Need for a Revised Notion of Concept Learning

- **Example:** Consider an environmental-cleanup robot that needs to inspect trees for damage from acid rain. We need to train this robot to recognize "trees".
- Suppose we train our robot in East Lansing. What will its performance in Florida be? Vice-versa, we train our robot in Florida. How will it do in East Lansing?
- **Result:** Robot trained in Florida will get confused by "fall colors". Vice-versa, robot trained in East Lansing will not recognize palm trees.

Probability Distribution on Instances

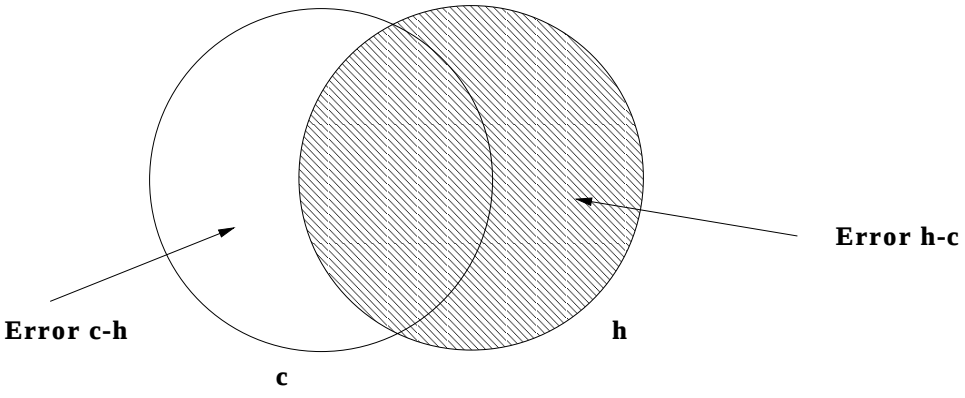
- For any given instance space, there is a **non-uniform** likelihood of seeing different instances. We can represent this situation by imagining that there is a **probability distribution** on the space of instances.
- The learner does not know this distribution ahead of time, but is allowed to assume that it is **fixed**. Thus, a learner trained on one particular distribution should only be tested on that distribution.

Approximate Concept Learning

- Requiring a learner to learn the *right* concept is too strict (e.g. is there a "right" concept of *tree*?).
- Instead, we relax this requirement and allow a learner to produce a **good approximation** to the actual concept.
- Let $P(x)$ be a fixed probability distribution on the instance space. Let c be the target concept, and let h be the concept produced by the learner.
- Let $S = \{x|c(x) \neq h(x)\}$ be the set of instances on which the target concept and the approximation disagree. Let ϵ be an error tolerance parameter where $0 < \epsilon < 1$. Then h is a good approximation (to within ϵ) of c if and only if:

$$\sum_{x \in S} P(x) \leq \epsilon$$

Approximation in Concept Learning



S = c-h U h-c

Approximate Learning using Version Spaces

- We say a version space is **exhausted** if the S and G sets are one and the same singleton set. We already know this is too hard.
- Given a hypothesis space H, a target concept c, a sequence of examples Q of c, and an error tolerance ϵ, the version space of Q (w.r.t. H) is ϵ-exhausted if it does not contain any hypothesis that has (**true**) error more than ϵ (w.r.t c).
- We will only require that the learner produce an ϵ-exhausted version space.
- Furthermore, we will solve the problem of exponentially large G sets by simply computing any one hypothesis h that has error $\leq \epsilon$.
- **Question:** How many examples are needed to ϵ-exhaust a version space?

Probabilistic Learning

- Assume training examples are drawn **independently and randomly** from an unknown but fixed distribution P on the instance space.
- We only require that the learner succeed in producing a good approximation to the target concept **with high probability**.
- Specifically, given a confidence parameter δ, we require the learner to be able to ϵ-exhaust a version space with probability at least $1 - \delta$.
- So how many examples are needed for the learner to ϵ-exhaust a version space with probability $\geq 1 - \delta$?

