

Computational Learning Theory

Mark Craven and David Page

Computer Sciences 760

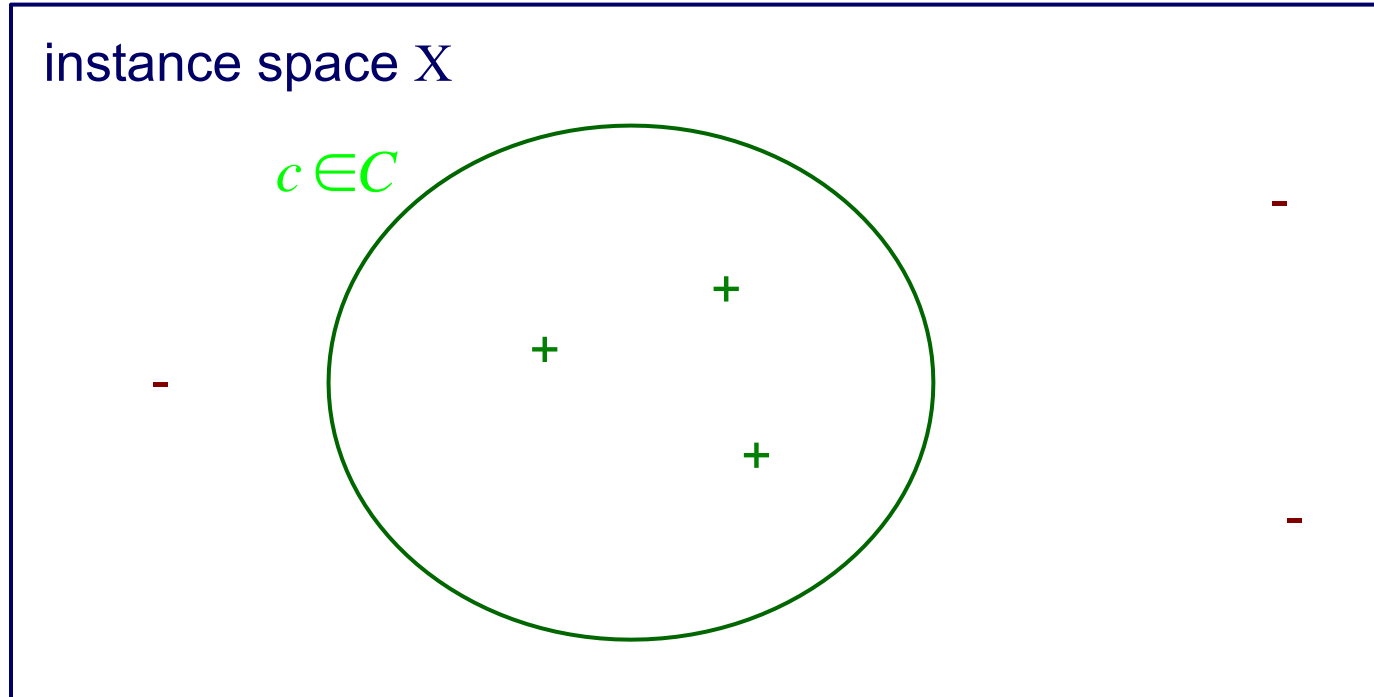
Spring 2019

Goals for the lecture

you should understand the following concepts

- PAC learnability
- consistent learners and version spaces
- sample complexity
- PAC learnability in the agnostic setting
- the VC dimension
- sample complexity using the VC dimension
- the on-line learning setting
- the mistake bound model of learnability
- the Halving algorithm
- the Weighted Majority algorithm

Learning setting #1



- set of instances X
- set of hypotheses (models) H
- set of possible target concepts C
- unknown probability distribution \mathcal{D} over instances

