Let us know if you have any questions. (email the TAs)

Grades Distribution

<table>
<thead>
<tr>
<th>Grade</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>A+</td>
<td>10</td>
</tr>
<tr>
<td>A</td>
<td>32</td>
</tr>
<tr>
<td>A-</td>
<td>23</td>
</tr>
<tr>
<td>B+</td>
<td>44</td>
</tr>
<tr>
<td>B</td>
<td>25</td>
</tr>
<tr>
<td>B-</td>
<td>7</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Grade</th>
<th>Count Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>A+</td>
<td>7%</td>
</tr>
<tr>
<td>A</td>
<td>22.5%</td>
</tr>
<tr>
<td>A-</td>
<td>16.1%</td>
</tr>
<tr>
<td>B+</td>
<td>30.2%</td>
</tr>
<tr>
<td>B</td>
<td>17.6%</td>
</tr>
<tr>
<td>B-</td>
<td>4.9%</td>
</tr>
<tr>
<td>C</td>
<td>0.7%</td>
</tr>
<tr>
<td>D</td>
<td>0.7%</td>
</tr>
</tbody>
</table>

Total Count: 161
HW3

Lessons:
- Understand Perceptron (x2), Winnow (x2), Adagrad
- Understand how to run ML experiments
  - Tuning Parameters; Comparisons

Your Assignment: Learn (l, m, n) functions
- Implement Algorithms
- Generate Data \((\text{Given})\)
- Think (Plan)
- Run three experiments. In each experiment:
  - Generate Data
  - Tune Parameters:
    - For each training dataset, For each algorithm,
  - Run experiment; plot/table results
  - Think and discuss what you see
Projects

- Term papers/Projects proposals are due on Friday, 10/16/15.
- Within a week we will give you an approval to continue with your project along with comments and/or a request to modify/augment/do a different project.
- Please start thinking and working on the project now; your proposal is limited to 1-2 pages, but needs to include references and, ideally, some of the ideas you have developed in the direction of the project so, ideally, you want you have some preliminary results.
- Any project that has a significant Machine Learning component is good.
- You may do experimental work, theoretical work, a combination of both or a critical survey of results in some specialized topic.
- The work has to include some reading. Even if you do not do a survey, you must read (at least) two related papers or book chapters and relate your work to it.
- Originality is not mandatory but is encouraged.
- Try to make it interesting!
Examples

Identify Multi-Word Expressions

Parse Multi-Word Expressions
Adj-Noun; Noun-Noun

Conference Program Chair Aid
• Identify if the paper is anonymous
• Author identification (especially difficult for multi-authored papers)
• Assign Editor/Reviewer
• Choose a (diverse) Program Committee

Twitter
• Identify Gender
• Identify Age
• Parse #hashtags: (&explain)
  • #dontwantmygradprogramfollowingme
  • #Rothhashana

Footstep recognition can ID you at 10 paces

Read more: Click here to read a longer version

Conference Program Chair Aid
• Identify if the paper is anonymous
• Author identification (especially difficult for multi-authored papers)
• Assign Editor/Reviewer
• Choose a (diverse) Program Committee

21 September 2011 by Lisa Grossman
Magazine issue 2831. Subscribe and

Where are we?

- Algorithmically:
  - Perceptron + Winnow
  - Gradient Descent

- Models:
  - Online Learning; Mistake Driven Learning

What do we know about Generalization? (to previously unseen examples?)
- How will your algorithm do on the next example?

Next we develop a theory of Generalization.
- We will come back to the same (or very similar) algorithms and show how the new theory sheds light on appropriate modifications of them, and provides guarantees.
Computational Learning Theory

- What general laws constrain inductive learning?
  - What learning problems can be solved?
  - When can we trust the output of a learning algorithm?

- We seek theory to relate
  - Probability of successful Learning
  - Number of training examples
  - Complexity of hypothesis space
  - Accuracy to which target concept is approximated
  - Manner in which training examples are presented
Quantifying Performance

- We want to be able to say something rigorous about the performance of our learning algorithm.

- We will concentrate on discussing the number of examples one needs to see before we can say that our learned hypothesis is good.
Learning Conjunctions

There is a hidden conjunction the learner (you) is to learn

\[ f = x_2 \land x_3 \land x_4 \land x_5 \land x_{100} \]

How many examples are needed to learn it? How?

- Protocol I: The learner proposes instances as queries to the teacher
- Protocol II: The teacher (who knows f) provides training examples
- Protocol III: Some random source (e.g., Nature) provides training examples; the Teacher (Nature) provides the labels (f(x))
Learning Conjunctions

- **Protocol I**: The learner proposes instances as queries to the teacher.

  Since we know we are after a **monotone conjunction**:

  - Is $x_{100}$ in? $<(1,1,1...,1,0), ?> f(x)=0$ (conclusion: Yes)
  - Is $x_{99}$ in? $<(1,1,...1,0,1), ?> f(x)=1$ (conclusion: No)
  - Is $x_{1}$ in? $<(0,1,...1,1,1), ?> f(x)=1$ (conclusion: No)

  A straightforward algorithm requires $n=100$ queries, and will produce as a result the hidden conjunction (exactly).

  $$ h = x_2 \land x_3 \land x_4 \land x_5 \land x_{100} $$

  What happens here if the conjunction is not known to be monotone? If we know of a positive example, the same algorithm works.
Learning Conjunctions

- **Protocol II:** The teacher (who knows \( f \)) provides training examples

- \( \langle 0,1,1,1,1,0,...,0,1 \rangle, 1 \rangle \) (We learned a superset of the good variables)

