Classification

**Classification**

- What is classification
- Simple methods for classification
- Classification by decision tree induction
- Classification evaluation
- Classification in Large Databases

**Decision tree induction**

- Decision tree generation consists of two phases
  - **Tree construction**
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - **Tree pruning**
    - Identify and remove branches that reflect noise or outliers
  - Prefer simplest tree (Occam’s razor)
    - The simplest tree captures the most generalization and hopefully represents the most essential relationships
Dataset subsets

- **Training set** – used in model construction
- **Test set** – used in model validation
- **Pruning set** – used in model construction
  - (30% of training set)
- Train/test (70% / 30%)

PRUNING TO AVOID OVERFITTING

Avoid Overfitting in Classification

- Ideal goal of classification:
  - Find the simplest decision tree that fits the data and generalizes to unseen data
  - Intractable in general
- The generated tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise, outliers or too little training data
    - erroneous attribute values
    - erroneous classification
    - too sparse training examples
    - insufficient set of attributes
  - Result in poor accuracy for unseen samples

Overfitting and accuracy

- Typical relation between tree size and accuracy:
Pruning to avoid overfitting

- **Prepruning:** Stop growing the tree when there is not enough data to make reliable decisions or when the examples are acceptably homogenous
  - Do not split a node if this would result in the goodness measure falling below a threshold (e.g. InfoGain)
  - Difficult to choose an appropriate threshold

- **Postpruning:** Grow the full tree, then remove nodes for which there is not sufficient evidence
  - Replace a split (subtree) with a leaf if the predicted validation error is no worse than the more complex tree (use ≠ dataset)

- Prepruning easier, but **postpruning works better**
- Prepruning - hard to know when to stop

Prepruning

- Based on statistical significance test
  - Stop growing the tree when there is no statistically significant association between any attribute and the class at a particular node

- Most popular test: **chi-squared test**
- ID3 used chi-squared test in addition to information gain
  - Only statistically significant attributes were allowed to be selected by information gain procedure

Methods for postpruning

- **Reduced-error pruning**
  - Split data into training & validation sets
  - Build full decision tree from training set
  - For every non-leaf node N
    - Prune subtree rooted by N, replace with majority class. Test accuracy of pruned tree on validation set, that is, check if the pruned tree performs no worse than the original over the validation set
    - Greedily remove the subtree that results in greatest improvement in accuracy on validation set
  - **Sub-tree raising** (more complex)
    - An entire sub-tree is raised to replace another sub-tree.

chi-squared test

- Allows to compare the cells frequencies with the frequencies that would be obtained if the variables were independent

<table>
<thead>
<tr>
<th>Civil status</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Married</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Widowed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Divorced</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- In this case, compare the overall frequency of yes and no with the frequencies in the attribute branches. Is the difference is small the attribute does not add enough value to de decision
Estimating error rates

- Prune only if it reduces the estimated error
- Error on the training data is NOT a useful estimator
  
  \[ Q: \text{Why it would result in very little pruning?} \]
- Use hold-out set for pruning ("reduced-error pruning")
- C4.5's method
  - Derive confidence interval from training data
  - Use a heuristic limit, derived from this, for pruning
  - Standard Bernoulli-process-based method
  - Shaky statistical assumptions (based on training data)

How to decide if we should replace a node by a leaf?

- Use an independent test set to estimate the error (reduced error pruning) -> less data to train the tree
- Make estimates of the error based on the training data (C4.5)
  - The majority class is chosen to represent the node
  - Count the number of errors, \( \frac{e}{N} = \text{error rate} \)
  - Establish a confidence interval and use the upper limit, pessimistic estimate of the error rate.
  - Compare such estimate with the combined estimate of the error estimates for the leaves
  - If the first is smaller replace the node by a leaf.

