CS 188: Artificial Intelligence
Spring 2006

Lecture 12: Learning Theory
2/23/2006

Dan Klein – UC Berkeley

Many slides from either Stuart Russell or Andrew Moore

Today

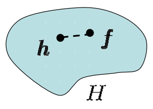
- A Taste of Learning Theory
- Sample Complexity
  - PAC-Learning
  - VC Dimension
- Mistake Bounds
- Note: goal of today is to illustrate what learning theory is like – you don't need to catch all the fine details!

Learning Theory

- Mathematical investigation of learning
- Kinds of things we can show:
  - Sample complexity bounds: how many examples needed to learn the target
  - Generalization bounds: how bad can test error be given training error
  - Mistake bounds (for online learning): how many errors can we make before we learn the target
- Often, make simplifying assumptions:
  - No noise in training labels
  - Target is realizable (i.e, f in H)
  - Test distribution same as training distribution

Realizable Learning

- Learn a realizable function from examples:
  - A hypothesis space H
  - A target function: $f \in H$
  - Examples: input-output pairs $(x, f(x))$
    - E.g. x is an email and $f(x)$ is spam / ham
  - Examples drawn from some distribution D



- Problem:
  - Given a training set of examples $T=\{x_i\}$ with labels $f(x_i)$
  - Find a hypothesis h such that $h \sim f$
  - $h \sim f$ means that the test error of h will be low (more soon)

Train and Test Errors

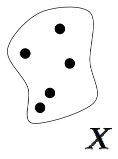
- Training error (or empirical error)
  - Error rate on training set:

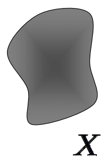
$$\text{ERR}_{\text{TRAIN}}(h,T) = \frac{1}{|T|}\sum_i I(f(x_i) \neq h(x_i))$$

  - Consistency: zero error on training set
- Test error (or true error)
  - Error rate on all examples from D:

$$\text{ERR}_{\text{TEST}}(h,D) = \sum_{x \in X} P_D(x) I(f(x) \neq h(x))$$

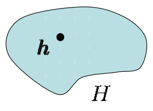
  - h is ε-good if its true error is less than ε
  - We usually have to minimize training error and hope for good generalization to test error



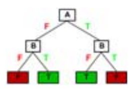


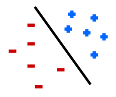
Reminder: Hypothesis Classes

- Hypothesis class H:
  - The set of functions a learner L can learn
  - Distinct from the learner, which has some method for choosing h from H



- Example (binary) hypothesis classes:
  - The constant functions, e.g. {true, false}
  - Decision stumps
  - All binary functions (decision trees)
  - Linear binary decision boundaries
    - NB, Perceptron both learn this class!



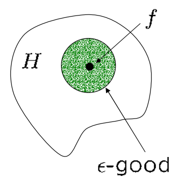


Learning a Target

- What do we mean by "learning" a target function?
- Older approach: learning in the limit
  - Insist on exactly identifying target (eventually)
  - Usually impossible (why?)
- Newer approach: just get "close"
  - Don't need the correct hypothesis
  - Only want one which has very low error (approximately correct)
  - Might draw a really crummy data set
  - Only require that learning usually works (probable learning)
- Probably approximately correct (PAC) learning

PAC Learning

- Setup:
  - Fix class H, learner L
  - Unknown realizable target f
  - Unknown example distribution D
  - L gets N examples from D
  - L picks some h consistent with examples
    - (Assumes this is both possible and efficient)



- Question: sample complexity
  - How many examples do we need before we know that h is probably approximately correct?
  - Formally: what is the smallest N such that with probability at least $1-\delta$ the test error of h will be ε-good?

