Computational Learning Theory

- What general laws constrain inductive learning?
- Seeking theory to relate
  - probability of successful learning
  - number of training examples
  - complexity of H
  - accuracy of approximations
  - manner in which examples are given

Prototypical concept learning task

- Given
  - X, c: X -> {0,1}, H
  - D = {<x1,c(x1)>,…,<xm,c(xm)>}
- Determine h s.t. h(x) = c(x)
  - for all x in D?
  - for all x in X?

Basic notions

X	- instance space (in general, an arbitrary set)
$c \subseteq X$	- concept (each subset of X is a concept)
$C \subseteq \{c \mid c \subseteq X\}$	- class of concepts to be learned
$T \in C$	- the unknown concept to be learned

Examples

X	$\mathbf{R}^2$	$\{0,1\}^n$
C	set of rectangles	CNF with n variables
T	a given rectangle	a given CNF

It may be more convenient for the learning algorithm to represent T in other way, and not simply as a subset of X.

Learning algorithm - inputs

P - fixed probability distribution defined on X

Learning algorithm receives a sequence of examples

$(x_0,v_0),(x_1,v_1),(x_2,v_2),(x_3,v_3),\ldots$

where $x_i \in X$ and v_i = "+", if $x_i \in T$, v_i = "–", if $x_i \notin T$, and the probability that x_i appears as a current element in the sequence is in accordance with P.

ε - accuracy parameter
δ - confidence parameter

Learning algorithm - inputs

A model where all v_i = “+” can be considered, in which case we say that an algorithm is learning from positive examples.

In general case we can say that algorithm is learning from positive and negative examples.

A learning from only negative examples can also be considered.

Learning algorithm - outputs

After the processing a finite sequence of inputs a learning algorithm outputs a concept $S \in C$.

$S \oplus T$ - the symmetric difference of S and T

$P(S \oplus T)$ - the error rate of concept S, i.e. the probability that T and S classify a random example differently.

PAC Learning: Results for Two Hypothesis Languages

- Unbiased Learner
  - Recall: sample complexity bound $m \geq 1/\varepsilon\ (\ln |H| + \ln (1/\delta))$
  - Sample complexity not always polynomial
  - Example: for unbiased learner, $|H| = 2^{|X|}$
  - Suppose X consists of n booleans (binary-valued attributes)
    - $|X| = 2^n$, $|H| = 2^{2^n}$
    - $m \geq 1/\varepsilon\ (2^n \ln 2 + \ln (1/\delta))$
    - Sample complexity for this H is exponential in n
- Monotone Conjunctions
  - Target function of the form $y = f(x_1, \ldots, x_n) = x'_1 \wedge \ldots \wedge x'_k$
  - Active learning protocol (learner gives query instances): n examples needed
  - Passive learning with a helpful teacher: k examples (k literals in true concept)
  - Passive learning with randomly selected examples (proof to follow):

Learnability - definition

A concept S is *approximately correct*, if $P(S \oplus T) \leq \varepsilon$

The learning algorithm is *probably approximately correct*, if the probability that the output S is approximately correct is at least $1-\delta$

In this case we say that a learning algorithm *pac-learns* the concept class C, and that the concept class C is *pac-learnable*

Polynomial learnability 1

A learning algorithm L is a *polynomial PAC-learning algorithm* for C, and C is *polynomially PAC-learnable*, if L PAC-learns C with time complexity (and sample complexity) which are polynomial in $1/\varepsilon$ and $1/\delta$.

It is useful to consider similar classes of concepts with different sizes n, and require polynomial complexity also in n, but then we must focus on specific instance spaces dependent from parameter n (eg. $X = \{0,1\}^n$).

Polynomial learnability 2

We consider $\mathbf{C} = \{(X_n, C_n) | n>0\}$

A learning algorithm L is a *polynomial PAC-learning algorithm* for **C**, and **C** is *polynomially PAC-learnable*, if L PAC-learns **C** with time complexity (and sample complexity) which are polynomial in n, $1/\varepsilon$ and $1/\delta$.

Sample Complexity

- How large D required
  - query model: learner proposes x, teacher gives c(x)
  - ‘tutorial’ model: teacher provides (good) <x,c(x)>
  - random: x drawn from some unknown distribution *D*, teacher provides c(x)

Sample complexity…

- Query model
  - assume c is in H
- Optimal query strategy
  - select x s.t. half of h in VS(H,D) classify it +, half -
  - log |H| queries
  - not always possible -> more queries

Sample complexity…

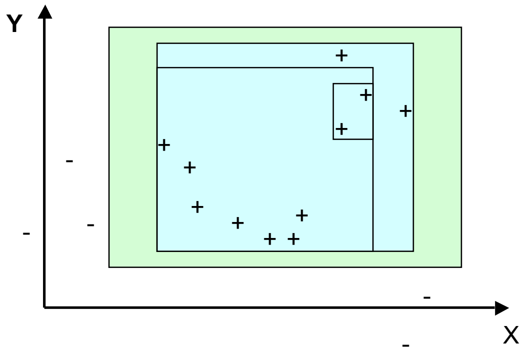
- Tutorial model
  - teacher knows c
- Optimal teaching strategy
  - depends on H
  - e.g. Boolean conjunctions of n literals
    - n + 1 examples suffice!