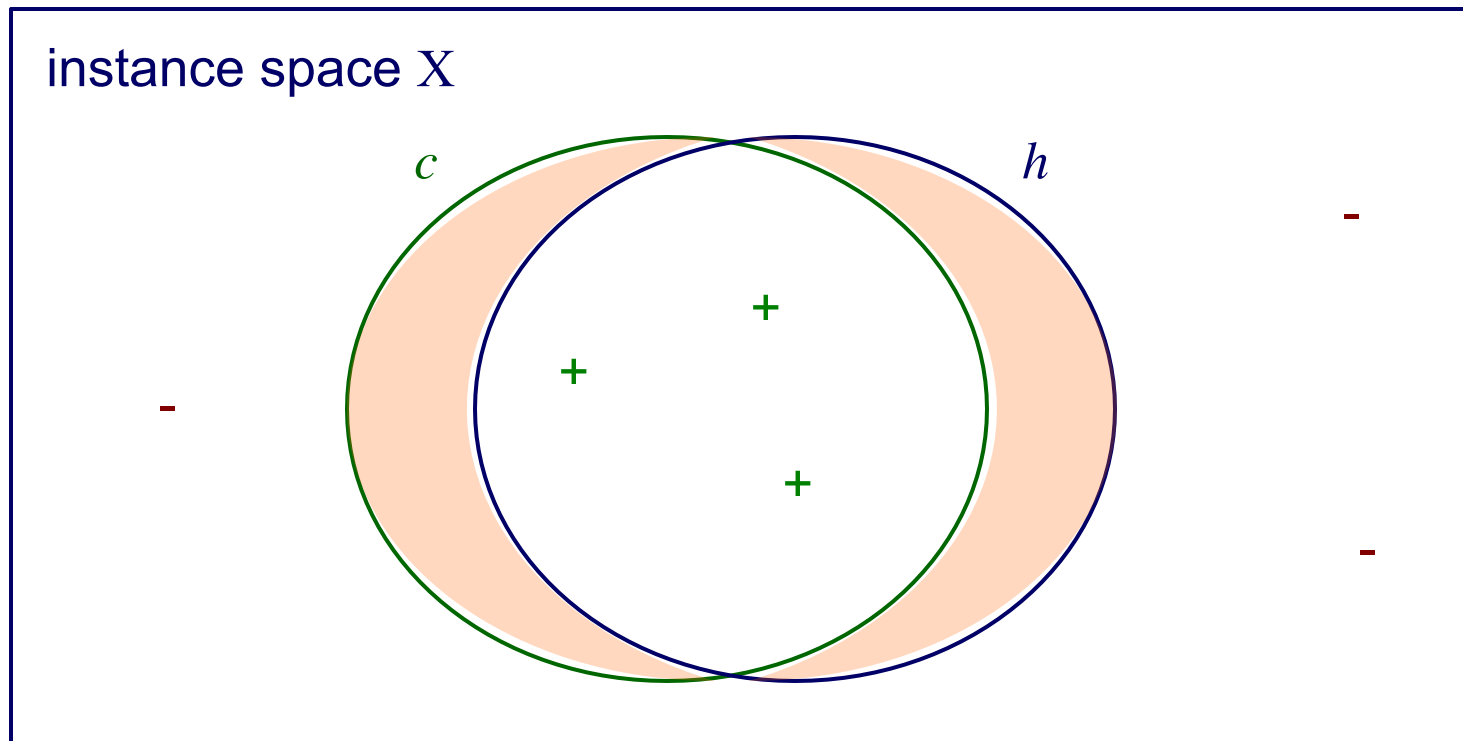
Learning setting #1

- learner is given a set D of training instances $\langle \mathbf{x}, c(\mathbf{x}) \rangle$ for some target concept c in C
 - each instance \mathbf{x} is drawn from distribution \mathcal{D}
 - class label $c(\mathbf{x})$ is provided for each \mathbf{x}
- learner outputs hypothesis h modeling c

True error of a hypothesis

the *true error* of hypothesis h refers to how often h is wrong on future instances drawn from probability distribution \mathcal{D}

$$\text{error}_{\mathcal{D}}(h) \equiv P_{\mathbf{x} \in \mathcal{D}} [c(\mathbf{x}) \neq h(\mathbf{x})]$$



Training error of a hypothesis

the *training error* of hypothesis h refers to how often h is wrong on instances in the training set D

$$error_D(h) \equiv P_{x \in D} [c(x) \neq h(x)] = \frac{\sum_{x \in D} \delta(c(x) \neq h(x))}{|D|}$$

Can we bound $error_{\mathcal{D}}(h)$ in terms of $error_D(h)$?

