For written notes on this lecture, please read chapter 3 of *The Practical Bioinformatician*. Alternatively, please read “Rule-Based Data Mining Methods for Classification Problems in Biomedical Domains”, a tutorial at *PKDD04* by Jinyan Li and Limsoon Wong, September 2004. http://www.comp.nus.edu.sg/~wongls/talks/pkdd04/

**CS2220: Introduction to Computational Biology**

**Unit 1b: Essence of Knowledge Discovery**

**Wong Limsoon**
Outline

• Overview of supervised learning
  – Decision trees

• Decision tree ensembles
  – Bagging

• Other methods
  – K-nearest neighbour
  – Support vector machines
  – Naïve Bayes
  – Hidden Markov models
Overview of supervised learning
Supervised learning

- Also called classification

- Learn from past experience, and use the learned knowledge to classify new data

- Knowledge learned by intelligent algorithms

- Examples:
  - Clinical diagnosis for patients
  - Cell type classification
Data

• Classification application involves > 1 class of data. E.g.,
  – Normal vs disease cells for a diagnosis problem

• Training data is a set of instances (samples, points, etc.) with known class labels

• Test data is a set of instances whose class labels are to be predicted
Notations

• Training data
  \{\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \ldots, \langle x_m, y_m \rangle \}
  where \( x_j \) are n-dimensional vectors and \( y_j \) are from a discrete space \( Y \).
  E.g., \( Y = \{\text{normal, disease}\} \)

• Test data
  \{\langle u_1, ? \rangle, \langle u_2, ? \rangle, \ldots, \langle u_k, ? \rangle \}
Training data: \( X \) \[ f(X) \] Class labels \( Y \)

A classifier, a mapping, a hypothesis

Test data: \( U \) \[ f(U) \] Predicted class labels
### Relational representation

**$m$ samples**

<table>
<thead>
<tr>
<th>gene_1</th>
<th>gene_2</th>
<th>gene_3</th>
<th>gene_4</th>
<th>...</th>
<th>gene_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{11}$</td>
<td>$x_{12}$</td>
<td>$x_{13}$</td>
<td>$x_{14}$</td>
<td>...</td>
<td>$x_{1n}$</td>
</tr>
<tr>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>$x_{23}$</td>
<td>$x_{24}$</td>
<td>...</td>
<td>$x_{2n}$</td>
</tr>
<tr>
<td>$x_{31}$</td>
<td>$x_{32}$</td>
<td>$x_{33}$</td>
<td>$x_{34}$</td>
<td>...</td>
<td>$x_{3n}$</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{m1}$</td>
<td>$x_{m2}$</td>
<td>$x_{m3}$</td>
<td>$x_{m4}$</td>
<td>...</td>
<td>$x_{mn}$</td>
</tr>
</tbody>
</table>

**$n$ features** (order of 1000)

**class**

- P
- N
- P
- N
Features (aka attributes)

- Categorical features
  - color = \{red, blue, green\}

- Continuous or numerical features
  - gene expression
  - age
  - blood pressure

- Discretization
# Example

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Windy</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>75</td>
<td>70</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>Sunny</td>
<td>80</td>
<td>90</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
<td>Don’t</td>
</tr>
<tr>
<td>Sunny</td>
<td>72</td>
<td>95</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>72</td>
<td>90</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>64</td>
<td>65</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>81</td>
<td>75</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Rain</td>
<td>65</td>
<td>70</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Rain</td>
<td>75</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Rain</td>
<td>68</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
<td>Play</td>
</tr>
</tbody>
</table>
Overall picture of supervised learning

Labelled Data + Algorithms

Biomedical
Financial
Government
Scientific

Decision trees
Emerging patterns
SVM
Neural networks

Classifiers (Medical Doctors)
Recap: Evaluation of a classifier

• Performance on independent blind test data
  – Blind test data properly represent real world

• K-fold cross validation
  – Given a dataset, divide it into k even parts, k-1 of them are used for training, and the rest one part treated as test data

• LOOCV, a special case of K-fold cross validation

• Accuracy, error rate, false positive rate, false negative rate, sensitivity, specificity, precision
Requirements of biomedical classification

• High accuracy, sensitivity, specificity, precision

• High comprehensibility
Importance of rule-based methods

• Systematic selection of a small number of features used for the decision making

⇒ Increase comprehensibility of the knowledge patterns

• C4.5 and CART are two commonly used rule induction algorithms---a.k.a. decision tree induction algorithms
Structure of decision trees

- If \( x_1 > a_1 \) & \( x_2 > a_2 \), then it’s A class
- C4.5, CART, two of the most widely used
- Easy interpretation, but accuracy maybe unattractive
Elegance of decision trees

Every path from root to a leaf forms a decision rule
Brief history of decision trees

CLS (Hunt et al. 1966) --- cost driven

CART (Breiman et al. 1984) --- Gini Index

ID3 (Quinlan, 1986) --- Information-driven

C4.5 (Quinlan, 1993) --- Gain ratio + Pruning ideas
### A simple dataset

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Windy</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>75</td>
<td>70</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>Sunny</td>
<td>80</td>
<td>90</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
<td>Don’t</td>
</tr>
<tr>
<td>Sunny</td>
<td>72</td>
<td>95</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>72</td>
<td>90</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>64</td>
<td>65</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>81</td>
<td>75</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Rain</td>
<td>65</td>
<td>70</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Rain</td>
<td>75</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Rain</td>
<td>68</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
<td>Play</td>
</tr>
</tbody>
</table>

9 Play samples
5 Don’t
A total of 14.
A decision tree

- Construction of a tree is equivalent to determination of root node of the tree and root nodes of its sub-trees

Exercise: What is the accuracy of this tree?
Food for thought

- What is the accuracy of this decision tree?