Sample complexity…

- Random model
  - X, H, C, *D*
- Task
  - output h estimating c
  - performance of h evaluated on new examples drawn from *D*
  - note: probabilistic selection of x, no noise in c(x)

PAC Learning:Rectangles

- Assume Target Concept Is An Axis Parallel (Hyper)rectangle



- Will We Be Able To Learn The Target Concept?
- Can We Come Close?

Valiant's results - k-CNF

$X = \{0,1\}^n$ (n boolean variables)

C = set of k-CNF formulas (ie, CNF formulas with at most k literals in each clause)

<u>Theorem</u>

For any positive integer k the class of k-CNF formulas is polynomially PAC-learnable from positive examples.

(Even when partial examples (with some variables missing) are allowed.)

Valiant's results - k-CNF - function L(r,m)

Definition

For any real number $r > 1$ and for any positive integer m the value L(r,m) is the smallest integer y, such that in y Bernoulli trials with probability of success at least 1/r, the probability of having fewer than m successes is less than 1/r.

Proposition

$L(r,m) \leq 2r(m+\ln r)$

True error...

- Our concern
  - can we bound true error of h given the training error?
  - may be 0 (consistent learners)
- VS(H,D) contains all consistent h
  - bound #examples needed to assure VS(h,D) contains no unacceptable h
  - applies to any consistent learner

Exhausting H

- VS(H,D) is ε-exhausted if
  - every h in VS(H,D)
  - has true error smaller than ε
- Surprise:
  - probabilistic argument allows us to bound the probability VS will be ε-exhausted after certain # of examples

Theorem (Haussler-88)

- If
  - H is finite
  - D indep. random examples, |D|=m
- then
  - P(exists h in VS(H,D), error(h) > ε)
  - is bounded by |H|e^(-εm)
- Ensure |H|e^(-εm) <= δ
  - m >= 1/ε(ln |H| + ln(1/δ))

Example: conj. of literals

- $|H| = 3^n$
  - $m >= 1/\varepsilon(\ln |H| + \ln(1/\delta))$
  - $m >= 1/\varepsilon(\ln 3^n + \ln(1/\delta))$
  - $m >= 1/\varepsilon(n \ln 3 + \ln(1/\delta))$
- EnjoySport example: $|H| = 973$
  - prob. of 95%, errors <= 0.10
  - $\varepsilon = 0.1$ and $\delta = 0.05$
  - get $m >= 98.8$

Agnostic learning

- Don't assume c is in H
- We want
  - h_best making fewest errors on D
- Sample complexity?
  - $m >= 1/(2\varepsilon^2)(\ln |H| + \ln(1/\delta))$
  - justification: Hoeffding bounds
    - $P[\text{true} > \text{sample} + \varepsilon] <= e^{(-2m\varepsilon^2)}$
    - $|H|$ alternatives to choose from

Some examples

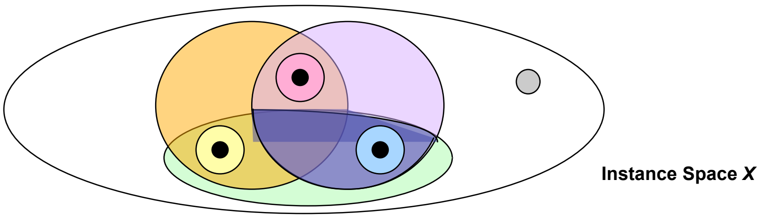
- Boolean conjunctions
  - use Find_S -> PAC
- Unbiased learners
  - too large sample
- k-term DNF & k-CNF
  - polynomial sample
  - DNF: NP-complete
  - k-CNF: polynomial

Shattering X

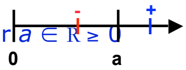
- VC(H): # of distinct instances of X that can be completely discriminated by H
- $S \subseteq X$ is *shattered* by H
  - for every dichotomy C of S
  - exists h consistent with C

VC Dimension: Examples of Shattered Sets

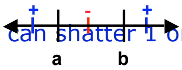
- Three Instances Shattered



- Intervals
  - Left-bounded intervals on the real axis: $[0, a)$, for $a \in \mathbb{R} \geq 0$



    - Sets of 2 points cannot be shattered
    - Given 2 points, can label so that no hypothesis will be consistent
  - Intervals on the real axis ($[a, b]$, $b \in \mathbb{R} > a \in \mathbb{R}$): can shatter 1 or 2 points, not 3



  - Half-spaces in the plane (non-collinear): 1? 2? 3? 4?

