Outline

• Decision Trees
  – Representation
  – Learning Algorithm
  – Potential pitfalls

• Experimental Methodology
Decision Trees

• Popular hypothesis space
  – Developed with learning in mind
  – Deterministic
  – Simple learning algorithm
  – Handles noise well
  – Produce comprehensible output
Decision Trees

• Effective hypothesis space
  – Variable sized hypotheses
  – Can represent any Boolean function
  – Can represent both discrete and continuous features
  – Equivalent to propositional DNF

• Classify learning algorithm as follows:
  – Constructive search: Learn by adding nodes
  – Eager
  – Batch [though online algorithms exist]
Good day for tennis?

Leaves = classification

Arcs = choice of value for parent attribute

Decision tree is equivalent to logic in disjunctive normal form

\[ \text{Play} \Leftrightarrow (\text{Sunny} \land \text{Normal}) \lor \text{Overcast} \lor (\text{Rain} \land \text{Weak}) \]
Use thresholds to convert numeric attributes into discrete values.
How Do Decision Trees Partition Feature Space?

Decisions divide feature space into axis parallel rectangles and labels each one with one of the K classes.
Decision Trees Provide Variable-Size Hypothesis Space

• As the number of nodes (or tree depth) increases, the hypothesis space grows
  – Depth 1 (decision “stumps”): Any Boolean function over one variable
  – Depth 2:
    • Any Boolean function over two variables
    • Some Boolean functions over three variables
e.g., \((x_1 \land x_2) \lor (!x_1 \land !x_3)\)
  – Etc.
Decision Trees Can Represent Any Boolean Function

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) 0 0</td>
<td>-</td>
</tr>
<tr>
<td>b) 0 1</td>
<td>+</td>
</tr>
<tr>
<td>c) 1 0</td>
<td>+</td>
</tr>
<tr>
<td>d) 1 1</td>
<td>-</td>
</tr>
</tbody>
</table>

However, in the worst case, the tree will require exponential many nodes.
Objective of DT Learning

**Goal**: Find the decision tree that minimizes the error rate on the training data

- **Solution 1**: For each training example, create one root-to-leaf path
- **Problem 1**: Just memorizes the training data
- **Solution 2**: Find smallest tree that minimizes our error function
- **Problem 2**: This is NP-hard
- **Solution 3**: Use a greedy approximation
DT Learning as Search

• Nodes
  Decision Trees:
    1) Internal: Attribute-value test
    2) Leaf: Class label

• Operators
  Tree Refinement: Sprouting the tree

• Initial node
  Smallest tree possible: a single leaf

• Heuristic?
  Information Gain

• Goal?
  Best tree possible (???)
Decision Tree Algorithm

\textbf{BuildTree}(TrainingData)
\hspace{1em} Split(TrainingData)

\textbf{Split}(D)
\hspace{1em} If (all points in D are of the same class)
\hspace{2em} Then Return
\hspace{1em} For each attribute A
\hspace{2em} Evaluate splits on attribute A
\hspace{2em} Use best split to partition D into D1, D2
\hspace{1em} \textbf{Split} (D1)
\hspace{1em} \textbf{Split} (D2)
What is the Simplest Tree?

<table>
<thead>
<tr>
<th>Day Outlook</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>s</td>
<td>h</td>
<td>h</td>
<td>w</td>
</tr>
<tr>
<td>d2</td>
<td>s</td>
<td>h</td>
<td>h</td>
<td>s</td>
</tr>
<tr>
<td>d3</td>
<td>o</td>
<td>h</td>
<td>h</td>
<td>w</td>
</tr>
<tr>
<td>d4</td>
<td>r</td>
<td>m</td>
<td>h</td>
<td>w</td>
</tr>
<tr>
<td>d5</td>
<td>r</td>
<td>c</td>
<td>n</td>
<td>w</td>
</tr>
<tr>
<td>d6</td>
<td>r</td>
<td>c</td>
<td>n</td>
<td>s</td>
</tr>
<tr>
<td>d7</td>
<td>o</td>
<td>c</td>
<td>n</td>
<td>s</td>
</tr>
<tr>
<td>d8</td>
<td>s</td>
<td>m</td>
<td>h</td>
<td>w</td>
</tr>
<tr>
<td>d9</td>
<td>s</td>
<td>c</td>
<td>n</td>
<td>w</td>
</tr>
<tr>
<td>d10</td>
<td>r</td>
<td>m</td>
<td>n</td>
<td>w</td>
</tr>
<tr>
<td>d11</td>
<td>s</td>
<td>m</td>
<td>n</td>
<td>s</td>
</tr>
<tr>
<td>d12</td>
<td>o</td>
<td>m</td>
<td>h</td>
<td>s</td>
</tr>
<tr>
<td>d13</td>
<td>o</td>
<td>h</td>
<td>n</td>
<td>w</td>
</tr>
<tr>
<td>d14</td>
<td>r</td>
<td>m</td>
<td>h</td>
<td>s</td>
</tr>
</tbody>
</table>