Is approximately correct good enough?



To say that our learner L has learned a concept, should we require $error_{\mathcal{D}}(h) = 0$?

this is not realistic:

- unless we've seen every possible instance, there may be multiple hypotheses that are consistent with the training set
- there is some chance our training sample will be unrepresentative

Probably approximately correct learning?



Instead, we'll require that

- the error of a learned hypothesis h is bounded by some constant ϵ
- the probability of the learner failing to learn an accurate hypothesis is bounded by a constant δ

Probably Approximately Correct (PAC) learning [Valiant, CACM 1984]

- Consider a class C of possible target concepts defined over a set of instances X of length n , and a learner L using hypothesis space H
- C is PAC learnable by L using H if, for *all*
 - $c \in C$
 - distributions \mathcal{D} over X
 - ε such that $0 < \varepsilon < 0.5$
 - δ such that $0 < \delta < 0.5$
- learner L will, with probability at least $(1-\delta)$, output a hypothesis $h \in H$ such that $error_{\mathcal{D}}(h) \leq \varepsilon$, provided time and sample size (from \mathcal{D}) polynomial in
 - $1/\varepsilon$
 - $1/\delta$
 - n
 - $size(c)$

PAC learning and consistency



- Suppose we can find hypotheses that are consistent with m training instances.
- We can analyze PAC learnability by determining whether
 1. m grows polynomially in the relevant parameters
 2. the processing time per training example is polynomial

Version spaces

- A hypothesis h is *consistent* with a set of training examples D of target concept c if and only if $h(\mathbf{x}) = c(\mathbf{x})$ for each training example $\langle \mathbf{x}, c(\mathbf{x}) \rangle$ in D

$$\text{consistent}(h, D) \equiv \left(\forall \langle \mathbf{x}, c(\mathbf{x}) \rangle \in D \right) h(\mathbf{x}) = c(\mathbf{x})$$

- The version space $VS_{H,D}$ with respect to hypothesis space H and training set D , is the subset of hypotheses from H consistent with all training examples in D

$$VS_{H,D} \equiv \{h \in H \mid \text{consistent}(h, D)\}$$



Exhausting the version space



- The version space $VS_{H,D}$ is ε -exhausted with respect to concept c and data set D if every hypothesis $h \in VS_{H,D}$ has true error $< \varepsilon$:

$$(\forall h \in VS_{H,D}) error_D(h) < \varepsilon$$

Exhausting the version space

- Suppose that every h in our version space $VS_{H,D}$ is consistent with m training examples
- The probability that $VS_{H,D}$ is not ε -exhausted (i.e. that it contains some hypotheses that are not accurate enough)

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

Proof: $(1 - \varepsilon)^m$ probability a particular hypothesis with error $> \varepsilon$ is consistent with m training instances

$k(1 - \varepsilon)^m$ there might be k such hypotheses

$|H|(1 - \varepsilon)^m$ k is bounded by $|H|$

$$\leq |H| e^{-\varepsilon m} \quad (1 - \varepsilon) \leq e^{-\varepsilon} \text{ when } 0 \leq \varepsilon \leq 1$$

Sample complexity for finite hypothesis spaces

[Blumer et al., *Information Processing Letters* 1987]

- choose m big enough to reduce this probability below δ

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

- solving for m we get desired result as long as:

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

log dependence on H  ε has stronger influence than δ 

PAC analysis example: learning conjunctions of Boolean literals

- each instance has n Boolean features
- learned hypotheses are of the form $Y = X_1 \wedge X_2 \wedge \neg X_5$

How many training examples suffice to ensure that with prob ≥ 0.99 , a consistent learner will return a hypothesis with error ≤ 0.05 ?

there are 3^n hypotheses (each variable can be present and unnegated, present and negated, or absent) in H

$$m \geq \frac{1}{.05} \left(\ln(3^n) + \ln\left(\frac{1}{.01}\right) \right)$$

for $n=10$, $m \geq 312$

for $n=100$, $m \geq 2290$

PAC analysis example: learning conjunctions of Boolean literals

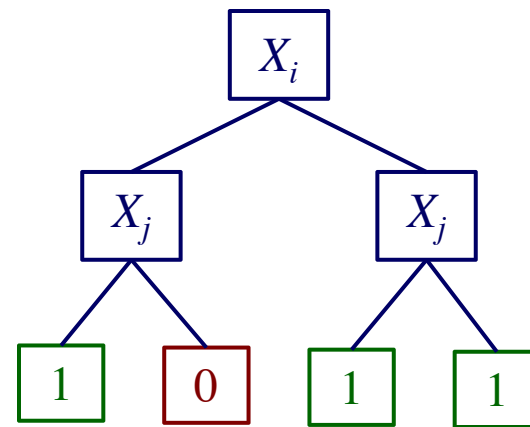
- we've shown that the sample complexity is polynomial in relevant parameters: $1/\epsilon$, $1/\delta$, n
- to prove that Boolean conjunctions are PAC learnable, need to also show that we can find a consistent hypothesis in polynomial time (the FIND-S algorithm in Mitchell, Chapter 2 does this)

FIND-S:

initialize h to the most specific hypothesis $l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \dots l_n \wedge \neg l_n$
for each positive training instance x
 remove from h any literal that is not satisfied by x
output hypothesis h

PAC analysis example: learning decision trees of depth 2

- each instance has n Boolean features
- learned hypotheses are DTs of depth 2 using only 2 variables



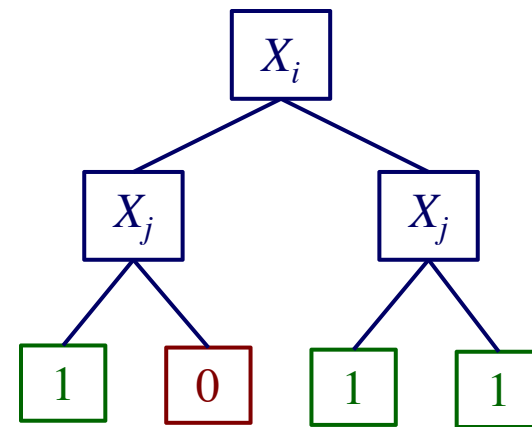
$$|H| = \binom{n}{2} \times 16 = \frac{n(n-1)}{2} \times 16 = 8n(n-1)$$

possible split choices

possible leaf labelings

PAC analysis example: learning decision trees of depth 2

- each instance has n Boolean features
- learned hypotheses are DTs of depth 2 using only 2 variables



How many training examples suffice to ensure that with prob ≥ 0.99 , a consistent learner will return a hypothesis with error ≤ 0.05 ?

$$m \geq \frac{1}{.05} \left(\ln(8n^2 - 8n) + \ln\left(\frac{1}{.01}\right) \right)$$

for $n=10$, $m \geq 224$

for $n=100$, $m \geq 318$

PAC analysis example:

K -term DNF is not PAC learnable

- each instance has n Boolean features
- learned hypotheses are of the form $Y = T_1 \vee T_2 \vee \dots \vee T_k$ where each T_i is a conjunction of n Boolean features or their negations