Bounding Failure Probability

- What does it take to for a learner to fail?
  - There has to be a lucky hypothesis
  - It aces the training data DESPITE being ☜-bad!
- How likely is it for an ☜-bad hypothesis to get one example right?

$$P(\text{one example right} \mid \epsilon\text{-bad}) \leq 1 - \epsilon$$

- How likely is it for a bad hypothesis to get all N examples right?

$$P(\text{all } N \text{ right} \mid \epsilon\text{-bad}) \leq (1-\epsilon)^N$$

- How likely that some hypothesis manages this feat of disguise?
  - At most |H| are bad, and each gets a shot at sneaking by:

$$P(\text{some bad } h \text{ gets all } N \text{ right}) \leq |H|(1-\epsilon)^N$$

Calculating The Sample Bound

- So, probability of failure is

$$P(\text{some bad } h \text{ gets all } N \text{ right}) \leq |H|(1-\epsilon)^N$$

- PAC learning requires failure to be below at most delta (user-supplied)
- So, we want $\delta \leq |H|(1-\epsilon)^N$
- If we solve for N:

$$N \geq \frac{1}{\epsilon}\left(ln|H| + ln\frac{1}{\delta}\right)$$

The Sample Bound

- Let's parse this bound!

$$N \geq \frac{1}{\epsilon}\left(ln|H| + ln\frac{1}{\delta}\right)$$

- Says that the number of samples we need depends on
  - The required epsilon, delta
  - The size of the hypothesis space
  - NOT the data distribution D!
- Shows formally that simpler hypothesis spaces require fewer samples to learn (which we've been suggesting all along)

Practice: Hypothesis Sizes

- Decision stumps over m binary attributes
  - Number: 4m
  - Sample complexity: logarithmic in m!
- Number of disjunctive hypotheses over m attrs
  - E.g.: *SomePatrons* ∨ *LittleWait* ∨ *NoChoice* ∨ *Hungry*
  - Number:
  - Sample complexity:

Practice: Lookup Tables

- H = all truth tables
- Question: if there are m attributes, what is the size of the complete set of hypotheses in f?

$$|H| = 2^{2^m}$$

- Why is this the same as the number of decision trees over m attributes (last class)?
- Sample complexity?

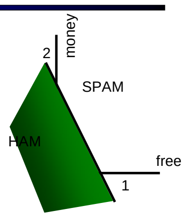
$$N \geq ln|H| = 2^m$$

Bad news!

X1	X2	X3	X4	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Practice: Linear Decisions

- Reminder: (binary) perceptrons learn linear separators
  - Add up the weights of the active features
  - If large enough, positive class
  - Otherwise, negative class
  - Decision boundary is a line / plane / hyperplane

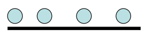


- So, what's |H| for 2-D linear separators?
  - Each hypothesis is a line (and a sign)
  - Number of lines in 2D? $|H| = \infty$
  - Sample complexity? $N \geq \infty$

VERY bad news!

Infinite Hypothesis Spaces

- With continuous parameters, H is infinite
  - E.g. perceptron, naïve Bayes
  - Yet, we never really need infinite samples
  - Explanation: linear separators can't represent very many behaviors on a fixed training set
- Example: N points in a plane
  - How many classifications can we actually make, using a threshold?
  - Only N+1
  - Most labelings can't be represented with this H



VC Dimension

- Vapnik-Chervonenkis (VC) dimension
  - A kind of measure of "effective" size of a hypothesis space |H|
  - Can be finite even in continuous spaces
  - (You will not need to know the details of this!)
- Definition: H shatters a data set T if any labeling of T can given by an h in H
- Example: points on a line, with H = threshold and positive direction

Example: Shattering

- Example: points on a plane
- In general: hyperplanes in R^n can shatter n+1 points

VC Dimension II

- Definition: the VC dimension of a hypothesis class H is the size of the largest set X it can shatter
- Example: VC dimension of the class of linear separators in n dimensions is n+1
- Example: circles around the origin

VC-Based Bounds

- Remember our PAC bound?

$$N \geq \frac{1}{\epsilon}\left(ln|H| + ln\frac{1}{\delta}\right)$$

- Can show a VC-based bound:

$$N \geq \frac{8}{\epsilon}\left(VC(H)\log\frac{13}{\epsilon} + \frac{1}{2}\log\frac{2}{\delta}\right)$$

- (Details and constants are NOT IMPORTANT)
- Modulo details: the log |H| has been replaced with VC(H)
- What does this mean (very loosely) for a perceptron over m features?
- What do you think happens in practice?