How good?

[9+, 5-]  

Majority class:  
correct on 9 examples  
incorrect on 5 examples
Successors

Yes

Humid

Wind

Outlook

Temp

Which attribute should we use to split?
Choosing the Best Attribute

One way to choose the best attribute is to perform a 1-step lookahead search and choose the attribute that gives the lowest error rate on the training data.

**CHOOSEBESTATTRIBUTE(S)**

choose $j$ to minimize $J_j$, computed as follows:

$S_0 =$ all $(x, y) \in S$ with $x_j = 0$;

$S_1 =$ all $(x, y) \in S$ with $x_j = 1$;

$y_0 =$ the most common value of $y$ in $S_0$

$y_1 =$ the most common value of $y$ in $S_1$

$J_0 =$ number of examples $(x, y) \in S_0$ with $y \neq y_0$

$J_1 =$ number of examples $(x, y) \in S_1$ with $y \neq y_1$

$J_j = J_0 + J_1$ (total errors if we split on this feature)

**return** $j$
Choosing the Best Attribute: Example

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) 0 0 0</td>
<td>+</td>
</tr>
<tr>
<td>b) 0 0 1</td>
<td>-</td>
</tr>
<tr>
<td>c) 0 1 0</td>
<td>+</td>
</tr>
<tr>
<td>d) 0 1 1</td>
<td>+</td>
</tr>
<tr>
<td>e) 1 0 0</td>
<td>-</td>
</tr>
<tr>
<td>f) 1 0 1</td>
<td>+</td>
</tr>
<tr>
<td>g) 1 1 0</td>
<td>-</td>
</tr>
<tr>
<td>h) 1 1 1</td>
<td>-</td>
</tr>
</tbody>
</table>

Diagram:

- \( X_1 \) with \( J = 2 \)
- \( X_2 \) with \( J = 4 \)
- \( X_3 \) with \( J = 4 \)
Choosing the Best Attribute: Example

This metric may not work well as it does not always detect cases where we are making progress towards the goal.
A Better Metric From Information Theory

Intuition: Disorder is bad and homogeneity is good
Entropy

50-50 class split
Maximum disorder

All positive
Pure distribution

% of example that are positive
Entropy (disorder) is bad
Homogeneity is good

- Let S be a set of examples
- Entropy(S) = \(-P \log_2(P) - N \log_2(N)\)
  - \(P\) is proportion of pos example
  - \(N\) is proportion of neg examples
  - \(0 \log 0 = 0\)
- Example: S has 9 pos and 5 neg
  Entropy([9+, 5-]) = -(9/14) \log_2(9/14) - (5/14)\log_2(5/14)
  = 0.940
Information Gain

- Measure of expected *reduction* in entropy
- Resulting from splitting along an attribute

\[
\text{Gain}(S,A) = \text{Entropy}(S) - \sum \frac{|S_v|}{|S|} \text{Entropy}(S_v)
\]
\[
\sum_{v \in \text{Values}(A)}
\]

Where \(\text{Entropy}(S) = -P \log_2(P) - N \log_2(N)\)
Example: “Good day for tennis”

• Attributes of instances
  – Outlook = \{rainy (r), overcast (o), sunny (s)\}
  – Temperature = \{cool (c), medium (m), hot (h)\}
  – Humidity = \{normal (n), high (h)\}
  – Wind = \{weak (w), strong (s)\}

