Where are we?

- Algorithmically:
  - Perceptron + Winnow
  - Gradient Descent
- Models:
  - Online Learning; Mistake Driven Learning
- What do we know about Generalization? (to previously unseen examples?)
  - How will your algorithm do on the next example?
- Next we develop a theory of Generalization.
  - We will come back to the same (or very similar) algorithms and show how the new theory sheds light on appropriate modifications of them, and provides guarantees.

Computational Learning Theory

- What general laws constrain inductive learning ?
  - What learning problems can be solved ?
  - When can we trust the output of a learning algorithm ?

- We seek theory to 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

Recall what we did earlier:

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 (you) 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
  - Protocol II: The teacher (who knows f) provides training examples
  - Protocol III: Some random source (e.g., Nature) provides training examples; the Teacher (Nature) provides the labels (f(x))

Learning Conjunctions

- Protocol I: The learner proposes instances as queries to the teacher
- Since we know we are after a monotone conjunction:
- Is x_{100} in? <(1,1,1…,1,0), ?> f(x)=0 (conclusion: Yes)
- Is x_{99} in? <(1,1,…1,0,1), ?> f(x)=1 (conclusion: No)
- Is x_1 in ? <(0,1,…1,1,1), ?> f(x)=1 (conclusion: No)

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

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

What happens here if the conjunction is not known to be monotone?
If we know of a positive example, the same algorithm works.

Learning Conjunctions

- Protocol II: The teacher (who knows f) provides training examples
- <(0,1,1,1,1,0,…,0,1), 1> (We learned a superset of the good variables)
- To show you that all these variables are required…
  - <(0,0,1,1,1,0,…,0,1), 0> need x_2
  - <(0,1,0,1,1,0,…,0,1), 0> need x_3
  - …..
  - <(0,1,1,1,1,0,…,0,0), 0> need x_{100}

Modeling Teaching
Is tricky

- A straight forward algorithm requires k = 6 examples to produce the hidden conjunction (exactly).

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

Learning Conjunctions

- Protocol III: Some random source (e.g., Nature) provides training examples
- Teacher (Nature) provides the labels (f(x))
  - <(1,1,1,1,1,1,…,1,1), 1>
  - <(1,1,1,0,0,0,…,0,0), 0>
  - <(1,1,1,1,1,0,...0,1,1), 1>
  - <(1,0,1,1,1,0,...0,1,1), 0>
  - <(1,1,1,1,1,0,...0,0,1), 1>
  - <(1,0,1,0,0,0,...0,1,1), 0>
  - <(1,1,1,1,1,1,…,0,1), 1>
  - <(0,1,0,1,0,0,...0,1,1), 0>

Learning Conjunctions

- Protocol III: Some random source (e.g., Nature) provides training 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> $f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge \ldots \wedge x_{100}$
- <(1,1,1,0,0,0,…,0,0), 0> learned nothing
- <(1,1,1,1,1,0,...0,1,1), 1> $f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{99} \wedge x_{100}$
- <(1,0,1,1,0,0,...0,0,1), 0> learned nothing
- <(1,1,1,1,1,0,...0,0,1), 1> $f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$
- <(1,0,1,0,0,0,...0,1,1), 0> Final hypothesis:
- <(1,1,1,1,1,1,…,0,1), 1> $h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$
- <(0,1,0,1,0,0,...0,1,1), 0>

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

Is that good ? Performance ?
# of examples ?

Two Directions

Two Directions

- Can continue to analyze the probabilistic intuition:
  - Never saw x_1 in positive examples, maybe we'll never see it?
  - And if we will, it will be with small probability, so the concepts we learn may be pretty good
  - Good: in terms of performance on future data
  - PAC framework
- Mistake Driven Learning algorithms
  - Update your hypothesis only when you make mistakes
  - Good: in terms of how many mistakes you make before you stop, happy with your hypothesis.
  - Note: not all on-line algorithms are mistake driven, so performance measure could be different.

Prototypical Concept Learning

- Instance Space: X
  - Examples
- Concept Space: C
  - Set of possible target functions: $f \in C$ is the hidden target function
  - All n-conjunctions; all n-dimensional linear functions.
- Hypothesis Space: H set of possible hypotheses
- Training instances Sx{0,1}: positive and negative examples of the target concept $f \in C$

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

- Determine: 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 $x \in X$?

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

Prototypical Concept Learning

- Instance Space: X
  - Examples
- Concept Space: C
  - Set of possible target functions: $f \in C$ is the hidden target function
  - All n-conjunctions; all n-dimensional linear functions.
- Hypothesis Space: H set of possible hypotheses
- Training instances S x{0,1}: positive and negative examples of the target concept $f \in C$. Training instances are generated by a fixed unknown probability distribution *D* over X

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

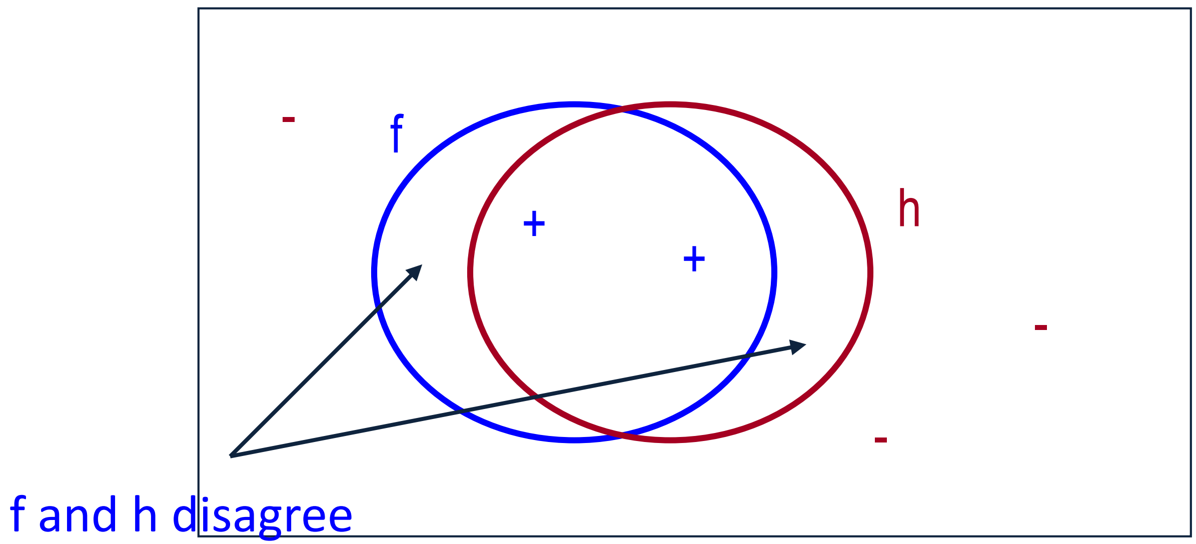
- Determine: A hypothesis $h \in H$ that estimates f, evaluated by its performance on subsequent instances $x \in X$ drawn according to *D*

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

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 our error will be small

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



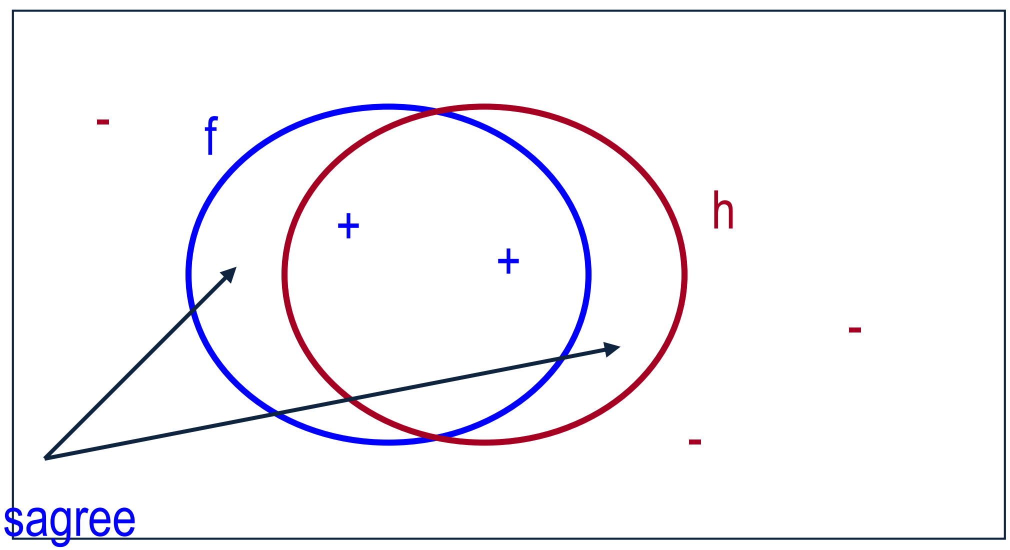
$$h = \underline{x_1} \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

The notion of error

Can we bound the Error

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

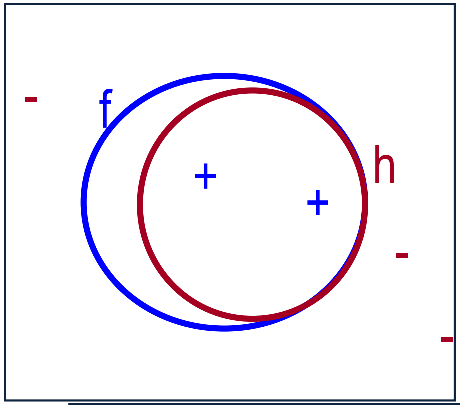
given what we know about the training instances ?



