Computational Learning Theory

www.cs.wisc.edu/~dpage/cs760/
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 $\langle x, c(x) \rangle$ for some target concept $c$ in $C$
  - each instance $x$ is drawn from distribution $D$
  - class label $c(x)$ is provided for each $x$

- 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 \( \mathcal{D} \)

\[
error_{\mathcal{D}}(h) \equiv P_{x \in \mathcal{D}} \left[ c(x) \neq h(x) \right]
\]
Training error of a hypothesis

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

\[
\text{error}_D(h) \equiv P_{x \in D} \left[ c(x) \neq h(x) \right] = \frac{\sum_{x \in D} \delta(c(x) \neq h(x))}{|D|}
\]

Can we bound \( \text{error}_\mathcal{D}(h) \) in terms of \( \text{error}_D(h) \) ?
Is approximately correct good enough?

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

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 $\varepsilon$
- the probability of the learner failing to learn an accurate hypothesis is bounded by a constant $\delta$
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 $X$
  $\varepsilon$ such that $0 < \varepsilon < 0.5$
  $\delta$ such that $0 < \delta < 0.5$

• learner $L$ will, with probability at least $(1-\delta)$, output a hypothesis $h \in H$ such that $\text{error}_\mathcal{D}(h) \leq \varepsilon$, provided time and sample size (from $\mathcal{D}$) polynomial in
  $1/\varepsilon$
  $1/\delta$
  $n$
  $\text{size}(c)$
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 \textit{consistent} with a set of training examples $D$ of target concept $c$ if and only if $h(x) = c(x)$ for each training example $\langle x, c(x) \rangle$ in $D$

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

• 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 $D$

\[
VS_{H,D} \equiv \left\{ h \in H \mid \text{consistent}(h,D) \right\}
\]
Exhausting the version space

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

$$\left( \forall h \in V_{S_{H,D}} \right) error_D(h) < \varepsilon$$
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 \( \epsilon \)-exhausted (i.e. that it contains some hypotheses that are not accurate enough)

\[
\leq |H| e^{-\epsilon m}
\]

Proof:

\[
(1 - \epsilon)^m \quad \text{probability a particular hypothesis with error} > \epsilon
\]

\[
k(1 - \epsilon)^m \quad \text{is consistent with} \ m \ \text{training instances}
\]

\[
|H|(1 - \epsilon)^m \quad \text{there might be} \ k \ \text{such hypotheses}
\]

\[
|H|(1 - \epsilon)^m \quad k \ \text{is bounded by} \ |H|
\]

\[
\leq |H| e^{-\epsilon m} \quad (1 - \epsilon) \leq e^{-\epsilon} \ \text{when} \ 0 \leq \epsilon \leq 1
\]
Sample complexity for finite hypothesis spaces
[Blumer et al., Information Processing Letters 1987]

- choose $m$ big enough to reduce this probability below $\delta$

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

- 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 $H$  
$\varepsilon$ has stronger influence than $\delta$
PAC analysis example:
learning conjunctions of Boolean literals

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

How many training examples suffice to ensure that with $\text{prob} \geq 0.99$, a consistent learner will return a hypothesis with error $\leq 0.05$?

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: $1/\varepsilon, 1/\delta, n$

• 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 $l_1 \land \neg l_1 \land l_2 \land \neg l_2 \ldots \land l_n \land \neg l_n$

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 $\geq 0.99$, a consistent learner will return a hypothesis with error $\leq 0.05$?

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

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 \( Y = T_1 \lor T_2 \lor \ldots \lor T_k \) 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 \textit{PAC-predictable}, or PAC learnable in terms of kCNF

- 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 \( \text{error}_D(h) \leq \text{error}_D(h) + \varepsilon \)?

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

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

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

then with probability at least $\delta$, $L$ outputs $h$ such that $\text{error}_D(h) > \epsilon$
Comments on PAC learning

• PAC analysis formalizes the learning task and allows for non-perfect learning (indicated by $\varepsilon$ and $\delta$)

• 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 $t = 1$ …

- 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 \( l_1 \land \neg l_1 \land l_2 \land \neg l_2 \ldots \land l_n \land \neg l_n \)

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

  \begin{itemize}
  \item \textit{Snow} (is it done on snow?)
  \item \textit{Water}
  \item \textit{Road}
  \item \textit{Mountain}
  \item \textit{Skis}
  \item \textit{Board}
  \item \textit{Ball} (does it involve a ball?)
  \end{itemize}
Example: using FIND-S to learn conjunctions

\[ t = 0 \quad h: \quad \text{snow} \land \neg\text{snow} \land \text{water} \land \neg\text{water} \land \text{road} \land \neg\text{road} \land \text{mountain} \land \neg\text{mountain} \land \text{skis} \land \neg\text{skis} \land \text{board} \land \neg\text{board} \land \text{ball} \land \neg\text{ball} \]

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

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

\[ t = 3 \quad x: \quad \text{snow}, \neg\text{water}, \neg\text{road}, \text{mountain}, \neg\text{skis}, \text{board}, \neg\text{ball} \]
\[ h(x) = \text{false} \quad c(x) = \text{true} \]
\[ h: \quad \text{snow} \land \neg\text{water} \land \neg\text{road} \land \text{mountain} \land \neg\text{ball} \]
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 \( \langle x_t, c(x_t) \rangle \)
    \( h'(x_t) = \text{MajorityVote}(VS_t, x_t) \)

    // eliminate all wrong \( h \) from version space (reduce the
    // size of the VS by at least half on mistakes)
    \( VS_{t+1} \leftarrow \{ h \in VS_t : h(x_t) = c(x_t) \} \)

return \( VS_{t+1} \)
Mistake bound for the Halving algorithm

the maximum # of mistakes the Halving algorithm will make = \left\lfloor \log_2 |H| \right\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 $a$
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 \( A = \{a_1 \ldots a_n\} \), learning rate \( 0 \leq \beta < 1 \)

for all \( i \) initialize \( w_i \leftarrow 1 \)

for each training instance \( \langle x, c(x) \rangle \)

initialize \( q_0 \) and \( q_1 \) to 0

for each predictor \( a_i \)

\[
\begin{align*}
\text{if } a_i(x) &= 0 \text{ then } q_0 &\leftarrow q_0 + w_i \\
\text{if } a_i(x) &= 1 \text{ then } q_1 &\leftarrow q_1 + w_i
\end{align*}
\]

if \( q_1 > q_0 \) then \( h(x) = 1 \)
else if \( q_0 > q_1 \) then \( h(x) = 0 \)
else if \( q_0 = q_1 \) then \( h(x) = 0 \) or \( 1 \) randomly chosen

for each predictor \( a_i \) do

\[
\text{if } a_i(x) \neq c(x) \text{ then } w_i \leftarrow \beta w_i
\]
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

\[
2.4(k + \log_2 n)
\]
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)
- some of the algorithms developed in this line of research have had practical impact (e.g. Weighted Majority, Winnow)