• Class value
  – Play Tennis? = \{don’t play (n), play (y)\}

• Feature = attribute with one value
  – E.g., outlook = \textit{sunny}

• Sample instance
  – outlook=\textit{sunny}, temp=\textit{hot}, humidity=\textit{high}, wind=\textit{weak}
Experience: “Good day for tennis”

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>s</td>
<td>h</td>
<td>h</td>
<td>w</td>
<td>n</td>
</tr>
<tr>
<td>d2</td>
<td>s</td>
<td>h</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
<tr>
<td>d3</td>
<td>o</td>
<td>h</td>
<td>h</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d4</td>
<td>r</td>
<td>m</td>
<td>h</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d5</td>
<td>r</td>
<td>c</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d6</td>
<td>r</td>
<td>c</td>
<td>n</td>
<td>s</td>
<td>n</td>
</tr>
<tr>
<td>d7</td>
<td>o</td>
<td>c</td>
<td>n</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d8</td>
<td>s</td>
<td>m</td>
<td>h</td>
<td>w</td>
<td>n</td>
</tr>
<tr>
<td>d9</td>
<td>s</td>
<td>c</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d10</td>
<td>r</td>
<td>m</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d11</td>
<td>s</td>
<td>m</td>
<td>n</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d12</td>
<td>o</td>
<td>m</td>
<td>h</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d13</td>
<td>o</td>
<td>h</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d14</td>
<td>r</td>
<td>m</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
</tbody>
</table>
Gain of Splitting on Wind

Values(wind)=weak, strong
S = [9+, 5-]
S_{weak} = [6+, 2-]
S_s = [3+, 3-]

Gain(S, wind)
= Entropy(S) - \sum (|S_v| / |S|) Entropy(S_v)
   \quad v \in \{\text{weak, s}\}
= Entropy(S) - 8/14 \text{Entropy}(S_{\text{weak}})
- 6/14 \text{Entropy}(S_s)
= 0.940 - (8/14) 0.811 - (6/14) 1.00
= .048
Evaluating Attributes

Gain(S, Humid) = 0.151

Gain(S, Outlook) = 0.246

Gain(S, Temp) = 0.029

Gain(S, Wind) = 0.048
Good day for tennis?

- **Sunny**: Don’t Play [2+, 3-]
- **Overcast**: Play [4+]
- **Rain**: Don’t Play [3+, 2-]
Recurse

Good day for tennis?

<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>n</td>
<td>weak</td>
<td>yes</td>
</tr>
<tr>
<td>d11</td>
<td>m</td>
<td>n</td>
<td>s</td>
<td>yes</td>
</tr>
</tbody>
</table>
Good day for tennis?

- **Outlook**: Sunny, Overcast, Rain
- **Humidity**: High, Normal

- Sunny: Play [4+]
- Overcast: Don’t Play [2+, 3-]
- Rain: Don’t Play [2+, 3-]
- High Humidity: Don’t play [3-]
- Normal Humidity: Play [2+]
Good day for tennis?

- **Outlook**
  - Sunny
    - Humidity
      - High: Don't play [3-]
      - Normal: Play [2+]
  - Overcast
    - Play [4+]
  - Rain
    - Wind
      - Strong: Don't play [2-]
      - Weak: Play [3+]
Issues

• Missing data
• Real-valued attributes
• Many-valued features
• Evaluation
• Overfitting
### 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>?</td>
<td>weak</td>
<td>yes</td>
</tr>
<tr>
<td>d11</td>
<td>m</td>
<td>n</td>
<td>s</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Assign most common value at this node**

? => h

**Assign most common value for class**

? => n
Missing Data 2

- 75% h and 25% n
- Use in 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

<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>?</td>
<td>weak</td>
<td>yes</td>
</tr>
<tr>
<td>d11</td>
<td>m</td>
<td>n</td>
<td>s</td>
<td>yes</td>
</tr>
</tbody>
</table>
Real-Valued Features

• Discretize?
  Wind  25 12 12|11 10 10|8 7 7 7 7|6 6 6 5
  Play  n y y n y n n y y y y y y y n

• Threshold split using observed values?
  Wind  8 25 7 6 6 10 12 5 7 7 12 10 7 11
  Play  n n y y y n y n y y y y y y y n

