Why Machine Learning Works: Explaining Generalization

Dan Roth
danroth@seas.upenn.edu|http://www.cis.upenn.edu/~danroth/|461C, 3401 Walnut

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

Administration (10/26/20)

Are we recording? YES!

Available on the web site

- Remember that all the lectures are available on the website before the class
  - Go over it and be prepared
  - A new set of written notes will accompany most lectures, with some more details, examples and, (when relevant) some code.
- HW 2: Due date extended to 10/22; late submissions are due today.
- Quizzes: Quiz 6 Statistics
- Mid-term is on 10/28; at the class time.

Midterm on 10/28/20

- Mid-term will be a Quiz style.
  - It will be done on Canvas.
  - We will open at 10:30am Eastern time, and close it at 11:30am
    - Except for people that have been approved for extended time
      - And have heard from me (some time today)
    - You need to be on zoom. If at all possible, open your video.
  - No one came forward with a time zone problem.
- Short questions
  - Multiple choice; a few will require filling in answers
  - There will be quite a few questions
  - Open books (but you may not have time to consult it too much)
  - We try to make it about understanding rather than memorization
- Questions? Please ask/comment during class; give us feedback

Projects

- CIS 519 students need to do a team project
  - Teams will be of size 2-4
  - We will help grouping if needed
- There will only be ~3 types of projects.
  - We will provide initial ideas and ask that you write a short proposal/plan for what you want to do.
- If you have an idea for a project that you would like to be one of these projects –
  - please send me a short write-up (< 1 page) with a description, motivation, relevant data available, and any other relevant information.
  - No later than Friday this week: 10/30/20
- Details will be available on the website
  - Start teaming up
- The project will require developing a machine learning system and running experiments with it
  - You will be given some data
  - Beyond running several algorithms on the data, the key part will require asking a question or proposing a hypothesis and investigating it.
    - Say that the data comes from multiple domains – it is enough to train on one of the domains?
  - The work has to include some reading of the literature .
  - Originality is not mandatory but is encouraged.
- Try to make it interesting!

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.

Why Learning Works?

- (A glimpse into) A theory of Generalization:
- The basic theorem we will discuss has the following form:

  - Error(f) [on sample from distribution D] < Training Error (f) +
    Complexity Term
    (size of hypothesis space, # of examples, how good you want it to be)

- **Key Condition:** Training data is sampled from the same distribution as the test data
  - IID: Independently, Identically distributed
- **Key question:** How we do estimate the complexity term?
  - What is the relation between what we see on the training data and what we'll see in the real world.
  - Note that you already know something about it, experimentally; but, can we quantify it?

Example:

- 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 $h \in H$

Error on the training data

Generalization: a function of the Hypothesis class size

$$Error_D < Error_{TR}(h) + \left[\frac{\log|H| + \log\left(\frac{1}{\delta}\right)}{2m}\right]^{\frac{1}{2}}$$

- What if H isn't finite? What other complexity parameters can be used?

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 via Learning Conjunctions

Recall what we did earlier:

- 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(III)

$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

- 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

- Is it good?
- Performance ?
- # of examples ?

<(1,1,1,1,1,1,...,1,1), 1>
<(1,1,1,0,0,0,...,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,...0,1,1), 0> Final hypothesis:
<(1,1,1,1,1,1,...,0,1), 1> $h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$
<(0,1,0,1,0,0,...0,1,1), 0>

Learning Conjunctions (III)

$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

- 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>
<(1,1,1,0,0,0,...,0,0), 0>
<(1,1,1,1,1,0,...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>
<(1,1,1,1,1,1,...,0,1), 1>
<(0,1,0,1,0,0,...0,1,1), 0>
<(0,1,0,1,0,0,...0,1,1), 0>

Final hypothesis:

$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

- Is it good
- Performance ?
- # of examples ?

- 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**.

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.
    - May be unsatisfactory, since we don't know **when** we will make the mistakes. We want a more robust notion of **performance in the future**.

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:
  - positive and negative examples of the target concept $f \in C$

$$< x_1, f(x_1) >, < x_2, f(x_2) >, \dots , < x_n, f(x_n) >$$

- 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 $x \in X$?

$$h = \underline{x_1} \land x_2 \land x_3 \land x_4 \land x_5 \land x_{100}$$

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

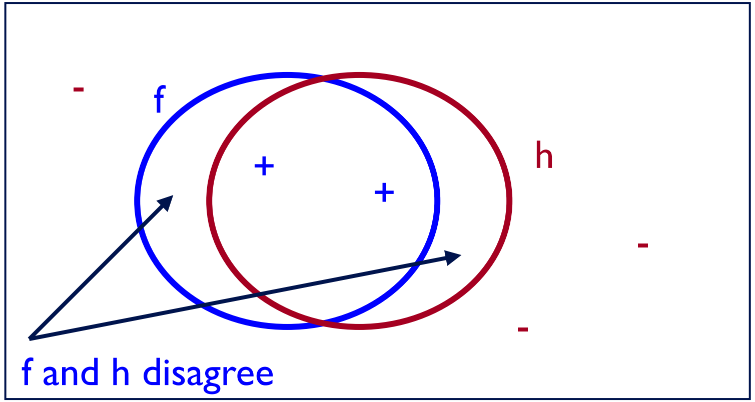
$$< x_1, f(x_1) >, < x_2, f(x_2) >, \dots, < x_n, f(x_n) >$$

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

$$h = \underline{x_1} \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

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



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

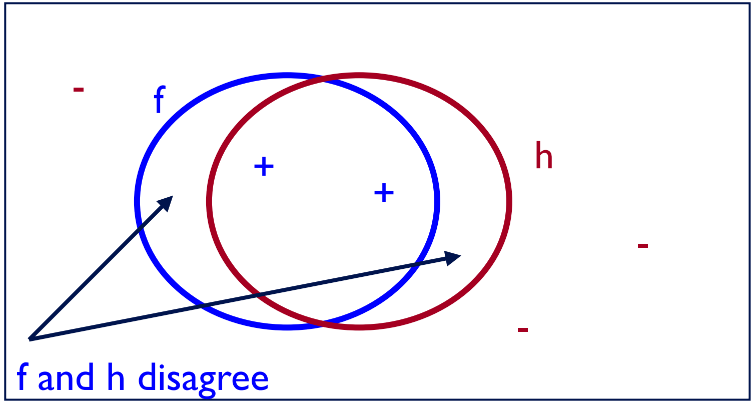
- $h = \underline{x_1} \land x_2 \land x_3 \land x_4 \land x_5 \land x_{100}$

The notion of error

- Can we bound the Error?

