CS 6375: Machine Learning Computational Learning Theory

Vibhav Gogate
The University of Texas at Dallas
Many slides borrowed from Ray Mooney

Learning Theory

- Theoretical characterizations of
  - Difficulty of several types of ML problems
  - Capabilities of several types of ML algorithms
- Questions:
  - Under what conditions is successful ML possible and impossible?
  - Under what conditions will a particular ML algorithm perform successfully?

Learning Theory

- Theorems that characterize
  - classes of learning problems or
  - specific algorithms in terms of
    - computational complexity or
    - ***sample complexity***, i.e. the number of training examples necessary or sufficient to learn hypotheses of a given accuracy.
- Complexity of a learning problem depends on:
  - Size or expressiveness of the hypothesis space.
  - Accuracy to which a target concept must be approximated.
  - Probability with which the learner must produce a successful hypothesis.
  - Manner in which training examples are presented, e.g. randomly or by query to an oracle.

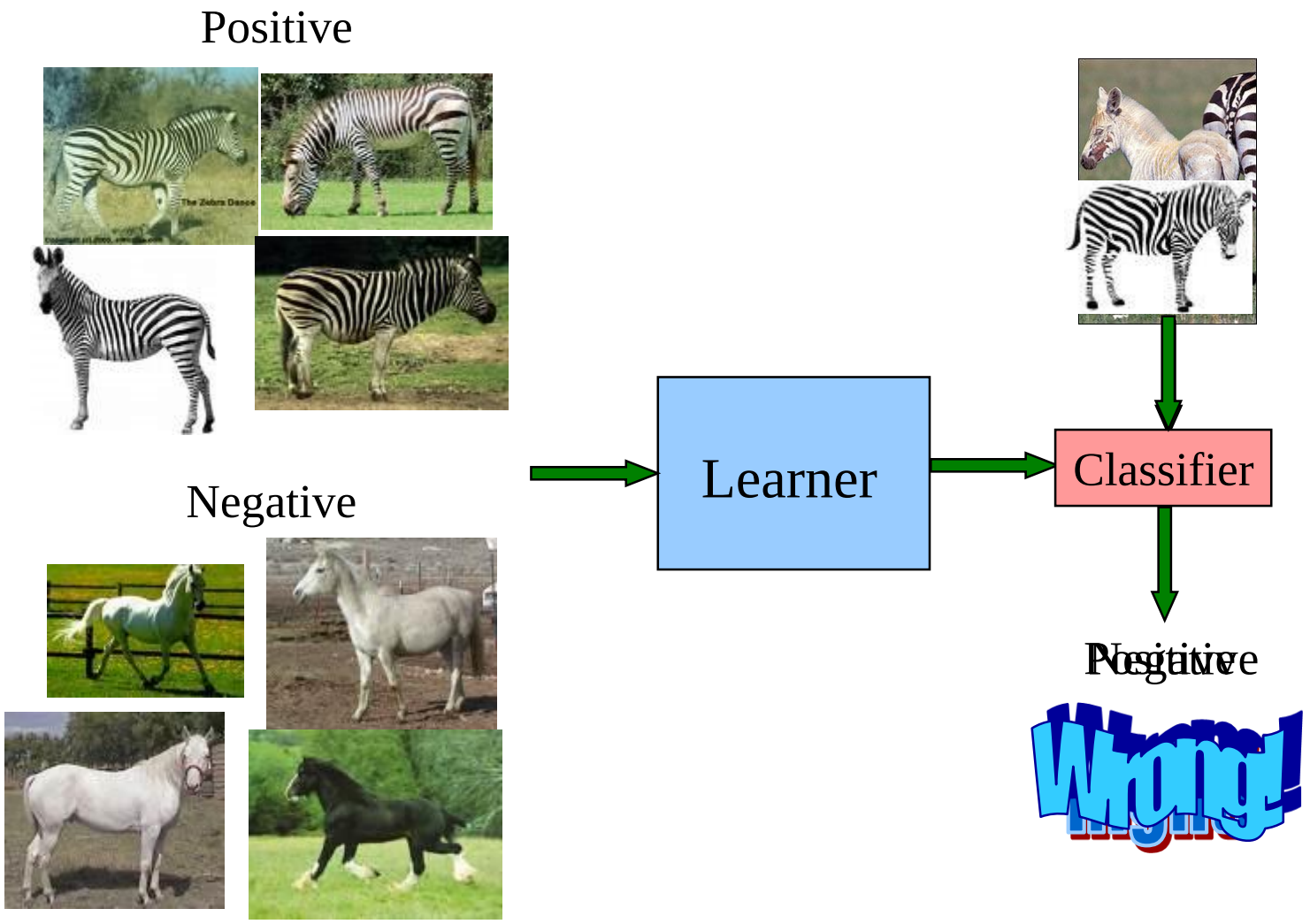
Types of Results

- We will not focus on specific learning algorithms but rather on broad classes of ML algorithms characterized by their hypothesis spaces, the presentation of training examples, etc.
- **Sample Complexity**: How many training examples are needed for a learner to construct (with high probability) a highly accurate concept?