Sample Complexity for Probably Approximate Version Spaces

- **Theorem:** Let H be a finite space of hypotheses, and denote its size by $|H|$. Given m independently drawn random examples (drawn using a fixed distribution P) of some concept c in H, for any $0 < \epsilon < 1$, the probability that the version space consistent with the m examples is not ϵ-exhausted is $\leq |H|e^{-\epsilon m}$.
- **Proof:** Let $h_1, \ldots, h_k$ be hypotheses in H that have error $> \epsilon$. We will not ϵ-exhaust the version space iff one of these h_i is consistent with all m training examples. Since each h_i has error $> \epsilon$, an individual example is consistent with a given h_i with probability $< 1 - \epsilon$. The same h_i is consistent with all m examples with probability $< (1 - \epsilon)^m$. Now the probability of any h being consistent with all m examples $< k(1 - \epsilon)^m$. Since $k < |H|$, and $(1 - \epsilon)^m < e^{-\epsilon m}$, the result follows.

How Many Examples are Needed for PAC Learning?

- **Corollary:** Given reliability δ and error ϵ, the number of examples needed to ϵ-exhaust a version space consistent with some c in H is

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

- **Proof:** We know from the theorem that

$$\delta \geq |H|e^{-\epsilon m}$$

Taking natural logarithms on either side, we get

$$ln(\delta) \geq ln(|H|) - \epsilon m$$

Transforming the above, we get

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

- **NOTE:** Bound is *logarithmic* in $|H|$!

Example

- Let the hypothesis space H be all pure conjunctive formulae over n boolean attributes $A_1, \ldots, A_n$. $|H| = 3^n$.

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

$$m = \frac{1}{\epsilon}(ln(\frac{1}{\delta}) + nln(3))$$

- Number of examples is linear in n.
- Consider $n = 10$, $\epsilon = 0.05$, $\delta = 0.05$, then

$$m = \frac{1}{0.05}(ln(\frac{1}{0.05}) + 10ln(3)) = 260$$

- This is quite good, considering that there are almost $60,000$ pure conjunctive concepts on 10 boolean variables.

Probably Approximately Correct (PAC) Learning

- An algorithm A is a **PAC learning algorithm** for a class of concepts F if
  1. A takes as input ϵ and δ (both > 0 and ≤ 1) and a natural number n (a "size" parameter)
  2. A can call a routine EXAMPLE which returns examples for some $f \in F$. Examples are selected randomly, independently, and according to some fixed distribution P on the space of instances I.
  3. For all concepts $f \in F$ and all probability distributions P on I, with probability $\geq 1 - \delta$ algorithm A outputs a concept $g \in F$ such that

$$\sum_{x \in \{i | f(i) \neq g(i)\}} P(x) \leq \epsilon$$

- A class of concepts F is **PAC learnable** if there exists a PAC learning algorithm for F.

- A class of concepts F is **polynomial-sample PAC learnable** if it is PAC learnable using atmost $p(\frac{1}{\epsilon}, \frac{1}{\delta}, n)$ examples, where $p()$ is a polynomial function of its arguments.
- A class of concepts F is **polynomial-time PAC learnable** if it is PAC learnable in time atmost $p(\frac{1}{\epsilon}, \frac{1}{\delta}, n)$, where $p()$ is a polynomial function of its arguments.

Generic PAC Learning Algorithm

- **PAC-Learn-H**(ϵ, δ):
  1. Call the EXAMPLE routine m times, where
  $$m = \frac{1}{\epsilon}(ln(\frac{1}{\delta}) + ln(|H|))$$
  2. Collect the examples in a set S.
  3. Output any $g \in H$ consistent with S.
- Note sample complexity theorem guarantees g output by algorithm will be PAC.
  1. How do we find a consistent hypothesis?
  2. How do we know if $ln(|H|)$ is polynomial in n?
  3. Can we do better than $ln(|H|)$?
  4. What if H is infinite?

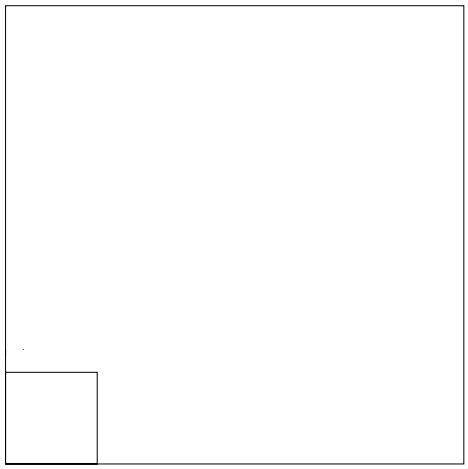
VC Dimension Examples

Calculate the VC-dimension of the following hypothesis spaces:

- All lines on a plane
  - For *some* set of 1,2, and 3 points, we can always find a line consistent with any labeling of these points.
  - However, there is no set of size 4 that can be shattered. **Why?**
- All hyperplanes in r dimensions (perceptron)
  - VC dimension of r dimensional hyperplane is $r + 1$. **Why?**
- Pure conjunctions (monomials) of n boolean literals
  - Consider $n = 3$ and let the set $S = \{011, 101, 110\}$.
  - There is a monomial consistent with any labeling of S.