Some Things We Won't Show

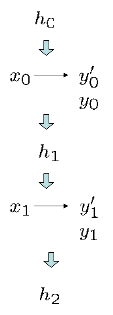
- VC dimension turns out to be very useful
- Many results from learning theory exploit it
- Can show generalization bounds, which bound the error on future examples using the training error and the VC dimension
- This is neat stuff (not always directly correlated with what works in practice, though)

Other Bounds

- Reset!
- So far: sample complexity bounds
- Other kinds of bounds:
  - Mistake bounds (now)
  - Generalization bounds (never)

Online Learning

- Online learning:
  - Receive examples one at a time
  - Make a prediction on each one
  - Learn the label and update hypothesis
  - Can't go back
  - Hopefully, stop making errors at some point
- We've already seen one online algorithm (what)?
- Main bound for online: maximum number of mistakes (ever!)
  - Only works if target realizable
  - In practice, online algorithms usually keep making mistakes forever (like any other method)



Learning Disjunctions

- Hypothesis space: disjunctions over n positive Boolean attributes (features)
- Example:
  - Attributes: *SomePatrons, FrenchFood, HasBar, …*
  - Target (WillEat): *SomePatrons* ∨ *LittleWait* ∨ *NoChoice* ∨ *Hungry*
- An algorithm:
  - Start with all variables in the disjunction
  - When we make a mistake, throw out any positive variables in negative example

Learning Disjunctions

- Example:
  - Hypothesis: *FullPatrons* ∨ *SomePatrons* ∨ *LittleWait* ∨ *NoChoice* ∨ *Hungry* ∨ *FrenchFood* ∨ *HasBar* ∨ *IsWeekend*
  - Example: *SomePatrons* ∧ *LittleWait* ∧ *FrenchFood : true*
  - Example: *FullPatrons* ∧ *FrenchFood* ∧ *IsWeekend : false*
  - Example: *HasBar* ∧ *Hungry : true*
  - Example: *FullPatrons* ∧ *HasBar : false*
- How many mistakes can we possibly make?
  - Each mistake throws out some variable (why?)
  - Can make at most n mistakes! (ever!)

Winnow

- A perceptron-like algorithm
  - We'll do the two-class case
- Algorithm:
  - Start with weight 1 on all features
  - For an example feature vector f(x), we calculate:

$$\sum_i w_i f_i(x) \geq n$$

  - If sum > n, output class 1, otherwise 0
  - If we make a mistake:

Guessed 0 (weights too low)

$$w_i = w_i \cdot 2^{f_i(x)}$$

Guessed 1 (weights too high)

$$w_i = w_i \cdot \frac{1}{2}^{f_i(x)}$$

Winnow Example

"win the match"

"win the vote"

"win the game"

w

```
BIAS  :
win   :
game  :
vote  :
match :
the   :
```

POLITICS is the + class

Winnow Mistake Bound

- Assume the target is a sparse disjunction:
  - k << n variables out of n
  - E.g. there are k spam words out of n total words
  - (rarely entirely true in practice)
- Can show: total mistakes is O(k log n)
- Much better than the previous algorithm!

That's It For Learning Theory

- Hopefully, you've gotten a taste of what LT is about!
- More sophisticated results take into account:
  - Unrealizable functions
  - Noisy labelings
  - Multiple learners (ensembles)
  - How to estimate generalization error
- What I concretely expect you to take away:
  - Understand ε-good, PAC learning criterion
  - Be able to show where the basic bound in |H| comes from
  - Be able to find the size of a (finite) hypothesis space
  - Know what online learning and mistake bounds are