Wind  25 12 12|11 10 10|8 7 7 7 7|6 6 6 5
Play  n y y n y n n y y y y y y y n

>= 12
Gain = 0.0004

>= 10
Gain = 0.048
Real-Valued Features

Note

Cannot discard numeric feature after use in one portion of d-tree
Many-Valued Attributes

FAVORS FEATURES WITH **HIGH BRANCHING FACTORS**
(i.e., many possible values)

Extreme Case:

At most one example per leaf and all I(.,.) scores for leafs equals zero, so gets perfect score! But generalizes very poorly (i.e., memorizes data)
Fix: Method 1

Convert all features to binary

e.g., Color = \{Red, Blue, Green\}

From 1 $N$-valued feature to $N$ binary features

- Color = Red?            \{True, False\}
- Color = Blue?           \{True, False\}
- Color = Green?          \{True, False\}

Used in Neural Nets and SVMs

D-tree readability probably less, but not necessarily
Fix 2: Gain Ratio

Gain Ratio(S,A) = Gain(S,A)/SplitInfo(S,A)

\[ \text{SplitInfo} = \sum_{v \in \text{Values}(A)} \left( \frac{|S_v|}{|S|} \right) \log_2 \left( \frac{|S_v|}{|S|} \right) \]

\[ \text{SplitInfo} \approx \text{entropy of S wrt values of A} \]

(Contrast with entropy of S wrt target value)

\[ \downarrow \text{attrs with many uniformly distrib values} \]

\[ \text{e.g. if A splits S uniformly into n sets} \]

\[ \text{SplitInformation} = \log_2(n) \ldots = 1 \text{ for Boolean} \]
Evaluation

• Question: How well will an algorithm perform on unseen data?

• Cannot score based on training data
  – Estimate will be overly optimistic about algorithm’s performance
Evaluation: Cross Validation

- Partition examples into $k$ disjoint sets
- 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 (2)

• Leave-one-out
  – Use if < 100 examples (rough estimate)
  – Hold out one example, train on remaining examples

• M of N fold
  – Repeat M times
  – Divide data into N folds, do N fold cross-validation
Consider adding a noisy training example:

*Sunny, Hot, Normal, Strong, PlayTennis=No*

What effect on tree?
Overfitting

Accuracy

On training data
On test data

Number of Nodes in Decision tree
Overfitting Definition

• 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

• Solutions
  – Reduced error pruning
  – Early stopping
  – Rule post pruning
Reduced Error Pruning

• Split data into train and validation set

• Repeat until pruning is harmful
  – Remove each subtree and replace it with majority class and evaluate on validation set
  – Remove subtree that leads to largest gain in accuracy
Reduced Error Pruning Example

Validation set accuracy = 0.75
Reduced Error Pruning Example

Validation set accuracy = 0.80
Reduced Error Pruning Example

- Outlook
  - Sunny
    - Humidity
      - High: Don’t play
      - Normal: Play
  - Overcast: Play
- Rain
  - Wind
    - Strong: Don’t play
    - Weak: Play
Reduced Error Pruning Example

Validation set accuracy = 0.70
Reduced Error Pruning Example

Use this as final tree
Early Stopping

Accuracy

Remember this tree and use it as the final classifier

On training data
On test data
On validation data

Number of Nodes in Decision tree
Post Rule Pruning

• Split data into train and validation set

• Prune each rule independently
  – Remove each pre-condition and evaluate accuracy
  – Pick pre-condition that leads to largest improvement in accuracy

• Note: ways to do this using training data and statistical tests
Conversion to Rule

Outlook = Sunny ∧ Humidity = High ⇒ Don’t play
Outlook = Sunny ∧ Humidity = Normal ⇒ Play
Outlook = Overcast ⇒ Play

…
Example

Outlook = Sunny \land Humidity = High \Rightarrow Don't play

Validation set accuracy = 0.68

\rightarrow Outlook = Sunny \Rightarrow Don't play \quad Validation set accuracy = 0.65

\rightarrow Humidity = High \Rightarrow Don't play \quad Validation set accuracy = 0.75

Keep this rule
15 Minute Break
Outline

• Decision Trees

• Experimental Methodology
  – Methodology overview
  – How to present results
  – Hypothesis testing
Experimental Methodology: A Pictorial Overview

Statistical techniques such as 10-fold cross validation and t-tests are used to get meaningful results.