- To show you that all these variables are required...
  - \( \langle 0,0,1,1,1,0,...,0,1 \rangle, 0 \rangle \) need \( x_2 \)
  - \( \langle 0,1,0,1,1,0,...,0,1 \rangle, 0 \rangle \) need \( x_3 \)
  - .....  
  - \( \langle 0,1,1,1,1,0,...,0,0 \rangle, 0 \rangle \) need \( x_{100} \)

- A straightforward algorithm requires \( k = 6 \) examples to produce the hidden conjunction (exactly).

\[
f = x_2 \land x_3 \land x_4 \land x_5 \land x_{100}
\]
Learning Conjunctions

- **Protocol III**: Some random source (e.g., Nature) provides training examples
- **Teacher (Nature) provides the labels (f(x))**
  - $<(1,1,1,1,1,1,...,1,1), 1>$$
  - $<(1,1,1,0,0,0,...,0,0), 0>$$
  - $<(1,1,1,1,0,...,0,0,1), 1>$$
  - $<(1,0,1,1,1,0,...,0,1,1), 0>$$
  - $<(1,1,1,1,0,...,0,1,1), 1>$$
  - $<(1,0,1,0,0,0,...,0,1,1), 0>$$
  - $<(1,1,1,1,1,...,0,0,1), 1>$$
  - $<(0,1,0,1,0,0,...0,1,1), 0>$$
Learning Conjunctions

- **Protocol III:** Some random source (e.g., Nature) provides training examples
  - Teacher (Nature) provides the labels \( f(x) \)

**Algorithm: Elimination**
- Start with the set of all literals as candidates
- Eliminate a literal that is not active (0) in a positive example
- \(<(1,1,1,1,1,...,1,1), 1> \)
  \[ f = x_1 \land x_2 \land x_3 \land x_4 \land x_5 \land ... \land x_{100} \]
- \(<(1,1,0,0,0,...,0,0), 0> \) learned nothing
- \(<(1,1,1,1,1,0,...0,1,1), 1> \)
  \[ f = x_1 \land x_2 \land x_3 \land x_4 \land x_5 \land x_{99} \land x_{100} \]
- \(<(1,0,1,1,0,...0,0,1), 0> \) learned nothing
- \(<(1,1,1,1,1,0,...0,0,1), 1> \)
- \(<(1,0,1,0,0,...0,1,1), 0> \)
- \(<(1,1,1,1,1,...,0,1), 1> \)
- \(<(0,1,0,1,0,0,...0,1,1), 0> \)

Final hypothesis:

\[ h = x_1 \land x_2 \land x_3 \land x_4 \land x_5 \land x_{100} \]

Is that good? Performance? # of examples?
Two Directions

- Can continue to analyze the probabilistic intuition:
  - Never saw $x_1$ in positive examples, maybe we’ll never see it?
  - And if we will, it will be with small probability, so the concepts we learn may be pretty good
  - **Good**: in terms of performance on future data
  - PAC framework

- Mistake Driven Learning algorithms
  - Update your hypothesis only when you make mistakes
  - **Good**: in terms of how many mistakes you make before you stop, happy with your hypothesis.
  - Note: not all on-line algorithms are mistake driven, so performance measure could be different.
Prototypical Concept Learning

- **Instance Space**: $X$
  - Examples

- **Concept Space**: $C$
  - Set of possible target functions: $f \in C$ is the hidden target function
  - All n-conjunctions; all n-dimensional linear functions.

- **Hypothesis Space**: $H$ set of possible hypotheses

- **Training instances** $S \times \{0,1\}$: positive and negative examples of the target concept $f \in C$
  \[
  < x_1, f(x_1) >, < x_2, f(x_2) >, \ldots, < x_n, f(x_n) >
  \]

- **Determine**: A hypothesis $h \in H$ such that $h(x) = f(x)$

- A hypothesis $h \in H$ such that $h(x) = f(x)$ \text{ for all } $x \in S$?

- A hypothesis $h \in H$ such that $h(x) = f(x)$ \text{ for all } $x \in X$?

\[
 h = x_1 \land x_2 \land x_3 \land x_4 \land x_5 \land x_{100}
\]
Prototypical Concept Learning

- Instance Space: $X$
  - Examples

- Concept Space: $C$
  - Set of possible target functions: $f \in C$ is the hidden target function
  - All $n$-conjunctions; all $n$-dimensional linear functions.

- Hypothesis Space: $H$ set of possible hypotheses

- Training instances $S \times \{0,1\}$: positive and negative examples of the target concept $f \in C$. Training instances are generated by a fixed unknown probability distribution $D$ over $X$

  \[ \langle x_1, f(x_1) \rangle, \langle x_2, f(x_2) \rangle, \ldots, \langle x_n, f(x_n) \rangle \]

- Determine: A hypothesis $h \in H$ that estimates $f$, evaluated by its performance on subsequent instances $x \in X$ drawn according to $D$

  \[ h = x_1 \land x_2 \land x_3 \land x_4 \land x_5 \land x_{100} \]
PAC Learning – Intuition

- We have seen many examples (drawn according to $D$).
- Since in all the positive examples $x_1$ was active, it is very likely that it will be active in future positive examples.
- If not, in any case, $x_1$ is active only in a small percentage of the examples so our error will be small.

$$\text{Error}_D = \Pr_{x \in D} [f(x) \neq h(x)]$$

$h = x_1 \land x_2 \land x_3 \land x_4 \land x_5 \land x_{100}$
The notion of error

Can we bound the Error

\[ \text{Error}_D = \Pr_{x \in D} [f(x) \neq h(x)] \]

given what we know about the training instances?

\[ h = x_1 \land x_2 \land x_3 \land x_4 \land x_5 \land x_{100} \]

f and h disagree
Learning Conjunctions– Analysis (1)

- Let \( z \) be a literal. Let \( p(z) \) be the probability that, in \( D \)-sampling an example, it is positive and \( z \) is false in it. Then: \( \text{Error}(h) \leq \sum_{z \in h} p(z) \)
  - \( p(z) \) is also the probability that a randomly chosen example is positive and \( z \) is deleted from \( h \).
  - If \( z \) is in the target concept, than \( p(z) = 0 \).

