Slides were created by Dan Roth (for CIS519/419 at Penn or CS446 at UIUC), Eric Eaton for CIS519/419 at Penn, or from other authors who have made their ML slides available.
Administration

- Midterm Exam next Thursday
  - In class
- Closed books
- We’ll give you some examples
- All the material covered in class and HW

- HW2 is due on Monday next week 2/26
  - Only 1 day of slack time since we want to release solutions and give you time to prepare for the mid-term.
- Recitations: Moore 216
  - Tue 6:30; Wed 5:30
- My Office hours: 5-6, Tue/Thur
Projects

- CIS 519 students need to do a team project
  - Teams will be of size 2-3
- Projects proposals are due on Friday 3/2/18
  - Details will be available on the website
  - We will give comments and/or requests to modify / augment / do a different project.
  - There may also be a mechanism for peer comments.
- 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 preliminary results/ideas.
- Any project with a significant Machine Learning component is good.
  - 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 of the literature.
  - Originality is not mandatory but is encouraged.
- Try to make it interesting!
Project Examples

- KDD Cup 2013:
  - "Author-Paper Identification": given an author and a small set of papers, we are asked to identify which papers are really written by the author.
  - “Author Profiling”: given a set of document, profile the author: identification, gender, native language, ....

- Caption Control: Is it gibberish? Spam? High quality text?
  - Adapt an NLP program to a new domain

- Work on making learned hypothesis more comprehensible
  - Explain the prediction

- Develop a (multi-modal) People Identifier

- Identify contradictions in news stories

- Large scale clustering of documents + name the cluster
  - E.g., cluster news documents and give a title to the document

- Deep Neural Networks: convert a state of the art NLP program to a NN
Where are we?

- Algorithmically:
  - Perceptron + Variations
  - (Stochastic) 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
  - $<(0,1,1,1,1,0,\ldots,0,1), 1>$ (We learned a superset of the good variables)
  - To show you that all these variables are required...
    - $<(0,0,1,1,1,0,\ldots,0,1), 0>$ need $x_2$
    - $<(0,1,0,1,1,0,\ldots,0,1), 0>$ need $x_3$
    - .....  
    - $<(0,1,1,1,1,0,\ldots,0,0), 0>$ need $x_{100}$

- A straightforward algorithm requires $k = 6$ examples to produce the hidden conjunction (exactly).
  \[
f = x_2 \land x_3 \land x_4 \land x_5 \land x_{100}\]
Protocol III: Some random source (e.g., Nature) provides training examples

Teacher (Nature) provides the labels \( f(x) \)

- \( \langle 1,1,1,1,1,...,1,1 \rangle, 1 \rangle \)
- \( \langle 1,1,1,0,0,0,...,0,0 \rangle, 0 \rangle \)
- \( \langle 1,1,1,1,0,...0,1,1 \rangle, 1 \rangle \)
- \( \langle 1,0,1,1,1,0,...0,1,1 \rangle, 0 \rangle \)
- \( \langle 1,1,1,1,1,0,...0,0,1 \rangle, 1 \rangle \)
- \( \langle 1,0,1,0,0,0,...0,1,1 \rangle, 0 \rangle \)
- \( \langle 1,1,1,1,1,...,0,1 \rangle, 1 \rangle \)
- \( \langle 0,1,0,1,0,0,...0,1,1 \rangle, 0 \rangle \)

How should we learn?

Skip
Learning Conjunctions (III)

- 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

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

- 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

Examples:
- \(<(1,1,1,1,1,...,1,1), 1>\>
- \(<(1,1,1,0,0,...,0,0), 0>\) learned nothing
- \(<(1,1,1,1,1,0,...0,1,1), 1>\>
- \(<(1,0,1,1,0,0,...0,0,1), 0>\) learned nothing
- \(<(1,1,1,1,1,0,...0,0,1), 1>\>
- \(<(1,0,1,0,0,...0,1,1), 0>\) learned nothing
- \(<(1,1,1,1,1,...,0,1,1), 1>\>
- \(<(0,1,0,1,0,0,...0,1,1), 0>\>

Final hypothesis:

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

- Is it good
- Performance?
- # of examples?
Learning Conjunctions (III)

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

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

- With the given data, we only learned an “approximation” to the true concept
- We don’t know how many examples we need to see to learn exactly. (do we care?)
- But we know that we can make a limited # of mistakes.