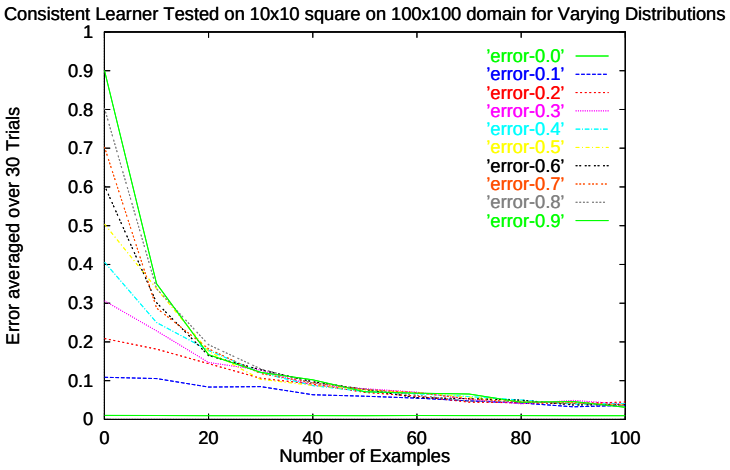
Learning Rectangles: Empirical Test of VC

Different distributions can be generated based on the proportion of positive and negative examples.

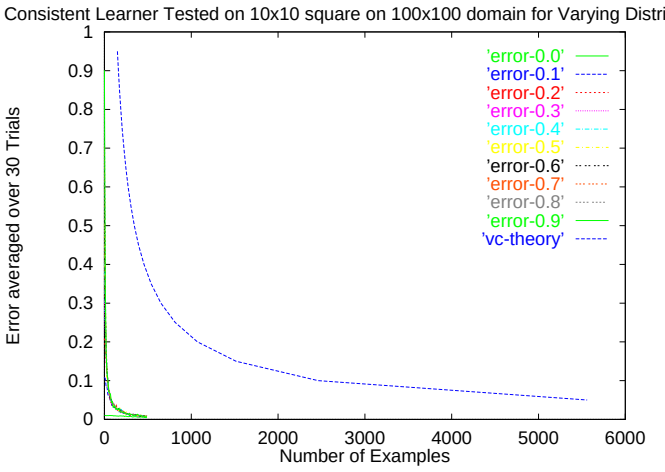


Empirical Test of VC Bounds

Each distribution shown here refers to the proportion of positive examples.



Empirical Test of VC Bounds



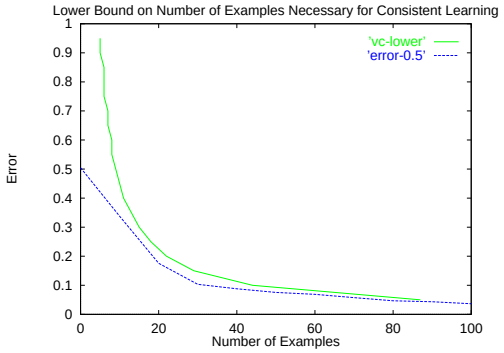
Small sample size behavior of Find-S algorithm

Samples	Error
0	0.504480
10	0.337607
20	0.175880
30	0.103427
40	0.087920
50	0.075420
60	0.068867
70	0.057460
80	0.046740

Lower Bound Prediction of PAC Framework

Any PAC learner who sees $< m$ examples, as given below, will fail sometimes (see book for exact theorem)

$$m \geq \max\left(\frac{1}{\epsilon}log(\frac{1}{\delta}), \frac{VC(C)-1}{32\epsilon}\right)$$



VC Theory for Perceptron Networks

Theorem: Let G be a layered DAG with n input nodes and $s \geq 2$ internal nodes, each of fan-in $\leq r$. Let each node in the network compute a concept $c \in C$ which is over R^r of VC-dim d. Let G_C be the composite concept computed by the overall network. Then, it can be shown that VC-dim$(G_C) \leq 2ds log(es)$.