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

given what we know about the training instances?



$$h = \underline{x_1} \land x_2 \land x_3 \land x_4 \land x_5 \land x_{100}$$

Is this the right picture?
Why? Why not?

Is the Venn diagram shown correct for monotone conjunctions? Answer [Yes/No, Why]

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

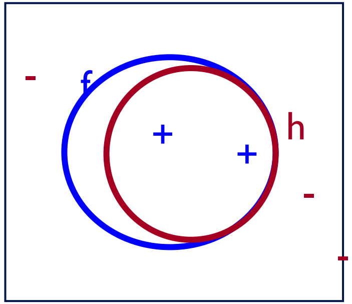
Learning Conjunctions– Analysis (1)

- **Claim 1:** Let z be a literal. Let $p(z)$ be the probability that, in D-sampling an example, the example is positive and z is false in it. Then:

$$Error(h) \leq \sum_{z\, \epsilon h} p(z)$$

- **Proof:**
  - During learning $p(z)$ is the probability that a randomly chosen example is positive and z is deleted from h.
  - If z is in the target concept, than $p(z) \;=\; 0$.
  - Note that 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.

$$\boxed{h = \underline{x_1} \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}}$$



Learning Conjunctions– Analysis (2)

Step 2 of the analysis:

- 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 $error(h) < \varepsilon$. Reason: $Error(h) \leq \sum_{z\, \epsilon h} p(z)$
- 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?

$$\Pr(z\ survives\ one\ example) = 1 - \Pr(z\ is\ eliminated\ by\ one\ example)$$
$$\leq 1 - p(z) < 1 - \frac{\varepsilon}{n}$$

- The probability that z will not be eliminated by m examples is therefore:

$$\Pr(z\ survives\ m\ independent\ examples) = (1 - p(z))^m < \left(1 - \frac{\varepsilon}{n}\right)^m$$

- There are at most n bad literals, so the probability that some bad literal survives m examples is bounded by $n(1 - \varepsilon/n)^m$

Learning Conjunctions– Analysis (3)

Step 3 of the analysis:

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

$$\Pr(z\ survives\ m\ example) = n\left(1-\frac{\varepsilon}{n}\right)^{m} < \delta$$

- Using $1 - x < e^{-x}$ $(x > 0)$ it is sufficient to require that $n\, e^{-\frac{m\varepsilon}{n}} < \delta$
- Therefore, we need :

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

examples to guarantee a probability of failure $(error > \epsilon)$ of less than δ.

- Theorem: If m is as above, then:
  - With probability $> 1 - \delta$, there are no bad literals; equivalently,
  - With probability $> 1 - \delta$, $Err(h) < \varepsilon$
- 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 $\delta = 0.01$

More Generally: 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: $Error_D = \Pr_{(x,y)\in D}[h(x) \neq y]$
- Empirical Error: $Error_S = \Pr_{(x,y) \in S}[h(x) \neq y] = \sum_{1,m}[h(x_i) \neq y_i]$
  - (Empirical Error == Observed Error)

This will allow us to ask: (1) Can we describe/bound Error_D given 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.

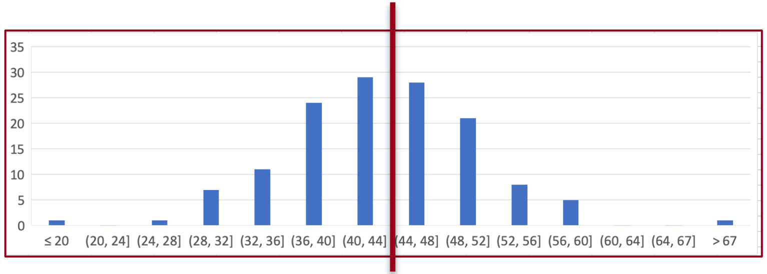
Those of you who cannot vote (and those who can):
A documentary: All In

Administration (11/2/20)

Are we recording? YES!

Available on the web site

- Remember that all the lectures are available on the website before the class
  - Go over it and be prepared
  - A new set of written notes will accompany most lectures, with some more details, examples and, (when relevant) some code.



- HW 3: Due on 11/16/
  - You cannot solve all the problems yet.
  - Less time consuming; no programming
- Mid-term:
  - Average: 43.4/81= 53.5%
  - Median: 43.5/81
  - Standard Deviation: 7.7

Expectation:

- Top 35-40% of the students = A
- Next 40% = B
- Next 20% = C
- Very few (hopefully none) who stop doing the work < C.

Projects

- CIS 519 students need to do a team project
  - Teams will be of size 2-4
  - We will help grouping if needed

- There will be 3 projects.
  - Natural Language Processing (Text)
  - Computer Vision (Images)
  - Speech (Audio)

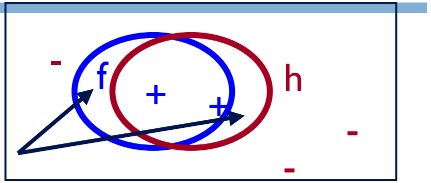
- In all cases, we will give you datasets and initial ideas
  - The problem will be multiclass classification problems
  - You will get annotated data only for some of the labels, but will also have to predict other labels
  - 0-zero shot learning; few-shot learning; transfer learning

- A detailed note will come out today.

- Timeline:
  - 11/9: Choose a project and team up
  - 11/23 Initial proposal describing what your team plans to do
  - 12/2 Progress report
  - 12/15-20 (TBD) Final paper + short video
- Try to make it interesting!

Probably Approximately Correct

We want a theory, so that we understand
(1) what observed performance says about future performance, and
(2) what contributes to this (gap in performance) .



