Announcement

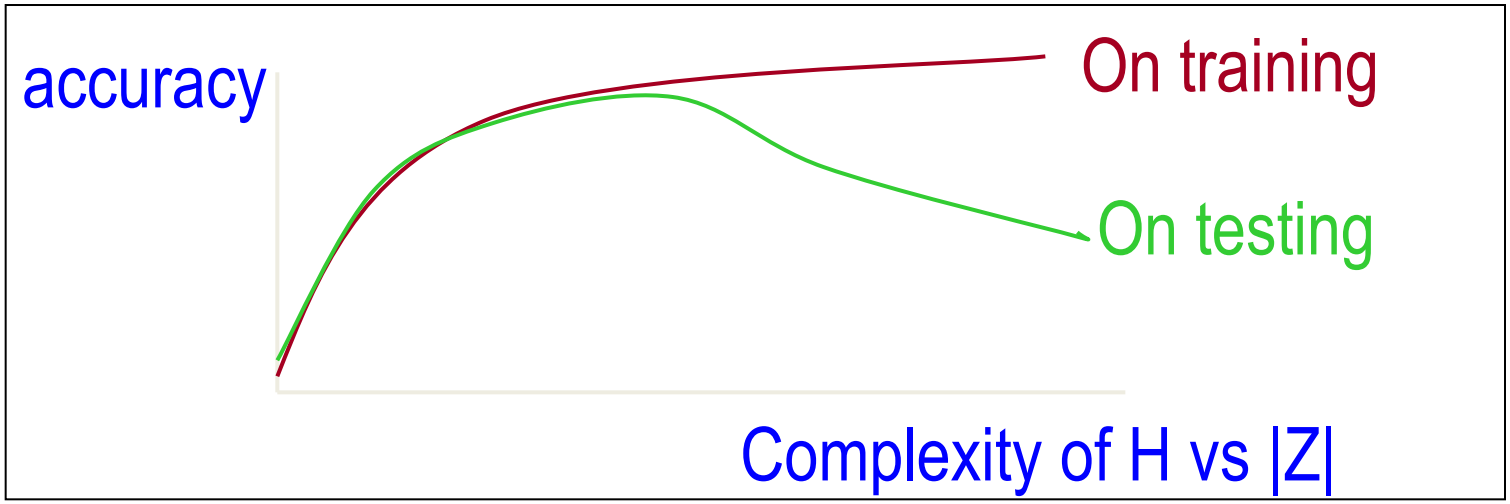
- Watch for next homework tomorrow or Thursday on web site
- On Naïve Bayes and Decision Trees

Overfitting
(very important & general phenomenon!)

- Concept performs
  - well on training data (drawn from X according to $\mathcal{D}$)
  - poorly on unseen examples of interest (same drawing)
- Excess flexibility in hypothesis space H
  - Finds training set pattern not in population
  - Concept selection from too little (insignificant) training data
- Confidence in selecting $c \in H$
  - Diversity of H reflects our ignorance
  - Training set Z provides information
  - Information(Z) $\geq$ Information need(H) [>> for high confidence]
- Often Learning algorithms cannot tell
- With low confidence, expected behavior of "best" concept (on Z) is poor on underlying population
- Extreme:
  - Rote learning of training data
  - No generalization

Overfitting

- Statistical significance of inference
- Decisions with insufficient evidence



Overfitting in Decision Trees

- A rich set of tests
  - Split to (nearly) homogeneity
  - Accurate on training set
  - Poor expected performance on new data
  - Consider noise (but even w/o…)
- With limited training examples, deep decision trees are bad
  - Low confidence in lower split choices (why?)
  - Hypothesis space H become too expressive

How to Reduce Overfitting in Decision Trees

- Restrict the expressiveness of H
- Limit the set of tests available
- Limit the depth
  - No deeper than N (say 3 or 12 or 86)
  - How to choose?
- Limit the minimum number of examples used to select a split
  - Need at least M (is 10 enough? 20?)
  - How to choose? (Adjust for number of tests? Their outcomes?)
  - Want significance; Statistical hypothesis testing can help
- BEST: Learn an overfit tree and prune
  - Partition the labeled examples
  - A: training data – grow the tree
  - B: pruning data – prune the tree from leaves
  - What would be the pruning algorithm?

Important Tradeoffs

- ML: bias – variance – accuracy tradeoff
- (In Statistics: bias – variance tradeoff)
- Learner chooses c $\in$ H based on Z (training set)
- Measures of interest:
  - **Accuracy**: c's performance over true population
  - **Bias**: c's choice not due to Z
  - **Variance**: c's change from a re-sampled Z'
- How likely would we build a similar concept
  - From a different training sample?
  - From this training set used differently?
  - What makes a concept similar?