- **Computational Complexity**: How much computational resources (time and space) are needed for a learner to construct (with high probability) a highly accurate concept?
  - High sample complexity implies high computational complexity, since learner at least needs to read the input data.
- **Mistake Bound**: Learning incrementally, how many training examples will the learner misclassify before constructing a highly accurate concept. (not covered)

Is Perfect Learning possible?

- Number of training examples needed to learn a hypothesis h for which error(h)=0
- Futile:
  - There may be multiple hypotheses that are consistent with the training data and the learner cannot be certain to pick the one that equals the target concept
  - Since training data is drawn randomly, there is always a chance that the training examples are misleading!

Cannot Learn Exact Concepts from Limited Data, Only Approximations



Cannot Learn Even Approximate Concepts from Pathological Training Sets

Positive

Negative

Learner

Classifier

Negative

Wrong!

PAC Learning

- Probably approximately correct (PAC)
  - Developed by Leslie Valiant (who later won the Turing award)
- The only reasonable expectation of a learner is that with ***high probability*** it learns a ***close approximation*** to the target concept.
- In the PAC model, we specify two small parameters, ε and δ, and require that with probability at least $(1 - \delta)$ a system learn a concept with error at most ε.

Problem Setting

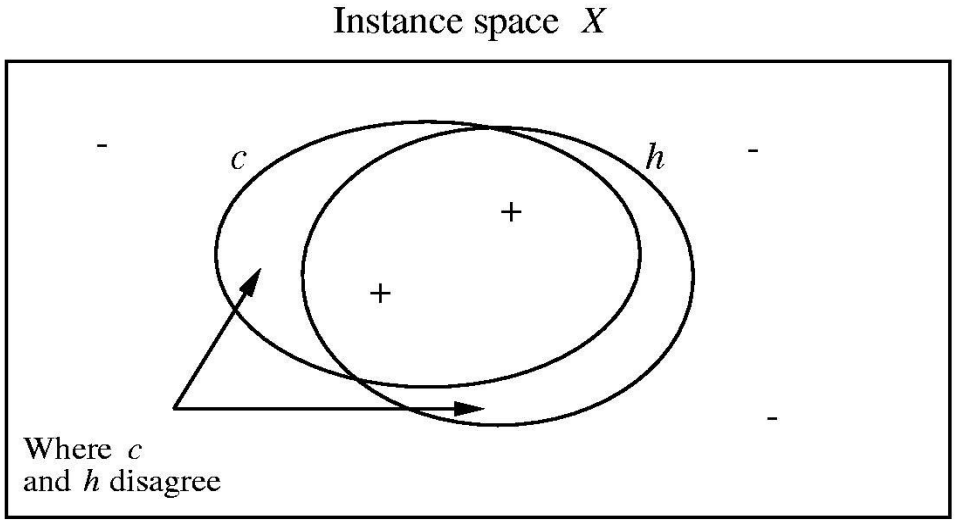
Given:

- Set of instances X
- Set of hypotheses H
- Set of possible target concepts C
- Training instances generated by a fixed, unknown probability distribution $\mathcal{D}$ over X

Learner observes sequence D of training examples $\langle x, c(x)\rangle$, for some target concept $c \in C$

- Instances x are drawn from distribution $\mathcal{D}$
- Teacher provides target value $c(x)$ for each

True Error of a Hypothesis



Definition: The **true error** (denoted $error_{\mathcal{D}}(h)$) of hypothesis h with respect to target concept c and distribution $\mathcal{D}$ is the probability that h will misclassify an instance drawn at random according to $\mathcal{D}$.

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$$

Two Notions of Error

Training error of hypothesis h with respect to target concept c

- How often $h(x) \neq c(x)$ over training instances

True error of hypothesis h with respect to c

- How often $h(x) \neq c(x)$ over future random instances

Our concern:

- Can we bound the true error of h given the training error of h?

Formal Definition of PAC

Given:

- A concept class C over an instance space X containing instances of length n
- A learner L, using a hypothesis space H.
- Two constants: $0 < \epsilon < 0.5$, $0 < \delta < 0.5$

C is said to be **PAC-learnable** by L using H iff for all $c \in C$, distributions $\mathcal{D}$ over X, learner L by sampling random examples from $\mathcal{D}$, will with probability at least $1 - \delta$ output a hypothesis $h \in H$ such that:

- $error_{\mathcal{D}}(h) \leq \epsilon$,
- time polynomial in $1/\epsilon$, $1/\delta$, n and $size(c)$

PAC Learnability

- PAC learnability seems to be concerned about computational resources required for learning
- In practice, we are only concerned about the number of training examples required
- The two are related
  - The computational limitation also imposes a polynomial constraint on the training set size, since a learner can process at most polynomial data in polynomial time.
  - The learner must visit each example at least once

Sample Complexity

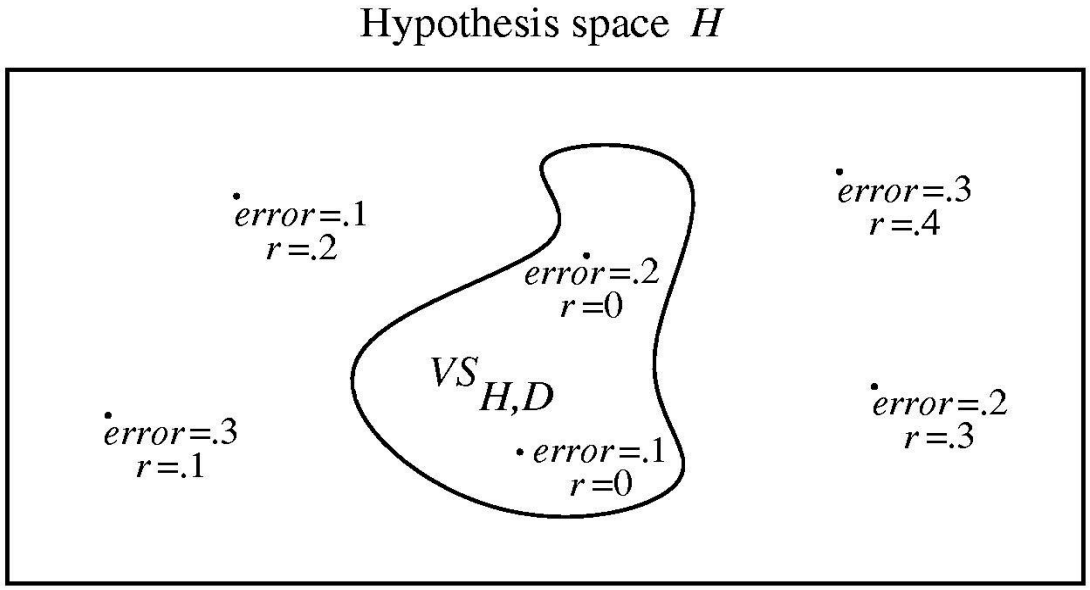
- Sample complexity is a useful notion
  - How many training examples are required for a problem of given size?
- How to prove PAC learnability:
  - First prove sample complexity of learning *C* using *H* is polynomial.
  - Second prove that the learner can train on a polynomial-sized data set in polynomial time.
- To be PAC-learnable, there must be a hypothesis in *H* with arbitrarily small error for every concept in *C*, generally $C \subseteq H$.