LEARNER

training examples

generate solutions

train' set

tune set

select best

testing examples

classifier

expected accuracy on future examples

collection of classified examples
Using Tuning Sets

• Often, an ML system has to choose when to stop learning, select among alternative answers, etc.

• One wants the model that produces the highest accuracy on future examples (“overfitting avoidance”)

• It is a “cheat” to look at the test set while still learning

• Better method
  – Set aside part of the training set
  – Measure performance on this “tuning” data to estimate future performance for a given set of parameters
  – Use best parameter settings, train with all training data (except test set) to estimate future performance on new examples
Proper Experimental Methodology Can Have a Huge Impact!

A 2002 paper in Nature (a major, major journal) needed to be corrected due to “training on the testing set”

Original report : 95% accuracy (5% error rate)
Corrected report (which still is buggy):
  73% accuracy (27% error rate)

Error rate increased over 400%!!!
Parameter Setting

Notice that each train/test fold may get different parameter settings!
  – That’s fine (and proper)

I.e., a “parameterless”* algorithm internally sets parameters for each data set it gets
Using Multiple Tuning Sets

• Using a **single** tuning set can be unreliable predictor, plus some data “wasted”
  Hence, often the following is done:

  1) For each possible set of parameters,
      a) Divide training data into **train’** and **tune** sets, using **N-fold cross validation**
      b) Score this set of parameter value, average **tune** set accuracy
  2) Use **best** set of parameter settings and **all (train’ + tune)** examples
  3) Apply resulting model to **test** set
Tuning a Parameter - Sample Usage

Step 1: Try various values for $k$ (e.g., neighborhood size/distance function in k-NN).

Use 10 train/tune splits for each $k$:

- **K=0**
  
  Tune set accuracy (ave. over 10 runs) = 92%

- **K=2**
  
  Tune set accuracy (ave. over 10 runs) = 97%

- **K=100**
  
  Tune set accuracy (ave. over 10 runs) = 80%

Step 2: Pick best value for $k$ (e.g., $k = 2$), then train using all training data.

Step 3: Measure accuracy on test set.
What to Do for the FIELDED System?

• Do not use any test sets
• Instead only use tuning sets to determine good parameters
  – Test sets used to estimate future performance
  – You can report this estimate to your “customer,” then use all the data to retrain a “product” to give them
What’s Wrong with This?

1. Do a cross-validation study to set parameters
2. Do another cross-validation study, using the best parameters, to estimate future accuracy
   • How will this relate to the “true” future accuracy?
   • Likely to be an overestimate

What about

1. Do a proper train/tune/test experiment
2. Improve your algorithm; goto 1
   (Machine Learning’s “dirty little” secret!)
Why Not Learn After Each Test Example?

• In “production mode,” this would make sense (assuming one received the correct label)

• In “experiments,” we wish to estimate
  
  **Probability we’ll label the next example correctly**
  
  need **several samples** to accurately estimate
Outline

• Decision Trees

• Experimental Methodology
  – Methodology overview
  – How to present results
  – Hypothesis testing
Scatter Plots
- Compare Two Algo’s on Many Datasets

Each dot is the error rate of the two algo’s on ONE dataset.
**Evaluation Metrics**

Called a confusion matrix or contingency table

<table>
<thead>
<tr>
<th></th>
<th>Predicted True</th>
<th>Predicted False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actually True</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>Actually False</td>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>

The number of times true is “confused” with false by the algorithm.
ROC Curves

• ROC: Receiver Operating Characteristics
• Started during radar research during WWII
• Judging algorithms on accuracy alone may not be good enough when getting a positive wrong costs more than getting a negative wrong (or vice versa)
  – Eg, medical tests for serious diseases
  – Eg, a movie-recommender (ala’ NetFlix) system
Evaluation Metrics

True positive rate (tpr) = \frac{TP}{TP + FN}

False positive rate (fpr) = \frac{FP}{TN + FP}

<table>
<thead>
<tr>
<th></th>
<th>Predicted True</th>
<th>Predicted False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actually True</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>Actually False</td>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>
ROC Curves Graphically