- 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 ε and δ, with probability at least $(1 - \delta)$ a learner produces a hypothesis with error at most ε
- 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 <u>PAC learnable</u> by L using H if
  - for all $f \epsilon C$,
  - 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 ε,

  $(Error_D = Pr_D[f(x) \neq h(x)])$ where m is polynomial in $1/\varepsilon$, $1/\delta$, n and $size(H)$
- C is <u>efficiently learnable</u> if L can produce the hypothesis in time polynomial in $1/\varepsilon$, $1/\delta$, n and $size(H)$

PAC Learnability

We want a theory, so that we understand
(1) what observed performance says about future performance, and
(2) what contributes to this (gap in performance) .

- We impose two limitations:
  - Polynomial sample complexity (a condition on m; information theoretic constraint)
    - Is there enough information in the sample to distinguish a hypothesis h that approximate f ?
  - Polynomial time complexity (a condition on the efficiency of L; 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)

Key Theorem (simple case): Generalizing the example we showed for conjunctions

Claim: The probability that there exists a hypothesis $h \epsilon\ H$ that

(1) is consistent with m examples and

(2) satisfies $error(h) > \varepsilon$ $\quad (\,Error_D(h) \;=\; Pr_{x\,\epsilon\,D}\,[f(x) \neq h(x)]\,)$

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

Note that this is an ideal situation – the learner is perfect on the training data. We call it the "consistent learner scheme".
First, we will ask "how good are we going to be in the future if we are perfect in training"; then we'll generalize to a more realistic scenario.

Proof: Let h be such a bad hypothesis.

- The probability that h is consistent with one example of f is

$$Pr_{x\,\epsilon D}[f(x) = h(x)] < 1 \; - \varepsilon$$

- 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 $|H|(1 \; - \varepsilon)^m$

So, what is 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 δ, that is:

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

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

(with $e^{-x} \;=\; 1 - x + \frac{x^2}{2} + \cdots;\; e^{-x} \;>\; 1 - x;$ → $\ln(1-\ \varepsilon) \;<\; -\varepsilon$; gives a safer δ)

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

(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 ($err < \varepsilon$)

(The appropriate m is a function of |H|)

- 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

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

  - 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), than h is close to the target function.

We did not need to show it directly. As shown above we could have just relied on the fact the H is not too large.

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)

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

(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,
  - $m \ > \ (ln(1/0.05) \ + 10ln(3))/0.1 \ = 140.$
- 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 δ)
- 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.

Example: K-CNF

- We will show that the class of K-CNF functions is PAC learnable.
  - Here is an example of a member of this class of functions:

$$f = \bigwedge_{i=1}^{r} (l_{i_1} \vee l_{i_2} \vee \cdots \vee l_{i_k})$$

- We will develop an Occam Algorithm (Consistent Learner algorithm) for a hidden $f \epsilon k - CNF$
- Draw a sample D of size m
- Find a hypothesis h that is consistent with all the examples in D
- Determine sample complexity:

$$f = C_1 \wedge C_2 \wedge \cdots \wedge C_m; \ldots \ldots \ldots; C_i = l_1 \vee l_2 \vee \cdots \vee l_k$$

$$\ln(|k - CNF|) = O\left(n^k\right) \ldots \ldots \ldots 2^{(2n)^k} \ldots \ldots \ldots (2n)^k$$

(that is, log|H| is polynomial in n; remember that k is just a fixed number)

(1) Due to the sample complexity result h is guaranteed to be a PAC hypothesis, if we can use the m examples to learn a consistent hypothesis.

How about an algorithm? how do we find the consistent hypothesis h?

Example: K-CNF (cont.)

$$f = \bigwedge_{i=1}^{\mathrm{r}} (l_{i_1} \vee l_{i_2} \vee \cdots \vee l_{i_k})$$

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

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

$$y_j = l_{i_1} \vee l_{i_2} \vee \cdots \vee l_{i_k}; j = 1,2, \ldots, n^k$$

- Use the algorithm for learning monotone conjunctions, over the new set of literals. We know that the algorithm is efficient.

Example: $n = 4$, $k = 2$; monotone k-CNF

$y_1 = x_1 \vee x_2$ $y_2 = x_1 \vee x_3$ $y_3 = x_1 \vee x_4$ $y_4 = x_2 \vee x_3$ $y_5 = x_2 \vee x_4$ $y_6 = x_3 \vee x_4$

- Original examples: $(0000, l)$ $(1010, l)$ $(1110, l)$ $(1111, l)$
- New examples: $(000000, l)$ $(111101, l)$ $(111111, l)$ $(111111, l)$

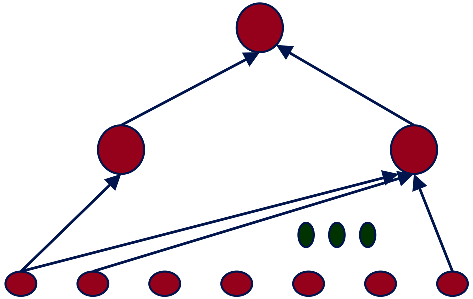
Distribution?

Negative Results – Examples

- Two types of non-learnability results:
- Complexity Theoretic (Time complexity, applies to the Efficient PAC condition)
  - 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 (Sample Complexity, applies to the basic PAC condition)
  - 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:

$$Err_{TR}(h) = \frac{1}{m}|\{\boldsymbol{x} \in training - examples; f(\boldsymbol{x}) \neq h(\boldsymbol{x})\}|$$

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

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

- 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 $h \in H$

Error on the training data

$$Error_D < Error_{TR}(h) + \left[\frac{\log|H| + \log\left(\frac{1}{\delta}\right)}{2m}\right]^{\frac{1}{2}}$$

Generalization: a function of the Hypothesis class size

- See slide 102 in the On-line Lecture
- So, what should m be?

