



CMSC726 Spring 2006: Learning Theory

readings: Mitchell ch. 7
sources: course slides are based on material from a variety of sources, including Tom Dietterich, Carlos Guestrin, Terran Lane, Rich Maclin, Ray Mooney, Andrew Moore, Andrew Ng, Jude Shavlik, and others.



Computational Learning Theory

- o Notions of interest:
 - efficiency,
 - accuracy,
 - complexity
- o Probably, Approximately Correct (PAC) Learning
- o Agnostic learning
- o VC Dimension and Shattering
- o Structural Risk Minimization
- o Mistake Bounds




Computational Learning Theory

What general laws constrain inductive learning?

Some potential areas of interest:

- o Probability of successful learning
- o Number of training examples
- o Complexity of hypothesis space
- o Accuracy to which target concept is approximated
- o Efficiency of learning process
- o Manner in which training examples are presented




The Concept Learning Task

Given

- o Instance space X – (e.g., possible faces described by attributes Hair, Nose, Eyes, etc.)
- o A unknown target function c – (e.g., Smiling : $X \rightarrow \{\text{yes, no}\}$)
- o A hypothesis space H : $H = \{ h : X \rightarrow \{\text{yes, no}\} \}$
- o A unknown, likely not observable probability distribution D over the instance space X

Determine


- o A hypothesis h in H such that $h(x) = c(x)$ for all x in D ?
- o A hypothesis h in H such that $h(x) = c(x)$ for all x in X ?



Variations on the Task – Data Sample


How many training examples sufficient to learn target concept?

1. Random process (e.g., nature) produces instances
 - Instances x generated randomly, teacher provides $c(x)$
2. Teacher (knows c) provides training examples
 - Teacher provides sequences of form $\langle x, c(x) \rangle$
3. Learner proposes instances, as queries to teacher
 - Learner proposes instance x , teacher provides $c(x)$



Noise-Free Case

- o **Assumption:** Examples are generated according to a probability distribution $p(x)$ and labeled according to an unknown function $y=f(x)$
- o **Learning Algorithm:** The learning algorithm is given a set of m examples, and it outputs a hypothesis $h \in H$ that is **consistent** with those examples (correctly labels all of them).
- o **Goal:** h should have a low error rate on new examples from the same distribution $p(x)$.



$\text{error}(h, f) = p\{f(x) \neq h(x)\}$

Probably-Approximately Correct Learning

We allow our algorithms to fail with probability δ .

Finding an approximately correct hypothesis with high probability

Imagine drawing a sample of m examples, running the learning algorithm, and obtaining h . Sometimes the sample will be unrepresentative, so we want to insist that $1 - \delta$ the time, the hypothesis will have error less than ϵ .

$$P(\text{error}(h,f) > \epsilon) < \delta$$

For example, we might want to obtain a 99% accurate hypothesis 90% of the time.

Case 1: Finite Hypothesis Space

- Suppose H is finite.
- Consider $h_1 \in H$ such that $\text{error}(h_1, f) < \epsilon$. What is the probability that it will correctly classify m training examples?
- If we draw one training example (x_1, y_1) , what is the probability that h_1 classifies it correctly?
 $1 - \epsilon$
- What is the probability that h_1 will be right m times?

Finite Hypothesis Case (2)

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

$$\begin{aligned} P[h_1 \vee h_2 \text{ survives}] &= P[h_1 \text{ survives}] + P[h_2 \text{ survives}] - P[h_1 \wedge h_2 \text{ survives}] \\ &\leq P[h_1 \text{ survives}] + P[h_2 \text{ survives}] \\ &\leq 2(1 - \epsilon)^m \end{aligned}$$

- So if there are k ϵ -bad hypotheses, the probability that any one of them will survive is

$$\leq k(1 - \epsilon)^m$$

- Since $k < |H|$, this is

$$\leq |H|(1 - \epsilon)^m$$

Finite Hypothesis Case (3)

- Factoid:** When $0 \leq \epsilon \leq 1$, $(1 - \epsilon) \leq e^{-\epsilon}$
 $|H|(1 - \epsilon)^m \leq |H|e^{-\epsilon m}$
- Lemma:** For a finite hypothesis space H , given a set of m training examples drawn independently according to p , the probability that there exists an hypotheses $h \in H$ with true error greater than ϵ consistent with the training examples is less than $|H|e^{-\epsilon m}$

- We want to ensure that this probability is less than δ :

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

- This will be true when $m \geq \frac{1}{\epsilon} \left[\ln |H| + \ln \frac{1}{\delta} \right]$

This is sometimes called the Blumer bound (Blumer, Ehrenfeucht, Haussler, and Warmuth, 1987).