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

Learning Conjunctions– Analysis (1)

- Let z be a literal. Let p(z) be the probability that, in D-sampling an example, it is positive and z is false in it. *Then: Error(h)* $\leq \sum_{z \in h} p(z)$
  - p(z) is also the probability that a randomly chosen example is positive and z is deleted from h.
  - If z is in the target concept, than p(z) = 0.
- 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 POS.



  - 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.

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

Learning Conjunctions– Analysis (2)

- Call a literal z in the hypothesis h bad if $p(z) > \varepsilon/n$.
- A bad literal is a literal that is not in the target concept and has a significant probability to appear false with a positive example.
- Claim: If there are no bad literals, than $error(h) < \varepsilon$. Reason: $Error(h) \leq \sum_{z \in h} p(z)$
- What if there are bad literals ?
  - Let z be a bad literal.
  - What is the probability that it will not be eliminated by a given example?

$$Pr(z \text{ survives one example}) = 1 - Pr(z \text{ is eliminated by one example}) \leq$$

$$\leq 1 - p(z) < 1 - \varepsilon/n$$

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

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

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

Learning Conjunctions– Analysis (3)

- We want this probability to be small. Say, 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 survives m example) = $n(1-\epsilon/n)^m < \delta$
- Using $1-x < e^{-x}$ (x>0) it is sufficient to require that $n e^{-m\epsilon/n} < \delta$
- Therefore, we need

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

examples to guarantee a probability of failure (error > ϵ) of less than δ.

- Theorem: If m is as above, then:
  - With probability > $1-\delta$, there are no bad literals; equivalently,
  - With probability > $1-\delta$, Err(h) < ϵ
- 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

Formulating Prediction Theory

- Instance Space X, Input to the Classifier; Output Space $Y = \{-1, +1\}$
- Making predictions with: $h: X \rightarrow Y$
- D: An unknown distribution over $X \times Y$
- S: A set of examples drawn independently from D; $m = |S|$, size of sample.

Now we can define:

- True Error: $Error_D = Pr_{(x,y) \in D} [h(x) \neg = y]$
- Empirical Error: $Error_S = Pr_{(x,y) \in S} [h(x) \neg = y] = \sum_{1,m} [h(x_i) \neg = y_i]$
  - (Empirical Error (Observed Error, or Test/Train error, depending on S))

This will allow us to ask: (1) Can we describe/bound $Error_D$ given $Error_S$?

- Function Space: C – A set of possible target concepts; target is: $f: X \rightarrow Y$
- Hypothesis Space: H – A set of possible hypotheses
- This will allow us to ask: (2) Is C learnable?
  - Is it possible to learn a given function in C using functions in H, given the supervised protocol?

Requirements of Learning

- Cannot expect a learner to learn a concept exactly, since
  - 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 since
  - Sometimes (only in rare learning situations, we hope) the training set will not be representative (will contain uncommon examples).
- Therefore, 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 Correct

- Cannot expect a learner to learn a concept exactly.
- Cannot always expect to learn a close approximation to the target concept
- Therefore, 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 reason we can hope for that is the Consistent Distribution assumption.

Learning

PAC Learnability

Definition

- 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 distributions D over X, and fixed $0 < \varepsilon, \delta < 1$,
- L, given a collection of m examples sampled independently according to D produces
  - with probability at least $(1-\delta)$ a hypothesis $h \in H$ with error at most ε, ($Error_D = Pr_D[f(x) := h(x)]$)
- where m is polynomial in $1/\varepsilon$, $1/\delta$, n and size(H)
- C is efficiently learnable if L can produce the hypothesis in time polynomial in $1/\varepsilon$, $1/\delta$, n and size(H)

PAC Learnability

Comments

- 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 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 (1)

Claim: The probability that there exists a hypothesis $h \in H$ that
(1) is consistent with m examples and
(2) satisfies $error(h) > \varepsilon$ ($Error_D(h) = Pr_{x \in D}[f(x) \neg= h(x)]$)
is less than $|H|(1-\varepsilon)^m$.

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 probability that h is consistent with m example of f is less than $(1-\varepsilon)^m$

- The probability that *some* hypothesis in H is consistent with m examples is less than $|H|(1-\varepsilon)^m$

Note that we don't need a true f for this argument; it can be done with h, relative to a distribution over $X \times Y$.

Occam's Razor (1)

We want this probability to be smaller than δ, that is:

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

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

(with $e^{-x} = 1-x+x^2/2+\ldots$; $e^{-x} > 1-x$; $\ln(1-\varepsilon) < -\varepsilon$; 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)

(gross over estimate)
It is called Occam's razor, because it indicates a preference towards small hypothesis spaces

What kind of hypothesis spaces do we want ? Large ? Small ?
To guarantee consistency we need $H \supseteq C$. But do we want the smallest H possible ?

Administration

Questions

- Hw4 will be out today.
  - Due on March 11 (Saturday)
  - No slack time since we want to release the solutions with enough time before the midterm.
  - You cannot solve all the problems yet.
- Quizzes:
  - Quiz 5 will be due before the Thursday lecture
  - Quiz 6 will be due before next Tuesday
- Midterm is coming in three weeks
  - 3/16, in class
- Project Proposals are due on 3/10.
  - Follow Piazza and the web site.

Consistent Learners

- Immediately 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
    - We showed that if m is large enough, a consistent hypothesis must be close enough to f
    - Check that m is not too large (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
- 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 directly that if we have sufficiently many examples (polynomial in the parameters), than h is close to the target function.

We did not need to show it directly. See above.

Examples

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

Since there are 3 choices for each feature

(not appear, appear positively or appear negatively)

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

(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) +10ln(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.

K-CNF

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

Occam Algorithm (=Consitent Learner 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:**

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

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

- **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 ?

K-CNF

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

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)

Distribution?

More Examples

Unbiased learning: **Consider the hypothesis space of all Boolean functions on n features.**

There are 2^{2^n} **different functions, and the bound is therefore exponential in n.**

k-CNF: **Conjunctions of any number of clauses where each disjunctive clause has at most k literals.** $f = C_1 \wedge C_2 \wedge .. \wedge .C_m ;........; C_i = l_1 \vee l_2 \vee ... \vee l_k$

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

k-clause-CNF: **Conjunctions of at most k disjunctive clauses.**

$$f = C_1 \wedge C_2 \wedge .. \wedge .C_k ;........; C_i = l_1 \vee l_2 \vee ... \vee l_m$$

$$\ln(|\ k - clause - CNF\ |) = O(kn) \qquad ....3^{kn}.......3^n.$$

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

k-term-DNF: **Disjunctions of at most k conjunctive terms.**

Sample Complexity

- All these classes can be learned using a polynomial size sample.
  - We want to learn a 2-term-DNF; what should our hypothesis class be?
- k-CNF: Conjunctions of any number of clauses where each disjunctive clause has at most k literals.

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

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

- k-clause-CNF: Conjunctions of at most k disjunctive clauses.

$$f = C_1 \wedge C_2 \wedge .. \wedge .C_k ;........; C_i = l_1 \vee l_2 \vee ... \vee l_m$$

$$\ln(|\ k-clause-CNF\ |) = O(kn)........3^{kn}.......3^n.$$

- k-DNF: Disjunctions of any number of terms where each conjunctive term has at most k literals.

$$f = T_1 \vee T_2 \vee .. \vee .T_m ;........; T_i = l_1 \wedge l_2 \wedge ... \wedge l_k$$

- k-term-DNF: Disjunctions of at most k conjunctive terms.