Question: How many examples are needed to train a 10-input network of 5 perceptrons to an accuracy of 90% with reliability $\geq 95\%$?

$$\begin{aligned} m &\geq \frac{1}{\epsilon}\left(4log(\frac{2}{\delta}) + 8VC(H)log(\frac{13}{\epsilon})\right) \\ &\geq \frac{1}{\epsilon}\left(4log(\frac{2}{\delta}) + 16(r+1)slog(es)log(\frac{13}{\epsilon})\right) \\ &\geq \frac{1}{0.1}\left(4log(\frac{2}{0.05}) + 16(11)(5)log(e5)log(\frac{13}{0.1})\right) \\ &\geq 10\left(4log(40) + 16(11)(5)log(e5)log(130)\right) \approx 22000! \end{aligned}$$

Applications of VC Theory to Neural Nets

Theorem: Let F be the class of all functions computed by feedforward nets defined on a fixed underlying graph G with E edges and $N \geq 2$ linear threshold nodes

Let $W = E + N$ be the total number of weights in the network (one for each edge and one threshold weight for each node).

Then, it can be shown that VC-dim$(F) \leq 2Wlog(eN)$.

Theorem: If a feedforward net with linear threshold units is given m examples, where $m \geq \frac{32W}{\epsilon} ln \frac{32N}{\epsilon}$, and weights can be found that classify at least $1 - \frac{\epsilon}{2}$ of these m examples, then it can be shown that the net has learned an approximation with error $\leq \epsilon$ with confidence $\delta \geq 1 - 8e^{-\frac{\epsilon m}{32}}$

What Net Size Produces Valid Generalization

Suppose we want to train a neural net with W weights to learn a high accuracy approximation with error $\leq \epsilon$, how many examples m do we need?

Roughly speaking, $m \approx \frac{W}{\epsilon}$.

So, if we want 90% accuracy on *test* examples from the same fixed distribution used to train the network, we need 10 times as many examples as there are weights in the network.

Is this borne out in practice? If we consider the neural net used in ALVINN, or the neural network for recognizing sunglasses, the number of weights is approximately

$$32 \times 30 \times 4 \approx 3600 \text{ weights!}$$

So, the theory predicts you need $> 30,000$ training examples!

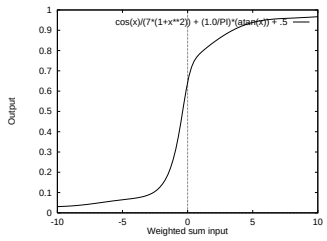
VC Results for Neural Nets: Summary

Network	VC bound	Authors
Single threshold unit	$n + 1$	Wenocur-Dudley
Feedforward threshold	$O(w\ log\ w)$	Baum-Haussler
Nonoverlapping threshold	$O(n\ log\ n)$	Schmitt
Feedforward sigmoidal	$O(w^4)$	Karpinski-Macintyre
Nonoverlapping sigmoidal	$O(n^4)$	Schmitt

Infinite dimension VC Bound Neural Nets exist!

Consider the smooth nonlinear activation function

$$o(x) = \frac{1}{\pi}\arctan(x) + \frac{\cos(x)}{7(1+x^2)} + \frac{1}{2}$$



Define N to be a neural net with 2 real-valued inputs, 2 hidden units with the above activation function, and a linear threshold gate as output

Theorem (Sontag, 1992): VC-dim$(N) = \infty$

Agnostic PAC Learning Model

The standard PAC theory assumes that the target concept $c \in H$ is some element of the space of hypotheses H (so a consistent learner is sufficient).

In many practical applications, this assumption is too stringent. The *agnostic* PAC framework assumes that learner outputs the hypothesis $h \in H$ that has the least error over the training data.

Theorem: After seeing m examples, an agnostic learner can output a good hypothesis h whose error is $\leq \epsilon$ with probability $\geq 1 - \delta$ if

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

Application of PAC Theory to Decision Trees

Define the *rank* of a decision tree as follows. If the tree is a single leaf, the rank is 0. If the rank of the left subtree is r_L equals the rank of the right subtree r_R, then the tree rank is $r_L + 1$. Otherwise, the rank is $\max r_L, r_R$.

Lemma: A decision tree with l leaves has rank atmost $log_2 l$.

Lemma: Rank r decision trees $\subset r$-decision lists.

Theorem: Constant rank decision trees are polynomial-time PAC learnable.

Theorem: Arbitrary decision trees of size s can be learned in time and number of examples $= O(n^{logs})$.