Different algorithms can work better in different parts of ROC space. This depends on cost of false + vs false -
Creating an ROC Curve - the Standard Approach

• You need an ML algorithm that outputs NUMERIC results such as prob(example is +)
• You can use ensembles (later) to get this from a model that only provides Boolean outputs
  – Eg, have 100 models vote & count votes
Algorithm for Creating ROC Curves

Step 1: Sort predictions on test set

Step 2: Locate a threshold between examples with opposite categories

Step 3: Compute TPR & FPR for each threshold of Step 2

Step 4: Connect the dots
### Plotting ROC Curves

#### Example

<table>
<thead>
<tr>
<th>ML Algo Output (Sorted)</th>
<th>Correct Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex 9 0.99</td>
<td>+</td>
</tr>
<tr>
<td>Ex 7 0.98</td>
<td>TPR=(2/5), FPR=(0/5) +</td>
</tr>
<tr>
<td>Ex 1 0.72</td>
<td>TPR=(2/5), FPR=(1/5) -</td>
</tr>
<tr>
<td>Ex 2 0.70</td>
<td>+</td>
</tr>
<tr>
<td>Ex 6 0.65</td>
<td>TPR=(4/5), FPR=(1/5) +</td>
</tr>
<tr>
<td>Ex 10 0.51</td>
<td>-</td>
</tr>
<tr>
<td>Ex 3 0.39</td>
<td>TPR=(4/5), FPR=(3/5) -</td>
</tr>
<tr>
<td>Ex 5 0.24</td>
<td>TPR=(5/5), FPR=(3/5) +</td>
</tr>
<tr>
<td>Ex 4 0.11</td>
<td>-</td>
</tr>
<tr>
<td>Ex 8 0.01</td>
<td>TPR=(5/5), FPR=(5/5) -</td>
</tr>
</tbody>
</table>

The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. Each point on the curve corresponds to a different threshold for classifying the output of the machine learning algorithm. The curve shows the trade-off between TPR and FPR as the threshold changes.
ROC’s and Many Models (not in the ensemble sense)

• It is not necessary that we learn one model and then threshold its output to produce an ROC curve
• You could learn different models for different regions of ROC space
• For example, see Goadrich, Oliphant, & Shavlik ILP ’04 and MLJ ‘06
Area Under ROC Curve

A common metric for experiments is to numerically integrate the ROC Curve.

AUC = Wilcoxon-Mann-Whitney Statistic
ROC’s & Skewed Data

• One strength of ROC curves is that they are a good way to deal with skewed data (|+| >> |-|) since the axes are fractions (rates) independent of the # of examples.

• You must be careful though!

• Low FPR * (many negative ex) = sizable number of FP

• Possibly more than # of TP
Evaluation Metrics: Precision and Recall

Recall = \[ \frac{TP}{TP + FN} \]

Precision = \[ \frac{TP}{TP + FP} \]

<table>
<thead>
<tr>
<th></th>
<th>Predicted True</th>
<th>Predicted False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actually True</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>Actually False</td>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>
ROC vs. Recall-Precision

You can get very different visual results on the same data

The reason for this is that there may be lots of \( \text{ex's} \) (eg, might need to include 100 neg's to get 1 more pos)
Two Highly Skewed Domains

Is an abnormality on a mammogram benign or malignant?

Do these two identities refer to the same person?
Diagnosing Breast Cancer

[Real Data: Davis et al. IJCAI 2005]
Diagnosing Breast Cancer

[Real Data: Davis et al. IJCAI 2005]
Predicting Aliases

[Synthetic data: Davis et al. ICIA 2005]
Predicting Aliases

[Synthetic data: Davis et al. ICIA 2005]

**PR Space**

- Algorithm 1
- Algorithm 2
- Algorithm 3
Four Questions about PR space and ROC space

• Q1: If a curve dominates in one space will it dominate in the other?
• Q2: What is the “best” PR curve?
• Q3: How do you interpolate in PR space?
• Q4: Does optimizing AUC in one space optimize it in the other space?
Definition: Dominance

![Graph showing precision and recall for Algorithm 1 and Algorithm 2]
A1: Dominance Theorem

For a fixed number of positive and negative examples, one curve dominates another curve in ROC space if and only if the first curve dominates the second curve in PR space.
Q2: What is the “best” PR curve?