Computational Complexity

- Even though from the sample complexity perspective things are good, they are not good from a computational complexity in this case.

What does it mean?

- Determining whether there is a 2-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)

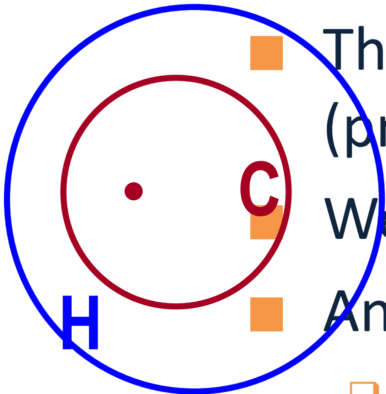
$$T_1 \vee T_2 \vee T_3 = \prod_{x \in T_1 y \in T_2 z \in T_3} \{x \vee y \vee z\}$$

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

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

Computational Complexity

- Determining whether there is a 2-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
- 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.**



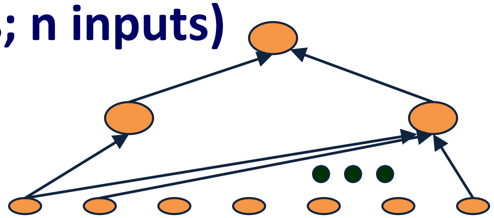
This result is analogous to an earlier observation that it's better to learn linear separators than conjunctions.

Negative Results – Examples

- Two types of nonlearnability results:
- Complexity Theoretic
  - Showing that various concepts classes cannot be learned, based on well-accepted assumptions from computational complexity theory.
  - E.g. : C cannot be learned unless P=NP
- Information Theoretic
  - 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?)
- Usually proofs are for EXACT learning, but apply for the distribution free case.

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:**
  - **DNF Formulas; CNF Formulas**
  - **Deterministic Finite Automata**
  - **Context Free Grammars**

We need to extend the theory in two ways:
(1) What if we cannot be completely consistent with the training data?
(2) What if the hypothesis class we work with is not finite?

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:

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

- 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. $\mathbf{Pr}[p > \hat{p} + \varepsilon] < e^{-2m\varepsilon^2}$
  - (p is the underlying probability of the binary variable (e.g., toss is Head) being 1)

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:

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

- Doing the same union bound game as before, with

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

- We get a generalization bound – a bound on how much will the true error E_D deviate from the observed (training) error E_{TR}.
- For any distribution D generating training and test instances, with probability at least $1-\delta$ over the choice of the training set of size m, (drawn IID), for all $h \in H$

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

Agnostic Learning

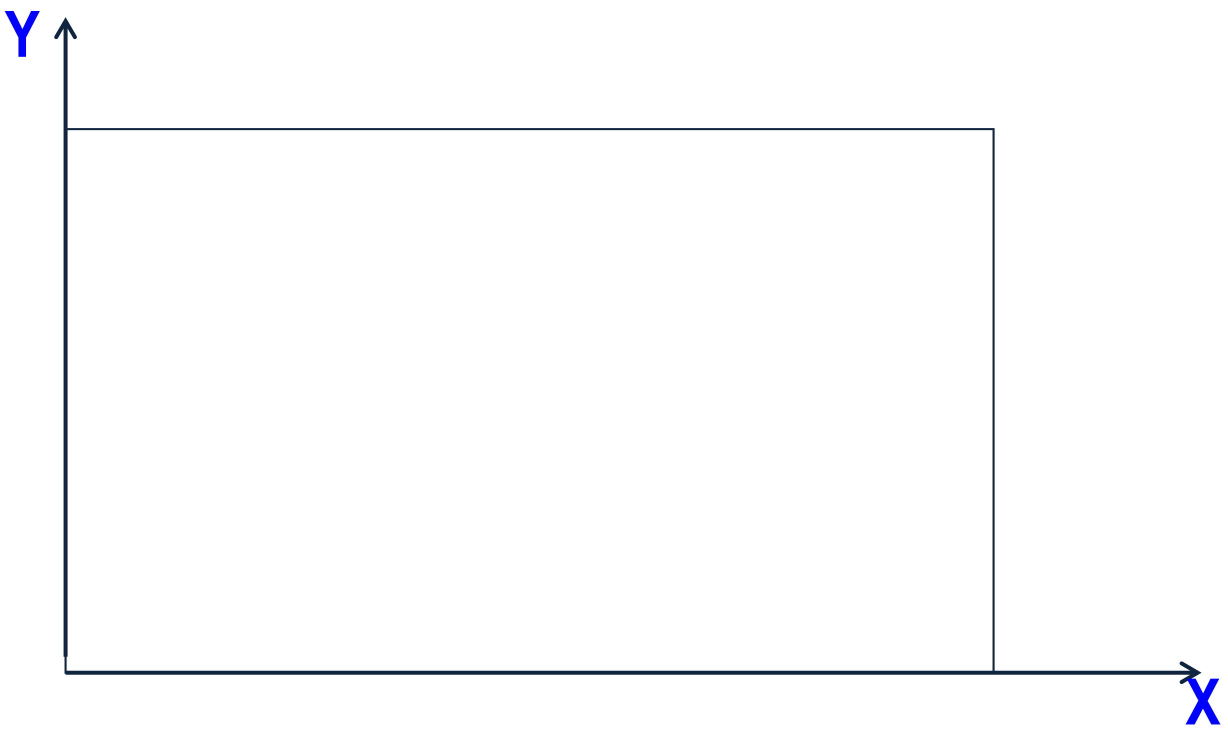
- An agnostic learner which makes no commitment to whether f is in H and returns the hypothesis with least training error over at least the following number of examples m can guarantee with probability at least (1-δ) that its training error is not off by more than ε from the true error.

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

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

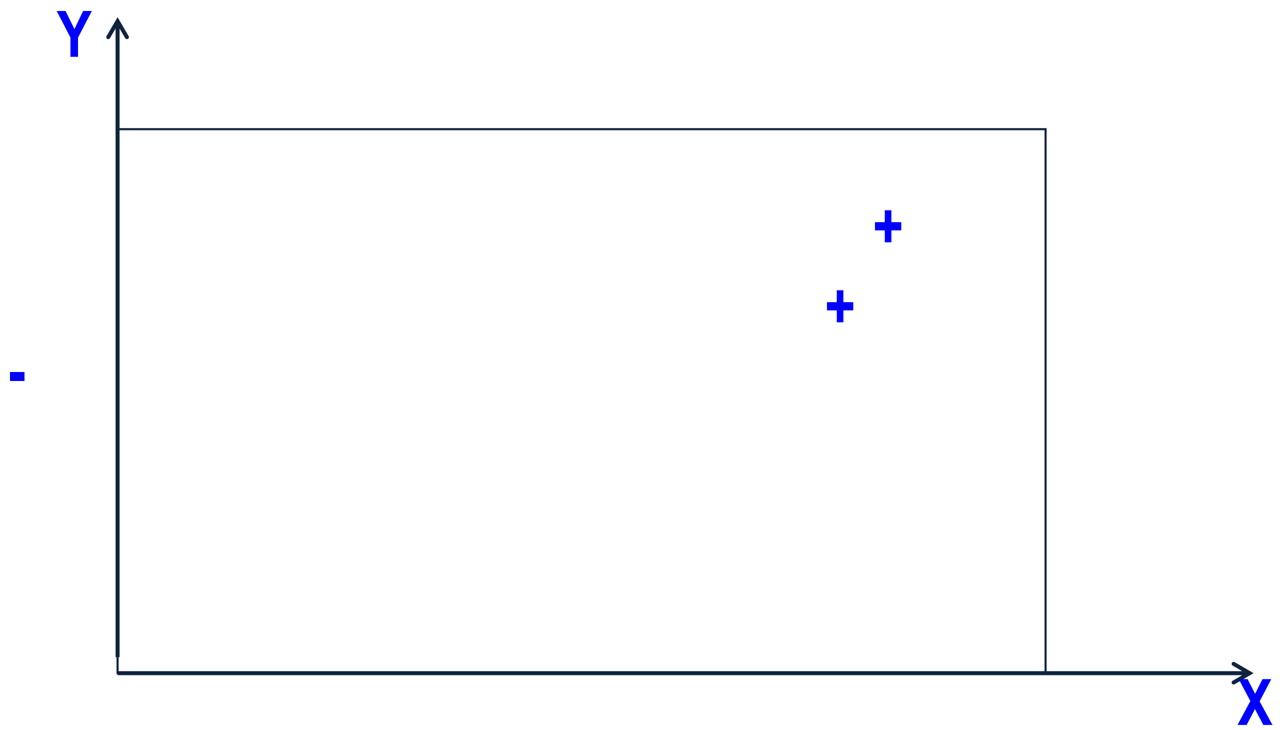
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