- Claim: \( h \) will make mistakes only on positive examples.
- A mistake is made only if a literal \( z \), that is in \( h \) but not in \( f \) is false in a positive example. In this case, \( h \) will say NEG, but the example is POS.
  - Thus, \( p(z) \) is also the probability that \( z \) causes \( h \) to make a mistake on a randomly drawn example from \( D \).
  - There may be overlapping reasons for mistakes, but the sum clearly bounds it.

\[
h = x_1 \land x_2 \land x_3 \land x_4 \land x_5 \land x_{100}
\]
Learning Conjunctions—Analysis (2)

- Call a literal $z$ in the hypothesis $h$ bad if $p(z) > \varepsilon/n$.
- A bad literal is a literal that has a significant probability to appear with a positive example but, nevertheless, it has not appeared with one in the training data.
- Claim: If there are no bad literals, then $\text{error}(h) < \varepsilon$. Reason: $\text{Error}(h) = \sum_{z \in h} p(z)$
- What if there are bad literals?
  - Let $z$ be a bad literal.
  - What is the probability that it will not be eliminated by a given example?
    $$\Pr(z \text{ survives one example}) = 1 - \Pr(z \text{ is eliminated by one example}) \leq 1 - p(z) < 1 - \varepsilon/n$$
  - The probability that $z$ will not be eliminated by $m$ examples is therefore:
    $$\Pr(z \text{ survives } m \text{ independent examples}) = (1 - p(z))^m < (1 - \varepsilon/n)^m$$
- There are at most $n$ bad literals, so the probability that some bad literal survives $m$ examples is bounded by $n(1 - \varepsilon/n)^m$
We want this probability to be small. Say, we want to choose \( m \) large enough such that the probability that some \( z \) survives \( m \) examples is less than \( \delta \).

(I.e., that \( z \) remains in \( h \), and makes it different from the target function)

\[
\Pr(z \text{ survives } m \text{ example}) = n(1 - \frac{\varepsilon}{n})^m < \delta
\]

Using \( 1-x < e^{-x} \) (\( x>0 \)) it is sufficient to require that \( n \cdot e^{-m\varepsilon/n} < \delta \)

Therefore, we need

\[
m > \frac{n}{\varepsilon} \left\{ \ln(n) + \ln(1/\delta) \right\}
\]

examples to guarantee a probability of failure (error > \( \varepsilon \)) of less than \( \delta \).

Theorem: If \( m \) is as above, then:

- With probability > 1-\( \delta \), there are no bad literals; equivalently,
- With probability > 1-\( \delta \), \( \text{Err}(h) < \varepsilon \)

With \( \delta=0.1, \varepsilon=0.1, \) and \( n=100, \) we need 6907 examples.

With \( \delta=0.1, \varepsilon=0.1, \) and \( n=10, \) we need only 460 example, only 690 for \( \delta=0.01 \)
Formulating Prediction Theory

- Instance Space $X$, Input to the Classifier; Output Space $Y = \{-1, +1\}$
- Making predictions with: $h: X \rightarrow Y$
- $D$: An unknown distribution over $X \times Y$
- $S$: A set of examples drawn independently from $D$; $m = |S|$, size of sample.

Now we can define:

- **True Error**: $\text{Error}_D = \Pr_{(x,y) \in D} [h(x) \neq y]$
- **Empirical Error**: $\text{Error}_S = \Pr_{(x,y) \in S} [h(x) \neq y] = \sum_{1,m} [h(x_i) \neq y_i]$
  - (Empirical Error (Observed Error, or Test/Train error, depending on $S$))

This will allow us to ask: **(1)** Can we describe/bound $\text{Error}_D$ given $\text{Error}_S$?

- **Function Space**: $C$ – A set of possible target concepts; target is: $f: X \rightarrow Y$
- **Hypothesis Space**: $H$ – A set of possible hypotheses

This will allow us to ask: **(2)** Is $C$ learnable?

  - Is it possible to learn a given function in $C$ using functions in $H$, given the supervised protocol?
Requirements of Learning

- Cannot expect a learner to learn a concept exactly, since
  - There will generally be multiple concepts consistent with the available data (which represent a small fraction of the available instance space).
  - Unseen examples could potentially have any label.
  - We “agree” to misclassify uncommon examples that do not show up in the training set.

- Cannot always expect to learn a close approximation to the target concept since
  - Sometimes (only in rare learning situations, we hope) the training set will not be representative (will contain uncommon examples).

- Therefore, the only realistic expectation of a good learner is that with high probability it will learn a close approximation to the target concept.
Cannot expect a learner to learn a concept exactly.

Cannot always expect to learn a close approximation to the target concept.