Sample Complexity for Consistent Learners

- A learner *L* using a hypothesis space *H* and training data D is said to be a **consistent learner** if it always outputs a hypothesis with zero error on D whenever *H* contains such a hypothesis.
- By definition, a consistent learner must produce a hypothesis in the version space for *H* given *D*.
- Therefore, to bound the number of examples needed by a consistent learner, we just need to bound the number of examples needed to ensure that the version-space contains no hypotheses with unacceptably high error.

Version Spaces

Version Space $VS_{H,D}$:
Subset of hypotheses in H consistent with training data D



(r = training error, $error$ = true error)

ε-Exhausted Version Space

- The version space, $VS_{H,D}$, is said to be ***ε-exhausted*** iff every hypothesis in it has true error less than or equal to ε.
- In other words, there are enough training examples to guarantee than any consistent hypothesis has error at most ε.
- One can never be sure that the version-space is ε-exhausted, but one can bound the probability that it is not.
- **Theorem 7.1** (Haussler, 1988): If the hypothesis space *H* is finite, and *D* is a sequence of $m \geq 1$ independent random examples for some target concept *c*, then for any $0 \leq \varepsilon \leq 1$, the probability that the version space $VS_{H,D}$ is ***not*** ε-exhausted is less than or equal to: $|H|e^{-\varepsilon m}$

Proof

- Let $H_{bad}=\{h_1,\ldots h_k\}$ be the subset of H with error > ε. The VS is not ε-exhausted if any of these are consistent with all *m* examples.
- A single $h_i \in H_{bad}$ is consistent with ***one*** example with probability:

$$P(\text{consist}(h_i, e_j)) \leq (1-\varepsilon)$$

- A single $h_i \in H_{bad}$ is consistent with ***all*** *m* independent random examples with probability:

$$P(\text{consist}(h_i, D)) \leq (1-\varepsilon)^m$$

- The probability that ***any*** $h_i \in H_{bad}$ is consistent with all *m* examples is:

$$P(\text{consist}(H_{bad}, D)) = P(\text{consist}(h_1, D) \vee \cdots \vee \text{consist}(h_k, D))$$

Proof (cont.)

- Since the probability of a disjunction of events is ***at most*** the sum of the probabilities of the individual events:

$$P(\text{consist}(H_{bad}, D)) \leq |H_{bad}|(1-\varepsilon)^m$$

- Since: $|H_{\text{bad}}| \leq |H|$ and $(1-\varepsilon)^m \leq e^{-\varepsilon m}$, $0 \leq \varepsilon \leq 1$, $m \geq 0$

$$P(\text{consist}(H_{bad}, D)) \leq |H|e^{-\varepsilon m}$$

Q.E.D

Sample Complexity Analysis

- Let δ be an upper bound on the probability of ***not*** exhausting the version space. So:

$$P(\text{consist}(H_{bad}, D)) \leq |H| e^{-\varepsilon m} \leq \delta$$

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

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

$$m \geq \left(-\ln\frac{\delta}{|H|}\right)/\varepsilon \quad \text{(flip inequality)}$$

$$m \geq \left(\ln\frac{|H|}{\delta}\right)/\varepsilon$$

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

Sample Complexity Result

- Therefore, any consistent learner, given at least:

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

  examples will produce a result that is PAC.
- Just need to determine the size of a hypothesis space to instantiate this result for learning specific classes of concepts.
- This gives a ***sufficient*** number of examples for PAC learning, but ***not*** a ***necessary*** number. Several approximations like that used to bound the probability of a disjunction make this a gross over-estimate in practice.