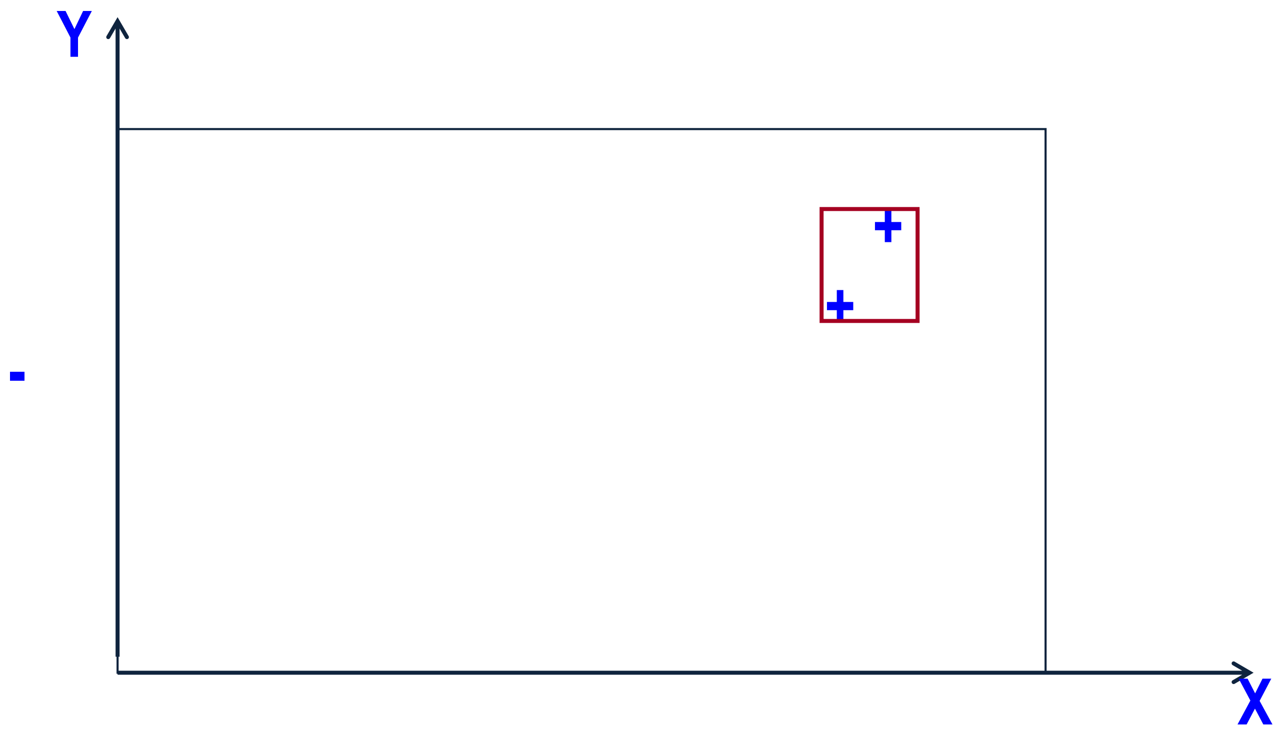
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



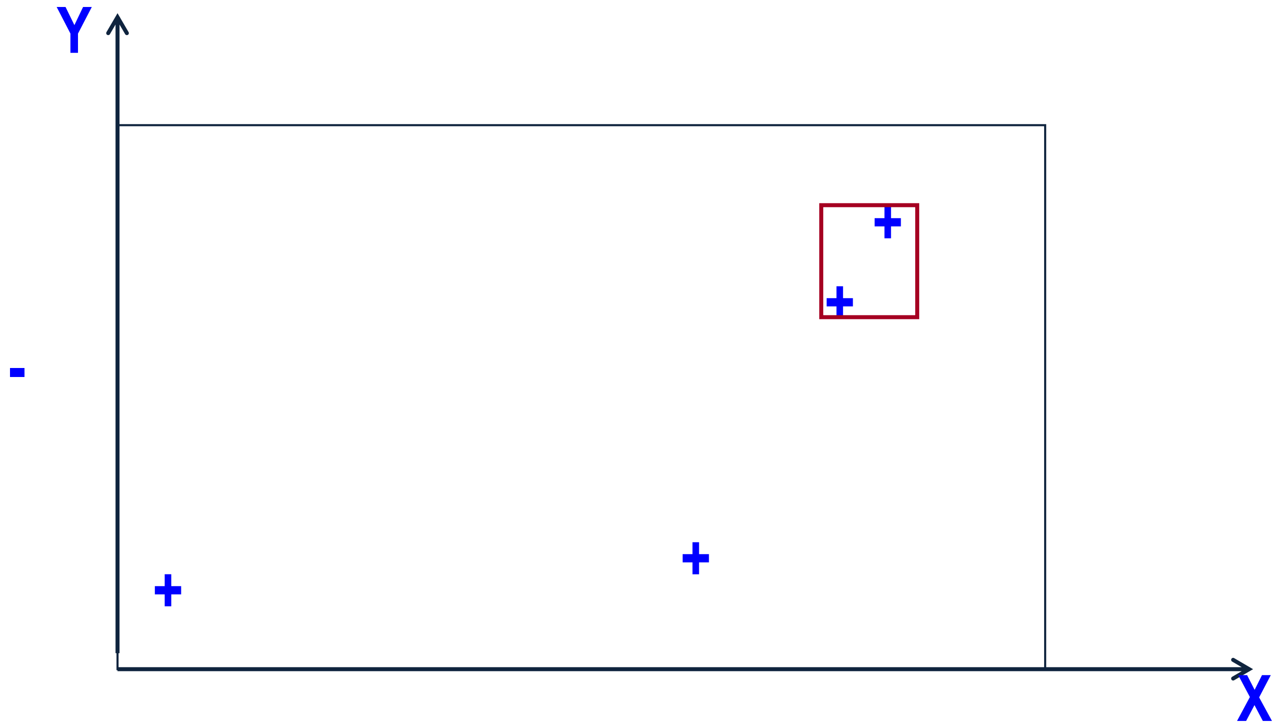
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



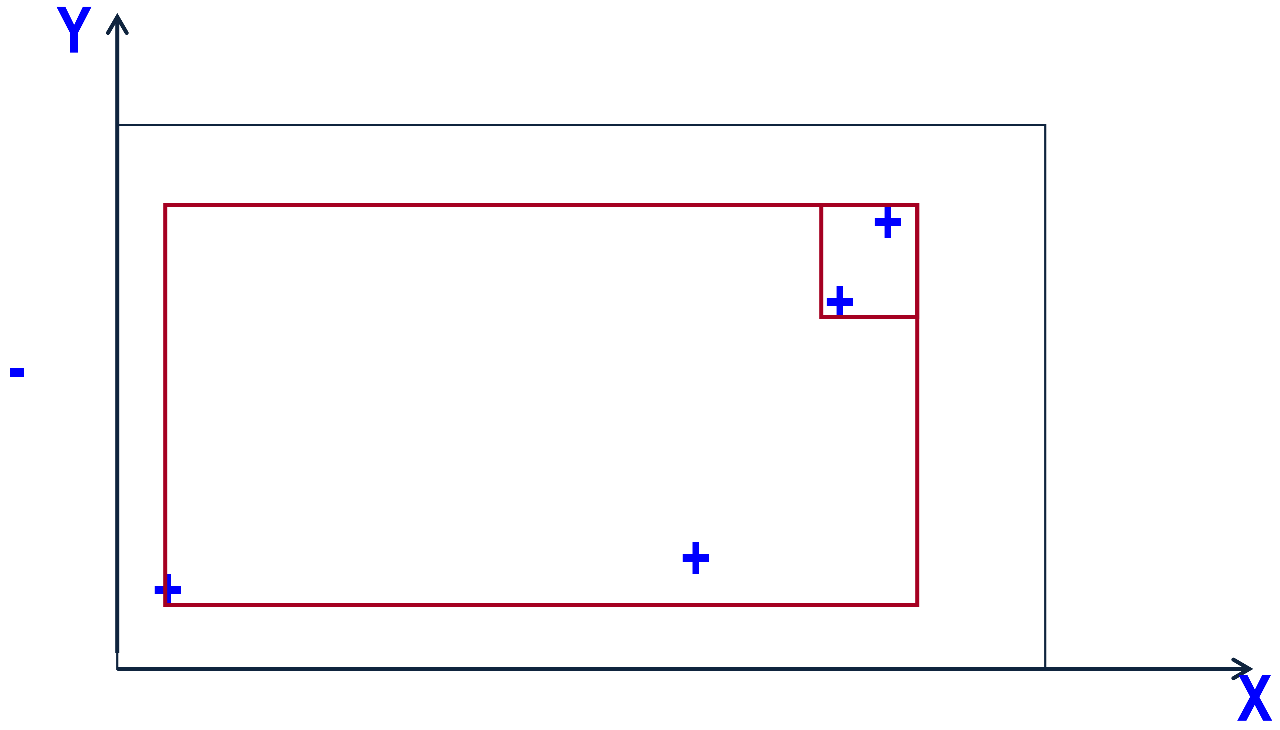
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



