Computational Learning Theory

CS534 - Machine Learning

Introduction

- Computational learning theory
  - Provides a theoretical analysis of learning
  - Shows when a learning algorithm can be expected to succeed
  - Shows when learning may be impossible
- Three primary questions include
  - **Sample Complexity:** How many examples do we need to find a good hypothesis?
  - **Computational Complexity:** How much computational power do we need to find a good hypothesis?
  - **Mistake Bound:** How many mistakes we will make before finding a good hypothesis?

Framework for Noise Free Learning

- **Assumptions for the noise-free case:**
  - Data is generated according to an unknown probability distribution D(**x**)
  - Data is labeled according to an unknown function f: y = f(**x**)
    (f is often referred to as the **target concept**)
  - Our hypothesis space *H* contains the target concept

Consistent-Learn

<u>Input</u>: access to a training example generator
sample size *m*
hypothesis space *H*

1) Draw a set *E* of *m* training examples.
   (drawn from the unknown distribution and labeled by the unknown target)
2) Find an $h \in H$ that agrees with all training examples in *E*.

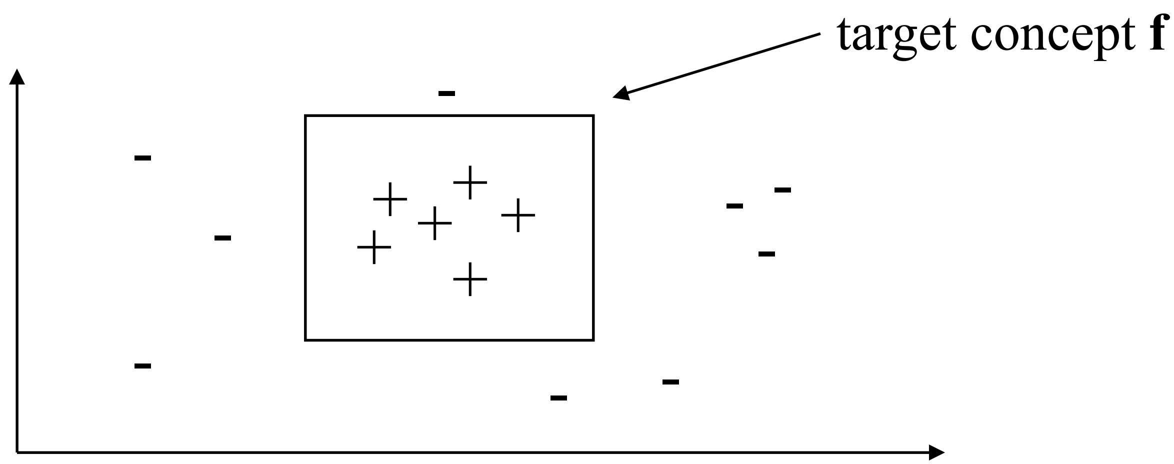
<u>Output</u>: *h*

- How good is the consistency learning algorithm?
  - **Training error** of h is zero, what about error on new examples drawn from the <u>same distribution</u> D (i.e. the **generalization error**)?

Example: Axis-Parallel Rectangles

Let's start with a simple problem:

Assume a two dimensional **input space** (R^2) with positive and negative **training examples**. Assume that the target function is some rectangle that separates the positive examples from the negatives. Instances inside the rectangle are positive and outside are negative.