$|H| \leq 3^{nk}$, so sample complexity is polynomial in the relevant parameters

$$m \geq \frac{1}{\varepsilon} \left(nk \ln(3) + \ln\left(\frac{1}{\delta}\right) \right)$$

however, the computational complexity (time to find consistent h) is not polynomial in m (e.g. graph 3-coloring, an NP-complete problem, can be reduced to learning 3-term DNF)

Extensions, Results, Questions

- k -term DNF not properly PAC-learnable, but *PAC-predictable*, or PAC learnable in terms of k CNF
- negative results for PAC-predictability more robust
- results not based on NP-hardness of consistency problem, but on hard cryptographic problems (Kearns & Valiant, 1994)
 - can't PAC-learn Boolean formulae (unless can crack RSA)
 - can't PAC-learn deterministic finite state machines (same)
- open PAC-learning questions include
 - DNF formulae
 - decision trees

What if the target concept is not in our hypothesis space?

- so far, we've been assuming that the target concept c is in our hypothesis space; this is not a very realistic assumption
- even if it is, might want to learn using another class (e.g., kCNF)
- *agnostic learning* setting
 - don't assume $c \in H$
 - learner returns hypothesis h that makes fewest errors on training data

- how many training instances suffice to ensure that $error_{\mathcal{D}}(h) \leq error_{\mathcal{D}}(h) + \varepsilon$?

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

What if the hypothesis space is not finite?

- **Q:** If H is infinite (e.g. the class of intervals on the real line), what measure of hypothesis-space complexity can we use in place of $|H|$?
- **A:** the largest subset of X for which H can guarantee zero training error, regardless of the target function.

this is known as the *Vapnik-Chervonenkis dimension* (VC-dimension)

Shattering and the VC dimension

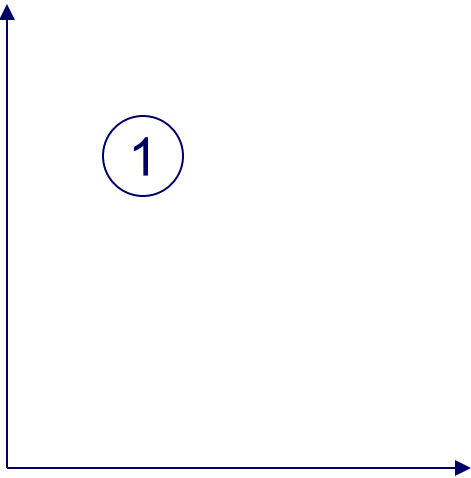


- a set of instances D is *shattered* by a hypothesis space H iff for every dichotomy of D there is a hypothesis in H consistent with this dichotomy
- the *VC dimension* of H defined over instance space X is the size of the largest finite subset of X shattered by H

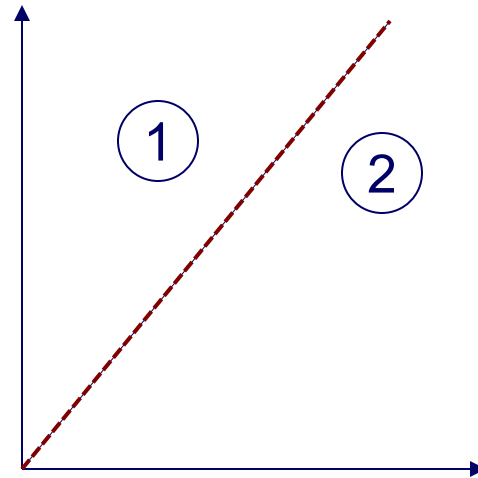
An infinite hypothesis space with a finite VC dimension

consider: H is set of lines (linear separators) in 2D

can find an h consistent with 1 instance no matter how it's labeled



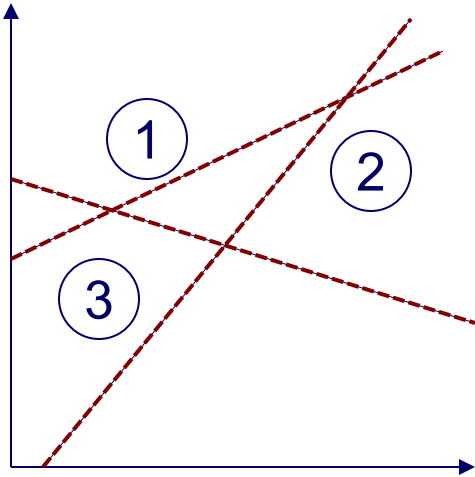
can find an h consistent with 2 instances no matter labeling