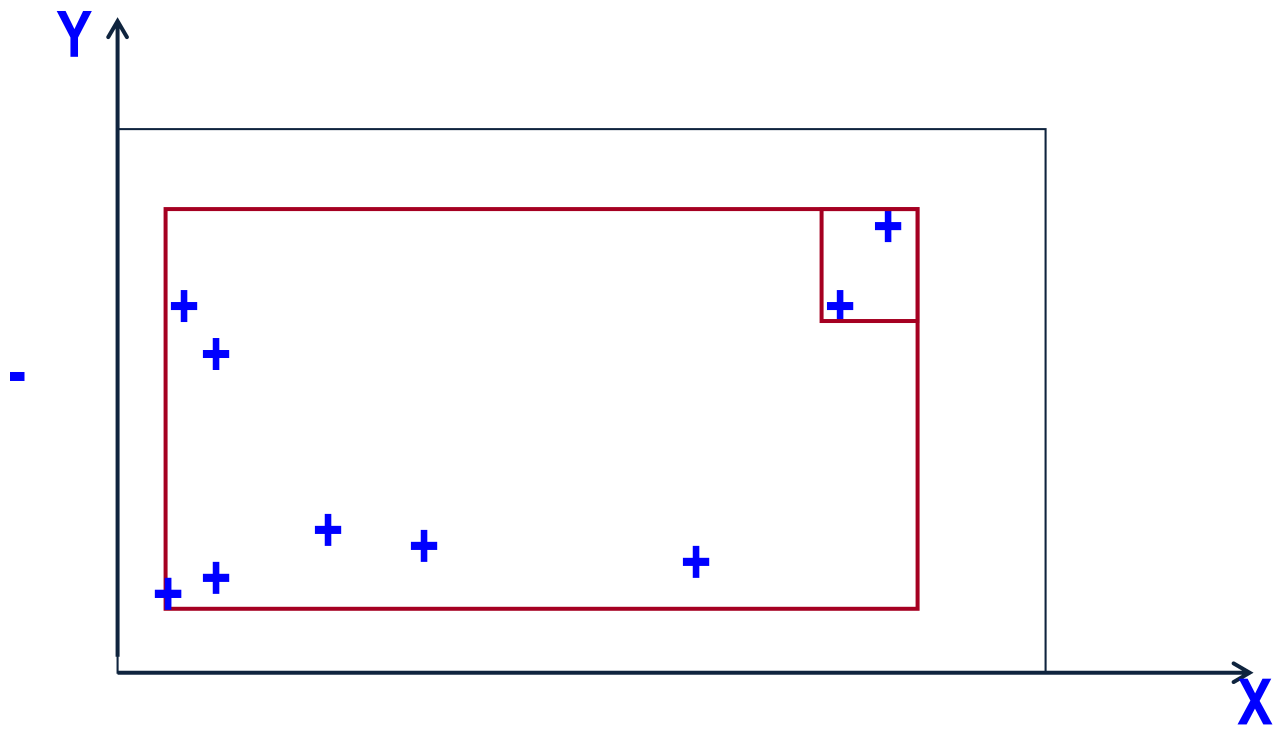
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



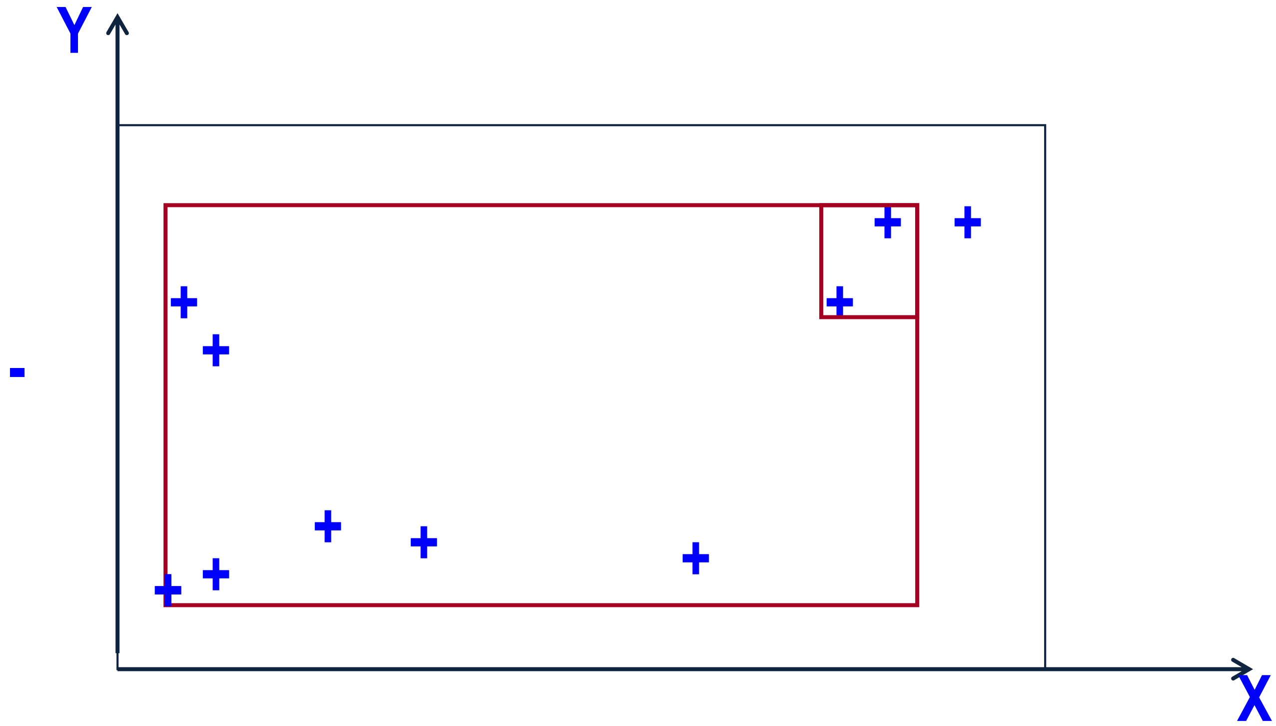
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



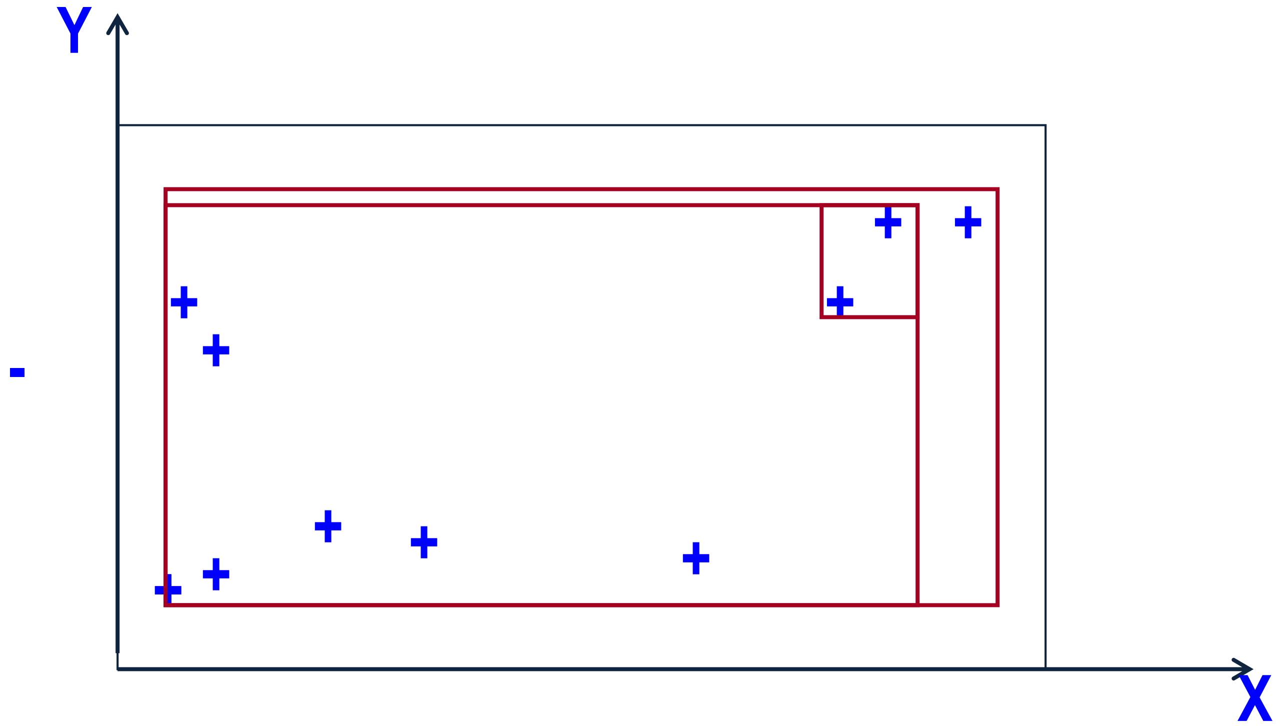
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