• The “best” curve in ROC space for a set of points is the convex hull [Provost et al ’98]
  – It is achievable
  – It maximizes AUC

Q: Does an analog to convex hull exist in PR space?

A2: Yes! We call it the Achievable PR Curve
Convex Hull

ROC Space

True Positive Rate

False Positive Rate

Original Points

Lecture #7, Slide 89
Convex Hull

ROC Space

True Positive Rate vs False Positive Rate

Convex Hull
Original Points
A2: Achievable Curve
A2: Achievable Curve
Constructing the Achievable Curve

Given: Set of PR points, fixed number positive and negative examples

- Translate PR points to ROC points
- Construct convex hull in ROC space
- Convert the curve into PR space

Corollary:

By dominance theorem, the curve in PR space dominates all other legal PR curves you could construct with the given points
Q3: Interpolation

- Interpolation in ROC space is easy
- Linear connection between points
Linear Interpolation Not Achievable in PR Space

- **Precision** interpolation is counterintuitive
  
  [Goadrich, et al., ILP 2004]

<table>
<thead>
<tr>
<th>TP</th>
<th>FP</th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Recall</th>
<th>Prec</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>500</td>
<td>0.50</td>
<td>0.06</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>750</td>
<td>4750</td>
<td>0.75</td>
<td>0.53</td>
<td>0.75</td>
<td>0.14</td>
</tr>
<tr>
<td>1000</td>
<td>9000</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.10</td>
</tr>
</tbody>
</table>

Example Counts  
ROC Curves  
PR Curves
Example Interpolation

Q: For each extra TP covered, how many FPs do you cover?

A: \( \frac{FP_B - FP_A}{TP_B - TP_A} \)

A dataset with 20 positive and 2000 negative examples
Example Interpolation

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>REC</th>
<th>PREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>5</td>
<td>0.25</td>
<td>0.5</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>30</td>
<td>0.5</td>
<td>0.25</td>
</tr>
</tbody>
</table>

A dataset with 20 positive and 2000 negative examples
Example Interpolation

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>REC</th>
<th>PREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>5</td>
<td>0.25</td>
<td>0.5</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>10</td>
<td>0.3</td>
<td>0.375</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>30</td>
<td>0.5</td>
<td>0.25</td>
</tr>
</tbody>
</table>

A dataset with 20 positive and 2000 negative examples
Example Interpolation

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>REC</th>
<th>PREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>5</td>
<td>0.25</td>
<td>0.5</td>
</tr>
<tr>
<td>.</td>
<td>6</td>
<td>10</td>
<td>0.3</td>
<td>0.375</td>
</tr>
<tr>
<td>.</td>
<td>7</td>
<td>15</td>
<td>0.35</td>
<td>0.318</td>
</tr>
<tr>
<td>.</td>
<td>8</td>
<td>20</td>
<td>0.4</td>
<td>0.286</td>
</tr>
<tr>
<td>.</td>
<td>9</td>
<td>25</td>
<td>0.45</td>
<td>0.265</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>30</td>
<td>0.5</td>
<td>0.25</td>
</tr>
</tbody>
</table>

A dataset with 20 positive and 2000 negative examples
Optimizing AUC

• Interest in learning algorithms that optimize Area Under the Curve (AUC)  
  [Ferri et al. 2002, Cortes and Mohri 2003, Joachims 2005,  
   Prati and Flach 2005, Yan et al. 2003, Herschtal and Raskutti 2004]

• Q: Does an algorithm that optimizes AUC-ROC also optimize AUC-PR?