**Final hypothesis:**

\[ h = x_1 \land x_2 \land x_3 \land x_4 \land x_5 \land x_{100} \]
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 $h(x) = f(x)$ for all $x \in S$?
- A hypothesis $h \in H$ such that $h(x) = f(x)$ 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 $X$
  
  $$< x_1, f(x_1) >, < x_2, f(x_2) >, ..., < x_n, f(x_n) >$$

- **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.

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

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

Can we bound the Error

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

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: $\text{Error}(h) \cdot \sum_z 2^h p(z)$
  - $p(z)$ 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 \( p(z) > \frac{\varepsilon}{n} \).
- A bad literal is a literal that is not in the target concept and has a significant probability to appear false with a positive example.
- Claim: If there are no bad literals, than \( \text{error}(h) < \varepsilon \). Reason: \( \text{Error}(h) \cdot \sum_{z \in h} p(z) \)
- 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?
    \[
    \Pr(\text{z survives one example}) = 1 - \Pr(\text{z is eliminated by one example}) = 1 - p(z) < 1 - \frac{\varepsilon}{n}
    \]
  - The probability that \( z \) will not be eliminated by \( m \) examples is therefore:
    \[
    \Pr(\text{z survives m independent examples}) = (1 - p(z))^m < (1 - \frac{\varepsilon}{n})^m
    \]
  - There are at most \( n \) bad literals, so the probability that some bad literal survives \( m \) examples is bounded by \( n(1 - \frac{\varepsilon}{n})^m \).
Learning Conjunctions—Analysis (3)

- 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 \( n \ e^{-\varepsilon m / n} < \delta \)
- Therefore, we need
  \[
  m > \frac{n}{\varepsilon} \left\{ \ln(n) + \ln\left(\frac{1}{\delta}\right) \right\}
  \]
  examples to guarantee a probability of failure (error > 2) of less than \( \delta \).

- **Theorem:** If \( m \) is as above, then:
  - With probability > 1 - \( \delta \) ±, there are no bad literals; equivalently,
  - With probability > 1 - \( \delta \) ±, \( \text{Err}(h) < \varepsilon^2 \)

- 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 $X \subseteq Y$
- $S$: A set of examples drawn independently from $D$; $m = |S|$, size of sample.

Now we can define:

- **True Error**: $\text{Error}_D = \Pr_{(x,y) \sim D} [h(x) : = y]$
- **Empirical Error**: $\text{Error}_S = \Pr_{(x,y) \sim S} [h(x) : = y] = \sum_{1,m} [h(x_i) : = y_i]$
  - (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.
Probably Approximately Correct

- 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 \( \text{error}(h) > \varepsilon \) \( \quad (\text{Error}_D(h) = \Pr_{x \sim D} [f(x) = h(x)]) \)
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
  \[ \Pr_{x \sim D} [f(x) = h(x)] < 1 - \varepsilon \]
- 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 \( X \times Y \).

So, what is \( m \)?
Occam’s Razor (1)

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

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

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

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

We showed that a $m$-consistent hypothesis generalizes well ($\text{err} < \varepsilon^2$) (Appropriate $m$ is a function of $|H|$..±)

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

(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?

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?
Why Should We Care?

- We now have a theory of generalization
  - We know what the important complexity parameters are,
  - We understand the dependence in the number of examples and in the size of the hypothesis class.

- We have a generic procedure for learning that is guaranteed to generalize well
  - Draw a sample of size m.
  - Develop an algorithm that is consistent with it.
  - It will be good
    - If m was large enough.
Consistent Learners

- 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
        \[
        m > 1/\varepsilon^2 (\ln |H| + \ln 1/\delta)
        \]
    - 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), than \( h \) is close to the target function.
Examples

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)

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

(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.
Why Should We Care?

- We now have a theory of generalization
  - We know what are the important complexity parameters
  - We understand the dependence in the number of examples and in the size of the hypothesis class

- We have a generic procedure for learning that is guaranteed to generalize well
  - Draw a sample of size $m$.
  - Develop an algorithm that is consistent with it.
  - It will be good.

- We have tools to prove that some hypothesis classes are learnable and some are not
Midterm Exam on Thursday

- Students whose name starts with A-K (inclusive)
  - Go to LRSM, Room: AUD
- Students whose name starts with L-Z (inclusive)
  - Come here, Levine, Wu & Chen

Please don’t be later!!!

Closed books

- 3 practice exams (on the web); will be discussed in recitations.
- Moore 216: Tue 6:30; Wed 5:30
- Covers: all the material covered in class and HW

HW2 + 1 day of slack time is due tonight.