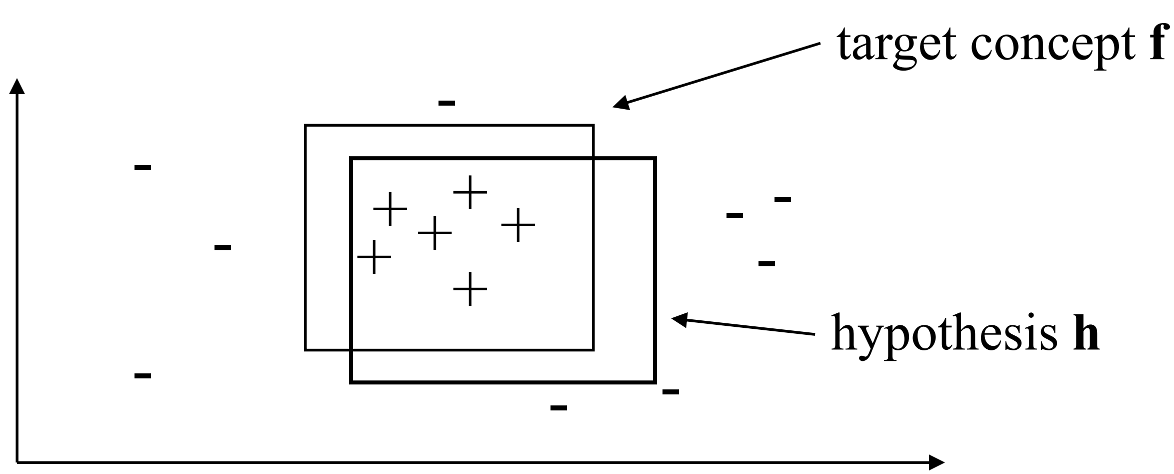


Examples drawn according to unknown distribution *D*.

Example: Hypothesis Generation

Consistent-Learn finds a hypothesis that is consistent with the training exmaples

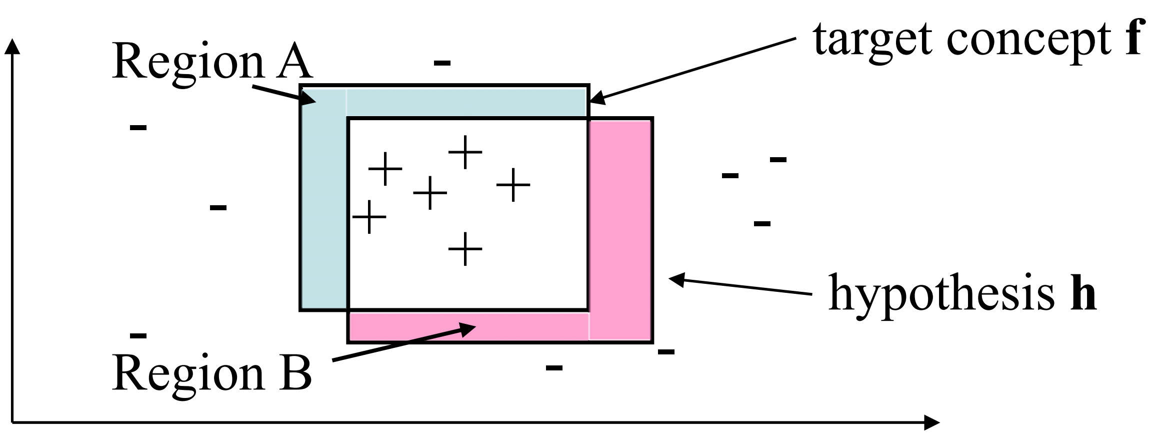
Note that it will generally not be the same as *f.* Here there are an infinite number of consistent hypotheses.



Example: Generalization Error

- The **generalization error** of a hypothesis h is the probability that h will make a mistaken on a new example randomly drawn from D

$$error(h, f) = P_D[f(\mathbf{x}) \neq h(\mathbf{x})]$$



True Error: the probability of regions A and B.
A : false negatives; B : false positives

The generalization error is the sum of probability of regions A and B

“Realistic Expectations” for Learning

- **Generalization Error:**
  - Many possible target functions and a small set of training examples
  - Therefore, we can’t expect algorithm to achieve zero generalization error
  - Rather we will be satisfied with an ***approximately correct*** hypothesis. That is, an h that has a small generalization error ε (that we specify)
- **Reliability:**
  - This is a non-zero probability that we draw a training set that's not representative of the target (e.g. in the worst case there is a non-zero probability that the training set contains a single repeated example).
  - Therefore, we can’t expect the algorithm to return an ε-good hypothesis every time it is run. Sometimes it will fail.
  - Rather we will be satisfied with if the algorithm returns an ε-good hypothesis with high prob. That is, the prob. that the algorithm fails is less than some threshold δ (that we specify)
- **Question:** How many training examples are required such that the algorithm is ***probably, approximately correct*** (PAC)?
  - That is, with probability at least 1- δ, the algorithm returns a hypothesis with generalization error less than ε (ε-good)
  - E.g. return a hypothesis with accuracy at least 95% ($\varepsilon = 0.05$) at least 99% ($\delta = 0.01$) of the time.