M is the **sample complexity**, the number of training examples needed for a learner to converge (with high probability) to a successful hypothesis

Finite Hypothesis Space Bound

- From the Blumer bound, we can get our first bound on the error rate:
- If $h \in H$ is consistent with all m examples drawn according to p , then the error rate on new data points can be estimated as

$$\epsilon = \frac{1}{m} \left[\ln |H| + \ln \frac{1}{\delta} \right]$$

PAC Learnable

- Given a concept class C defined over a set of instances X of length n and a hypothesis space H . C is PAC-learnable using H if for all $c \in C$, all distributions over X , we can guarantee that with high probability $(1 - \delta)$ we output a hypothesis h with error at most ϵ , for $\delta \leq 1/2$, $\epsilon > 0$, in time polynomial in $1/\epsilon$, $1/\delta$, n and $\text{size}(C)$.
- issues:
 - sample complexity:** How many training examples are needed for a learner to converge with high probability?
 - computational complexity:** how much computational effort is needed for a learner to converge to a successful hypothesis
 - mistake bound:** how many training examples will the learner misclassify before converging to a successful hypothesis?

● ● ● | Example 1

- Boolean Conjunctions over n features
 - $|H| = ?$
 - Each feature can appear $x_j, \neg x_j$ or missing
 - $|H| = 3^n$

$$m = \frac{1}{\epsilon} \left[n \ln 3 + \ln \frac{1}{\delta} \right]$$

Boolean Conjunctions are PAC-learnable

● ● ● | Example 2

- K-DNF formulas
 - 2-DNF: $(x_1 \wedge \neg x_2) + (x_3 \wedge \neg x_4) + (x_2 \wedge x_3) \dots$
 - number of disjuncts (k-terms):
 - $\leq (2n)^k$
 - number of formulas
 - $|H| \leq 2^{(2n)^k}$
 - $\lg|H| = (2n)^k$

$$m = \frac{1}{\epsilon} O \left[n^k + \ln \frac{1}{\delta} \right]$$

K-DNF is PAC-learnable

● ● ● | Example 3

- unbiased learner
 - arbitrary boolean formula
 - size of hypothesis space
 - $|H| = 2^{2^n}$
 - $\lg|H| = 2^n$

$$m = \frac{1}{\epsilon} \left[2^n + \ln \frac{1}{\delta} \right]$$

Arbitrary boolean formulas are **not** PAC-learnable

● ● ● | Finite Hypothesis Space: Inconsistent Hypothesis

- Suppose that h does not perfectly fit the data (is not consistent), but rather has an error rate of ϵ_T . Then the following holds:

$$\epsilon \leq \epsilon_T + \sqrt{\frac{\ln|H| + \ln \frac{1}{\delta}}{2m}}$$

- This makes it clear that the error rate on test data is usually going to be larger than the error rate on the training data.

● ● ● | But what if hypothesis space not finite?

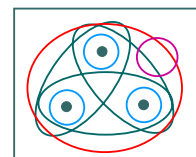
What if $|H|$ can not be determined?

- It is still possible to come up with estimates based not on counting how many hypotheses, but based on how many instances can be completely discriminated by H
- Use the notion of a shattering of a set of instances to measure the complexity of a hypothesis space
- VC Dimension measures this notion and can be used as a stand in for $|H|$

● ● ● | Shattering a Set of Instances

- **Definition:** a *dichotomy* of a set S is a partition of S into two disjoint subsets.
- **Definition:** a set of instances S is *shattered* by hypothesis space H iff for every dichotomy of S there exists some hypothesis in H consistent with this dichotomy.

Example:
3 instances
shattered



Instance space X

● ● ● | The Vapnik-Chervonenkis Dimension

- **Definition:** the **Vapnik-Chervonenkis (VC) 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 sets of X can be shattered by H , then $VC(H) = \infty$.
- Example: VC dimension of linear decision surfaces is 3.



● ● ● | Infinite Hypothesis Spaces

- The following bound is analogous to the Blumer bound. If h is a hypothesis that makes no errors on the training data set of size m , and h is drawn from an hypothesis space H with VC-dimension d , then with probability $1 - \delta$, h will have error rate less than ϵ if

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

● ● ● | Error Bound for Infinite Hypothesis Spaces

- Theorem: Suppose H has a VC-dimension of d and a learning algorithm find an $h \in H$ with error rate ϵ_T on a sample of size m . Then with probability $1 - \delta$, the error rate ϵ on new data points is

$$\epsilon \leq 2\epsilon_T + \frac{4}{m} \left[d \log_2 \left(\frac{2em}{d} \right) + \log_2 \left(\frac{4}{\delta} \right) \right]$$

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

● ● ● | Comments

- VC dimension is distribution-free; it is independent of the probability distribution from which the instances are drawn
- In this sense, it gives us a worse case complexity (pessimistic)
- However, this is still useful for providing bounds, such as the sample complexity of a hypothesis class.
- In general, there is a connection between the VC dimension (which we would like to minimize) and the error on the training set (empirical risk)

● ● ● | How do we characterize “power”?

