For Monday

• No reading
• No homework
Version Space
Find-S
Learning Theory

• Theorems that characterize classes of learning problems or specific algorithms in terms of computational complexity or \textit{sample complexity}, i.e. the number of training examples necessary or sufficient to learn hypotheses of a given accuracy.

• Complexity of a learning problem depends on:
  – Size or expressiveness of the hypothesis space.
  – Accuracy to which target concept must be approximated.
  – Probability with which the learner must produce a successful hypothesis.
  – Manner in which training examples are presented, e.g. randomly or by query to an oracle.
Types of Results

- **Learning in the limit**: Is the learner guaranteed to converge to the correct hypothesis in the limit as the number of training examples increases indefinitely?
- **Sample Complexity**: How many training examples are needed for a learner to construct (with high probability) a highly accurate concept?
- **Computational Complexity**: How much computational resources (time and space) are needed for a learner to construct (with high probability) a highly accurate concept?
  - High sample complexity implies high computational complexity, since learner at least needs to read the input data.
- **Mistake Bound**: Learning incrementally, how many training examples will the learner misclassify before constructing a highly accurate concept.
Learning in the Limit

• Given a continuous stream of examples where the learner predicts whether each one is a member of the concept or not and is then is told the correct answer, does the learner eventually converge to a correct concept and never make a mistake again?

• No limit on the number of examples required or computational demands, but must eventually learn the concept exactly.

• By simple enumeration, concepts from any known finite hypothesis space are learnable in the limit, although typically requires an exponential (or doubly exponential) number of examples and time.

• Class of total recursive (Turing computable) functions is not learnable in the limit.
Unlearnable Problem

• Identify the function underlying an ordered sequence of natural numbers \((t: \mathbb{N} \to \mathbb{N})\), guessing the next number in the sequence and then being told the correct value.

• For any given learning algorithm \(L\), there exists a function \(t(n)\) that it cannot learn in the limit.

Given the learning algorithm \(L\) as a Turing machine:

\[
\text{D} \xrightarrow{L} h(n)
\]

Construct a function it cannot learn:

\[
\begin{cases}
\langle t(0), t(1), \ldots, t(n-1) \rangle \\
t(n) < t(0), t(1), \ldots, t(n-1) + 1
\end{cases}
\]

Example Trace

<table>
<thead>
<tr>
<th>Oracle:</th>
<th>1</th>
<th>3</th>
<th>6</th>
<th>11</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learner:</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

\(h(n) = h(n-1) + n + 1\)
Learning in the Limit vs. PAC Model

• Learning in the limit model is too strong.
  – Requires learning correct exact concept

• Learning in the limit model is too weak
  – Allows unlimited data and computational resources.

• PAC Model
  – Only requires learning a *Probably Approximately Correct* Concept: Learn a decent approximation most of the time.
  – Requires polynomial sample complexity and computational complexity.
Cannot Learn Exact Concepts from Limited Data, Only Approximations
Cannot Learn Even Approximate Concepts from Pathological Training Sets
PAC Learning

• The only reasonable expectation of a learner is that with high probability it learns a close approximation to the target concept.

• In the PAC model, we specify two small parameters, \( \epsilon \) and \( \delta \), and require that with probability at least \( 1 - \delta \) a system learn a concept with error at most \( \epsilon \).
Formal Definition of PAC-Learnable

• Consider a concept class $C$ defined over an instance space $X$ containing instances of length $n$, and a learner, $L$, using a hypothesis space, $H$.

• $C$ is said to be **PAC-learnable** by $L$ using $H$ iff for all $c \in C$, distributions $D$ over $X$, $0<\varepsilon<0.5$, $0<\delta<0.5$; learner $L$ by sampling random examples from distribution $D$, will with probability at least $1-\delta$ output a hypothesis $h \in H$ such that $\text{error}_D(h) \leq \varepsilon$, in time polynomial in $1/\varepsilon$, $1/\delta$, $n$ and $\text{size}(c)$.