Therefore, the only realistic expectation of a good learner is that with high probability it will learn a close approximation to the target concept.

In Probably Approximately Correct (PAC) learning, one requires that given small parameters $\varepsilon$ and $\delta$, with probability at least $(1 - \delta)$ a learner produces a hypothesis with error at most $\varepsilon$.

The reason we can hope for that is the Consistent Distribution assumption.
### PAC Learnability

- 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 **PAC learnable** by $L$ using $H$ if
  - for all $f \in C$,
  - for all distributions $D$ over $X$, and fixed $0 < \varepsilon, \delta < 1$,
  - $L$, given a collection of $m$ examples sampled independently according to $D$ produces
    - with probability at least $(1 - \delta)$ a hypothesis $h \in H$ with error at most $\varepsilon$, ($\text{Error}_D = \Pr_D[f(x) = h(x)]$)
  - where $m$ is polynomial in $1/\varepsilon$, $1/\delta$, $n$ and $\text{size}(H)$

- $C$ is **efficiently learnable** if $L$ can produce the hypothesis in time polynomial in $1/\varepsilon$, $1/\delta$, $n$ and $\text{size}(H)$
PAC Learnability

- We impose two limitations:
  - **Polynomial sample complexity** (information theoretic constraint)
    - Is there enough information in the sample to distinguish a hypothesis $h$ that approximate $f$?
  - **Polynomial time complexity** (computational complexity)
    - Is there an efficient algorithm that can process the sample and produce a good hypothesis $h$?
- To be PAC learnable, there must be a hypothesis $h \in H$ with arbitrary small error for every $f \in C$. We generally assume $H \supseteq C$. (Properly PAC learnable if $H=C$)
- **Worst Case definition**: the algorithm must meet its accuracy
  - for every distribution (The distribution free assumption)
  - for every target function $f$ in the class $C$
Occam’s Razor (1)

**Claim:** The probability that there exists a hypothesis \( h \in H \) that

1. is **consistent** with \( m \) examples and
2. satisfies \( \text{error}(h) > \varepsilon \quad (\text{Error}_D(h) = \Pr_{x \in D} [f(x) \neq h(x)]) \)

is less than \( |H|(1-\varepsilon)^m \).

**Proof:** Let \( h \) be such a bad hypothesis.

- The probability that \( h \) is consistent with one example of \( f \) is
  \[
  \Pr_{x \in D} [f(x) = h(x)] < 1 - \varepsilon
  \]

- Since the \( m \) examples are drawn independently of each other, the probability that \( h \) is consistent with \( m \) example of \( f \) is less than \( (1 - \varepsilon)^m \).

- The probability that **some** hypothesis in \( H \) is consistent with \( m \) examples is less than \( |H| (1 - \varepsilon)^m \).

Note that we don’t need a true \( f \) for this argument; it can be done with \( h \), relative to a distribution over \( X \times Y \).
Occam’s Razor (1)

We want this probability to be smaller than $\delta$, that is:

$$|H|(1-\epsilon)^m < \delta$$

$$\ln(|H|) + m \ln(1-\epsilon) < \ln(\delta)$$

(with $e^x = 1-x+x^2/2+...$; $e^{-x} > 1-x$; $\ln(1-\epsilon) < -\epsilon$; gives a safer $\delta$)

$$m > \frac{1}{\epsilon} \{\ln(|H|) + \ln(1/\delta)\}$$

(gross over estimate)

It is called Occam’s razor, because it indicates a preference towards small hypothesis spaces

What do we know now about the Consistent Learner scheme?

We showed that a $m$-consistent hypothesis generalizes well ($\text{err} < \epsilon$)
(Appropriate $m$ is a function of $|H|$, $\epsilon$, $\delta$)

What kind of hypothesis spaces do we want? Large? Small?

To guarantee consistency we need $H \supseteq C$. But do we want the smallest $H$ possible?
HW3 is due on Thursday
- Read the write-up given carefully
- Ask questions

Project Proposals
- Due 10/16
- Submission instructions to follow
Immediately from the definition, we get the following general scheme for PAC learning:

- Given a sample $D$ of $m$ examples
  - Find some $h \in H$ that is consistent with all $m$ examples
    - We showed that if $m$ is large enough, a consistent hypothesis must be close enough to $f$
    - Check that $m$ is not too large (polynomial in the relevant parameters): we showed that the “closeness” guarantee requires that
      $$m > \frac{1}{\epsilon} \left( \ln |H| + \ln \frac{1}{\delta} \right)$$
  - Show that the consistent hypothesis $h \in H$ can be computed efficiently

In the case of conjunctions
- We used the Elimination algorithm to find a hypothesis $h$ that is consistent with the training set (easy to compute)
- We showed directly that if we have sufficiently many examples (polynomial in the parameters), then $h$ is close to the target function

We did not need to show it directly. See above.