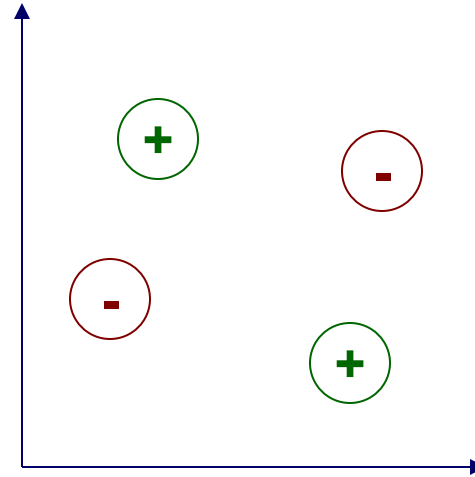
An infinite hypothesis space with a finite VC dimension

consider: H is set of lines in 2D

can find an h consistent with 3 instances no matter labeling (assuming they're not colinear)



cannot find an h consistent with 4 instances for some labelings

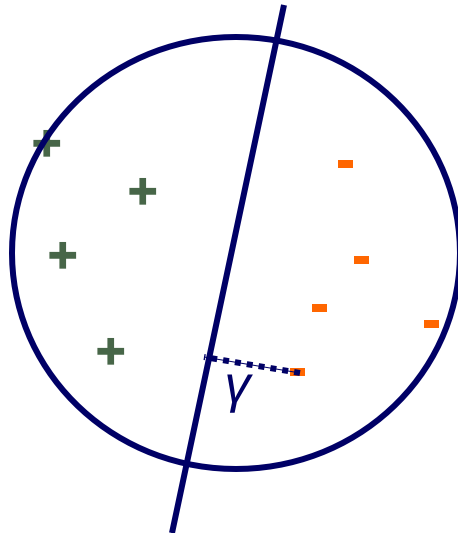


can shatter 3 instances, but not 4 \setminus the $\text{VC-dim}(H) = 3$

more generally, the VC-dim of hyperplanes in n dimensions = $n+1$

Interesting aside

- VC-dim of hyperplane in n dimension is $n+1$
- Let R be radius of smallest hypersphere circumscribing the data, and let γ (margin) be smallest distance of any data point to hyperplane
- Can replace n in VC-dim with $(R/\gamma)^2$ if smaller



VC dimension for finite hypothesis spaces

for finite H , $\text{VC-dim}(H) \leq \log_2 |H|$

Proof:

suppose $\text{VC-dim}(H) = d$

for d instances, 2^d different labelings possible

therefore H must be able to represent 2^d hypotheses

$$2^d \leq |H|$$

$$d = \text{VC-dim}(H) \leq \log_2 |H|$$

Sample complexity and the VC dimension

- using $\text{VC-dim}(H)$ as a measure of complexity of H , we can derive the following bound [Blumer et al., *JACM* 1989]

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

m grows $\log \times$ linear in ε

can be used for both finite and infinite hypothesis spaces

Lower bound on sample complexity

[Ehrenfeucht et al., *Information & Computation* 1989]

- there exists a distribution \mathcal{D} and target concept in C such that if the number of training instances given to L is

$$m < \max \left[\frac{1}{\varepsilon} \log \left(\frac{1}{\delta} \right), \frac{\text{VC-dim}(C) - 1}{32\varepsilon} \right]$$

then with probability at least δ , L outputs h such that $\text{error}_{\mathcal{D}}(h) > \varepsilon$