Sample Complexity of Conjunction Learning

- Consider conjunctions over *n* boolean features. There are 3^n of these since each feature can appear positively, appear negatively, or not appear in a given conjunction. Therefore $|H| = 3^n$, so a sufficient number of examples to learn a PAC concept is:

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

- Concrete examples:
  - δ=ε=0.05, *n*=10 gives 280 examples
  - δ=0.01, ε=0.05, *n*=10 gives 312 examples
  - δ=ε=0.01, *n*=10 gives 1,560 examples
  - δ=ε=0.01, *n*=50 gives 5,954 examples

How About *PlayTennis*?

1 attribute with 3 values (outlook)
9 attributes with 2 values (temp, humidity, wind, etc.)
Language: Conjunction of features or null concept

$$|H| = 4 \times 3^9 + 1 = 78733$$

$$m \geq \frac{1}{\epsilon}(\ln 78733 + \ln(1/\delta))$$

If we want to ensure that with probability 95%,
VS contains only hypotheses with $error_{\mathcal{D}}(h) \leq 10\%$,
then it is sufficient to have m examples, where

$$m \geq \frac{1}{0.1}(\ln 78733 + \ln(1/.05)) = 143$$

(# examples in domain: $3 \times 2^9 = 1536$)

Sample Complexity of Learning Arbitrary Boolean Functions

- Consider any boolean function over *n* boolean features such as the hypothesis space of DNF or decision trees. There are 2^{2^n} of these, so a sufficient number of examples to learn a PAC concept is:

$$\left(\ln \frac{1}{\delta} + \ln 2^{2^n} \right) / \varepsilon = \left(\ln \frac{1}{\delta} + 2^n \ln 2 \right) / \varepsilon$$

- Concrete examples:
  - δ=ε=0.05, *n*=10 gives 14,256 examples
  - δ=ε=0.05, *n*=20 gives 14,536,410 examples
  - δ=ε=0.05, *n*=50 gives $1.561 \text{x} 10^{16}$ example

Agnostic Learning

So far, assumed $c \in H$

Agnostic learning setting: don't assume $c \in H$

- What can we say in this case?
  - Hoeffding bounds:

    $Pr[error_{\mathcal{D}}(h) > error_D(h) + \epsilon] \leq e^{-2m\epsilon^2}$

  - For hypothesis space H:

    $Pr[error_{\mathcal{D}}(h_{best}) > error_D(h_{best}) + \epsilon] \leq |H|e^{-2m\epsilon^2}$

- What is the sample complexity in this case?

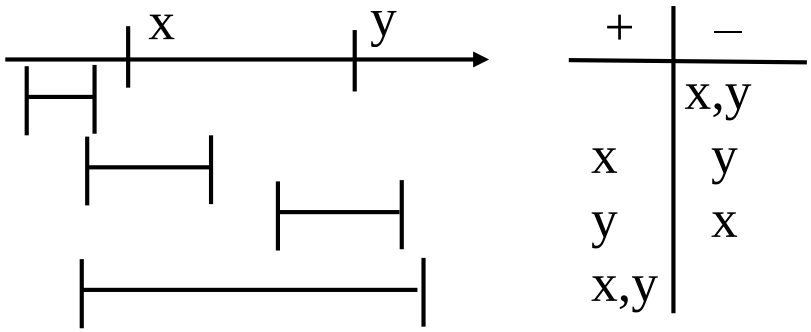
$$m \geq \frac{1}{2\epsilon^2}(\ln |H| + \ln(1/\delta))$$

Infinite Hypothesis Spaces

- The preceding analysis was restricted to finite hypothesis spaces. Moreover, the bounds were quite weak.
- Some infinite hypothesis spaces (such as those including real-valued thresholds or parameters) are more expressive than others.
  - Compare a rule allowing one threshold on a continuous feature (length<3cm) vs one allowing two thresholds (1cm<length<3cm).
- Need some measure of the expressiveness of infinite hypothesis spaces.
- The ***Vapnik-Chervonenkis*** (***VC***) ***dimension*** provides just such a measure, denoted VC(*H*).
- Analogous to ln|*H*|, there are bounds for sample complexity using VC(*H*).