• A: No. Can easily construct counterexample
Outline

• Decision Trees

• Experimental Methodology
  – Methodology overview
  – How to present results
  – Hypothesis testing
Alg 1 vs. Alg 2

- Alg 1 has accuracy 80%, Alg 2 82%
- Is this difference significant?
- Depends on how many test cases these estimates are based on
- The test we do depends on how we arrived at these estimates
The Binomial Distribution

- Distribution over the number of successes in a fixed number of independent trials (with same probability of success in each)

$$\Pr(x) = \binom{n}{x} p^x (1 - p)^{n-x}$$

![Binomial distribution graph with p=0.5, n=10]
Leave-One-Out: Sign Test

• Suppose we ran leave-one-out cross-validation on a data set of 100 cases
• Divide the cases into (1) Alg 1 won, (2) Alg 2 won, (3) Ties (both wrong or both right); Throw out the ties
• Suppose 10 ties and 50 wins for Alg 1
• Ask: Under (null) binomial(90,0.5), what is prob of 50+ or 40- successes?
What about 10-fold?

- Difficult to get significance from sign test of 10 cases
- We’re throwing out the **numbers** (accuracy estimates) for each fold, and just asking which is larger
- Use the numbers... t-test... designed to test for a difference of means
Paired Student $t$-tests

• Given
  – 10 training/test sets
  – 2 ML algorithms
  – Results of the 2 ML algo’s on the 10 test-sets

• Determine
  – Which algorithm is better on this problem?
  – Is the difference statistically significant?
Paired Student \( t \)-Tests (cont.)

**Example**

<table>
<thead>
<tr>
<th>Algorithm 1:</th>
<th>Algorithm 2:</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>80% 50 75 ... 99</td>
<td>79 49 74 ... 98</td>
<td>( +1 \ +1 \ +1 \ ... \ +1 )</td>
</tr>
</tbody>
</table>

- Algorithm 1’s mean is better, but the two std. Deviations will clearly overlap
- But algorithm1 is always better than algorithm 2
Consider random variable

\[ \delta_i = \text{Algo A's test-set}_i \text{ error} - \text{Algo B's test-set}_i \text{ error} \]

Notice we’re “factoring out” test-set difficulty by looking at relative performance. In general, one tries to explain variance in results across experiments. Here we’re saying that

\[ \text{Variance} = f(\text{Problem difficulty}) + g(\text{Algorithm strength}) \]
More on the Paired $t$-Test

Our **NULL HYPOTHESIS** is that the two ML algorithms have **equivalent average accuracies**

– That is, differences (in the scores) are due to the “random fluctuations” about the mean of zero

We compute the probability that the observed $\delta$ arose from the null hypothesis

– If this probability is **low** we **reject** the null hypo and say that the two algo’s appear different

– ‘Low’ is usually taken as **prob $\leq 0.05$**
The Null Hypothesis Graphically

1. Assume zero mean and use the sample’s variance (sample = experiment)

   \[
   \frac{1}{2} (1 - M) \text{ probability mass in each tail (i.e., } M \text{ inside)}
   \]

   Typically \( M = 0.95 \)

   Does our measured \( \delta \) lie in the regions indicated by arrows? If so, reject null hypothesis, since it is unlikely we’d get such a \( \delta \) by chance
Some Jargon: $P$–values

$P$-Value = Probability of getting one’s results or greater, given the NULL HYPOTHESIS

(We usually want $P \leq 0.05$ to be confident that a difference is statistically significant)
“Accepting” the Null Hypothesis

Note: even if the $p$–value is high, we cannot assume the null hypothesis is *true*

Eg, if we flip a coin twice and get one head, can we statistically infer the coin is *fair*?

Vs. if we flip a coin 100 times and observe 10 heads, we can statistically infer coin is *unfair* because that is very unlikely to happen with a fair coin

How would we show a coin *is* fair?
Performing the t-Test

• Easiest way: Excel:
  – `ttest(array1, array2, 2, 1)`
  – Returns p-value
Assumptions of the t-Test

• Test statistical is normally distributed
  – Reasonable if we are looking at classifier accuracy
  – Not reasonable if we are looking at AUC
    • Use Wilcoxon signed-rank test

• Independent sample of test-examples
  – Violate this with 10-fold cross-validation
Next Class

• Homework 1 is due!

• Bayesian learning
  – Bayes rule
  – MAP hypothesis

• Bayesian networks
  – Representation
  – Learning
  – Inference
Summary

• Decision trees are a very effective classifier
  – Comprehensible to humans
  – Constructive, deterministic, eager
  – Make axis-parallel cuts through feature space

• Having the right experimental methodology is crucial
  – Don’t train on the test data!!
  – Many different ways to present results
end