Comments on PAC learning

- PAC analysis formalizes the learning task and allows for non-perfect learning (indicated by ε and δ)
- finding a consistent hypothesis is sometimes easier for larger concept classes (*PAC-prediction*)
 - e.g. although k -term DNF is not PAC learnable, the more general class k -CNF is
- PAC analysis has been extended to explore a wide range of cases
 - noisy training data
 - learner allowed to ask queries (active learning)
 - restricted distributions (e.g. uniform) \mathcal{D}
- most analyses are worst case -> negative results, very restricted concept classes for positive results
- sample complexity bounds are generally not tight
- contributed major insights to ensembles, active learning, SVMs, ...

Learning setting #2: on-line learning

Now let's consider learning in the *on-line* learning setting:

for $t = 1 \dots$

- learner receives instance x_t

- learner predicts $h(x_t)$

- learner receives label $c(x_t)$ and updates model h

The *mistake bound* model of learning

How many mistakes will an on-line learner make in its predictions before it learns the target concept?

the *mistake bound model* of learning addresses this question

Mistake bound example: learning conjunctions with FIND-S

consider the learning task

- training instances are represented by n Boolean features
- target concept is conjunction of up to n Boolean literals (variable or its negation)

FIND-S:

initialize h to the most specific hypothesis $l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \dots l_n \wedge \neg l_n$

for each positive training instance x

 remove from h any literal that is not satisfied by x

output hypothesis h

Example: using FIND-S to learn conjunctions

- suppose we're learning a concept representing the sports someone likes
- instances are represented using Boolean feature that characterize the sport

Snow (is it done on snow?)

Water

Road

Mountain

Skis

Board

Ball (does it involve a ball?)

Example: using FIND-S to learn conjunctions

$t = 0$ $h:$ $snow \wedge \neg snow \wedge water \wedge \neg water \wedge road \wedge \neg road \wedge$
 $mountain \wedge \neg mountain \wedge skis \wedge \neg skis \wedge board$
 $\wedge \neg board \wedge ball \wedge \neg ball$

$t = 1$ $x:$ $snow, \neg water, \neg road, mountain, skis, \neg board, \neg ball$
 $h(x) = \text{false}$ $c(x) = \text{true}$
 $h:$ $snow \wedge \neg water \wedge \neg road \wedge mountain \wedge skis \wedge \neg board \wedge \neg ball$

$t = 2$ $x:$ $snow, \neg water, \neg road, \neg mountain, skis, \neg board, \neg ball$
 $h(x) = \text{false}$ $c(x) = \text{false}$

$t = 3$ $x:$ $snow, \neg water, \neg road, mountain, \neg skis, board, \neg ball$
 $h(x) = \text{false}$ $c(x) = \text{true}$
 $h:$ $snow \wedge \neg water \wedge \neg road \wedge mountain \wedge \neg ball$



Mistake bound example: learning conjunctions with FIND-S

the maximum # of mistakes FIND-S will make = $n + 1$

Proof:

- FIND-S will never mistakenly classify a negative (h is always at least as specific as the target concept)
- initial h has $2n$ literals
- the first mistake on a positive instance will reduce the initial hypothesis to n literals
- each successive mistake will remove at least one literal from h

Halving algorithm

// initialize the version space to contain all $h \in H$

$VS_1 \leftarrow H$

for $t \leftarrow 1$ to T do

 given training instance $\langle \mathbf{x}_t, c(\mathbf{x}_t) \rangle$

$h'(\mathbf{x}_t) = \text{MajorityVote}(VS_t, \mathbf{x}_t)$

 // eliminate all wrong h from version space (reduce the
 size of the VS by at least half on mistakes)

$VS_{t+1} \leftarrow \{h \in VS_t : h(\mathbf{x}_t) = c(\mathbf{x}_t)\}$

return VS_{t+1}

Mistake bound for the Halving algorithm

the maximum # of mistakes the Halving algorithm will make = $\lfloor \log_2 |H| \rfloor$