Learning Rectangles

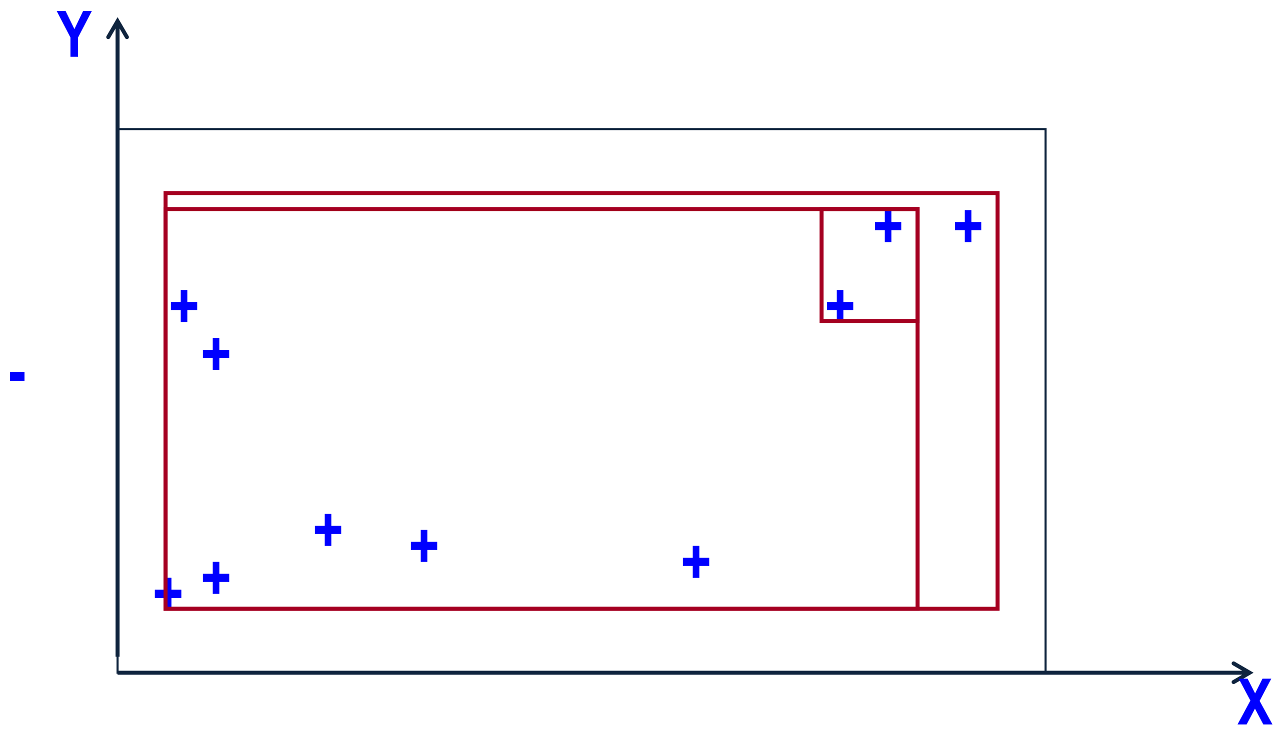
- Assume the target concept is an axis parallel rectangle



- Will we be able to learn the target rectangle ?

Learning Rectangles

- Assume the target concept is an axis parallel rectangle

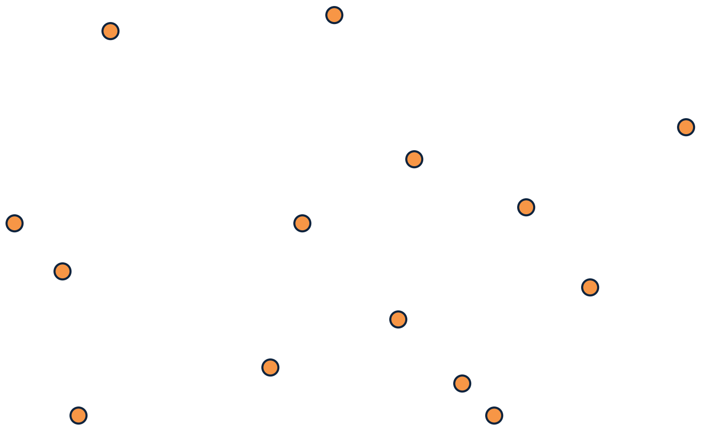


- Will we be able to learn the target rectangle ?
- Can we come close ?

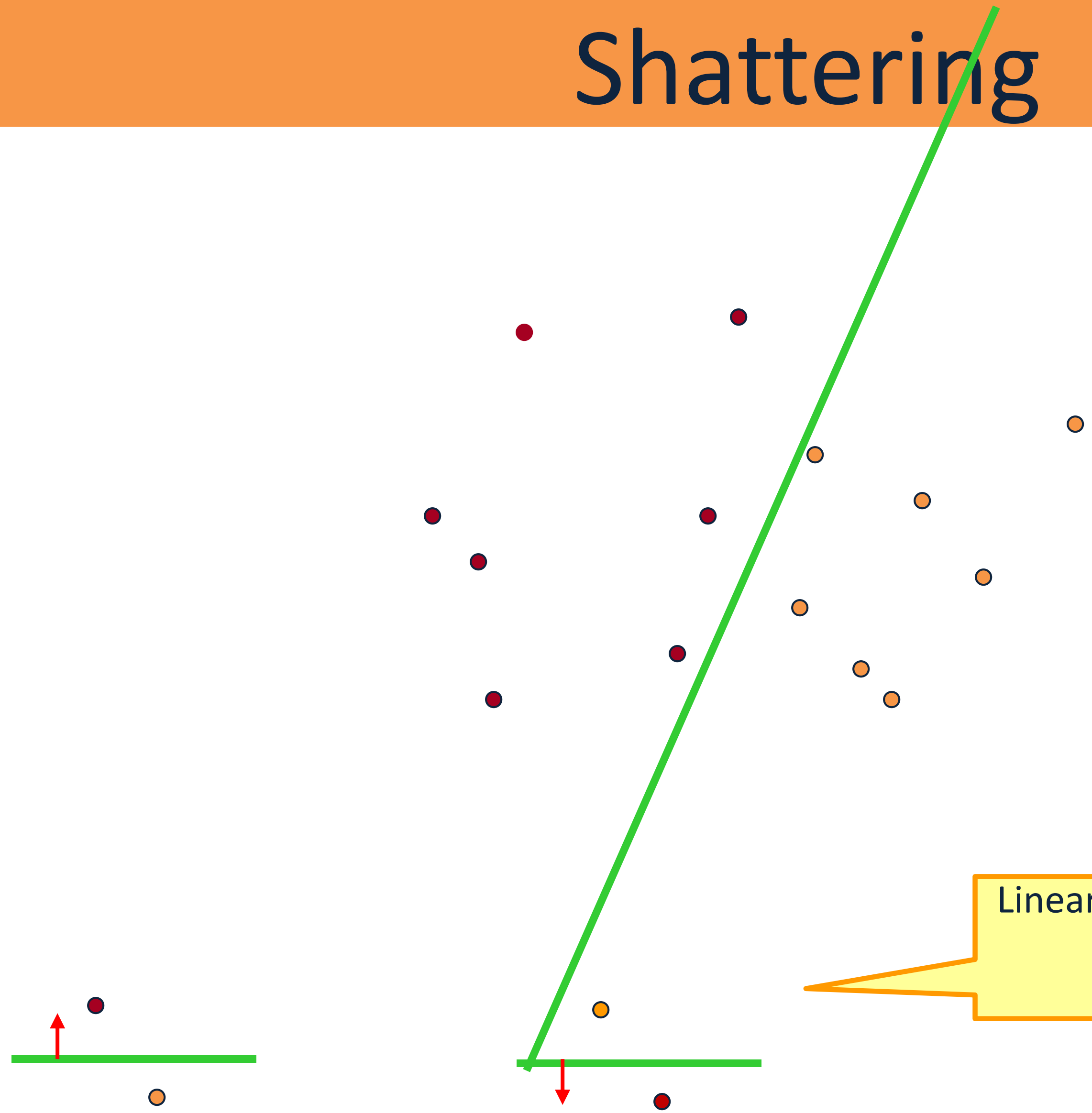
Infinite Hypothesis Space

- 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 conjunction 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.
- Analogous to $|H|$, there are bounds for sample complexity using VC(H)