Agnostic Learning [Details]

- 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:

$$Err_{TR}(h) = \frac{1}{m}|\{\boldsymbol{x} \in training - examples; f(\boldsymbol{x}) \neq h(\boldsymbol{x})\}|$$

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

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

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

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

$$\Pr[Err_D(h) > Err_{TR}(h) + \varepsilon] < e^{-2m\varepsilon^2}$$

- Doing the same union bound game as before, with $\delta = |H|e^{-2m\varepsilon^2}$ (from here, we can now isolate m, or ε)
- 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 ge est instances, with probability at least $1 - \delta$ over the choice of the training se for all $h \in H$

Error on the training data

Generalization: a function of the Hypothesis class size

$$Error_D < Error_{TR}(h) + \left[\frac{\log|H| + \log\left(\frac{1}{\delta}\right)}{2m}\right]^{\frac{1}{2}}$$

Agnostic Learning

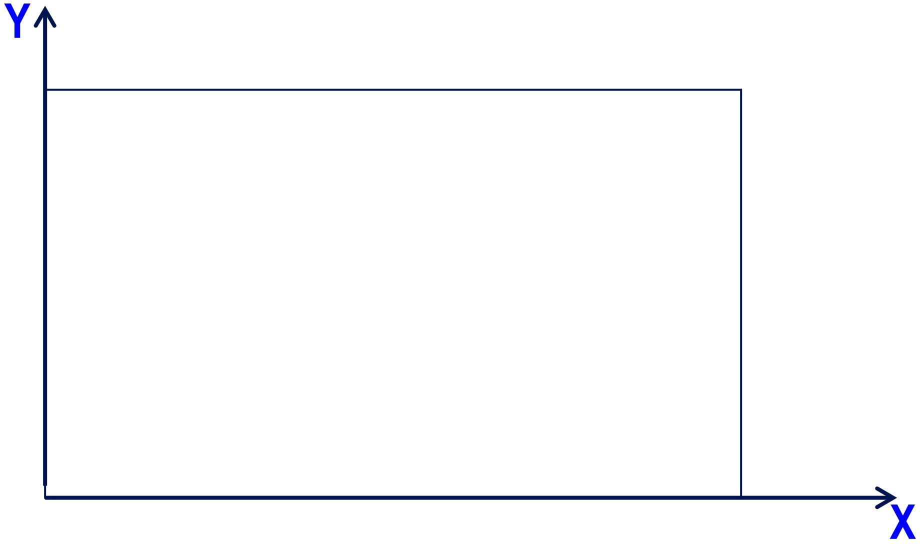
- An agnostic learner
  - which makes no commitment to whether f is in H, and
- returns the hypothesis with least training error over at least the following number of examples m
- can guarantee with probability at least $(1 - \delta)$ that its training error is not off by more than ε from the true error.

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

Learnability depends on the log of the size of the hypothesis space

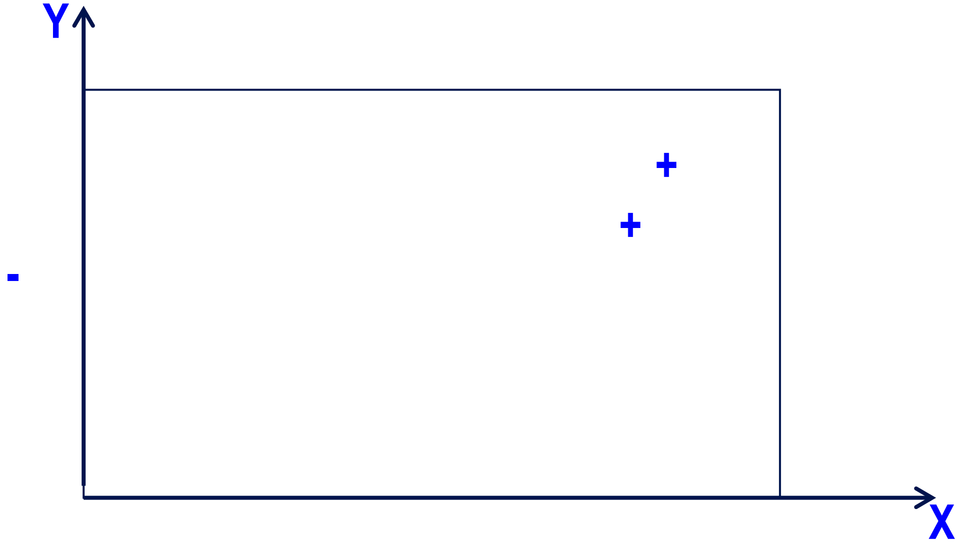
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



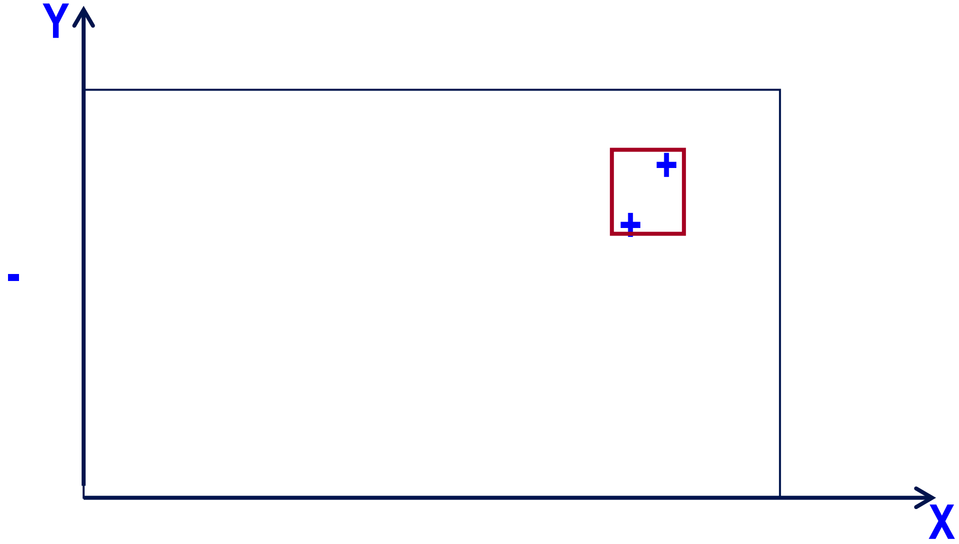
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