VC(H)

- Size of largest S that can be shattered by H
  - one 'large' S suffices
  - infinite if any S is shattered
- Note
  - If VC(H) = d, then |H| >= 2^d
  - VC(H) <= log|H| for finite H

VC(H) for ANNs…

- Network of perceptrons
  - internal units have VC(C) = r+1
  - VC(net) <= 2(r+1)s log(es)
  - apply this to count upper bound on # of required training examples
- Note
  - not applicable to sigmoid units
  - inductive bias of BP (small weights) reduces the effective VC dimension

VC (Vapnik - Chervonenkis) dimension

C	- nonempty concept class
$s \subseteq X$	- set
$\Pi_C(s) = \{s \cap c \mid c \in C\}$	- all subsets of s, that can be obtaining by intersecting s with a concept from c

We say that s is *shattered* by C, if $\Pi_C(s) = 2^s$.

The VC (Vapnik-Chervonenkis) dimension of C is the cardinality of the largest finite set of points $s \subseteq X$ that is shattered by C (if arbitrary large finite sets are shattered, we say that VC dimension of C is infinite).

VC dimension - examples

Example 1

$X = \mathbf{R}$
C = set of all (open or closed) intervals

If $s = \{x_1,x_2\}$, then there exist c_1, c_2, c_3, $c_4 \in C$, such that $c_1 \cap s = \{x_1\}$, $c_2 \cap s = \{x_2\}$, $c_3 \cap s = \varnothing$, and $c_{14} \cap s = s$.

If $s = \{x_1,x_2,x_3\}$, $x_1 \le x_2 \le x_3$, then there is no concept $c \in C$, such that $x_1 \in c$, $x_3 \in c$ and $x_2 \notin c$. Thus the VC dimension of C is 2.

VC dimension - examples

Example 2

C = any finite concept class

It requires 2^d distinct concepts to shatter a set of d points, therefore no set of cardinality larger that log|C| can be shattered. Hence the VC dimension of C is at most log |C|.

VC dimension - examples

Example 3

X = **R**
C = set of all finite unions of (open or closed) intervals

Any finite s∈X can be shattered, thus the VC dimension of C is infinite.

VC dimension - relation to PAC

Theorem [A.Blumer, A.Ehrenfeucht, D.Haussler, M.Warmuth]

C is PAC-learnable if and only if the VC dimension of C is finite.

Theorem also gives an upper and lower bounds of number of examples needed for learning.

Learnability and the Vapnik-Chervonenkis dimension, Journal of the ACM, vol. 36, 4, 1989, pp. 929-965.

VC dimension - relation to PAC

Let d be the VC dimension of C, then:

- no algorithm can PAC-learn class C with less than $\max((1-\varepsilon)/\varepsilon \ln 1/\delta, d(1 - 2(\varepsilon(1-\delta)+\delta)))$ examples.

- any consistent algorithm can PAC-learn class C with $\max(4/\varepsilon \log 2/\delta, 8d/\varepsilon \log 13/\varepsilon)$ examples.

VC dimension - relation to PAC - example

$X = R^2$
C = set of all triangles in X

VC dimension of C is 4, thus C is PAC-learnable.

VC Dimension: Relation to Sample Complexity

- *VC*(*H*) as A Measure of Expressiveness
  - Prescribes 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 $m > 1/\varepsilon\ (8\ VC(H) \lg 13/\varepsilon + 4 \lg (2/\delta))$, then with probability at least $(1 - \delta)$, *h* has true error less than ε
- Significance
  - If *m* is polynomial, we have a PAC learning algorithm
  - To be efficient, we need to produce the hypothesis *h* efficiently
- Note
  - $|H| > 2^m$ required to shatter *m* examples
  - Therefore $VC(H) \leq \lg(H)$

Occam's Razor and PAC Learning [1]