Last Time:
- We defined PAC Learning (24)
- Proved the Occam’s Razor Result (27)
- Concluded: the “consistent learning scheme”
Conjunction (general): The size of the hypothesis space is $3^n$
Since there are 3 choices for each feature
(not appear, appear positively or appear negatively)

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

(slightly different than previous bound)

• If we want to guarantee a 95% chance of learning a hypothesis of at least 90% accuracy,
  with $n=10$ Boolean variable, $m > (\ln(1/0.05) + 10\ln(3))/0.1 = 140$.
• If we go to $n=100$, this goes just to 1130, (linear with $n$)
• but changing the confidence to 1% it goes just to 1145 (logarithmic with $\delta$)

These results hold for any consistent learner.
K-CNF

\[ f = \bigwedge_{i=1}^{m} (l_{i_1} \lor l_{i_2} \lor \ldots \lor l_{i_k}) \]

Occam Algorithm for \( f \in k\text{-CNF} \)

- Draw a sample \( D \) of size \( m \)
- Find a hypothesis \( h \) that is consistent with all the examples in \( D \)
- Determine sample complexity:

\[
\ln(\mid k-CNF \mid) = O(n^k) \quad \ldots \ldots \quad 2^{(2n)^k} \quad \ldots \ldots (2n)^k
\]

- Due to the sample complexity result \( h \) is guaranteed to be a PAC hypothesis; but we need to learn a consistent hypothesis.

How do we find the consistent hypothesis \( h \)?
How do we find the consistent hypothesis \( h \)?

- Define a new set of features (literals), one for each clause of size \( k \)
  \[
y_j = l_{i_1} \lor l_{i_2} \lor \ldots \lor l_{i_k} \ ; \ j = 1, 2, \ldots, n^k
\]

- Use the algorithm for learning monotone conjunctions, over the new set of literals

**Example:** \( n=4, \ k=2; \) monotone \( k \)-CNF

\[
y_1 = x_1 \lor x_2 \quad y_2 = x_1 \lor x_3 \quad y_3 = x_1 \lor x_4 \\
y_4 = x_2 \lor x_3 \quad y_5 = x_2 \lor x_4 \quad y_6 = x_3 \lor x_4
\]

Original examples: \((0000,l) \ (1010,l) \ (1110,l) \ (1111,l)\)

New examples: \((000000,l) \ (111101,l) \ (111111,l) \ (111111,l)\)
More Examples

Unbiased learning: Consider the hypothesis space of all Boolean functions on $n$ features.

There are $2^{2^n}$ different functions, and the bound is therefore exponential in $n$.

$k$-CNF: Conjunctions of any number of clauses where each disjunctive clause has at most $k$ literals.

$$f = C_1 \land C_2 \land \ldots \land C_m; \ldots; C_i = l_1 \lor l_2 \lor \ldots \lor l_k$$

$$\ln(|k-CNF|) = O(n^k)$$

$k$-clause-CNF: Conjunctions of at most $k$ disjunctive clauses.

$$f = C_1 \land C_2 \land \ldots \land C_k; \ldots; C_i = l_1 \lor l_2 \lor \ldots \lor l_m$$

$$\ln(|k-clause-CNF|) = O(kn)$$

$k$-DNF: Disjunctions of any number of terms where each conjunctive term has at most $k$ literals.

$$f = T_1 \lor T_2 \lor \ldots \lor T_m; \ldots; T_i = l_1 \land l_2 \land \ldots \land l_k$$

$k$-term-DNF: Disjunctions of at most $k$ conjunctive terms.
Sample Complexity

- All these classes can be learned using a polynomial size sample.

  - We want to learn a 2-term-DNF; what should our hypothesis class be?

  - **k-CNF:** Conjunctions of any number of clauses where each disjunctive clause has at most k literals.
    
    \[ f = C_1 \land C_2 \land \ldots \land C_m; \ldots. \; C_i = l_1 \lor l_2 \lor \ldots \lor l_k \]
    
    \[ \ln(|k-CNF|) = O(n^k) \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots 2^{(2n)^k} \ldots \ldots \ldots (2n)^k \]

  - **k-clause-CNF:** Conjunctions of at most k disjunctive clauses.
    
    \[ f = C_1 \land C_2 \land \ldots \land C_k; \ldots. \; C_i = l_1 \lor l_2 \lor \ldots \lor l_m \]
    
    \[ \ln(|k-clause-CNF|) = O(kn) \ldots \ldots 3^{kn} \ldots \ldots 3^n \]

  - **k-DNF:** Disjunctions of any number of terms where each conjunctive term has at most k literals.
    
    \[ f = T_1 \lor T_2 \lor \ldots \lor T_m; \ldots. \; T_i = l_1 \land l_2 \land \ldots \land l_k \]

  - **k-term-DNF:** Disjunctions of at most k conjunctive terms.
Computational Complexity

- Even though from the sample complexity perspective things are good, they are not good from a computational complexity in this case.

- Determining whether there is a 2-term DNF consistent with a set of training data is NP-Hard. Therefore the class of k-term-DNF is not efficiently (properly) PAC learnable due to computational complexity.

- But, we have seen an algorithm for learning k-CNF.

- And, k-CNF is a superset of k-term-DNF
  - (That is, every k-term-DNF can be written as a k-CNF)

\[
T_1 \lor T_2 \lor T_3 = \prod_{x \in T_1, y \in T_2, z \in T_3} \left\{ x \lor y \lor z \right\} \\
(a \land b \land c) \lor (b \land d \land e).
\]

\[
\land \left\{ a \lor b; a \lor d; a \lor e; \quad b; b \lor d; b \lor e; c \lor b; c \lor d; c \lor e; \right\}
\]
Determining whether there is a 2-term DNF consistent with a set of training data is NP-Hard.

Therefore the class of k-term-DNF is not efficiently (properly) PAC learnable due to computational complexity.

We have seen an algorithm for learning k-CNF.

And, k-CNF is a superset of k-term-DNF

- (That is, every k-term-DNF can be written as a k-CNF)

Therefore, C=k-term-DNF can be learned as using H=k-CNF as the hypothesis space.

