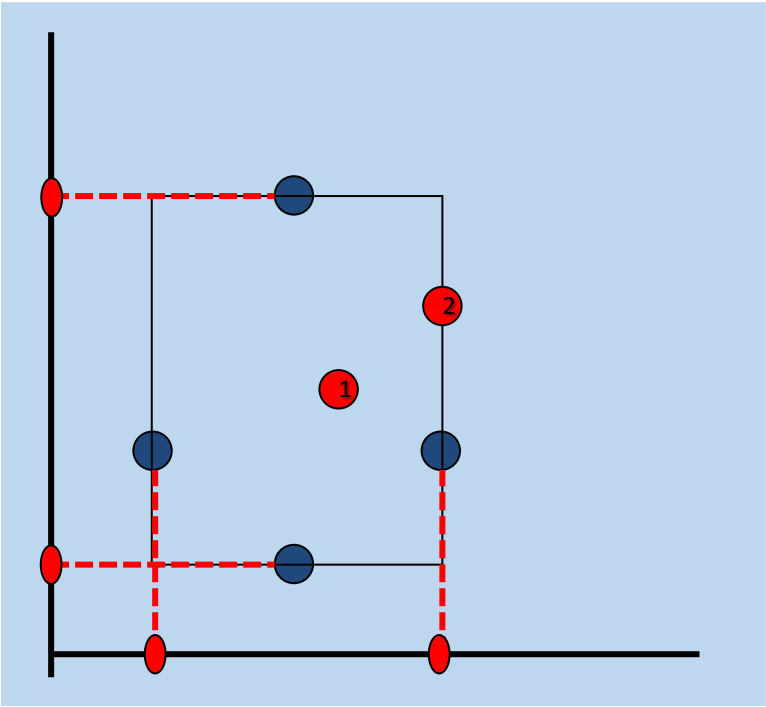


Consider the rectangle defined by *the min-x,max-x, min-y, max-y* values of the four extreme points. The fifth point must lie either inside the rectangle or on the edges

➔ 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?

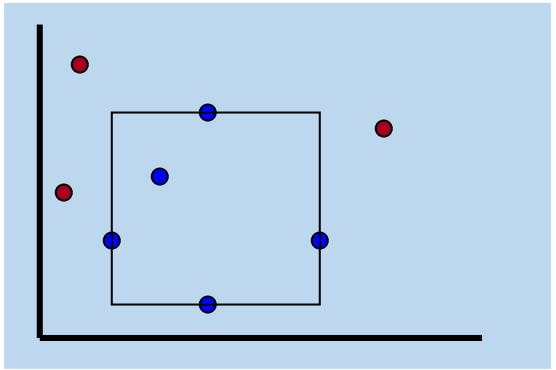
No set of five points can be shattered!

➔ VC(H) = 4

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

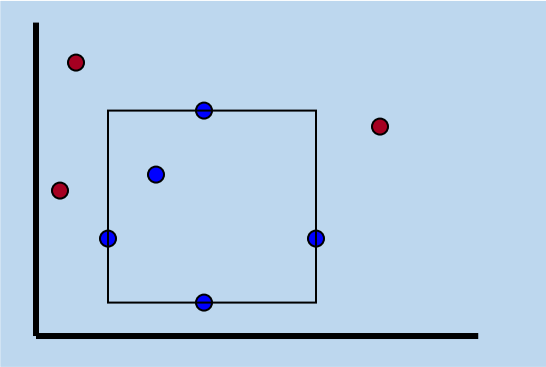
Learning Rectangles

- 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