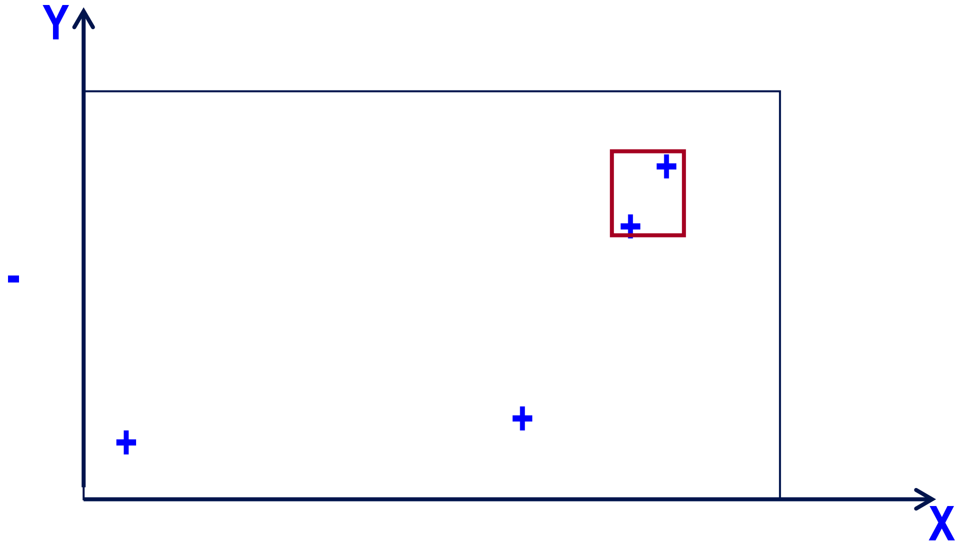
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



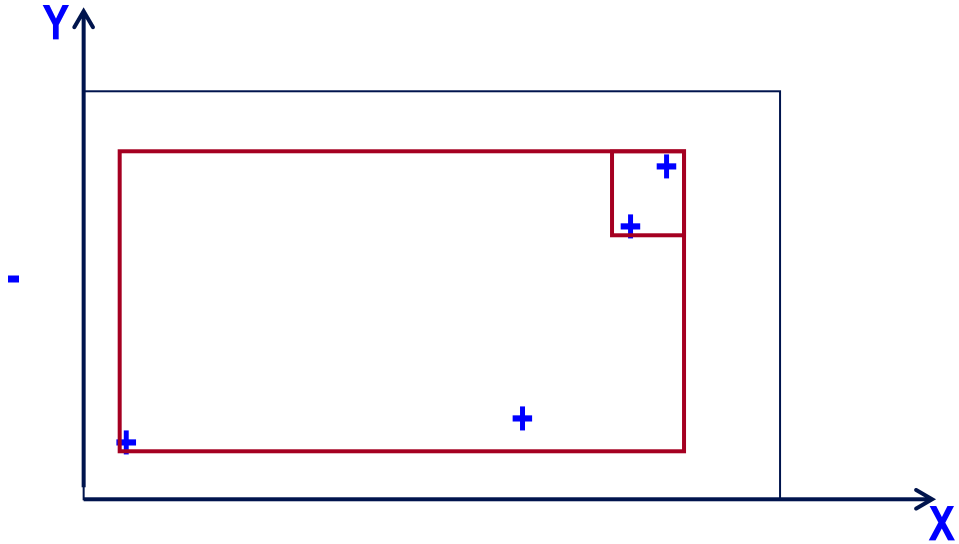
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



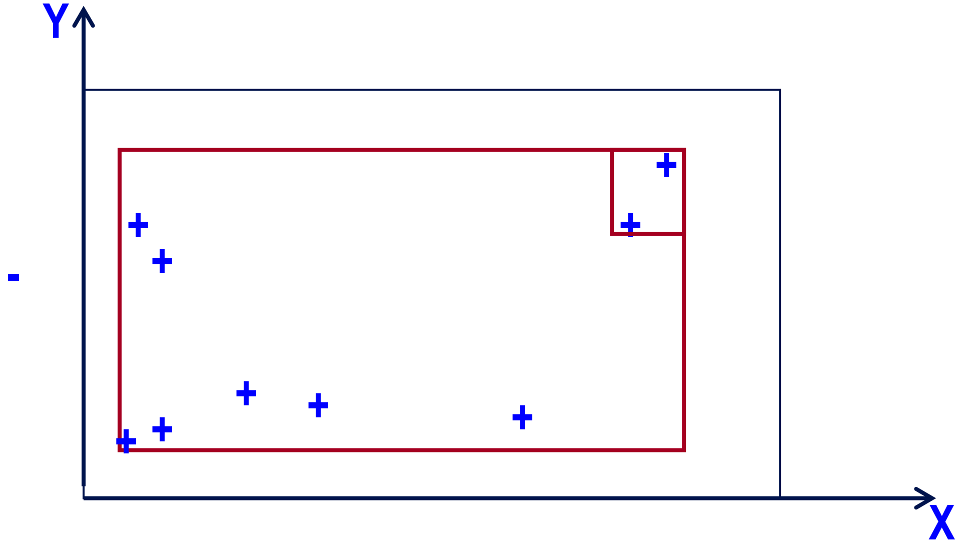
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