Case 1: Finite Hypothesis Space

- Assume our hypothesis space H is finite - start with the simple case
- Consider a hypotheiss $h_1 \in$ H and its error > ε (ε-bad).
  - We would like to bound the probability that *the consistency learning algorithm* fails by returning such a hypothesis.
  - What is the probability that h_1 will be consistent with *m* training examples drawn from distribution D?
- Let's start with one randomly drawn training example, what is the probability that h_1 will correctly classify it?

$$P_D\,[h_1(\mathbf{x}) = y] \leq (1 - \varepsilon) \quad [\textit{given that its error} > \varepsilon]$$

- What is the probability that *h* will be consistent with *m* examples drawn ***independently*** from D?

$$P^m{}_D[h_1(\mathbf{x}_1) = y_1,\ldots,\, h_1(\mathbf{x}_m) = y_m]$$
$$= P_D[h_1(\mathbf{x}_1) = y_1]*\ldots*P_D[h_1(\mathbf{x}_m) = y_m]$$
$$\leq (1 - \varepsilon)^m$$

Finite Hypothesis Spaces (2)

Abbreviate $P^m_D[h(\mathbf{x}_1) = y_1, \ldots, h(\mathbf{x}_m) = y_m] = P^m_D[h \text{ survives}]$

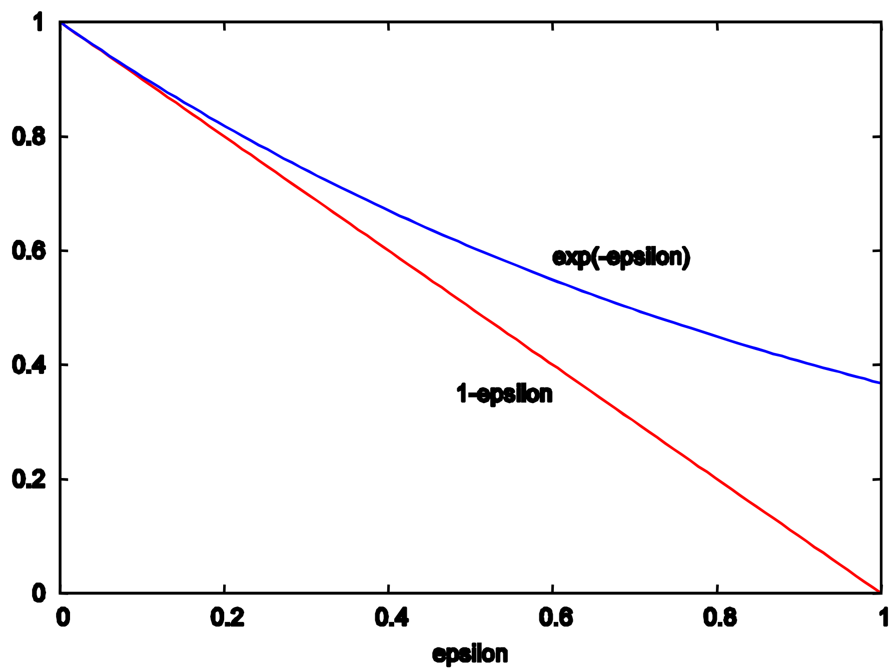
- Now consider a second hypothesis h_2 that is also ε-bad. What is the probability that <u>either</u> h_1 or h_2 will survive the m training examples?