Estimating the error rate

- Transformed value for \( f \):
  \[ f = \frac{f - \mu}{\sqrt{p(1-p)/N}} \]
  (i.e. subtract the mean and divide by the standard deviation)
- Resulting equation:
  \[ \Pr[-z \leq \frac{f - \mu}{\sqrt{p(1-p)/N}} \leq z] = c \]
- Solving for \( p \):
  \[ p = \left( f + \frac{z^2}{2N} \right) \pm \frac{f - f + \frac{z^2}{4N}}{4N} \]

\[
\frac{\sqrt{p(1-p)}}{N} = z(\alpha / 2) \cdot \sqrt{\left( \frac{Y}{N} \right) \cdot \left( 1 - \frac{Y}{N} \right)}
\]

\[
\frac{\sqrt{p(1-p)}}{N} = z(\alpha / 2) \cdot \sqrt{\left( \frac{Y}{N} \right) \cdot \left( 1 - \frac{Y}{N} \right)}
\]

So, prune!
**Subtree raising**
- Delete node
- Redistribute instances
- Slower than subtree replacement
  *(Worthwhile?)*

---

**Extracting classification rules from trees**

- Simple way:
  - Represent the knowledge in the form of IF-THEN rules
  - One rule is created for each path from the root to a leaf
  - Each attribute-value pair along a path forms a conjunction
  - The leaf node holds the class prediction
  - Rules are easier for humans to understand

---

**Extracting classification rules from trees**

*If-then rules*
- IF Outlook=Sunny \(\cap\) Humidity=Normal
  THEN PlayTennis=Yes
- IF Outlook=Overcast
  THEN PlayTennis=Yes
- IF Outlook=Rain \(\cap\) Wind=Weak
  THEN PlayTennis=Yes
- IF Outlook=Sunny \(\cap\) Humidity=High
  THEN PlayTennis=No
- IF Outlook=Rain \(\cap\) Wind=Strong
  THEN PlayTennis=No

*Is Saturday morning OK for playing tennis?*
- Outlook=Sunny, Temperature=Hot, Humidity=High, Wind=Strong
- PlayTennis = No, because Outlook=Sunny \(\cap\) Humidity=High
From trees to rules

- C4.5rules: greedily prune conditions from each rule if this reduces its estimated error
  - Can produce duplicate rules
  - Check for this at the end
- Then
  - look at each class in turn
  - consider the rules for that class
  - find a “good” subset (guided by MDL)
- Then rank the subsets to avoid conflicts
- Finally, remove rules (greedily) if this decreases error on the training data

C4.5rules: choices and options

- C4.5rules slow for large and noisy datasets
- Commercial version C5.0rules uses a different technique
  - Much faster and a bit more accurate
- C4.5 has two parameters
  - Confidence value (default 25%): lower values incur heavier pruning
  - Minimum number of instances in the two most popular branches (default 2)

Classification Accuracy

- How predictive is the model we learned?
- Error on the training data is **not** a good indicator of performance on future data
  - **Q: Why?**
    - **A:** Because new data will probably not be exactly the same as the training data!
- **Overfitting** – fitting the training data too precisely - usually leads to poor results on new data
Overfitting

How well is the model going to predict future data?

Evaluation on “LARGE” data

- If many (thousands) of examples are available, including several hundred examples from each class, then a simple evaluation is sufficient
  - Randomly split data into training and test sets
    - (usually 2/3 for train and 1/3 for test)
  
- Build a classifier using the train set and evaluate it using the test set.

Evaluation - usual procedure

Typical proportions

Problem with using “Pruning Set”: less data for “Growing Set”
**Evaluation on “small” data**

- **Cross-validation**
  - First step: data is split into $k$ subsets of equal size
  - Second step: each subset in turn is used for testing and the remainder for training

- This is called **$k$-fold cross-validation**
- Often the subsets are stratified before the cross-validation is performed
- The error estimates are averaged to yield an overall error estimate

**Tree evaluation - cross validation**

- Method for training and testing on the same set of data
- Useful when training data is limited
  1. Divide the training set into $N$ partitions (usually 10)
  2. Do $N$ experiments: each partition is used once as the validation set, and the other $N-1$ partitions are used as the training set.

The best model is chosen

**Ten Easy Pieces**

- Divide data into 10 equal pieces $P_1\ldots P_{10}$.
- Fit 10 models, each on 90% of the data.
- Each data point is treated as an out-of-sample data point by exactly one of the models.

**Ten Easy Pieces**

- Collect the scores from the red diagonal and compute the average accuracy
- Index the models by the chosen accuracy parameter and choose the best one
Evaluation of classification systems

- **Training Set**: examples with class values for learning.
- **Test Set**: examples with class values for evaluating.
- **Evaluation**: Hypotheses are used to infer classification of examples in the test set; inferred classification is compared to known classification.
- **Accuracy**: percentage of examples in the test set that are classified correctly.