**Importance of representation:**

- Concepts that cannot be learned using one representation can be learned using another (more expressive) representation.
Negative Results – Examples

- Two types of nonlearnability results:
  - Complexity Theoretic
    - Showing that various concepts classes cannot be learned, based on well-accepted assumptions from computational complexity theory.
    - E.g. : C cannot be learned unless P=NP
  - Information Theoretic
    - The concept class is sufficiently rich that a polynomial number of examples may not be sufficient to distinguish a particular target concept.
    - Both type involve “representation dependent” arguments.
    - The proof shows that a given class cannot be learned by algorithms using hypotheses from the same class. (So?)
- Usually proofs are for EXACT learning, but apply for the distribution free case.
Negative Results for Learning

- **Complexity Theoretic:**
  - k-term DNF, for $k > 1$ (k-clause CNF, $k > 1$)
  - Neural Networks of fixed architecture (3 nodes; n inputs)
  - “read-once” Boolean formulas
  - Quantified conjunctive concepts

- **Information Theoretic:**
  - DNF Formulas; CNF Formulas
  - Deterministic Finite Automata
  - Context Free Grammars
Agnostic Learning

Assume we are trying to learn a concept $f$ using hypotheses in $H$, but $f \not\in H$.

In this case, our goal should be to find a hypothesis $h \in H$, with a small training error: $$Err_{TR}(h) = \frac{1}{m} | \{x \in \text{training examples}; f(x) \neq h(x)\} |$$

We want a guarantee that a hypothesis with a small training error will have a good accuracy on unseen examples:

$$Err_D(h) = Pr_{x \in D}[f(x) \neq h(x)]$$

Hoeffding bounds characterize the deviation between the true probability of some event and its observed frequency over $m$ independent trials. $Pr[p > \hat{p} + \varepsilon] < e^{-2m\varepsilon^2}$

($p$ is the underlying probability of the binary variable (e.g., toss is Head) being 1)
Agnostic Learning

Therefore, the probability that an element in $H$ will have training error which is off by more than $\varepsilon$ can be bounded as follows:

$$\Pr[\text{Err}_D(h) > \text{Err}_{TR}(h) + \varepsilon] < e^{-2m\varepsilon^2}$$

Doing the same union bound game as before, with

$$\delta = |H| e^{-2m\varepsilon^2}$$

We get a generalization bound – a bound on how much will the true error $E_D$ deviate from the observed (training) error $E_{TR}$.

For any distribution $D$ generating training and test instances, with probability at least $1-\delta$ over the choice of the training set of size $m$, (drawn IID), for all $h \in H$

$$\text{Err}_D(h) < \text{Err}_{TR}(h) + \sqrt{\frac{\log|H| + \log(1/\delta)}{2m}}$$
An agnostic learner which makes no commitment to whether \( f \) is in \( H \) and returns the hypothesis with least training error over at least the following number of examples \( m \) can guarantee with probability at least \((1-\delta)\) that its training error is not off by more than \( \varepsilon \) from the true error.

\[
m > \frac{1}{2\varepsilon^2} \{ \ln(|H|) + \ln(1/\delta) \}
\]

Learnability depends on the log of the size of the hypothesis space.
Learning Rectangles

- Assume the target concept is an axis parallel rectangle
Learning Rectangles

- Assume the target concept is an axis parallel rectangle
Learning Rectangles

- Assume the target concept is an axis parallel rectangle
Learning Rectangles

• Assume the target concept is an axis parallel rectangle
Learning Rectangles

- Assume the target concept is an axis parallel rectangle
Learning Rectangles

• Assume the target concept is an axis parallel rectangle
Learning Rectangles

- Assume the target concept is an axis parallel rectangle
Learning Rectangles

• Assume the target concept is an axis parallel rectangle

• Will we be able to learn the target rectangle?
Learning Rectangles

• Assume the target concept is an axis parallel rectangle

• Will we be able to learn the target rectangle?
• Can we come close?

Y

X
Infinite Hypothesis Space

- The previous analysis was restricted to finite hypothesis spaces.
- Some infinite hypothesis spaces are more expressive than others:
  - E.g., Rectangles, vs. 17-sided convex polygons vs. general convex polygons
  - Linear threshold function vs. a conjunction of LTUs
- Need a measure of the expressiveness of an infinite hypothesis space other than its size.
- The Vapnik-Chervonenkis dimension (VC dimension) provides such a measure.
- Analogous to $|\mathcal{H}|$, there are bounds for sample complexity using $\text{VC}(\mathcal{H})$. 
Shattering
Linear functions are expressive enough to shatter 2 points (4 options; not all shown).
Linear functions are not expressive enough to shatter 13 points.
Shattering

• We say that a set $S$ of examples is shattered by a set of functions $H$ if for every partition of the examples in $S$ into positive and negative examples there is a function in $H$ that gives exactly these labels to the examples (Intuition: A rich set of functions shatters large sets of points)
We say that a set $S$ of examples is shattered by a set of functions $H$ if for every partition of the examples in $S$ into positive and negative examples there is a function in $H$ that gives exactly these labels to the examples (Intuition: A rich set of functions shatters large sets of points)

Left bounded intervals on the real axis: $[0,a)$, for some real number $a>0$
Shattering

- We say that a set $S$ of examples is shattered by a set of functions $H$ if for every partition of the examples in $S$ into positive and negative examples there is a function in $H$ that gives exactly these labels to the examples.