Shattering

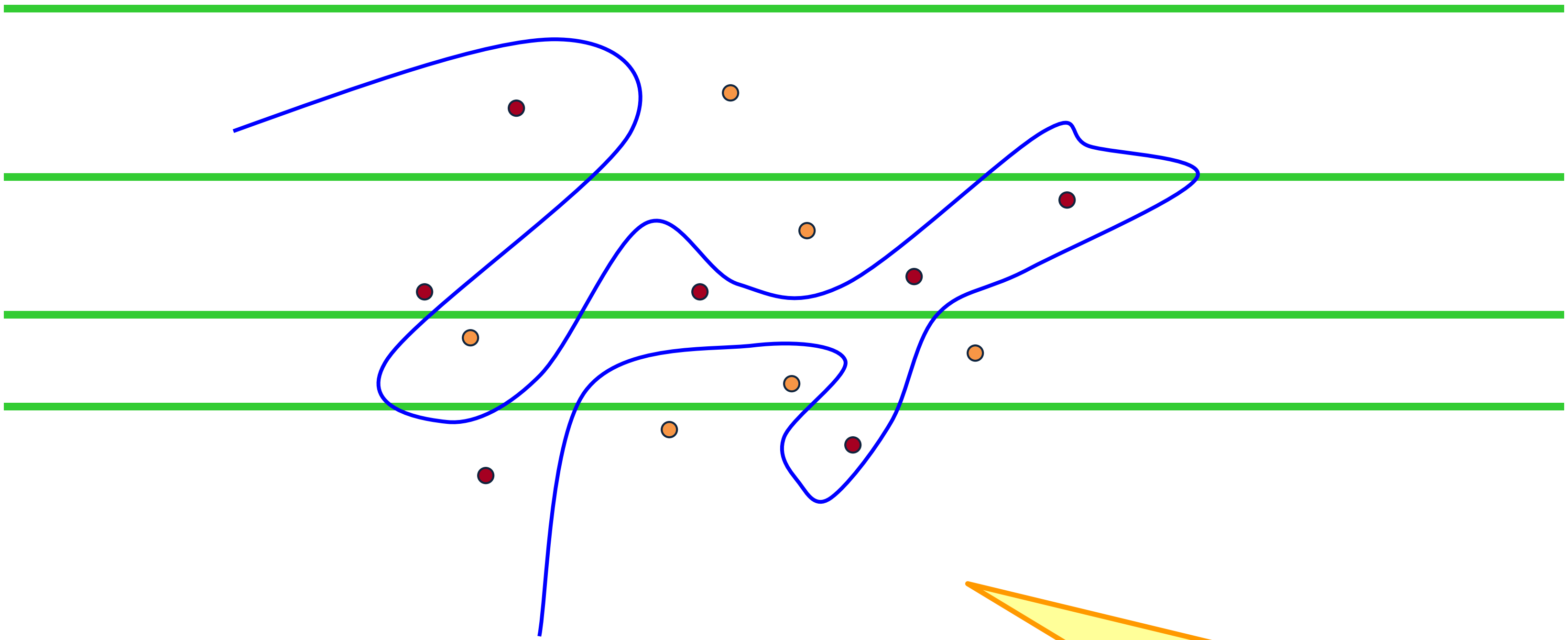


Shattering



Linear functions are expressive enough to shatter 2 points
(4 options; not all shown)

Shattering



Linear functions are not expressive enough to shatter 13 points

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 rich set of functions shatters large sets of points)

Shattering

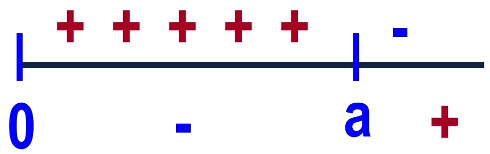
- We say that a set S of examples is shattered by a set of functions H if for every partition of the examples in S into positive and negative examples there is a function in H that gives exactly these labels to the examples (Intuition: A rich set of functions shatters large sets of points)

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

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 rich set of functions shatters large sets of points)

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



Sets of two points cannot be shattered
(we mean: given two points, you can label them in such a way that no concept in this class will be consistent with their labeling)

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

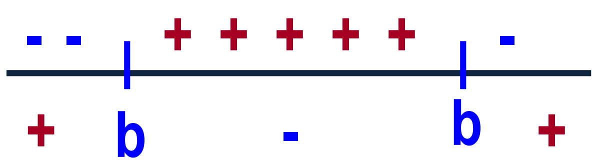
This is the set of functions (concept class) considered here

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

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

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

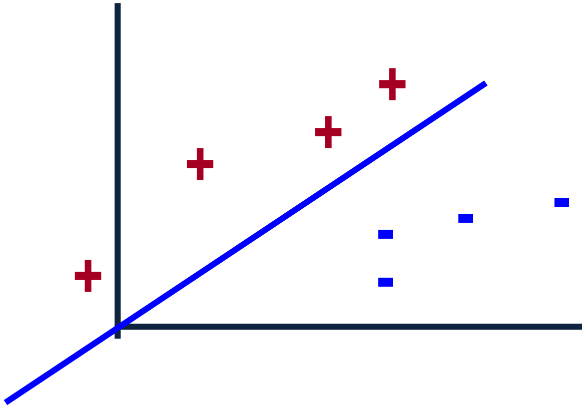


All sets of one or two points can be shattered
but sets of three points cannot be shattered

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

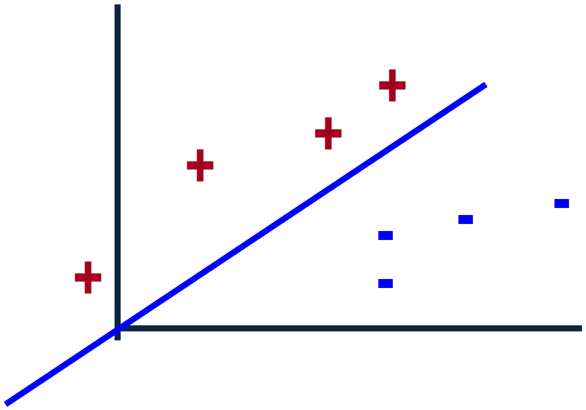
Half-spaces in the plane:



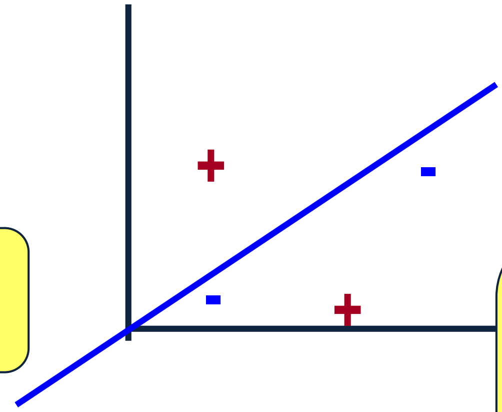
Shattering

- 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:



All sets of three?



1. If the 4 points form a convex polygon… (if not?)
2. If one point is inside the convex hull defined by the other three… (if not?)

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

VC Dimension

- An unbiased hypothesis space H shatters the entire instance space X, i.e, it is able to 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.

VC Dimension

- 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

- 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)

Some are shattered, but some are not

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
- If
- $$m > \frac{1}{\varepsilon}\{8VC(H)\log\frac{13}{\varepsilon} + 4\log(\frac{2}{\delta})\}$$
- Then with probability at least $(1-\delta)$, h has error less than ε.

What if H is finite?