$$
\begin{aligned}
&P^m_D[h_1 \text{ survives} \vee h_2 \text{ survives}] \\
&\quad = P^m_D[h_1 \text{ survives}] + P^m_D[h_2 \text{ survives}] - P^m_D[h_1 \wedge h_2 \text{ survives}] \\
&\quad \leq P^m_D[h_1 \text{ survives}] + P^m_D[h_2 \text{ survives}], \quad \text{(the union bound)} \\
&\quad \leq 2(1-\varepsilon)^m
\end{aligned}
$$

- Suppose the hypothesis space contains k ε-bad hypotheses, the probability that any one of them will survive m training examples is $\leq k(1-\varepsilon)^m$
- Since $k \leq |H|$, this is $\leq |H|\,(1-\varepsilon)^m$

Finite Hypothesis Spaces (3)

- Fact: When $0 \le \varepsilon \le 1$, $(1 - \varepsilon) \le e^{-\varepsilon}$
  therefore $|H|(1 - \varepsilon)^m \le |H|\, e^{-\varepsilon m}$



Blumer Bound
(Blumer, Ehrenfeucht, Haussler, Warmuth)

- Thus we have shown the following lemma

- **Lemma** (Blumer Bound)
  For a finite hypothesis space H, given a set of *m* training examples drawn independently according to D, the probability that there exists an hypothesis *h* in H that has generalization error greater than ε and is cons istent with the training examples is less than $|H|e^{-\varepsilon m}$

- This implies that the probability that *Consistent-Learn* fails to return an ε accurate hypothesis given *m* examples is less than $|H|e^{-\varepsilon m}$

- Note that based on PAC learning requirement, we want this probability to be less than δ.

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

Sample Complexity Bound

- To ensure that *Consistent-Learn* outputs a good hypothesis (error < ε) with high (>1-δ) probability, a sufficient number of samples is:

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

- Key property: the sample complexity grows linearly in 1/ε and logarithmic in |H| and 1/δ

- **Corollary**: If $h \in$ H is consistent with all *m* examples drawn according to D, then with probability at least 1-δ the generalization error ε of *h* is no greater than

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

PAC Learnability

- Let C be a class of possible target concepts, e.g., all possible conjunctions over 10 boolean variables.

Definition (PAC learnability): A concept class C is PAC-learnable iff there exists an algorithm *Learn* such that, for any distribution D, for any target concept $c \in C$, for any $0<\delta<1$, and for any $0<\varepsilon<1$, with probability at least $(1 - \delta)$ *Learn* outputs a hypothesis $h \in C$, such that $error_D(h,c) \leq \varepsilon$, and *Learn* runs in time polynomial in $1/\varepsilon$, $1/\delta$, n, and size(c).

- Learn can draw training examples labeled by the unknown c and drawn from the unknown D (but only polynomially many).
- An adversary can pick D and c (so PAC learnability requires handling the worst case).

PAC Consistency Learning

PAC-Consistent-Learn

Input: ε, δ, and description of the concept class C

1) Draw a set E of $m \geq \frac{1}{\epsilon}\left(\ln|C| + \ln\frac{1}{\delta}\right)$ training examples. (these are labeled by the unknown target and drawn from the unknown distribution)
2) Find an $h \in C$ that agrees with all training examples in E.

Output: h

We can show that C is PAC-learnable via *PAC-Consistent-Learn* if

1) $ln|C|$ is polynomial so that we only need to draw a polynomial number of examples to meet PAC accuracy requirements
2) Step 2 must be polynomial in the size of E so that the computational complexity meets the PAC requirement

Examples

- Exa. 1: Conjunctions (allow negation) over n Boolean features.
  - Hypothesis space $|H|=3^n$: each features can appear positively, appear negatively, or not appear

$$m \geq \frac{1}{\epsilon}(n \ln 3 + \ln \frac{1}{\delta})$$

  - Furthermore one can find a consistent hypothesis efficiently (How?)
  - So the concept class of conjunctions is PAC learnable.