Types of possible outcomes

- **Spam example**
- **Two types of errors**:
  - False positive: classify a good email as spam
  - False negative: classify a spam email as good
- **Two types of good decisions**:
  - True positive: classify a spam email as spam
  - True negative: classify a good email as good

Confusion matrix

Confusion matrix entries are counts of correct classifications and counts of errors.

Example Misclassification Costs Diagnosis of Appendicitis

- **Cost Matrix**: \( C(i,j) = \text{cost of predicting class } i \text{ when the true class is } j \)

<table>
<thead>
<tr>
<th>Predicted State of Patient</th>
<th>True State of Patient</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
</tr>
<tr>
<td>Positive</td>
<td>1</td>
</tr>
<tr>
<td>Negative</td>
<td>100</td>
</tr>
</tbody>
</table>
**Estimating Expected Misclassification Cost**

- Let $M$ be the confusion matrix for a classifier: $M(i,j)$ is the number of test examples that are predicted to be in class $i$ when their true class is $j$.

<table>
<thead>
<tr>
<th>Predicted State of Patient</th>
<th>True State of Patient</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
</tr>
<tr>
<td>Positive</td>
<td>1</td>
</tr>
<tr>
<td>Negative</td>
<td>100</td>
</tr>
</tbody>
</table>

Cost = $1 \times 40 + 1 \times 16 + 100 \times 8 + 0 \times 36$

**Evaluating Classification**

- Which goal we have:
  - minimize the number of classification errors
  - minimize the total cost of misclassifications
- In some cases FN and FP have different associated costs
  - spam vs. non-spam
  - medical diagnosis
- We can define a cost matrix in order to associate a cost with each type of result. This way we can replace the success rate by the corresponding average cost.

**Reduce the 4 numbers to two rates**

- **True Positive Rate** $= TP = (#TP)/(#P)$
- **False Positive Rate** $= FP = (#FP)/(#N)$

<table>
<thead>
<tr>
<th>True</th>
<th>Predicted</th>
<th>Predicted</th>
<th>Predicted</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>pos</td>
<td>40</td>
<td>60</td>
<td>70</td>
<td>30</td>
</tr>
<tr>
<td>neg</td>
<td>30</td>
<td>70</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classifier 1</th>
<th>Classifier 2</th>
<th>Classifier 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP = 0.4</td>
<td>TP = 0.7</td>
<td>TP = 0.6</td>
</tr>
<tr>
<td>FP = 0.3</td>
<td>FP = 0.5</td>
<td>FP = 0.2</td>
</tr>
</tbody>
</table>

**Direct Marketing Paradigm**

- Find most likely prospects to contact
- Not everybody needs to be contacted
- Number of targets is usually much smaller than number of prospects
- Typical Applications
  - retailers, catalogues, direct mail (and e-mail)
  - customer acquisition, cross-sell, attrition prediction
  - ...
Direct Marketing Evaluation

- Accuracy on the entire dataset is not the right measure
- Approach
  - develop a target model
  - score all prospects and rank them by decreasing score
  - select top P% of prospects for action
- How to decide what is the best selection?

Model-Sorted List

Use a model to assign score to each customer
Sort customers by decreasing score
Expect more targets (hits) near the top of the list

<table>
<thead>
<tr>
<th>No</th>
<th>Score</th>
<th>Target</th>
<th>Cust ID</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.97</td>
<td>Y</td>
<td>1746</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.95</td>
<td>N</td>
<td>1024</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.94</td>
<td>Y</td>
<td>2478</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.93</td>
<td>Y</td>
<td>3820</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.92</td>
<td>N</td>
<td>4897</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>0.11</td>
<td>N</td>
<td>2734</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0.06</td>
<td>N</td>
<td>2422</td>
<td></td>
</tr>
</tbody>
</table>

CPH (Cumulative Percentage Hits)

Definition:
CPH(P,M) = % of all targets in the first P% of the list scored by model M
CPH frequently called Gains

Q: What is expected value for CPH(P,Random)?
A: Expected value for CPH(P,Random) = P

CPH: Random List vs Model-ranked list

5% of random list have 5% of targets, but 5% of model ranked list have 21% of targets CPH(5%, model)=21%
Comparing models by measuring lift