(Intuition: A rich set of functions shatters large sets of points)

Left bounded intervals on the real axis: $[0,a)$, for some real number $a>0$

Sets of two points cannot be shattered

(we mean: given two points, you can label them in such a way that no concept in this class will be consistent with their labeling)
Shattering

• We say that a set $S$ of examples is shattered by a set of functions $H$ if for every partition of the examples in $S$ into positive and negative examples there is a function in $H$ that gives exactly these labels to the examples.

Intervals on the real axis: $[a,b]$, for some real numbers $b>a$
Shattering

• We say that a set $S$ of examples is shattered by a set of functions $H$ if for every partition of the examples in $S$ into positive and negative examples there is a function in $H$ that gives exactly these labels to the examples.

Intervals on the real axis: $[a, b]$, for some real numbers $b > a$

All sets of one or two points can be shattered but sets of three points cannot be shattered.
Shattering

- We say that a set $S$ of examples is shattered by a set of functions $H$ if for every partition of the examples in $S$ into positive and negative examples there is a function in $H$ that gives exactly these labels to the examples.

Half-spaces in the plane:
Shattering

• We say that a set $S$ of examples is shattered by a set of functions $H$ if for every partition of the examples in $S$ into positive and negative examples there is a function in $H$ that gives exactly these labels to the examples.

Half-spaces in the plane:

sets of one, two or three points can be shattered but there is no set of four points that can be shattered

1. If the 4 points form a convex polygon... (if not?)
2. If one point is inside the convex hull defined by the other three... (if not?)
VC Dimension

• An unbiased hypothesis space $H$ **shatters** the entire instance space $X$, i.e., it is able to induce every possible partition on the set of all possible instances.

• The larger the subset $X$ that can be shattered, the more expressive a hypothesis space is, i.e., the less biased.
VC Dimension

• We say that a set $S$ of examples is shattered by a set of functions $H$ if for every partition of the examples in $S$ into positive and negative examples there is a function in $H$ that gives exactly these labels to the examples.

• The VC dimension of hypothesis space $H$ over instance space $X$ is the size of the largest finite subset of $X$ that is shattered by $H$.

• If there exists a subset of size $d$ that can be shattered, then $\text{VC}(H) \geq d$.
• If no subset of size $d$ can be shattered, then $\text{VC}(H) < d$.

$\text{VC}(\text{Half intervals}) = 1$ (no subset of size 2 can be shattered)
$\text{VC}(\text{Intervals}) = 2$ (no subset of size 3 can be shattered)
$\text{VC}(\text{Half-spaces in the plane}) = 3$ (no subset of size 4 can be shattered)
Sample Complexity & VC Dimension

- Using VC(H) as a measure of expressiveness we have 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
  
  \[
  m > \frac{1}{\varepsilon} \left\{ 8VC(H) \log \frac{13}{\varepsilon} + 4 \log \left( \frac{2}{\delta} \right) \right\}
  \]

  - Then with probability at least \((1-\delta)\), \( h \) has error less than \( \varepsilon \).

  (that is, if \( m \) is polynomial we have a PAC learning algorithm; to be efficient, we need to produce the hypothesis \( h \) efficiently.

- Notice that to shatter \( m \) examples it must be that: \(|H| > 2^m\), so \( \log(|H|) \geq VC(H) \)
Learning Rectangles

- Consider axis parallel rectangles in the real plan
- Can we PAC learn it?
Learning Rectangles

- Consider axis parallel rectangles in the real plan
- Can we PAC learn it?
  (1) What is the VC dimension?
Learning Rectangles

- Consider axis parallel rectangles in the real plan
- Can we PAC learn it?
  - (1) What is the VC dimension?

- Some four instance can be shattered

(need to consider here 16 different rectangles) Shows that $\text{VC}(H) \geq 4$
Learning Rectangles

- Consider axis parallel rectangles in the real plan
- Can we PAC learn it?
  1. What is the VC dimension?

- Some four instance can be shattered and some cannot

(need to consider here 16 different rectangles) Shows that \( \text{VC}(H) \geq 4 \)
Learning Rectangles

• Consider axis parallel rectangles in the real plan
• Can we PAC learn it?
  (1) What is the VC dimension?

• But, no five instances can be shattered
Learning Rectangles

• Consider axis parallel rectangles in the real plan
• Can we PAC learn it?
  (1) What is the VC dimension?

• But, no five instances can be shattered

There can be at most 4 distinct extreme points (smallest or largest along some dimension) and these cannot be included (labeled +) without including the 5th point.

Therefore \( VC(H) = 4 \)

As far as sample complexity, this guarantees PAC learnability.
Learning Rectangles

- Consider axis parallel rectangles in the real plane
- Can we PAC learn it?
  1. What is the VC dimension?
  2. Can we give an efficient algorithm?
Learning Rectangles

• Consider axis parallel rectangles in the real plan
• Can we PAC learn it?
  (1) What is the VC dimension?
  (2) Can we give an efficient algorithm?

Find the smallest rectangle that contains the positive examples (necessarily, it will not contain any negative example, and the hypothesis is consistent.

Axis parallel rectangles are efficiently PAC learnable.
Sample Complexity Lower Bound

• There is also a general lower bound on the minimum number of examples necessary for PAC learning in the general case.

• Consider any concept class $C$ such that $VC(C) > 2$, any learner $L$ and small enough $\varepsilon$, $\delta$. Then, there exists a distribution $D$ and a target function in $C$ such that if $L$ observes less than $m$ examples, then with probability at least $\delta$, $L$ outputs a hypothesis having error($h$) $> \varepsilon$.

