Machine Learning II
Decision Tree Induction

CSE 573

Logistics

- Reading
  - Ch 13
  - Ch 14 thru 14.3
- Project
  - Writeups due Wednesday November 10
  - … 9 days to go …

Learning from Training Experience

- Credit assignment problem:
  - **Direct** training examples:
    - E.g. individual checker boards + correct move for each
    - Supervised learning
  - **Indirect** training examples :
    - E.g. complete sequence of moves and final result
    - Reinforcement learning
- Which examples:
  - Random, teacher chooses, learner chooses
- Unsupervised Learning

Machine Learning Outline

- Machine learning:
  - √ Function approximation
  - √ Bias
- Supervised learning
  - √ Classifiers & concept learning
  - Decision-trees induction (pref bias)
- Overfitting
- Ensembles of classifiers
- Co-training

Need for Bias

- Example space: 4 Boolean attributes
- How many ML hypotheses?

Two Strategies for ML

- **Restriction bias**: use prior knowledge to specify a restricted hypothesis space.
  - Version space algorithm over conjunctions.
- **Preference bias**: use a broad hypothesis space, but impose an ordering on the hypotheses.
  - Decision trees.

Decision Trees

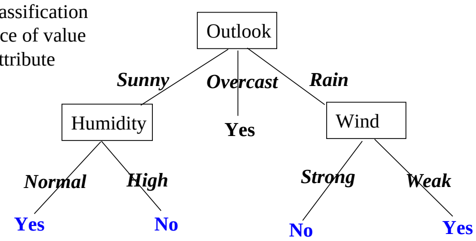
- Convenient Representation
  - Developed with learning in mind
  - Deterministic
- Expressive
  - Equivalent to propositional DNF
  - Handles discrete and continuous parameters
- Simple learning algorithm
  - Handles noise well
  - Classify as follows
    - **Constructive (build DT by adding nodes)**
    - **Eager**
    - **Batch (but incremental versions exist)**

Decision Tree Representation

Good day for tennis?

Leaves = classification

Arcs = choice of value for parent attribute



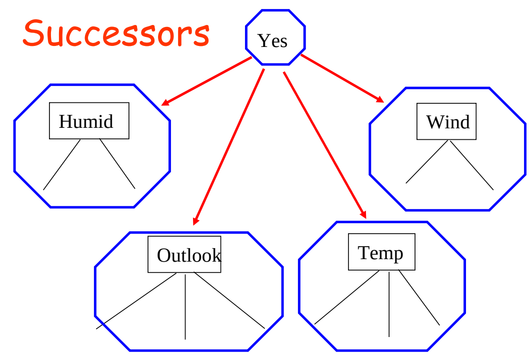
Decision tree is equivalent to logic in disjunctive normal form

G-Day ⇔ (Sunny ∧ Normal) ∨ Overcast ∨ (Rain ∧ Weak)

DT Learning as Search

- Nodes
  - **Decision Trees**
- Operators
  - **Tree Refinement: Sprouting the tree**
- Initial node
  - **Smallest tree possible: a single leaf**
- Heuristic?
  - **Information Gain**
- Goal?
  - **Best tree possible (???)**
- Type of Search?
  - **Hill climbing**

Successors



Which attribute should we use to split?

Decision Tree Algorithm

BuildTree(TraingData)
    Split(TrainingData)

Split(D)
    If (all points in D are of the same class)
        Then Return
    For each attribute A
        Evaluate splits on attribute A
    Use best split to partition D into D1, D2
    Split (D1)
    Split (D2)

Movie Recommendation

- Features?

Rambo								
Matrix								
Rambo 2								
•								
•								
•								

Key Questions

- How to choose best attribute?
  - Mutual Information (Information gain)
    - Entropy (disorder)
- When to stop growing tree?
- Non-Boolean attributes
- Missing data

Issues

- Content *vs.* Social
- Non-Boolean Attributes
- Missing Data
- Scaling up

Missing Data 1

Day	Temp	Humid	Wind	Tennis?
d1	h	h	weak	n
d2	h	h	s	n
d8	m	h	weak	n
d9	c		weak	yes
d11	m	n	s	yes

- Don't use this instance for learning?
- Assign attribute …
  - most common value at node, or
  - most common value, … given classification