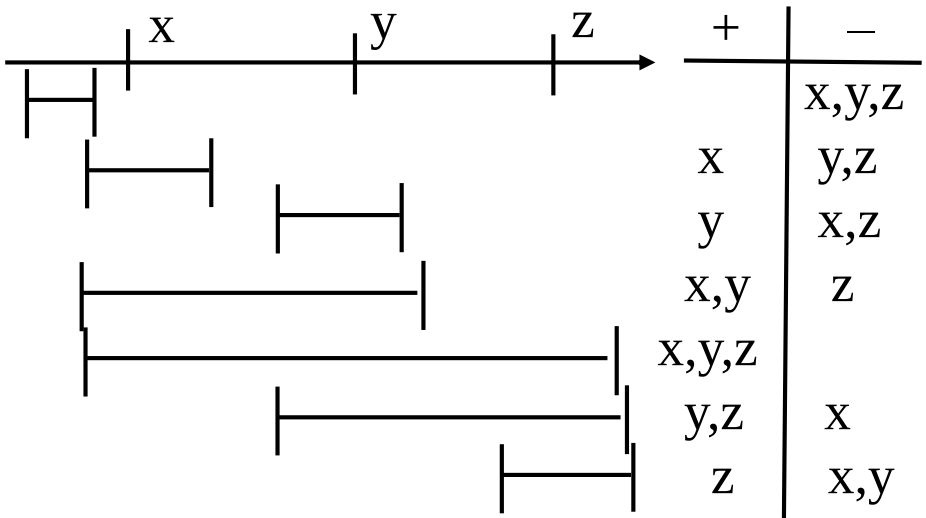
Shattering Instances

- A hypothesis space is said to shatter a set of instances iff for every partition of the instances into positive and negative, there is a hypothesis that produces that partition.
- For example, consider 2 instances described using a single real-valued feature being shattered by intervals.



Shattering Instances (cont)

- But 3 instances cannot be shattered by a single interval.

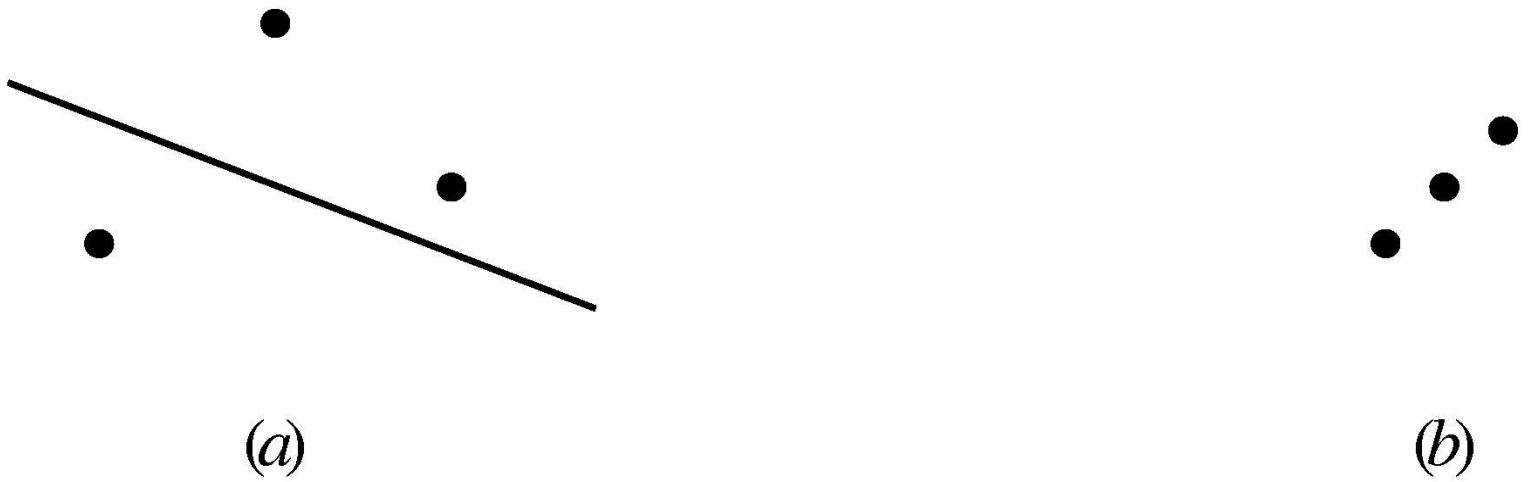


- Since there are 2^m partitions of m instances, in order for H to shatter instances: $|H| \geq 2^m$.