Ignoring constant factors, the lower bound is the same as the upper bound, except for the extra $\log(1/\varepsilon)$ factor in the upper bound.

$$m = \max\left[ \frac{1}{\varepsilon} \log\left( \frac{1}{\delta} \right), \frac{VC(C) - 1}{32\varepsilon} \right]$$
The **PAC framework** provides a reasonable model for theoretically analyzing the effectiveness of learning algorithms.

The **sample complexity** for any consistent learner using the hypothesis space, \( H \), can be determined from a measure of \( H \)'s expressiveness (\(|H|, \text{VC}(H)|\)).

If the sample complexity is tractable, then the **computational complexity** of finding a consistent hypothesis governs the complexity of the problem.

Sample complexity bounds given here are far from being tight, but separates **learnable classes** from **non-learnable classes** (and show what’s important).

**Computational complexity** results exhibit cases where information theoretic learning is feasible, but finding good hypothesis is intractable.

The theoretical framework allows for a concrete analysis of the **complexity of learning** as a function of various assumptions (e.g., relevant variables).
Many additional models have been studied as extensions of the basic one:

- Learning with noisy data
- Learning under specific distributions
- Learning probabilistic representations
- Learning neural networks
- Learning finite automata
- Active Learning; Learning with Queries
- Models of Teaching

An important extension: PAC-Bayesians theory.

- In addition to the Distribution Free assumption of PAC, makes also an assumption of a prior distribution over the hypothesis the learner can choose from.
Theoretical results shed light on important issues such as the importance of the bias (representation), sample and computational complexity, importance of interaction, etc.

Bounds can guide model selection even when not practical.

A lot of recent work is on data dependent bounds.

The impact the theory of learning has had on practical learning system in the last few years has been very significant: SVMs; Winnow (Sparsity), Boosting.
COLT approach to explaining Learning

- No Distributional Assumption
- Training Distribution is the same as the Test Distribution
- IID Assumption is the basis for theoretical support
  - Relaxations have been studied
- Generalization bounds depend on this view and affect model selection.
- As presented, the VC dimension is a combinatorial parameter that is associated with a class of functions.
- Next we see that this notion can be refined to depend on a given data set, and this way directly affect the hypothesis chosen for this data set.
Data Dependent VC dimension

- So far we discussed VC dimension in the context of a fixed class of functions.
- We can also parameterize the class of functions in interesting ways.
- Consider the class of linear functions, parameterized by their margin. Note that this is a data dependent notion.
Let $X = \mathbb{R}^2$, $Y = \{+1, -1\}$

Which of these classifiers would be likely to generalize better?
VC and Linear Classification

- Recall the VC based generalization bound:

\[ \text{Err}(h) \leq \text{err}_{\text{TR}}(h) + \{\text{VC}(H)(\log 2m/\text{VC}(H) +1) +\log(4/\delta)/m\}^{1/2} \]

- Here we get the same bound for both classifier:

\[ \text{Err}_{\text{TR}}(h_1) = \text{Err}_{\text{TR}}(h_2) = 0 \]

\[ h_1, h_2 \in H_{\text{lin}(2)}, \text{VC}(H_{\text{lin}(2)}) = 3 \]

- How, then, can we explain our intuition that \( h_2 \) should give better generalization than \( h_1 \)?
Although both classifiers separate the data, the distance with which they separate is achieved is different:
The margin $\gamma_i$ of a point $x_i \in \mathbb{R}^n$ with respect to a linear classifier $h(x) = \text{sign}(w \cdot x + b)$ is defined as the distance of $x_i$ from the hyperplane $w \cdot x + b = 0$:

$$\gamma_i = \frac{|(w \cdot x_i + b)|}{||w||}$$

The margin of a set of points $\{x_1, \ldots, x_m\}$ is defined as the margin of the point closest to the hyperplane:

$$\gamma = \min_i \gamma_i = \min_i \frac{|(w \cdot x_i + b)|}{||w||}$$
VC and Linear Classification

- If $H_\gamma$ is the space of all linear classifiers in $\mathbb{R}^n$ that separate the training data with margin at least $\gamma$, then:

  $$\text{VC}(H_\gamma) \leq \min(R^2/\gamma^2, n) + 1,$$

- Where $R$ is the radius of the smallest sphere (in $\mathbb{R}^n$) that contains the data.

- Thus, for such classifiers, we have a bound of the form:

  $$\text{Err}(h) \leq \text{err}_{\text{TR}}(h) + \{ (O(R^2/\gamma^2) + \log(4/\delta))/m \}^{1/2}$$
Data Dependent VC dimension

- Namely, when we consider the class $H_{\gamma}$ of linear hypotheses that separate a given data set with a margin $\gamma$,
- We see that
  - Large Margin $\gamma \Rightarrow$ Small VC dimension of $H_{\gamma}$
- Consequently, our goal could be to find a separating hyperplane $w$ that maximizes the margin of the set $S$ of examples.
- A key observation that drives an algorithmic approach is that:
  - Small $||w|| \Rightarrow$ Large Margin
  (next week)
- This leads to an algorithm: from among all those $w$’s that agree with the data, find the one with the minimal size $||w||$
Theoretical results shed light on important issues such as the importance of the bias (representation), sample and computational complexity, importance of interaction, etc.

Bounds can guide model selection even when not practical.

A lot of recent work is on data dependent bounds.

The impact the theory of learning has had on practical learning system in the last few years has been very significant: SVMs; Winnow (Sparsity), Boosting.