• Example:
  – $X$: instances described by $n$ binary features
  – $C$: conjunctive descriptions over these features
  – $H$: conjunctive descriptions over these features
  – $L$: most-specific conjunctive generalization algorithm (Find-S)
  – $\text{size}(c)$: the number of literals in $c$ (i.e. length of the conjunction).
Issues of PAC Learnability

• The computational limitation also imposes a polynomial constraint on the training set size, since a learner can process at most polynomial data in polynomial time.

• How to prove PAC learnability:
  – First, prove sample complexity of learning $C$ using $H$ is polynomial.
  – Second, prove that the learner can train on a polynomial-sized data set in polynomial time.

• To be PAC-learnable, there must be a hypothesis in $H$ with arbitrarily small error for every concept in $C$, generally $C \subseteq H$. 
Consistent Learners

• A learner $L$ using a hypothesis $H$ and training data $D$ is said to be a consistent learner if it always outputs a hypothesis with zero error on $D$ whenever $H$ contains such a hypothesis.

• By definition, a consistent learner must produce a hypothesis in the version space for $H$ given $D$.

• Therefore, to bound the number of examples needed by a consistent learner, we just need to bound the number of examples needed to ensure that the version-space contains no hypotheses with unacceptably high error.
ε-Exhausted Version Space

• The version space, VS_{H,D}, is said to be ε-exhausted iff every hypothesis in it has true error less than or equal to ε.

• In other words, there are enough training examples to guarantee than any consistent hypothesis has error at most ε.

• One can never be sure that the version-space is ε-exhausted, but one can bound the probability that it is not.

• Theorem 7.1 (Haussler, 1988): If the hypothesis space H is finite, and D is a sequence of m≥1 independent random examples for some target concept c, then for any 0≤ε≤1, the probability that the version space VS_{H,D} is not ε-exhausted is less than or equal to:

\[ |H| e^{-\epsilon m} \]
Sample Complexity Analysis

- Let $\delta$ be an upper bound on the probability of *not* exhausting the version space. So:

$$P(\text{consist}(H_{bad}, D)) \leq |H| e^{-\varepsilon m} \leq \delta$$

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

$$-\varepsilon m \leq \ln\left(\frac{\delta}{|H|}\right)$$

$$m \geq \left(-\ln\left(\frac{\delta}{|H|}\right)\right)/\varepsilon \quad \text{(flip inequality)}$$

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

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

• Therefore, any consistent learner, given at least:

\[ \left( \ln \frac{1}{\delta} + \ln |H| \right) / \varepsilon \]

equations will produce a result that is PAC.

• Just need to determine the size of a hypothesis space to instantiate this result for learning specific classes of concepts.

• This gives a **sufficient** number of examples for PAC learning, but **not** a **necessary** number. Several approximations like that used to bound the probability of a disjunction make this a gross over-estimate in practice.
Sample Complexity of Conjunction Learning

• Consider conjunctions over $n$ boolean features. There are $3^n$ of these since each feature can appear positively, appear negatively, or not appear in a given conjunction. Therefore $|H|=3^n$, so a sufficient number of examples to learn a PAC concept is:

$$\left( \ln \frac{1}{\delta} + \ln 3^n \right) / \varepsilon = \left( \ln \frac{1}{\delta} + n \ln 3 \right) / \varepsilon$$

• Concrete examples:
  – $\delta=\varepsilon=0.05$, $n=10$ gives 280 examples
  – $\delta=0.01$, $\varepsilon=0.05$, $n=10$ gives 312 examples
  – $\delta=\varepsilon=0.01$, $n=10$ gives 1,560 examples
  – $\delta=\varepsilon=0.01$, $n=50$ gives 5,954 examples

• Result holds for any consistent learner.
Sample Complexity of Learning Arbitrary Boolean Functions

- Consider any boolean function over $n$ boolean features such as the hypothesis space of DNF or decision trees. There are $2^{2^n}$ of these, so a sufficient number of examples to learn a PAC concept is:

$$\left(\ln \frac{1}{\delta} + \ln 2^{2^n}\right)/\varepsilon = \left(\ln \frac{1}{\delta} + 2^n \ln 2\right)/\varepsilon$$