VC Dimension

- An unbiased hypothesis space shatters the entire instance space.
- The larger the subset of X that can be shattered, the more expressive the hypothesis space is, i.e. the less biased.
- The Vapnik-Chervonenkis dimension, VC(H). of hypothesis space H defined over instance space X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite subsets of X can be shattered then VC(H) = ∞
- If there exists at least one subset of X of size d that can be shattered then VC(H) $\geq d$. If no subset of size d can be shattered, then VC(H) $< d$.
- For a single intervals on the real line, all sets of 2 instances can be shattered, but no set of 3 instances can, so VC(H) = 2.
- Since $|H| \geq 2^m$, to shatter m instances, VC(H) $\leq \log_2 |H|$

VC Dim. of Linear Decision Surfaces

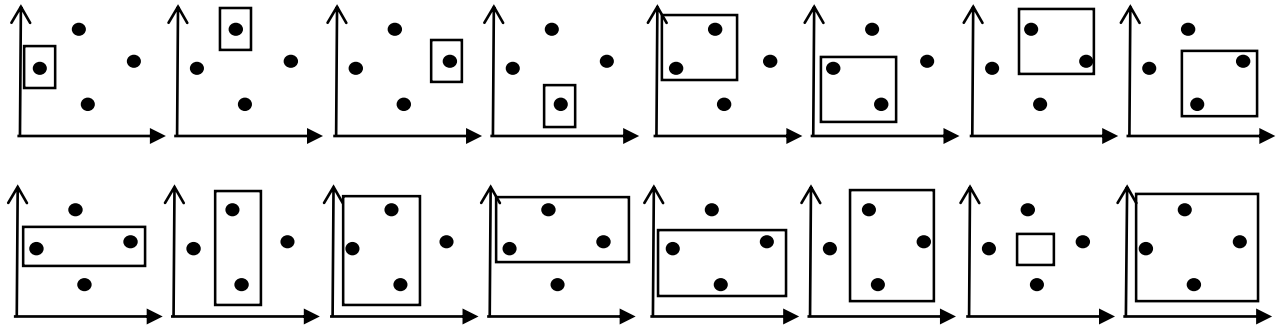


VC dim. of hyperplane in d-dimensional space is $d + 1$

The VC dimension of hypothesis space H defined over instance space X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite subsets of X can be shattered then VC(H) = ∞

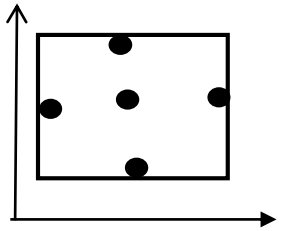
VC Dimension Example

- Consider axis-parallel rectangles in the real-plane, i.e. conjunctions of intervals on two real-valued features. Some 4 instances can be shattered.



VC Dimension Example (cont)

- No five instances can be shattered since there can be at most 4 distinct extreme points (min and max on each of the 2 dimensions) and these 4 cannot be included without including any possible 5^{th} point.



- Therefore VC(*H*) = 4
- Generalizes to axis-parallel hyper-rectangles (conjunctions of intervals in *n* dimensions): VC(*H*)=2*n*.

Upper Bound on Sample Complexity with VC

- Using VC dimension as a measure of expressiveness, the following number of examples have been shown to be sufficient for PAC Learning (Blumer *et al.*, 1989).

$$\frac{1}{\varepsilon}\left(4\log_2\left(\frac{2}{\delta}\right)+8VC(H)\log_2\left(\frac{13}{\varepsilon}\right)\right)$$

- Compared to the previous result using $\ln|H|$, this bound has some extra constants and an extra $\log_2(1/\varepsilon)$ factor. Since $VC(H) \leq \log_2|H|$, this can provide a tighter upper bound on the number of examples needed for PAC learning.