- Exp. 2: k-DNF formulas: unlimited number of disjunctions of k-term conjunctions, e.g., $(x_1 \wedge x_3) \vee (x_2 \wedge x_4) \vee (x_1 \wedge x_4)$
  - There are at most $(2n)^k$ distinct conjunctions, so

$$|H| = 2^{(2n)^k} \text{ and } \log|H| = O(n^k)$$

  - Finding a consistent k-DNF formula is an NP-hard problem.
  - So we can't use *Consist-Learn* to prove PAC-learnability of k-DNF

Examples

- Exp 3: Space of all boolean functions over *n* Boolean features.
  - There is a polynomial time algorithm for finding a consistent hypothesis in the size of the training set (How?)
  - The size of the hypothesis space is 2^{2^n}
  - So a sufficient number of examples is

    $$m \geq \frac{1}{\epsilon}(2^n + \ln \frac{1}{\delta})$$

    which is exponential.
  - So we can't use PAC-Consistent-Learn to show PAC learnability for this class due to the sample complexity

What we have seen so far

Finite hypothesis space with consistent hypothesis

1. We have shown that for the learner Consistent-Learn,
   - Sample complexity (number of training examples needed to ensure at least 1-δ prob. of finding a ε good hypothesis)

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

   - Given sample size m, with at least 1-δ prob. generalization error of the learned h is bounded by

     $$\varepsilon \leq \frac{1}{m}\left(\ln|H| + \ln\frac{1}{\delta}\right).$$

What if there is no consistent hypothesis?

- Suppose our learner outputs an *h* which has training error ε_T>0, what can we say about its generalization error?
  - We will use the Hoeffding bound for this purpose.

Additive Hoeffding Bound

- Let Z be a binary random variable with P(Z=1) = p
- An let z_i i=1,…,m be i.i.d. samples of Z
- The Hoeffding bound gives a bound on the probability that the average of the z_i are far from E[Z]=1*p + 0*(1-p) = p

Let {z_i | i=1,…, m} be i.i.d. samples of binary random variable Z, with P(Z=1) = p, then for any $\gamma \in [0,1]$

$$P\left(p - \frac{1}{m}\sum z_i > \gamma\right) \leq \exp(-2\gamma^2 m)$$
$$P\left(p - \frac{1}{m}\sum z_i < -\gamma\right) \leq \exp(-2\gamma^2 m)$$

Hoeffding Bound for Generalization Error

- Suppose an *h* has training error $\varepsilon_T>0$, what can we say about its generalization error?
- Let Z be a Bernoulli random variable defined as follows:
  - Draw a sample **x** from *D*, *Z=1* if h(**x**) ≠ f(**x**), *Z=0* otherwise
- The training error of h is simply

$$\epsilon_T = \frac{1}{m}\sum_{i=1}^{m} Z_i$$

  i.e., the observed frequency of *Z=1*
- The true error of h is simply $\varepsilon = P(Z=1)$,
- From the *Hoeffding bounds*: $P(\varepsilon - \varepsilon_T > \gamma) \leq \exp(-2\gamma^2 m)$
- As the training set grows the probability that the training error underestimates the generalization error decreases exponentially fast

Error Bound: Inconsistent Hypothesis

- Thus for a random *h* in *H*, if the training error of *h* is ε_T, then the probability that its true generalization error ε is larger than ε_T by a large margin ($>\gamma$) is bounded by:

$$P(\varepsilon - \varepsilon_T > \gamma) \leq \exp(-2\gamma^2 m)$$

- Now we would like to bound this for all *h*'s in *H* simultaneously
  - That is, want to guarantee for any learned *h* that generalization error $\varepsilon(h)$ is close to the training error $\varepsilon_T(h)$

$$P(\exists\, h \in H:\ \varepsilon(h) - \varepsilon_T(h) > \gamma)$$

$$= P((\varepsilon(h_1) - \varepsilon_T(h_1) > \gamma) \vee \cdots \vee (\varepsilon(h_k) - \varepsilon_T(h_k) > \gamma))$$

$$\leq \sum_{i=1\cdots|H|} P(\varepsilon(h_i) - \varepsilon_T(h_i) > \gamma) = |H| \exp(-2\gamma^2 m)$$

- Now suppose we bound this probability by δ, and that we have m samples, it is thus guaranteed with probability at 1-δ that for all h in H:

$$\varepsilon(h) < \varepsilon_T(h) + \gamma = \varepsilon_T(h) + \sqrt{\frac{1}{2m} \log \frac{|H|}{\delta}}$$