Fractional Values

Day	Temp	Humid	Wind	Tennis?
d1	h	h	weak	n
d2	h	h	s	n
d8	m	h	weak	n
d9	c		weak	yes
d11	m	n	s	yes

[0.75+, 3-]

[1.25+, 0-]

- 75% h and 25% n
- Use in information gain calculations
- Further subdivide if other missing attributes
- Same approach to classify test ex with missing attr
  - Classification is most probable classification
  - Summing over leaves where it got divided

Non-Boolean Features

- Features with multiple discrete values
  - Construct a multi-way split
  - Test for one value *vs.* all of the others?
  - Group values into two disjoint subsets?
- Real-valued Features
  - Discretize?
  - Consider a threshold split using observed values?

Attributes with many values

Problem:

- If attribute has many values, *Gain* will select it
- Imagine using $Date = Jun\_3\_1996$ as attribute

- So many values that it
  - Divides examples into tiny sets
  - Each set is likely ***uniform*** → high info gain
  - But poor predictor…
- Need to penalize these attributes

One approach: Gain ratio

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

SplitInfo ≅ entropy of S *wrt* values of A

(Contrast with entropy of S *wrt* **target** value)

⇓ attribs with many uniformly distrib values

e.g. if A splits S uniformly into n sets

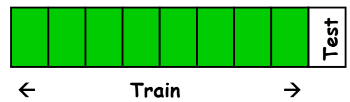
SplitInformation = $\log_2(n)$... = 1 for Boolean

Cross validation

- Partition examples into *k* disjoint equiv classes
- Now create *k* training sets

  Each set is union of all equiv classes ***except one***

  So each set has (k-1)/k of the original training data

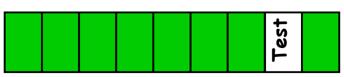


Cross Validation

- Partition examples into *k* disjoint equiv classes
- Now create *k* training sets

  Each set is union of all equiv classes ***except one***

  So each set has (k-1)/k of the original training data

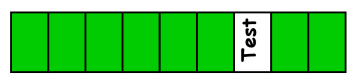


Cross Validation

- Partition examples into *k* disjoint equiv classes
- Now create *k* training sets

  Each set is union of all equiv classes ***except one***

  So each set has (k-1)/k of the original training data



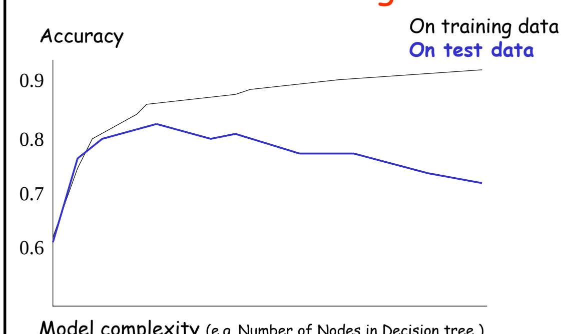
Machine Learning Outline

- Machine learning:
- Supervised learning
- Overfitting

  What is the problem?

  Reduced error pruning
- Ensembles of classifiers
- Co-training

Overfitting



Overfitting…

- DT is *overfit* when exists another DT' and
  - DT has ***smaller*** error on training examples, but
  - DT has ***bigger*** error on test examples
- Causes of overfitting
  - Noisy data, or
  - Training set is too small

Avoiding Overfitting

How can we avoid overfitting?

- Stop growing when data split not statistically significant
- Grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- Add complexity penalty to performance measure

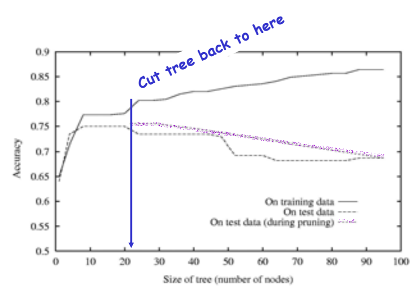
Reduced-Error Pruning

Split data into *training* and *validation* set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