(that is, if m is polynomial we have a PAC learning algorithm; to be efficient, we need to produce the hypothesis h efficiently.

- Notice that to shatter m examples it must be that: $|H|>2^m$, so $\log(|H|)\geq VC(H)$

Administration

Questions

- Hw4 is out.
  - Due on March 11 (Saturday)
  - No slack time since we want to release the solutions with enough time before the midterm.
  - You cannot solve all the problems yet.
- Quizzes:
  - Quiz 5 is done.
  - Quiz 6 will be due before next Tuesday
- Midterm is coming in three weeks
  - 3/16, in class
- Project Proposals are due on 3/10.
  - Follow Piazza and the web site.

Learning Rectangles

- Consider axis parallel rectangles in the real plan
- Can we PAC learn it ?

Learning Rectangles

- Consider axis parallel rectangles in the real plan
- Can we PAC learn it ?

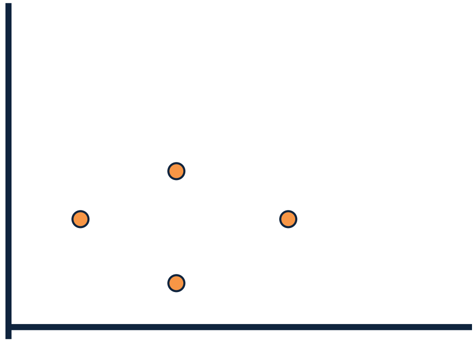
  (1) What is the VC dimension ?

Learning Rectangles

- Consider axis parallel rectangles in the real plan
- Can we PAC learn it ?

  (1) What is the VC dimension ?

- Some four instance can be shattered



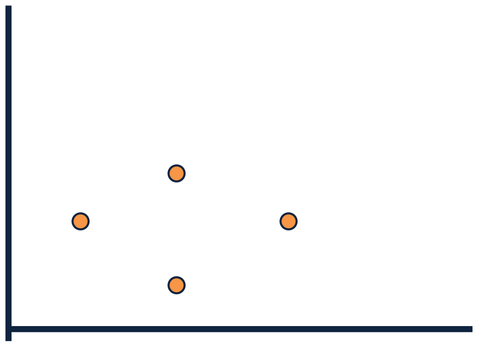
(need to consider here 16 different rectangles) Shows that VC(H)>=4

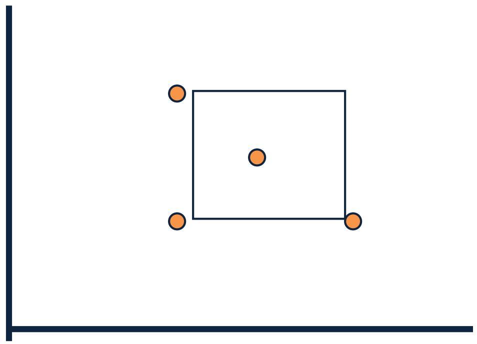
Learning Rectangles

- Consider axis parallel rectangles in the real plan
- Can we PAC learn it ?

  (1) What is the VC dimension ?

- Some four instance can be shattered and some cannot





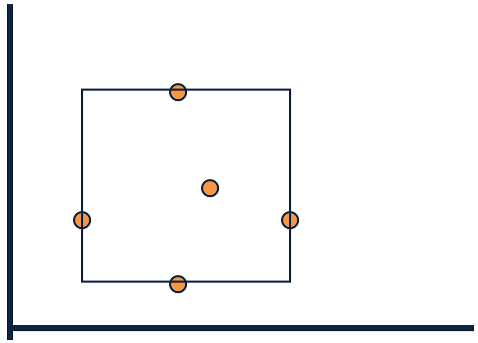
(need to consider here 16 different rectangles) Shows that VC(H)>=4

Learning Rectangles

- Consider axis parallel rectangles in the real plan
- Can we PAC learn it ?

  (1) What is the VC dimension ?

- But, no five instances can be shattered

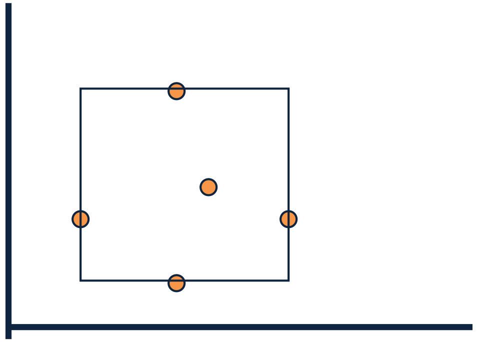


Learning Rectangles

- Consider axis parallel rectangles in the real plan
- Can we PAC learn it ?

  (1) What is the VC dimension ?

- But, no five instances can be shattered



There can be at most 4 distinct extreme points (smallest or largest along some dimension) and these cannot be included (labeled +) without including the 5th point.

Therefore VC(H) = 4

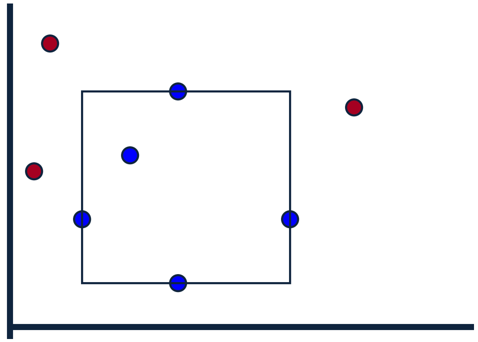
As far as sample complexity, this guarantees PAC learnabilty.

Learning Rectangles

- Consider axis parallel rectangles in the real plan
- Can we PAC learn it ?

  (1) What is the VC dimension ?

  (2) Can we give an efficient algorithm ?

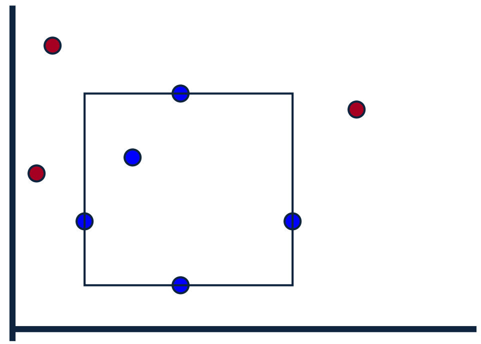


Learning Rectangles

- Consider axis parallel rectangles in the real plan
- 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, and the hypothesis is consistent.

Axis parallel rectangles are efficiently PAC learnable.

Sample Complexity Lower Bound

- There is also a general lower bound on the minimum number of examples necessary for PAC leaning in the general case.

- Consider any concept class C such that VC(C)>2, any learner L and small enough ε, δ.
  Then, there exists a distribution *D* and a target function in C such that if L observes less than

$$m = \max\left[\frac{1}{\varepsilon}\log\left(\frac{1}{\delta}\right), \frac{VC(C)-1}{32\varepsilon}\right]$$

  examples, then with probability at least δ,
  L outputs a hypothesis having error(h) > ε .

Ignoring constant factors, the lower bound is the same as the upper bound, except for the extra log(1/ε) factor in the upper bound.

COLT Conclusions

- The PAC framework provides a reasonable model for theoretically analyzing the effectiveness of learning algorithms.
- The sample complexity for any consistent learner using the hypothesis space, H, can be determined from a measure of H's expressiveness (|H|, VC(H))
- If the sample complexity is tractable, then the computational complexity of finding a consistent hypothesis governs the complexity of the problem.
- Sample complexity bounds given here are far from being tight, but separate learnable classes from non-learnable classes (and show what's important).
- Computational complexity results exhibit cases where information theoretic learning is feasible, but finding good hypothesis is intractable.
- The theoretical framework allows for a concrete analysis of the complexity of learning as a function of various assumptions (e.g., relevant variables)

COLT Conclusions (2)

- Many additional models have been studied as extensions of the basic one:
  - Learning with noisy data
  - Learning under specific distributions
  - Learning probabilistic representations
  - Learning neural networks
  - Learning finite automata
  - Active Learning; Learning with Queries
  - Models of Teaching
- An important extension: PAC-Bayesians theory.
  - In addition to the Distribution Free assumption of PAC, makes also an assumption of a prior distribution over the hypothesis the learner can choose from.

COLT Conclusions (3)

- Theoretical results shed light on important issues such as the importance of the bias (representation), sample and computational complexity, importance of interaction, etc.
- Bounds guide model selection even when not practical.
- A lot of recent work is on data dependent bounds.
- The impact COLT has had on practical learning system in the last few years has been very significant:
  - SVMs;
  - Winnow (Sparsity),
  - Boosting
  - Regularization