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

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.

Need for Bias

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

\[ G-Day \Leftrightarrow (Sunny \land Normal) \lor Overcast \lor (Rain \land Weak) \]

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(TrainingData)

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?**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rambo</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Matrix</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Rambo 2</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Day</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Tennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>h</td>
<td>h</td>
<td>weak</td>
<td>n</td>
</tr>
<tr>
<td>d2</td>
<td>h</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
<tr>
<td>d8</td>
<td>m</td>
<td>h</td>
<td>weak</td>
<td>n</td>
</tr>
<tr>
<td>d9</td>
<td>c</td>
<td>weak</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>d11</td>
<td>m</td>
<td>n</td>
<td>s</td>
<td>yes</td>
</tr>
</tbody>
</table>

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

Fractional Values

<table>
<thead>
<tr>
<th>Day</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Tennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>h</td>
<td>h</td>
<td>weak</td>
<td>n</td>
</tr>
<tr>
<td>d2</td>
<td>h</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
<tr>
<td>d8</td>
<td>m</td>
<td>h</td>
<td>weak</td>
<td>n</td>
</tr>
<tr>
<td>d9</td>
<td>c</td>
<td>weak</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>d11</td>
<td>m</td>
<td>n</td>
<td>s</td>
<td>yes</td>
</tr>
</tbody>
</table>

- 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 \( \rightarrow \) high info gain
  - But poor predictor...

- Need to penalize these attributes
One approach: Gain ratio

\[ \text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)} \]

\[ \text{SplitInformation}(S, A) = -\sum_{i} \frac{|S_i|}{|S|} \log_2 \left( \frac{|S_i|}{|S|} \right) \]

where \( S_i \) is subset of \( S \) for which \( A \) has value \( a_i \)

\( \text{SplitInfo} \equiv \text{entropy of } S \text{ wrt values of } A \)
(Contrast with entropy of \( S \) wrt target value)

\[ \downarrow \text{attrs with many uniformly distrib values} \]
e.g. if \( A \) splits \( S \) uniformly into \( n \) sets
\[ \text{SplitInformation} = \log_2(n) \ldots = 1 \text{ for Boolean} \]

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

Machine Learning Outline

- Machine learning:
- Supervised learning
- Overfitting
  - What is the problem?
  - Reduced error pruning
- Ensembles of classifiers
- Co-training

Overfitting

Accuracy

On training data

On test data

Model complexity (e.g. Number of Nodes in Decision tree)
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,
  - 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

- Suppose each instance has two parts:
  - \(x = [x_1, x_2]\)
  - \(x_1, x_2\) conditionally independent given \(f(x)\)
- Suppose each half can be used to classify instance
  - \(\exists f_1, f_2\) such that \(f_1(x_1) = f_2(x_2) = f(x)\)
- Suppose \(f_1, f_2\) are learnable
  - \(f_1 \in H_1, f_2 \in H_2\).
  - \(\exists\) learning algorithms \(A_1, A_2\)

Unlabeled Instances  \([x_1, x_2]\)  \(A_1\)  \(<[x_1, x_2], f_1(x_1)>\)  \(A_2\)  \(f_2\)
Labeled Instances  \(\sim\)
Hypothesis

Observations

- Can apply \(A_1\) to generate as much training data as one wants
  - If \(x_1\) is conditionally independent of \(x_2 / f(x)\), then the error in the labels produced by \(A_1\) will look like random noise to \(A_2\) !!!
- Thus no limit to quality of the hypothesis \(A_2\) can make

It really works!

- Learning to classify web pages as course pages
  - \(x_1 = \text{bag of words on a page}\)
  - \(x_2 = \text{bag of words from all anchors pointing to a page}\)
- Naïve Bayes classifiers
  - 12 labeled pages
  - 1039 unlabeled

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Vote</td>
<td>7.0</td>
<td>7.0</td>
<td>7.0</td>
</tr>
<tr>
<td>Co-training</td>
<td>12.0</td>
<td>12.0</td>
<td>12.0</td>
</tr>
<tr>
<td>Unlabeled</td>
<td>6.0</td>
<td>6.0</td>
<td>6.0</td>
</tr>
</tbody>
</table>

Table 1: Error rates in percent for classifying web pages as course home pages. The top row shows errors when training on only the labeled instances. Bottom row shows errors when co-training using both labeled and unlabeled instances.