My Office hours: 5-6, Tue/Thur
K-CNF

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

Occam Algorithm (=Consistent Learner 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:

\[
\ln(|k-CNF|) = O(n^k) \quad \text{and} \quad 2^{(2n)^k} \quad (2n)^k
\]

- (1) 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 \( h \)?
K-CNF

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

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

- Define a new set of features (literals), one for each clause of size \( k \)

\[ y_j = l_{i_1} \lor l_{i_2} \lor \ldots \lor l_{i_k}; \quad j = 1,2,\ldots,n^k \]

- 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)
Negative Results – Examples

- Two types of non-learnability 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

We need to extend the theory in two ways:
(1) What if we cannot be *completely consistent* with the training data?
(2) What if the hypothesis class we work with is *not finite*?
Agnostic Learning

- Assume we are trying to learn a concept \( f \) using hypotheses in \( H \), but \( f \notin H \)
- In this case, our goal should be to find a hypothesis \( h \in H \), with a small training error:
  \[
  Err_{TR}(h) = \frac{1}{m} \left| \{x \in \text{training examples}; f(x) \neq h(x)\} \right|
  \]
- 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. \( \Pr[p > p_{emp} + \varepsilon] < e^{-2m\varepsilon^2} \)
  - (\( p \) is the underlying probability of the binary variable (e.g., toss is Head) being 1; \( p_{emp} \) is what we observe empirically – empirical error)
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 \( \delta = |H|e^{-2m\varepsilon^2} \) (from here, we can now isolate \( m \), or \( \varepsilon \))

- We get a generalization bound – a bound on how much 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 \( h \in H \)

\[
\text{Error}_D(h) < \text{Error}_{TR}(h) + \sqrt{\frac{\log|H| + \log(1/\delta)}{2m}}
\]

See slide 71 in the On-line Lecture
Summary
(on-line lecture #71)

- Introduced multiple versions of on-line algorithms
- All turned out to be Stochastic Gradient Algorithms
  - For different loss functions
- Some turned out to be mistake driven
- We suggested generic improvements via:
  - Regularization via adding a term that forces a “simple hypothesis”
    \[ J(w) = \sum_{i,m} Q(z_i, w_i) + \lambda R_i (w_i) \]
  - Regularization via the Averaged Trick
    - “Stability” of a hypothesis is related to its ability to generalize
  - An improved, adaptive, learning rate (Adagrad)
- Dependence on function space and the instance space properties.
- Today:
  - A way to deal with non-linear target functions (Kernels)
  - Beginning of Learning Theory.

\( w^T x = \theta \)
Agnostic Learning

- 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 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?
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-sides 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 $|H|$, there are bounds for sample complexity using $\mathcal{VC}(H)$
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)
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 $a>0$.
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.

This is the set of functions (concept class) considered here.

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

\[ \begin{array}{cccccc}
- & - & + & + & + & + \\
\hline
 a & b
\end{array} \]
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:

![Diagram of half-spaces in the plane with positive and negative examples.](image-url)
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.
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 $H$.

**Two steps to proving that $VC(H) = d$:**

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

VC(Half intervals) = 1  
VC( Intervals) = 2  
VC(Half-spaces in the plane) = 3

(no subset of size 2 can be shattered)  
(no subset of size 3 can be shattered)  
(no subset of size 4 can be shattered)

Some are shattered, but some are not
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
    \[
    m > \frac{1}{\varepsilon} \left\{ 8 \text{VC}(H) \log \frac{13}{\varepsilon} + 4 \log \left( \frac{2}{\delta} \right) \right\}
    \]
    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 \(\log(|H|) \geq \text{VC}(H)\)

What if \( H \) is finite?
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 instances 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 $\text{VC}(H) = 4$

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.

$$m = \max\left[\frac{1}{\varepsilon} \log\left(\frac{1}{\delta}\right), \frac{VC(C) - 1}{32\varepsilon}\right]$$
COLT Conclusions

- 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 ($|H|$, 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 separate **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).
COLT Conclusions (2)

- 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.
COLT Conclusions (3)

- Theoretical results shed light on important issues such as the importance of the bias (representation), sample and computational complexity, importance of interaction, etc.
- Bounds guide model selection even when not practical.
- A lot of recent work is on data dependent bounds.
- The impact COLT has had on practical learning system in the last few years has been very significant:
  - SVMs;
  - Winnow (Sparsity),
  - Boosting
  - Regularization