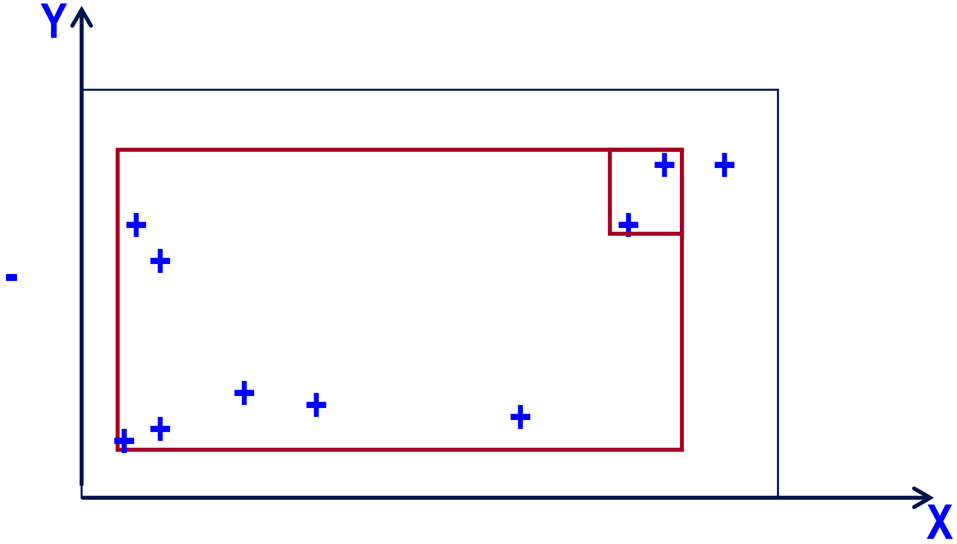
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



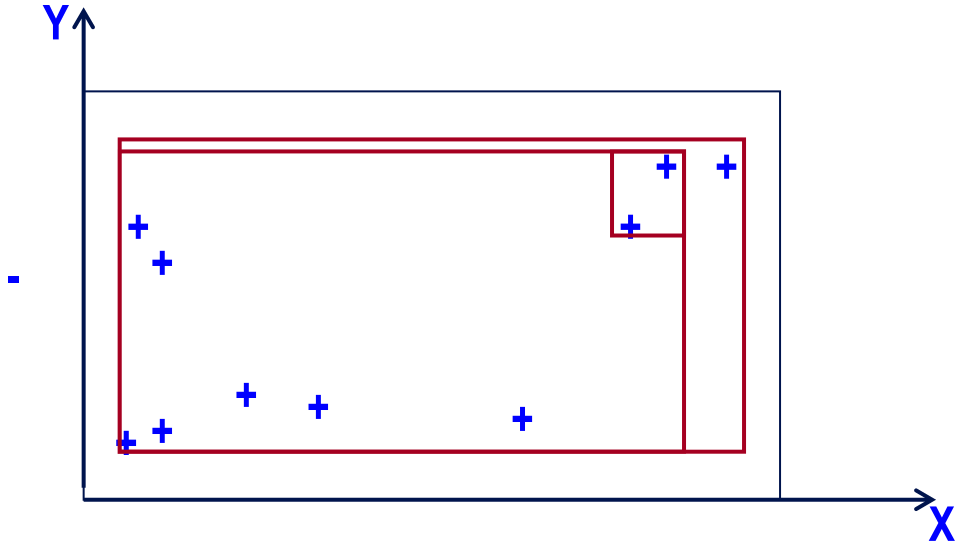
Learning Rectangles

- Assume the target concept is an axis parallel rectangle



Learning Rectangles

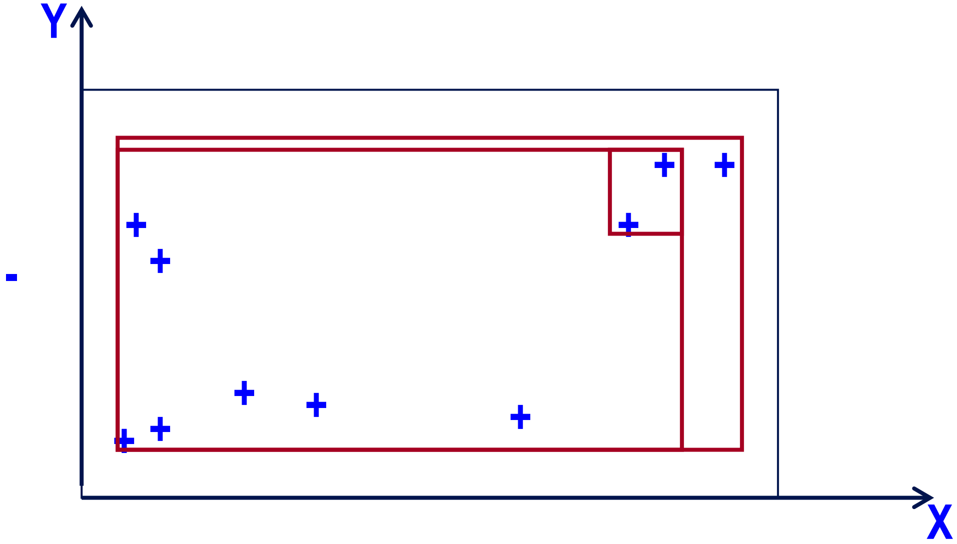
- Assume the target concept is an axis parallel rectangle



- Will we be able to learn the Rectangle?

Learning Rectangles

- Assume the target concept is an axis parallel rectangle

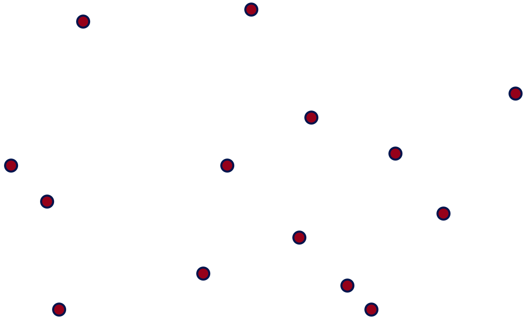


- Will we be able to learn the target rectangle ?
- Can we come close ?