Best Possible Hypothesis in H

- **Theorem:** Consider a learner that always outputs *h* to minimize training error, i.e., $h_L = \text{argmin}_{h\ \text{H}}\varepsilon_\text{T}(h)$. Let m be the size of the training set, with probability 1-δ, we have

$$\varepsilon(h_L) \le \varepsilon(h^*) + 2\sqrt{\frac{1}{2m} \ \ln\frac{|H|}{\delta}}$$

$$h_L = \arg\min_{h\in H} \varepsilon_T(h), \qquad h^* = \arg\min_{h\in H} \varepsilon(h)$$

- **Interpretation:** by selecting h_L, we are not too much worse off than the best possible hypothesis h^*.
  - The difference gets smaller as we increases sample size

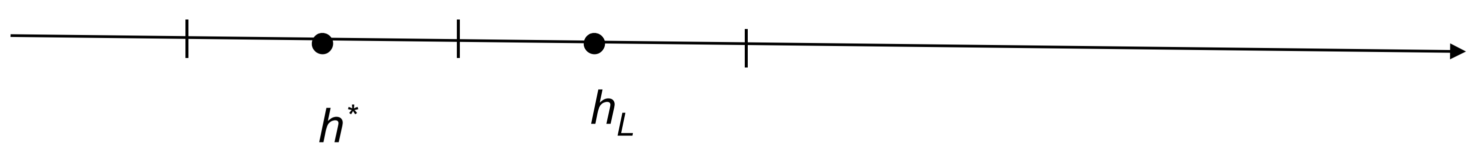
Proof

$$\varepsilon(h_L) \le \varepsilon_T(h_L) + \sqrt{\frac{1}{2m} \ln\frac{|H|}{\delta}}$$

$$\text{because } h_L = \arg\min_{h\in H} \varepsilon_T(h)$$

$$\varepsilon_T(h_L) \le \varepsilon_T(h^*) \le \varepsilon(h^*) + \sqrt{\frac{1}{2m} \ln\frac{|H|}{\delta}}$$

$$\varepsilon(h_L) \le \varepsilon(h^*) + 2\sqrt{\frac{1}{2m} \ln\frac{|H|}{\delta}}$$



Interpretation

$$\varepsilon(h_L) \leq \varepsilon(h^*) + 2\sqrt{\frac{1}{2m} \ln\frac{|H|}{\delta}}$$

True error of h_L

Best Possibe Error in H

Penalty for Size of H

Fundamental tradeoff in selecting Hypothesis space

- Bigger hypothesis space causes the 1st term to decrease (this is sometimes called the “bias” of H)
- However, as |H| increases, the second term increases (this is related to the “variance” of the learning algorithm)

What we have seen so far

Finite hypothesis space

1. Learner always find a consistent hypothesis
   - Sample complexity (number of training examples needed to ensure at least 1-δ prob. of finding a ε good hypothesis)

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

   - Given sample size m, with at least 1-δ prob. generalization error of the learned h_L is bounded by

     $$\varepsilon(h_L) \leq \frac{1}{m}\left(\ln|H| + \ln\frac{1}{\delta}\right)$$

2. Learner finds the hypothesis that minimizes training error

$$\varepsilon(h) \leq \varepsilon_T(h) + \sqrt{\frac{1}{2m} \ln\frac{|H|}{\delta}} \qquad \varepsilon(h_L) \leq \min_{h\in H} \varepsilon(h) + 2\sqrt{\frac{1}{2m} \ln\frac{|H|}{\delta}}$$

What about Infinite Hypothesis Space

- Most of our classifiers (LTUs, neural networks, SVMs) have continuous parameters and therefore, have infinite hypothesis spaces

- For some, despite their infinite size, they have limited expressive power, so we should be able to prove something

- We need to characterize the learner's ability to model complex concepts
  - For finite spaces the complexity of a hypothesis space was characterized roughly by $\ln|H|$
  - Instead for infinite spaces we will characterize a hypothesis space by its VC-dimension

Definition: VC-dimension

- Consider S = {$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$}, a set of *m* points in the input space, a hypothesis space H is said to <u>shatter</u> S if for every possible way of labeling the points in S, there exists an *h* in H that gives this labeling.