- Concrete examples:
  - $\delta=\varepsilon=0.05$, $n=10$ gives 14,256 examples
  - $\delta=\varepsilon=0.05$, $n=20$ gives 14,536,410 examples
  - $\delta=\varepsilon=0.05$, $n=50$ gives $1.561 \times 10^{16}$ examples
Other Concept Classes

- **$k$-term DNF**: Disjunctions of at most $k$ unbounded conjunctive terms: $T_1 \lor T_2 \lor \cdots \lor T_k$
  
  - $\ln(|H|) = O(kn)$

- **$k$-DNF**: Disjunctions of any number of terms each limited to at most $k$ literals: $((L_1 \land L_2 \land \cdots \land L_k) \lor (M_1 \land M_2 \land \cdots \land M_k)) \lor \cdots$
  
  - $\ln(|H|) = O(n^k)$

- **$k$-clause CNF**: Conjunctions of at most $k$ unbounded disjunctive clauses: $C_1 \land C_2 \land \cdots \land C_k$
  
  - $\ln(|H|) = O(kn)$

- **$k$-CNF**: Conjunctions of any number of clauses each limited to at most $k$ literals: $((L_1 \lor L_2 \lor \cdots \lor L_k) \land (M_1 \lor M_2 \lor \cdots \lor M_k)) \land \cdots$
  
  - $\ln(|H|) = O(n^k)$

Therefore, all of these classes have polynomial sample complexity given a fixed value of $k$. 
Basic Combinatorics Counting

<table>
<thead>
<tr>
<th></th>
<th>dups allowed</th>
<th>dups not allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>order relevant</td>
<td>samples</td>
<td>permutations</td>
</tr>
<tr>
<td>order irrelevant</td>
<td>selections</td>
<td>combinations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>samples</th>
<th>permutations</th>
<th>selections</th>
<th>combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>ab</td>
<td>aa</td>
<td>ab</td>
</tr>
<tr>
<td>ab</td>
<td>ba</td>
<td>ab</td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td></td>
<td>bb</td>
<td></td>
</tr>
<tr>
<td>bb</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pick 2 from \{a,b\}

**k - samples:** \(n^k\)

**k - permutations:** \(\frac{n!}{(n-k)!}\)

**k - selections:** \(\binom{n+k-1}{k} = \frac{(n+k-1)!}{k!(n-1)!}\)

**k - combinations:** \(\binom{n}{k} = \frac{n!}{k!(n-k)!}\)

All \(O(n^k)\)
Computational Complexity of Learning

- However, determining whether or not there exists a $k$-term DNF or $k$-clause CNF formula consistent with a given training set is NP-hard. Therefore, these classes are not PAC-learnable due to computational complexity.

- There are polynomial time algorithms for learning $k$-CNF and $k$-DNF. Construct all possible disjunctive clauses (conjunctive terms) of at most $k$ literals (there are $O(n^k)$ of these), add each as a new constructed feature, and then use FIND-S (FIND-G) to find a purely conjunctive (disjunctive) concept in terms of these complex features.

Sample complexity of learning $k$-DNF and $k$-CNF are $O(n^k)$
Training on $O(n^k)$ examples with $O(n^k)$ features takes $O(n^{2k})$ time
Enlarging the Hypothesis Space to Make Training Computation Tractable

• However, the language $k$-CNF is a superset of the language $k$-term-DNF since any $k$-term-DNF formula can be rewritten as a $k$-CNF formula by distributing AND over OR.

• Therefore, $C = k$-term DNF can be learned using $H = k$-CNF as the hypothesis space, but it is intractable to learn the concept in the form of a $k$-term DNF formula (also the $k$-CNF algorithm might learn a close approximation in $k$-CNF that is not actually expressible in $k$-term DNF).
  
  – Can gain an exponential decrease in computational complexity with only a polynomial increase in sample complexity.

• Dual result holds for learning $k$-clause CNF using $k$-DNF as the hypothesis space.
Probabilistic Algorithms

- Since PAC learnability only requires an approximate answer with *high probability*, a probabilistic algorithm that only halts and returns a consistent hypothesis in polynomial time with a high-probability is sufficient.