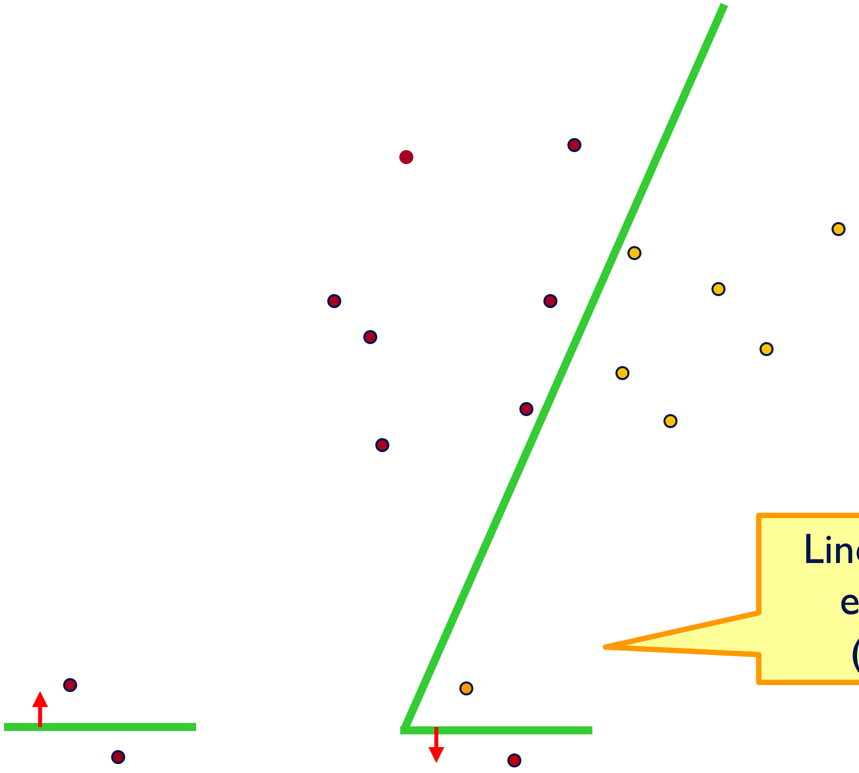
Infinite Hypothesis Space

- 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 $VC(H)$

Shattering

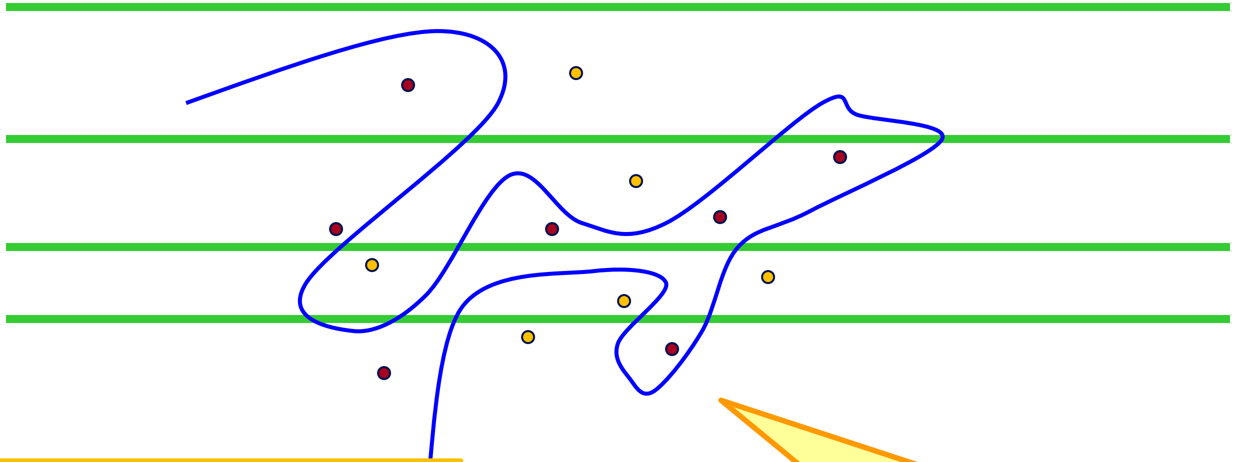


Shattering



Linear functions are expressive enough to shatter 2 points (4 options; not all shown)

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

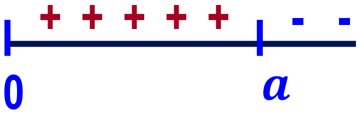
Linear functions are not expressive enough to shatter 13 points

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)

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

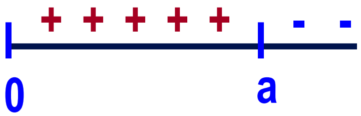


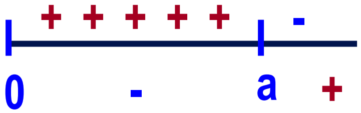
Shattering

- 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 $a > 0$





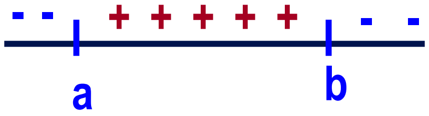
- Sets of two points cannot be shattered (we mean: given two points, you can label them in such a way that no concept in this class will be consistent with their labeling)

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 $b > a$



What is the smallest set of points that CANNOT be shattered by the set of functions defined by Real intervals [a,b]?

1 point

2 points

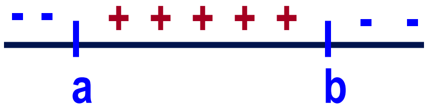
3 points

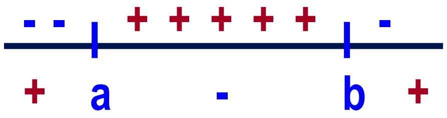
4 points

None of the above

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 $b > a$

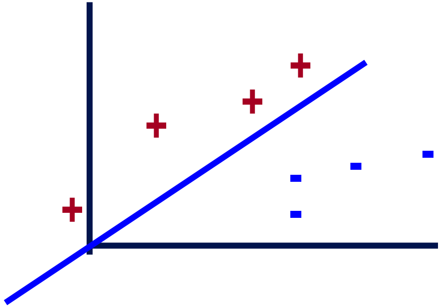




- All sets of one or two points can be shattered but sets of three points cannot be shattered
- Why?
  - Give a labeling configuration of three points that cannot be expressed by any function in this class of functions.