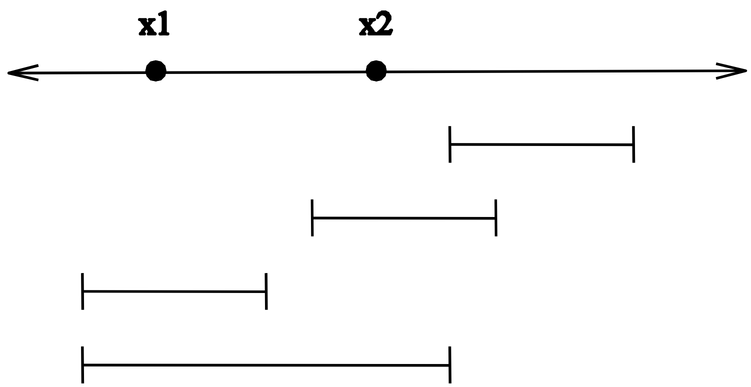
You can view this as a game, you choose a set of points (their locations), and your opponent chooses the labels, you need to be able to find an h to correctly label the points.

- Definition: The <u>Vapnik-Chervonenkis</u> dimension (**VC-dimension**) of an hypothesis space H is the size of the largest set S that can be shattered by H.
  - A hypothesis space can "trivially fit" S if it can shatter S

As long as you can find ONE set of points with size m that can be shattered by H, we have that VC(H)>=m
It does not matter if there exist other size m sets that can not be shattered by H.

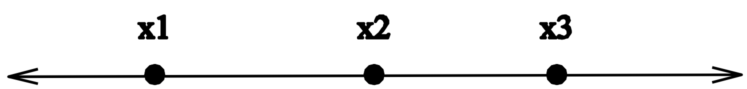
VC-dimension Example: 1D Intervals

- Let H be the set of intervals on the real line such that $h(\mathbf{x}) = 1$ iff $\mathbf{x}$ is in the interval. H can shatter any pair of examples:



x1	x2
0	0
0	1
1	0
1	1

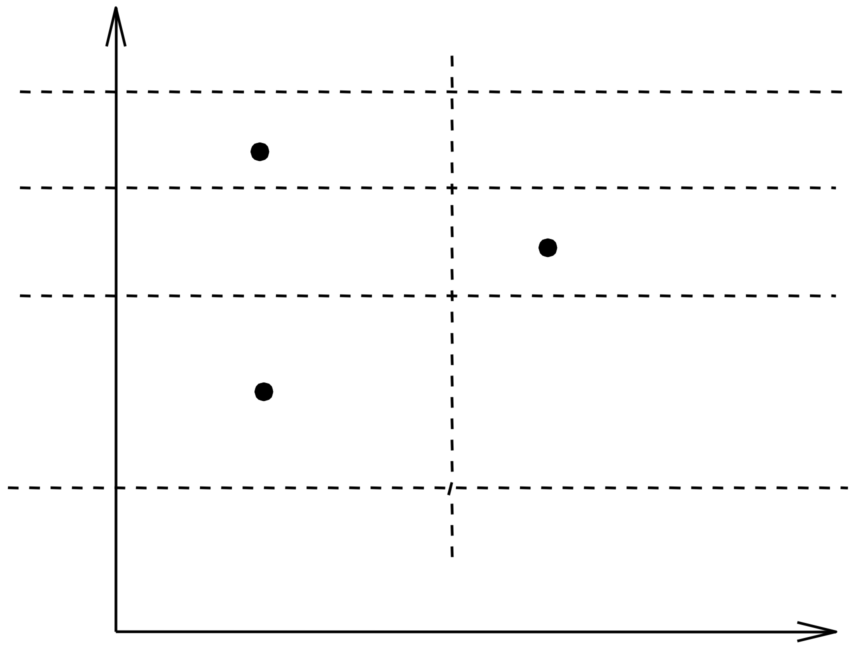
- However, H can not shatter any set of three examples. Therefore the VC-dimension of H is 2