Absolute number of true positives, instead of a percentage

- Instances are sorted according to their predicted probability of being a true positive:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Predicted probability</th>
<th>Actual class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.95</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>0.93</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>0.93</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>0.88</td>
<td>Yes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- In lift chart, x axis is sample size and y axis is number of true positives

Steps in Building a Lift Chart

1. First, produce a ranking of the data, using your learned model (classifier, etc):
   - Rank 1 means most likely in + class,
   - Rank n means least likely in + class
2. For each ranked data instance, label with Ground Truth label:
   - This gives a list like
     - Rank 1, +
     - Rank 2, -
     - Rank 3, +
     - Etc.
3. Count the number of true positives (TP) from Rank 1 onwards
   - Rank 1, + TP=1
   - Rank 2, - TP=1
   - Rank 3, + TP=2
   - Etc.
4. Plot # of TP against % of data in ranked order (if you have 10 data instances, then each instance is 10% of the data):
   - 10%, TP=1
   - 20%, TP=1
   - 30%, TP=2,
   - Etc.
   - This gives a lift chart.

*ROC curves

- ROC curves are similar to CPH (gains) charts
  - Stands for “receiver operating characteristic”
  - Used in signal detection to show tradeoffs between hit rate and false alarm rate over noisy channel
- Differences from gains chart
  - y axis shows percentage of true positives in sample
    - rather than absolute number
  - x axis shows percentage of false positives in sample
    - rather than sample size

To understand ROC curves go to → http://www.anaesthetist.com/mnm/stats/roc/
The plot of a ROC curve is obtained by varying the position of the cut-off point and estimating the ratio of TP and FP for each cut-off value.

For a given classifier we can instead vary the target sample size and estimate the ratio between TP and FN in the target sample.

*A sample ROC curve*
*ROC curves for two schemes

- For a small, focused sample, use method A
- For a larger one, use method B
- In between, choose between A and B with appropriate probabilities

Evaluating numeric prediction

- Same strategies: independent test set, cross-validation, significance tests, etc.
- Difference: error measures
- Actual target values: $a_1, a_2, ..., a_n$
- Predicted target values: $p_1, p_2, ..., p_n$
- Most popular measure: mean-squared error

\[
\frac{(p_1 - a_1)^2 + ... + (p_n - a_n)^2}{n}
\]

- Easy to manipulate mathematically

Other measures

- The root mean-squared error:

\[
\sqrt{\frac{(p_1 - a_1)^2 + ... + (p_n - a_n)^2}{n}}
\]

- The mean absolute error is less sensitive to outliers than the mean-squared error:

\[
\frac{|p_1 - a_1| + ... + |p_n - a_n|}{n}
\]

- Sometimes relative error values are more appropriate (e.g. 10% for an error of 50 when predicting 500)

Different Costs

- In practice, true positive and false negative errors often incur different costs
- Examples:
  - Medical diagnostic tests: does X have leukaemia?
  - Loan decisions: approve mortgage for X?
  - Web mining: will X click on this link?
  - Promotional mailing: will X buy the product?
  - ...
Cost-sensitive learning

- Most learning schemes do not perform cost-sensitive learning
  - They generate the same classifier no matter what costs are assigned to the different classes
  - Example: standard decision tree learner
- Simple methods for cost-sensitive learning:
  - Re-sampling of instances according to costs
  - Weighting of instances according to costs
- Some schemes are inherently cost-sensitive, e.g. naïve Bayes

Summary

- Classification is an extensively studied problem (mainly in statistics, machine learning & neural networks)
- Classification is probably one of the most widely used data mining techniques with a lot of extensions
- Knowing how to evaluate different classifiers is essential for the process of building a model that is adequate for a given problem

References

- Jiawei Han and Micheline Kamber, "Data Mining: Concepts and Techniques", 2000
- J. Shafer, R. Agrawal, and M. Mehta. “SPRINT: A scalable parallel classifier for data mining”. In VLDB’96, pp. 544-555,
- Robert Holt “Cost-Sensitive Classifier Evaluation” (ppt slides)
- James Guszcza, “The Basics of Model Validation”, CAS Predictive Modeling Seminar, September, 2005

Thank you !!!