- However, it is generally assumed that NP complete problems cannot be solved even with high probability by a probabilistic polynomial-time algorithm, i.e. \( \text{RP} \neq \text{NP} \).

- Therefore, given this assumption, classes like \( k \)-term DNF and \( k \)-clause CNF are not PAC learnable in that form.
Infinite Hypothesis Spaces

• The preceding analysis was restricted to finite hypothesis spaces.
• Some infinite hypothesis spaces (such as those including real-valued thresholds or parameters) are more expressive than others.
  – Compare a rule allowing one threshold on a continuous feature (length<3cm) vs one allowing two thresholds (1cm<length<3cm).
• Need some measure of the expressiveness of infinite hypothesis spaces.
• The Vapnik-Chervonenkis (VC) dimension provides just such a measure, denoted VC(H).
• Analagous to ln|H|, there are bounds for sample complexity using VC(H).
Shattering Instances

• A hypothesis space is said to shatter a set of instances iff for every partition of the instances into positive and negative, there is a hypothesis that produces that partition.

• For example, consider 2 instances described using a single real-valued feature being shattered by intervals.
Shattering Instances (cont)

• But 3 instances cannot be shattered by a single interval.

Cannot do

• Since there are $2^m$ partitions of $m$ instances, in order for $H$ to shatter instances: $|H| \geq 2^m$. 
VC Dimension

• An unbiased hypothesis space shatters the entire instance space.
• The larger the subset of $X$ that can be shattered, the more expressive the hypothesis space is, i.e. the less biased.
• The Vapnik-Chervonenkis 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 subsets of $X$ can be shattered then $VC(H) = \infty$
• If there exists at least one subset of $X$ of size $d$ that can be shattered then $VC(H) \geq d$. If no subset of size $d$ can be shattered, then $VC(H) < d$.
• For a single intervals on the real line, all sets of 2 instances can be shattered, but no set of 3 instances can, so $VC(H) = 2$.
• Since $|H| \geq 2^m$, to shatter $m$ instances, $VC(H) \leq \log_2 |H|$
Consider axis-parallel rectangles in the real-plane, i.e. conjunctions of intervals on two real-valued features. Some 4 instances can be shattered.

Some 4 instances cannot be shattered:
VC Dimension Example (cont)

- No five instances can be shattered since there can be at most 4 distinct extreme points (min and max on each of the 2 dimensions) and these 4 cannot be included without including any possible 5\(^{th}\) point.

\[
\text{Therefore } VC(H) = 4
\]

- Generalizes to axis-parallel hyper-rectangles (conjunctions of intervals in } n \text{ dimensions): } VC(H) = 2n.
Upper Bound on Sample Complexity with VC

- Using VC dimension as a measure of expressiveness, the following number of examples have been shown to be sufficient for PAC Learning (Blumer et al., 1989):
  \[
  \frac{1}{\varepsilon} \left( 4 \log_2 \left( \frac{2}{\delta} \right) + 8 \text{VC}(H) \log_2 \left( \frac{13}{\varepsilon} \right) \right)
  \]

- Compared to the previous result using \( \ln |H| \), this bound has some extra constants and an extra \( \log_2(1/\varepsilon) \) factor. Since \( \text{VC}(H) \leq \log_2 |H| \), this can provide a tighter upper bound on the number of examples needed for PAC learning.
Conjunctive Learning with Continuous Features

- Consider learning axis-parallel hyper-rectangles, conjunctions on intervals on \( n \) continuous features.
  - \( 1.2 \leq \text{length} \leq 10.5 \land 2.4 \leq \text{weight} \leq 5.7 \)
- Since \( \text{VC}(H)=2n \) sample complexity is
  \[
  \frac{1}{\varepsilon} \left( 4 \log_2 \left( \frac{2}{\delta} \right) + 16n \log_2 \left( \frac{13}{\varepsilon} \right) \right)
  \]
- Since the most-specific conjunctive algorithm can easily find the tightest interval along each dimension that covers all of the positive instances \( (f_{\text{min}} \leq f \leq f_{\text{max}}) \) and runs in linear time, \( O(|D|n) \), axis-parallel hyper-rectangles are PAC learnable.
Sample Complexity Lower Bound with VC