Conjunctive Learning with Continuous Features

- Consider learning axis-parallel hyper-rectangles, conjunctions on intervals on *n* continuous features.
  - $1.2 \leq \text{length} \leq 10.5 \wedge 2.4 \leq \text{weight} \leq 5.7$
- Since VC(*H*)=2*n* sample complexity is

$$\frac{1}{\varepsilon}\left(4\log_2\left(\frac{2}{\delta}\right)+16n\log_2\left(\frac{13}{\varepsilon}\right)\right)$$

- Since the most-specific conjunctive algorithm can easily find the tightest interval along each dimension that covers all of the positive instances ($f_{\min} \leq f \leq f_{\max}$) and runs in linear time, O(|*D*|*n*), axis-parallel hyper-rectangles are PAC learnable.

Sample Complexity Lower Bound with VC

- There is also a general lower bound on the minimum number of examples necessary for PAC learning (Ehrenfeucht, *et al*., 1989):

  Consider any concept class *C* such that VC(*H*)≥2 any learner *L* and any 0<ε<1/8, 0<δ<1/100. Then there exists a distribution *D* and target concept in *C* such that if *L* observes fewer than:

$$\max\left(\frac{1}{\varepsilon}\log_2\left(\frac{1}{\delta}\right), \frac{VC(C)-1}{32\varepsilon}\right)$$

  examples, then with probability at least δ, *L* outputs a hypothesis having error greater than ε.

- Ignoring constant factors, this lower bound is the same as the upper bound except for the extra $\log_2(1/\varepsilon)$ factor in the upper bound.

No Free Lunch Theorem

- If we are interested solely in the generalization performance, are there any reasons to prefer one classifier over the other?
- If we make no prior assumptions about the nature of the classification task, can we expect any classification method to be superior overall?
- Can we find an algorithm that is superior to random guessing?
- Answer is No

Implications of No free Lunch theorem

- No context-independent or usage-independent reasons to prefer one classifier over the other
- If one algorithm seems to outperform another in a particular situation, it is a consequence of its (over??)fit to the particular domain
  - Does not imply general superiority
- In practice what matters most is
  - Prior knowledge
  - Data distribution
  - Amount of training data
- Try different approaches and compare
- Off-training-set error: The test-set should be such that it contains no examples that are in the training set

No Free Lunch, Overfitting Avoidance and Occam's Razro

- We saw how to avoid overfitting
  - Regularization, pruning, inclusion of penalty, etc.
- Occam's razor: one should not use classifiers that are more complicated than necessary
- No free lunch invalidates these techniques
  - If there is no reason to prefer one over the other why are overfitting techniques and simpler models universally advocated

So Why are these techniques successful?

- Classes of problems addressed so far have certain properties
- Evolution: we have a strong selection pressure to be computationally simple.
- Satisficing (Herb Simon): Creating an adequate though possibly non-optimal solution
  - Underlying much of machine learning and human cognition

COLT Conclusions

- The PAC framework provides a theoretical framework for analyzing the effectiveness of learning algorithms.
- The sample complexity for any consistent learner using some hypothesis space, H, can be determined from a measure of its expressiveness $|H|$ or VC(H), quantifying bias and relating it to generalization.
- If sample complexity is tractable, then the computational complexity of finding a consistent hypothesis in H governs its PAC learnability.
- Constant factors are more important in sample complexity than in computational complexity, since our ability to gather data is generally not growing exponentially.
- Experimental results suggest that theoretical sample complexity bounds over-estimate the number of training instances needed in practice since they are worst-case upper bounds.

What you should know?

- Given a machine learning algorithm figure out the size of the hypothesis space and the VC dimension
- The sample complexity equations
  - Will be provided on the test but you should remember the notation and the background.