- Different machines have different amounts of “power”.
- Tradeoff between:
 - More power: Can model more complex classifiers but might overfit.
 - Less power: Not going to overfit, but restricted in what it can model.
- How do we characterize the amount of power?

● ● ● | Structural Risk Minimization

- Let $\phi(f)$ = the set of functions representable by f .
- Suppose $\phi(f_1) \subseteq \phi(f_2) \subseteq \dots \subseteq \phi(f_n)$
- Then $h(f_1) \leq h(f_2) \leq \dots \leq h(f_n)$
- To decide which machine to use: train each machine and make a table...

$$\text{TESTERR}(\alpha) \leq \text{TRAINERR}(\alpha) + \frac{h(\log(2R/h) + 1) - \log(\eta/4)}{R}$$

i	f_i	TRAINER R	VC-Conf	Probable upper bound on TESTERR	Choice
1	f_1				
2	f_2				
3	f_3				⊗
4	f_4				
5	f_5				
6	f_6				

● ● ● | SVMs and PAC Learning

- Theorems connect PAC theory to the size of the *margin*
- Basically, the *larger* the margin, the better the expected accuracy
- See, for example, Chapter 4 of *Support Vector Machines* by Christianini and Shawe-Taylor, Cambridge University Press, 2002

● ● ● | PAC and the Number of Support Vectors

- The fewer the support vectors, the better the generalization will be
- Recall, non-support vectors are
 - Correctly classified
 - Don't change the learned model if left out of the training set
- So

$$\text{leave - one - out error rate} \leq \frac{\# \text{ support vectors}}{\# \text{ training examples}}$$

● ● ● | VC-dimension of an SVM

- Very very very loosely speaking there is some theory which under some different assumptions puts an upper bound on the VC dimension as

$$\left\lceil \frac{\text{Diameter}}{\text{Margin}} \right\rceil$$

- where
 - *Diameter* is the diameter of the smallest sphere that can enclose all the high-dimensional term-vectors derived from the training set.
 - *Margin* is the smallest margin we'll let the SVM use
- This can be used in SRM (Structural Risk Minimization) for choosing the polynomial degree, RBF σ , etc.
 - But most people just use Cross-Validation

Copyright © 2001, 2003, Andrew W. Moore

● ● ● | Mistake Bounds

So far: how many examples needed to learn?
 What about: how many mistakes before convergence?

Consider setting similar to PAC learning:

- Instances drawn at random from X according to distribution D
- Learner must classify each instance before receiving correct classification from teacher

Can we bound the number of mistakes learner makes before converging?

● ● ● | Mistake Bounds: Find-S

Consider Find-S when H = conjunction of boolean literals

Find-S:

- Initialize h to the most specific hypothesis:

$$l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \wedge l_3 \wedge \neg l_3 \wedge \dots \wedge l_n \wedge \neg l_n$$
- For each positive training instance x
 - Remove from h any literal that is not satisfied by x
- Output hypothesis h

How many mistakes before converging to correct h ?

● ● ● | Mistakes in Find-S

- Assuming $c \in H$
 - Negative examples – can never be mislabeled as positive, the current hypothesis h is always at least as specific as target concept c
 - Positive examples – can be mislabeled as negative (concept not general enough, consider initial)
 - First positive example, $2n$ terms in literal (positive and negative of each feature), n will be eliminated
 - Each subsequent mislabeled positive example – will eliminate at least one term
 - Thus at most $n+1$ mistakes

● ● ● | Mistake Bounds: Halving Algorithm

Consider the Halving Algorithm

- Learn concept using version space candidate elimination algorithm
- Classify new instances by majority vote of version space members
- How many mistakes before converging to correct h ?
- ... in worst case?
- ... in best case?

● ● ● | Mistakes in Halving

- At each point, predictions are made based on a majority of the remaining hypotheses
- A mistake can be made only when at least half of the hypotheses are wrong
- Thus the size of H decreases by half for each mistake
- Thus, worst case bound is related to $\log_2 |H|$
- How about best case?
 - Note, prediction of the majority could be correct but number of remaining hypotheses can decrease
 - Possible for the number of hypotheses to reach one with no mistakes

● ● ● | Summary: Learning Theory

- Characterizations:
 - sample complexity
 - computational complexity
 - mistake bound
- The complexity of a hypothesis space is measured by the VC-dimension
- There is a tradeoff between ϵ , δ and m
- Structural Risk Minimization