• There is also a general lower bound on the minimum number of examples necessary for PAC learning (Ehrenfeucht, et al., 1989):

Consider any concept class $C$ such that $\text{VC}(H) \geq 2$ any learner $L$ and any $0 < \varepsilon < 1/8$, $0 < \delta < 1/100$. Then there exists a distribution $D$ and target concept in $C$ such that if $L$ observes fewer than:

$$\max\left(\frac{1}{\varepsilon} \log_2 \left(\frac{1}{\delta}\right), \frac{\text{VC}(C) - 1}{32\varepsilon}\right)$$

examples, then with probability at least $\delta$, $L$ outputs a hypothesis having error greater than $\varepsilon$.

• Ignoring constant factors, this lower bound is the same as the upper bound except for the extra $\log_2(1/\varepsilon)$ factor in the upper bound.
Analyzing a Preference Bias

• Unclear how to apply previous results to an algorithm with a preference bias such as simplest decisions tree or simplest DNF.
• If the size of the correct concept is $n$, and the algorithm is guaranteed to return the minimum sized hypothesis consistent with the training data, then the algorithm will always return a hypothesis of size at most $n$, and the effective hypothesis space is all hypotheses of size at most $n$.

• Calculate $|H|$ or $VC(H)$ of hypotheses of size at most $n$ to determine sample complexity.
Computational Complexity and Preference Bias

• However, finding a minimum size hypothesis for most languages is computationally intractable.

• If one has an approximation algorithm that can bound the size of the constructed hypothesis to some polynomial function, $f(n)$, of the minimum size $n$, then can use this to define the effective hypothesis space.

• However, no worst case approximation bounds are known for practical learning algorithms (e.g. ID3).
“Occam’s Razor” Result  
(Blumer et al., 1987)

• Assume that a concept can be represented using at most \( n \) bits in some representation language.
• Given a training set, assume the learner returns the consistent hypothesis representable with the least number of bits in this language.
• Therefore the effective hypothesis space is all concepts representable with at most \( n \) bits.
• Since \( n \) bits can code for at most \( 2^n \) hypotheses, \(|H|=2^n\), so sample complexity is bounded by:

\[
\left( \ln \frac{1}{\delta} + \ln 2^n \right) / \varepsilon = \left( \ln \frac{1}{\delta} + n \ln 2 \right) / \varepsilon
\]

• This result can be extended to approximation algorithms that can bound the size of the constructed hypothesis to at most \( n^k \) for some fixed constant \( k \) (just replace \( n \) with \( n^k \))
Interpretation of “Occam’s Razor” Result

- Since the encoding is unconstrained it fails to provide any meaningful definition of “simplicity.”
- Hypothesis space could be any sufficiently small space, such as “the $2^n$ most complex boolean functions, where the complexity of a function is the size of its smallest DNF representation”
- Assumes that the correct concept (or a close approximation) is actually in the hypothesis space, so assumes *a priori* that the concept is simple.
- Does not provide a theoretical justification of Occam’s Razor as it is normally interpreted.
COLT Conclusions

• The PAC framework provides a theoretical framework for analyzing the effectiveness of learning algorithms.
• The sample complexity for any consistent learner using some hypothesis space, $H$, can be determined from a measure of its expressiveness $|H|$ or $VC(H)$, quantifying bias and relating it to generalization.
• If sample complexity is tractable, then the computational complexity of finding a consistent hypothesis in $H$ governs its PAC learnability.
• Constant factors are more important in sample complexity than in computational complexity, since our ability to gather data is generally not growing exponentially.
• Experimental results suggest that theoretical sample complexity bounds over-estimate the number of training instances needed in practice since they are worst-case upper bounds.
• Additional results produced for analyzing:
  – Learning with queries
  – Learning with noisy data
  – Average case sample complexity given assumptions about the data distribution.
  – Learning finite automata
  – Learning neural networks

• Analyzing practical algorithms that use a preference bias is difficult.

• Some effective practical algorithms motivated by theoretical results:
  – Winnow
  – Boosting
  – Support Vector Machines (SVM)