Variance Reduction: Bagging

- General technique
- Easy, Effective, Useful
- Average over a set of quasi-independent concepts
- Quasi-independent???
- Partition training set Z different ways
- With each, grow & prune a decision tree
- Classify new examples by vote of concepts
- “Decision Forest”

Cross Validation

(Popular & Useful)

- Estimating Parameters
  - Classification Accuracy
  - Blending or other Learning Parameters
- Partition training data into subsets for
  - Training, Testing
- Each partitioning yields quasi-independent learning
- How to partition?
  - Often into N sets
    - N=5, 10,…
    - N=|Z| (Leave One Out)
  - Train on N-1 sets, Test on the remaining one
  - All N ways
- Agreement is evidence for confidence
  Can combine results (e.g., averaging)

Cross Validation

(one approach, DT training using all the data)

- Partition training set N ways
- Train on N-1, Prune on 1 to best tree
- Repeat N times
- Yield quasi-independent estimates of best tree
- Estimate best depth (poor)
- Estimate best number of leaves (better)
- Estimate a model of good decisions (best)
- Decide on a split termination (using above)
- Use all the data, stop at split termination

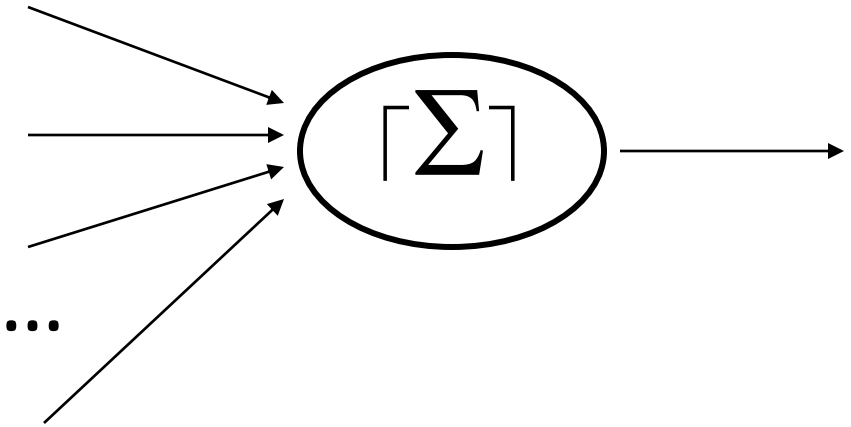
Decision Trees

- Discriminative or Generative?
- Learnability issues
- Decision lists
- Another popular classifier: perceptrons
  - Neural networks
  - Support Vector Machines

Perceptron

Linear Separator, Linear Discriminant, …

- Rosenblatt ~1960
- Analog of a neuron
  McCulloch & Pitts (1943),
  Hebb (1949),
  Widrow (1960)…
  Minsky & Papert (1969)
  Rumelhart, Hinton,… ~1985



- Switching device
- Weighted sum of inputs
- Compare to threshold

Perceptron Example Space

Each inputs values n features

An input is a vector of n components

If input is a vector of Booleans

The n-dimensional Boolean hypercube

If the input is a vector of real numbers

n-dimensional real space $\Re^n$

Perceptron Decision Boundary

Compare weighted sum of inputs to a threshold

$$\sum_{i=1}^{n} w_i \cdot x_i > \theta$$

Without loss of generality

set $x_0 = -1$ then w_0 is θ

$$\sum_{i=0}^{n} w_i \cdot x_i > 0$$

This defines a decision surface

$$\sum_{i=0}^{n} w_i \cdot x_i = \mathbf{w} \cdot \mathbf{x} = 0$$

Which is the equation of

a hyperplane

Perceptron Hypothesis Space

- Perceptron as a classifier
  - An input **x** is labeled
    - Positive + if it is above the hyperplane,
    - else Negative –
- Parameterized family of functions
- Each function: $\mathbf{x} \rightarrow \{+,-\}$
- Family is parameterized by **w**
- So the hypothesis space is…
- The set of linearly bounded half-spaces

Perceptron Concepts

Given any collection of points in $\Re^n$

Given any assignment of + / -

For which is there a perceptron that is consistent with them?

IFF the +'s are linearly separable from the –'s

How many such perceptrons will there be?

Perceptron Concepts

Even though an infinite supply of different concepts, there are some + / - configurations that we cannot handle

Given a perceptron, we are committed to a labeling of all possible inputs

Perceptron Learning

Given a training set of linearly separable + / - labeled points

Can we find a consistent perceptron efficiently?

Trying out lots of **w**'s is not efficient

Thought question:

<How good will it be on future points?>

<Remember we do NOT care (directly) about getting the training set right>

Perceptron Learning

(Widrow-Hoff or delta rule)

percep$_{\mathbf{w}}$(**x**) assigns + or 1 if **w** · **x** > 0 (vector dot product)
else it assigns – or 0

err = label(**x**) – percep$_{\mathbf{w}}$(**x**)
0: correct -1: false pos 1: false neg

Here, false neg: **w** · **x** < 0 but it should be > 0

loss = distance from boundary = - err **w** · **x**

Want to adjust w_i's to reduce this loss

Loss fcn *gradient* is direction of greatest increase in loss with **w**

Want the opposite: step **w** in direction $-\nabla_w$ loss