Proof:

- initial version space contains $|H|$ hypotheses
- each mistake reduces version space by at least half

$\lfloor a \rfloor$ is the largest integer
not greater than a




Optimal mistake bound

[Littlestone, *Machine Learning* 1987]

let C be an arbitrary concept class

$$VC(C) \leq M_{opt}(C) \leq M_{Halving}(C) \leq \log_2(|C|)$$

 # mistakes by best algorithm
(for hardest $c \in C$, and
hardest training sequence)

 # mistakes by Halving algorithm

The Weighted Majority algorithm

given: a set of predictors $A = \{a_1 \dots a_n\}$, learning rate $0 \leq \beta < 1$

```
for all  $i$  initialize  $w_i \leftarrow 1$ 
for each training instance  $\langle \mathbf{x}, c(\mathbf{x}) \rangle$ 
    initialize  $q_0$  and  $q_1$  to 0
    for each predictor  $a_i$ 
        if  $a_i(\mathbf{x}) = 0$  then  $q_0 \leftarrow q_0 + w_i$ 
        if  $a_i(\mathbf{x}) = 1$  then  $q_1 \leftarrow q_1 + w_i$ 
    if  $q_1 > q_0$  then  $h(\mathbf{x}) = 1$ 
    else if  $q_0 > q_1$  then  $h(\mathbf{x}) = 0$ 
    else if  $q_0 = q_1$  then  $h(\mathbf{x}) = 0$  or  $1$  randomly chosen

    for each predictor  $a_i$  do
        if  $a_i(\mathbf{x}) \neq c(\mathbf{x})$  then  $w_i \leftarrow \beta w_i$ 
```


The Weighted Majority algorithm

- predictors can be individual features or hypotheses or learning algorithms
- if the predictors are all of the $h \in H$, then WM is like a weighted voting version of the Halving algorithm
- WM learns a linear separator, like a perceptron
- weight updates are multiplicative instead of additive (as in perceptron/neural net training)
 - multiplicative is better when there are many features (predictors) but few are relevant
 - additive is better when many features are relevant
- approach can handle noisy training data

Notes

- Halving algorithm eliminates inconsistent predictors on *every* round
- Two versions of weighted majority
 - Original only down-weights predictors on rounds where overall prediction is wrong
 - Also a version that down-weights wrong predictors on every round
 - Following bound applies to both versions

Relative mistake bound for Weighted Majority

Let

- D be any sequence of training instances
- A be any set of n predictors
- k be minimum number of mistakes made by best predictor in A for training sequence D
- the number of mistakes over D made by Weighted Majority using $\beta = 1/2$ is at most

$$2.4(k + \log_2 n)$$

Any Efficient Mistake-Bounded Learner L for Concept Class C is a PAC Learner

- Run L on data
- Halt and return current h if right on $\frac{1}{\epsilon} \ln\left(\frac{M}{\delta}\right)$ consecutive examples, where M is mistake bound
- Efficient means each update is made in polynomial time
- Algorithm terminates within $\frac{M}{\epsilon} \ln\left(\frac{M}{\delta}\right)$ examples
- Blumer-like argument ensures low error with high confidence
- Note: converse not true. Consider learning single interval on real line.

Comments on mistake bound learning

- we've considered mistake bounds for learning the target concept exactly
- learning with polynomial mistake bound and polynomial update time implies PAC learning (can turn any such mistake bounded learner into a PAC learner), but not vice-versa
- learning from *Equivalence Queries*: telescope *MBB* to the mistakes
 - On every iteration, ask: is this h equivalent to the hidden target concept c ?
 - EQ oracle's answer is either "yes" or a counterexample \mathbf{x} where $h(\mathbf{x}) \neq c(\mathbf{x})$
- some of the algorithms developed in this line of research have had practical impact (e.g. Weighted Majority, Winnow) [Blum, *Machine Learning* 1997]