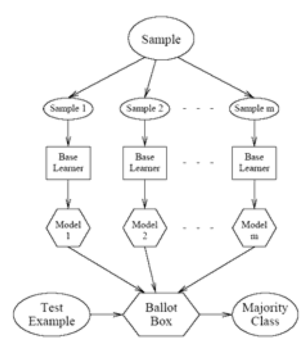
Effect of Reduced-Error Pruning



Machine Learning Outline

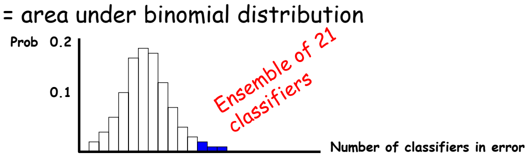
- Machine learning:
- Supervised learning
- Overfitting
- Ensembles of classifiers
  - Bagging
  - Cross-validated committees
  - Boosting
  - Stacking
- Co-training

Voting



Ensembles of Classifiers

- Assume
  - Errors are independent (suppose 30% error)
  - Majority vote
- Probability that majority is wrong…
  - = area under binomial distribution

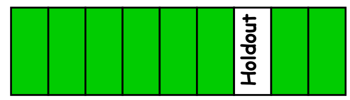


- If individual area is 0.3
- **Area under curve for ≥11 wrong is 0.026**
- Order of magnitude improvement!

Constructing Ensembles

Cross-validated committees

- Partition examples into *k* disjoint equiv classes
- Now create *k* training sets
  - Each set is union of all equiv classes ***except one***
  - So each set has (k-1)/k of the original training data
- Now train a classifier on each set



Ensemble Construction II

Bagging

- Generate k sets of training examples
- For each set
  - Draw m examples randomly (with replacement)
  - From the original set of m examples
- Each training set corresponds to
  - 63.2% of original
  - (+ duplicates)
- Now train classifier on each set

Ensemble Creation III

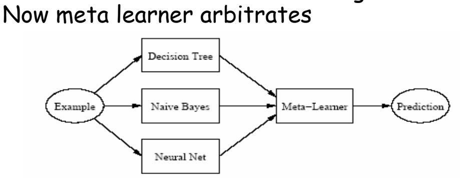
Boosting

- Maintain prob distribution over set of training ex
- Create k sets of training data iteratively:
- On iteration *i*
  - Draw m examples randomly (like bagging)
  - But use probability distribution to bias selection
  - Train classifier number *i* on this training set
  - Test partial ensemble (of *i* classifiers) on all training exs
  - Modify distribution: increase P of each error ex
- Create harder and harder learning problems...
- "Bagging with ***optimized*** choice of examples"

Ensemble Creation IV

Stacking

- Train several base learners
- Next train meta-learner
  - Learns when base learners are right / wrong
  - Now meta learner arbitrates



Train using cross validated committees
- Meta-L inputs = base learner predictions
- Training examples = 'test set' from cross validation

Machine Learning Outline

- Machine learning:
- Supervised learning
- Overfitting
- Ensembles of classifiers
- Co-training

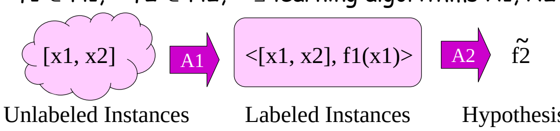
Co-Training Motivation

- Learning methods need labeled data
  - Lots of <x, f(x)> pairs
  - Hard to get… (who wants to label data?)
- But unlabeled data is usually plentiful…
  - Could we use this instead??????

Co-training

Small labeled data needed

- Suppose each instance has two parts:
  - x = [x1, x2]
  - x1, x2 conditionally independent given f(x)
- Suppose each half can be used to classify instance
  - ∃f1, f2 such that f1(x1) = f2(x2) = f(x)
- Suppose f1, f2 are learnable
  - f1 ∈ H1, f2 ∈ H2, ∃ learning algorithms A1, A2



Observations

- Can apply A1 to generate as much training data as one wants
  - If x1 is conditionally independent of x2 / f(x), then the error in the labels produced by A1
    - ***will look like random noise to A2 !!!***
- Thus no limit to quality of the hypothesis A2 can make

It really works!

- Learning to classify web pages as course pages
  - x1 = bag of words on a page
  - x2 = bag of words from all anchors pointing to a page
- Naïve Bayes classifiers
  - 12 labeled pages
  - 1039 unlabeled

	Page-based classifier	Hyperlink-based classifier	Combined classifier
Supervised training	12.9	12.4	11.1
Co-training	6.2	11.6	5.0

Table 2: Error rate in percent for classifying web pages as course home pages. The top row shows errors when training on only the labeled examples. Bottom row shows errors when co-training, using both labeled and unlabeled examples.