- Bad Hypothesis
  - $error_D(h) \equiv \Pr_{x \in D}[c(x) \neq h(x)]$
  - Want to bound: probability that there exists a hypothesis $h \in H$ that
    - is consistent with *m* examples
    - satisfies $error_D(h) > \varepsilon$
  - Claim: the probability is less than $|H|\,(1 - \varepsilon)^m$
- Proof
  - Let *h* be such a bad hypothesis
  - The probability that *h* is consistent with one example <*x*, *c*(*x*)> of *c* is $\Pr_{x \in D}[c(x) = h(x)] < 1 - \varepsilon$
  - Because the *m* examples are drawn independently of each other, the probability that *h* is consistent with *m* examples of *c* is less than $(1 - \varepsilon)^m$
  - The probability that *some* hypothesis in *H* is consistent with *m* examples of *c* is less than $|H|\,(1 - \varepsilon)^m$, Quod Erat Demonstrandum

Occam's Razor and PAC Learning [2]

- Goal
  - 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 $\ln (1 - \varepsilon) \leq \varepsilon$: $m \geq 1/\varepsilon (\ln | H | + \ln (1/\delta))$
  - This is the result from last time [Blumer *et al*, 1987; Haussler, 1988]
- Occam's Razor
  - "Entities should not be multiplied without necessity"
  - So called because it indicates a preference towards a small *H*
  - Why do we want small *H*?
    - Generalization capability: explicit form of inductive bias
    - Search capability: more efficient, compact
  - To guarantee consistency, need $H \supseteq C$ – really want the smallest *H* possible?

Mistake Bounds: Rationale and Framework

- So Far: How Many Examples Needed To Learn?
- Another Measure of Difficulty: How Many <u>Mistakes Before Convergence</u>?
- Similar Setting to PAC Learning Environment
  - Instances drawn at random from *X* according to distribution D
  - Learner must classify each instance before receiving correct classification from teacher
  - Can we bound number of mistakes learner makes before converging?
  - Rationale: suppose (for example) that *c* = fraudulent credit card transactions

Mistake Bounds: *Find-S*

- Scenario for Analyzing Mistake Bounds
  - Suppose H = conjunction of Boolean literals
  - *Find-S*
    - Initialize h to the most specific hypothesis $l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \wedge \ldots \wedge l_n \wedge \neg l_n$
    - 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?
  - Once a literal is removed, it is never put back (monotonic relaxation of h)
  - No false positives (started with most restrictive h): count false negatives
  - First example will remove n candidate literals (which don't match x_1's values)
  - Worst case: every remaining literal is also removed (incurring 1 mistake each)
  - For this concept ($\forall x \,.\, c(x) = 1$, *aka* "true"), Find-S makes $n + 1$ mistakes

Mistake Bounds: Halving Algorithm

- Scenario for Analyzing Mistake Bounds
  - Halving Algorithm: learn concept using version space
    - e.g., *Candidate-Elimination* algorithm (or *List-Then-Eliminate*)
  - Need to specify performance element (how predictions are made)
    - Classify new instances by *majority vote of version space members*
- How Many Mistakes before Converging to Correct h?
  - … in worst case?
    - Can make a mistake when the majority of hypotheses in $VS_{H,D}$ are wrong
    - But then we can remove at least half of the candidates
    - Worst case number of mistakes: $\lfloor \log_2 |H| \rfloor$
  - … in best case?
    - Can get away with *no* mistakes!
    - (If we were lucky and majority vote was right, $VS_{H,D}$ still shrinks)

Optimal Mistake Bounds

- Upper Mistake Bound for A Particular Learning Algorithm
  - Let $M_A(C)$ be the max number of mistakes made by algorithm A to learn concepts in C
    - Maximum over $c \in C$, all possible training sequences D
    - $M_A(C) \equiv \max_{c \in C}[M_A(c)]$
- Minimax Definition
  - Let C be an arbitrary non-empty concept class
  - The optimal mistake bound for C, denoted $Opt(C)$, is the minimum over all possible learning algorithms A of $M_A(C)$
  - $Opt(C) \equiv \min_{A \in \text{learning algorithms}}[M_A(c)]$
  - $VC(C) \leq Opt(C) \leq M_{Halving}(C) \leq lg(C)$

COLT Conclusions

- PAC Framework
  - Provides reasonable model for theoretically analyzing effectiveness of learning algorithms
  - *Prescribes* things to do: enrich the hypothesis space (search for a less restrictive H); make H more flexible (e.g., hierarchical); incorporate knowledge
- Sample Complexity and Computational Complexity
  - Sample complexity for any consistent learner using H can be determined from measures of H's expressiveness ($|H|$, $VC(H)$, etc.)
  - If the sample complexity is tractable, then the computational complexity of finding a consistent h governs the complexity of the problem
  - Sample complexity bounds are not tight! (But they separate learnable classes from non-learnable classes)
  - Computational complexity results exhibit cases where information theoretic learning is feasible, but finding a good h is intractable
- COLT: Framework For Concrete Analysis of the Complexity of L
  - Dependent on various assumptions (e.g., $x \in X$ contain relevant variables)