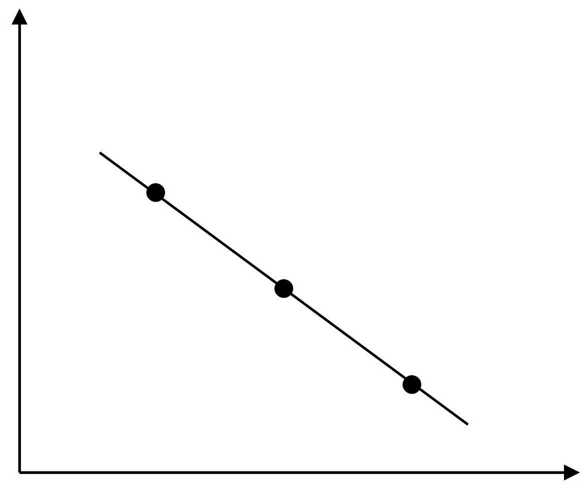
x1	x2	x3
1	0	1

VC-dimension: Linear Separators

- Let H be the space of linear separators in the 2-D plane.



These 3 points can be shattered

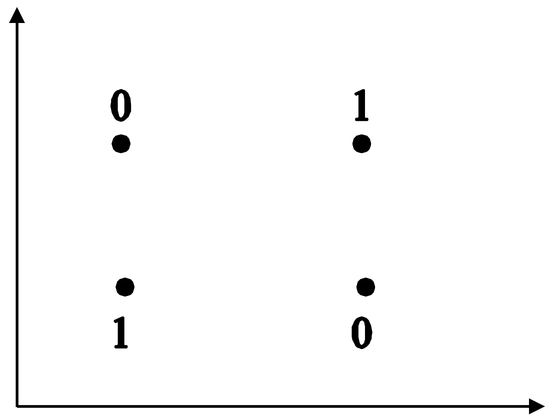


These 3 points can not be shattered

However since there is at least one set of size three that can be shattered, we have that VC(H) $\geq$ 3

VC-dimension: Linear Separators

- We can not shatter any set of 4 points



XOR function

- The VC-dimension of linear separators in 2-d space is 3.
- In general, the VC-dimension for linear separators in *n*-dimensional space can be shown to be *n+1*.
- A good initial guess is often that the VC-dimension is equal to the number of tunable parameters in the model (unless the parameters are redundant)

Property of VC dimension

$$\mathbf{VC(H) \leq \log_2 |H|}$$

- For set of *m* points, there are 2^m distinct ways to label them

- Thus for H to shatter the set we must have $|H| \geq 2^m$ which implies the bound

Bounds for *Consistent* Hypotheses

- The following bound is analogous to the Blumer bound but more complicated to prove
- If *h* (in H) is a hypothesis consistent with a training set of size m, and VC(H) = *d*, then with probability at least 1 – δ, *h* will have an error rate less than ε if

$$m \geq \frac{1}{\epsilon}\left(4\log_2\frac{2}{\delta} + 8d\log_2\frac{13}{\epsilon}\right)$$

- Compared to the **previous result** using *ln|H|*,

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

this bound has some extra constants and an extra $log_2(1/\varepsilon)$ factor. Since $VC(H) \leq log_2(H)$, this may be a tighter upper bound on the number of examples sufficient for PAC learning.

Bound for *Inconsistent* Hypotheses

- **Theorem**. Suppose VC(H)=*d* and a learning algorithm finds *h* with error rate ε_T on a training set of size *m*. Then with probability 1 – δ, the true error rate ε of *h* is

$$\varepsilon \le \varepsilon_T + \sqrt{\frac{d(\log\frac{2m}{d}+1)+\log\frac{4}{\delta}}{m}}$$

- **Empirical Risk Minimization Principle**
  - If you have a fixed hypothesis space H, then your learning algorithm should minimize ε_T: the error on the training data. (ε_T is also called the "empirical risk")