Shattering

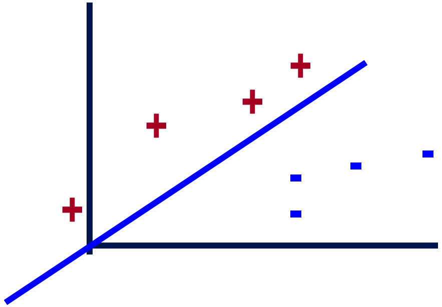
- We say that a set S of examples is shattered by a set of functions H if for every partition of the examples in S into positive and negative examples there is a function in H that gives exactly these labels to the examples
- Half-spaces in the plane:



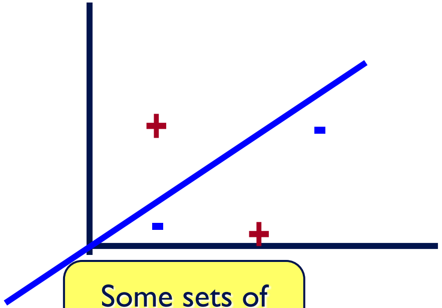
Shattering

- 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:

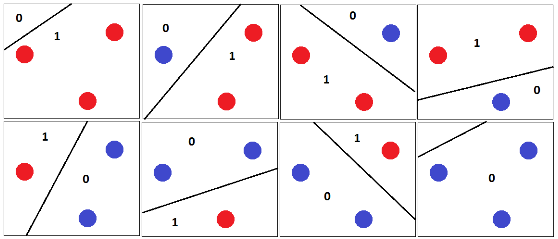


All sets of three points?



Some sets of three points?

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?)



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

VC Dimension: Motivation

- 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 of 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.

Even if only one subset of this size does it!

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 $d + 1$ can be shattered, then $VC(H) < d + 1$

VC(Half intervals) = 1 (no subset of size 2 can be shattered)

VC(Intervals) = 2 (no subset of size 3 can be shattered)

VC(Half-spaces in the plane) = 3 (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 can get an Occam algorithm for infinite hypothesis spaces.

- Given a sample D of m examples, find some $h \epsilon H$ that is consistent with all m examples
- If $m > \frac{1}{\varepsilon}\{8VC(H)\log\frac{13}{\varepsilon} + 4\log\left(\frac{2}{\delta}\right)\}$

What if H is finite?

- Then with probability at least $(1-\delta)$, h has error less than ε. (that is, if m is polynomial we have a PAC learning algorithm; to be efficient, we need to produce the hypothesis h efficiently.

- Note that the notion of VC applies also to finite hypothesis spaces:
  - Assume that H shatters k examples.
  - Notice that to shatter k examples it must be that: $|H| > 2^k$ (why?)
    - So,

$$\log(|H|) \geq VC(H)$$

Assume that H shatters k points; how many different functions must be in H? Respond with: [#, reason]

Learning Rectangles

- Consider axis parallel rectangles in the real plane
- Can we PAC learn it ?

Learning Rectangles

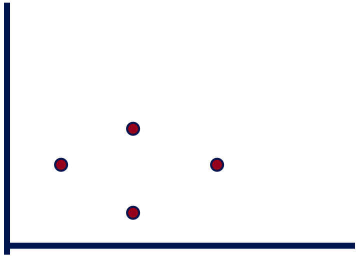
- Consider axis parallel rectangles in the real plane
- Can we PAC learn it ?

  (1) What is the VC dimension ?

Learning Rectangles

- Consider axis parallel rectangles in the real plane
- Can we PAC learn it ?

  (1) What is the VC dimension ?
- Some four instance can be shattered



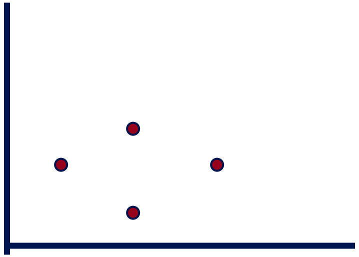
- (need to consider here 16 different rectangles) Shows that $VC(H) \geq 4$

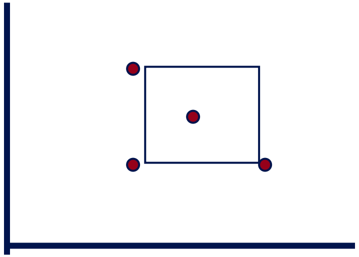
Learning Rectangles

- Consider axis parallel rectangles in the real plane
- Can we PAC learn it ?

  (1) What is the VC dimension ?

- Some four instance can be shattered and some cannot



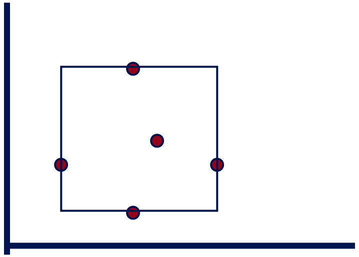


- (need to consider here 16 different rectangles)
- Shows that $VC(H) \geq 4$

Learning Rectangles

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

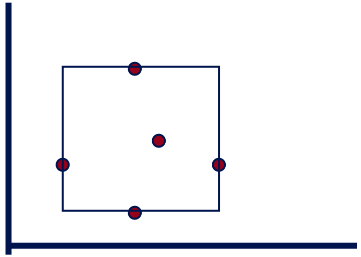
  (1) What is the VC dimension ?
- But, no five instances can be shattered



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.

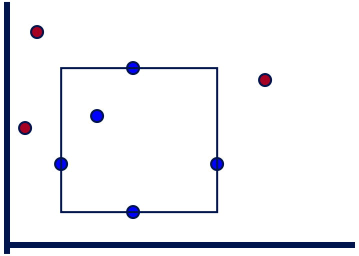
- Therefore $VC(H) = 4$. As far as sample complexity, this guarantees PAC learnability.

Learning Rectangles

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

(1) What is the VC dimension ?

(2) Can we give an efficient algorithm ?

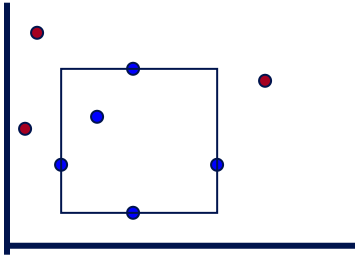


Learning Rectangles

- 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 leaning in the general case.

- Consider any concept class C such that $VC(C) > 2$, any learner L and small enough ε, δ. Then, there exists a distribution D and a target function in C such that if L observes less than

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

  examples, then with probability at least δ, 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 $log\,\frac{1}{\varepsilon}$ factor in the upper bound.

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). They also guide us to try and use smaller hypothesis spaces.
- 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