**Exercise #1**
An example

Source: Anthony Tung
Most discriminatory feature

- Every feature can be used to partition the training data

- If the partitions contain a pure class of training instances, then this feature is most discriminatory
Example of partitions

- **Categorical feature**
  - Number of partitions of the training data is equal to the number of values of this feature

- **Numerical feature**
  - Two partitions
<table>
<thead>
<tr>
<th>Instance #</th>
<th>Humidity</th>
<th>Windy</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>70</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>2</td>
<td>90</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>3</td>
<td>85</td>
<td>false</td>
<td>Don’t</td>
</tr>
<tr>
<td>4</td>
<td>95</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>5</td>
<td>70</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>7</td>
<td>78</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>8</td>
<td>65</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>9</td>
<td>75</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>10</td>
<td>80</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>11</td>
<td>70</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>12</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>13</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>14</td>
<td>96</td>
<td>false</td>
<td>Play</td>
</tr>
</tbody>
</table>
A categorical feature is partitioned based on its number of possible values.

<table>
<thead>
<tr>
<th>instance #</th>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Windy</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>75</td>
<td>70</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>80</td>
<td>90</td>
<td>true</td>
<td>Don't</td>
</tr>
<tr>
<td>3</td>
<td>Sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
<td>Don't</td>
</tr>
<tr>
<td>4</td>
<td>Sunny</td>
<td>72</td>
<td>95</td>
<td>true</td>
<td>Don't</td>
</tr>
<tr>
<td>5</td>
<td>Sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
<td>Don't</td>
</tr>
<tr>
<td>6</td>
<td>Overcast</td>
<td>72</td>
<td>90</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
<td>Don't</td>
</tr>
<tr>
<td>8</td>
<td>Overcast</td>
<td>64</td>
<td>65</td>
<td>false</td>
<td>Don’t</td>
</tr>
<tr>
<td>9</td>
<td>Overcast</td>
<td>81</td>
<td>75</td>
<td>false</td>
<td>Don’t</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>11</td>
<td>Rain</td>
<td>65</td>
<td>70</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>12</td>
<td>Rain</td>
<td>75</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>13</td>
<td>Rain</td>
<td>68</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
<td>Play</td>
</tr>
</tbody>
</table>

Outlook = sunny
1,2,3,4,5
P,D,D,D,D,P

Outlook = overcast
6,7,8,9
P,P,P,P

Outlook = rain
10,11,12,13,14
D, D, P, P, P
A numerical feature is generally partitioned by choosing a "cutting point".

### Total 14 training instances

<table>
<thead>
<tr>
<th>Instance #</th>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Windy</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>75</td>
<td>70</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>80</td>
<td>90</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>3</td>
<td>Sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
<td>Don’t</td>
</tr>
<tr>
<td>4</td>
<td>Sunny</td>
<td>72</td>
<td>95</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>5</td>
<td>Sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>6</td>
<td>Overcast</td>
<td>72</td>
<td>90</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>8</td>
<td>Overcast</td>
<td>64</td>
<td>65</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>9</td>
<td>Overcast</td>
<td>81</td>
<td>75</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>11</td>
<td>Rain</td>
<td>65</td>
<td>70</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>12</td>
<td>Rain</td>
<td>75</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>13</td>
<td>Rain</td>
<td>68</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
<td>Play</td>
</tr>
</tbody>
</table>

- **Temperature ≤ 70**
  - Instances: 5, 8, 11, 13, 14
  - Classes: P, P, D, P, P

- **Temperature > 70**
  - Instances: 1, 2, 3, 4, 6, 7, 9, 10, 12
  - Classes: P, D, D, D, P, P, P, D, P
Decision tree construction

• Select the “best” feature as root node of the whole tree

• Partition dataset into subsets using this feature so that the subsets are as “pure” as possible

• After partition by this feature, select the best feature (wrt the subset of training data) as root node of this sub-tree

• Recursively, until the partitions become pure or almost pure
Let’s construct a decision tree

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Windy</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>75</td>
<td>70</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>Sunny</td>
<td>80</td>
<td>90</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
<td>Don’t</td>
</tr>
<tr>
<td>Sunny</td>
<td>72</td>
<td>95</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>72</td>
<td>90</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>64</td>
<td>65</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>81</td>
<td>75</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Rain</td>
<td>65</td>
<td>70</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Rain</td>
<td>75</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Rain</td>
<td>68</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
<td>Play</td>
</tr>
</tbody>
</table>

Exercise #2

Ask the class to pick root node and construct the tree recursively with them… How good is that tree?
Three measures to evaluate which feature is best

- Gini index
- Information gain
- Information gain ratio
Gini index

\[ gini(S) = \frac{\text{diff of two arbitrary specimen in } S}{\text{mean specimen in } S} \]

\[ = \text{prob}(\text{getting two specimen of diff class in } S) \]

\[ = 1 - \text{prob}(\text{getting two specimen of same class in } S) \]

\[ = 1 - \sum_i \text{prob}(\text{getting specimen of class } i \text{ in } S)^2 \]

- Gini index is the expected value of the ratio of the diff of two arbitrary specimens to the mean value of all specimens

- Closer to 1, feature is similar to “background distribution”. Closer to 0, feature is “unexpected”
Let \( \mathcal{U} = \{C_1, \ldots, C_k\} \) be all the classes. Suppose we are currently at a node and \( D \) is the set of those samples that have been moved to this node. Let \( f \) be a feature and \( d[f] \) be the value of the feature \( f \) in a sample \( d \). Let \( S \) be a range of values that the feature \( f \) can take. Then the Gini index for \( f \) in \( D \) for the range \( S \) is defined as

\[
gini_f^D(S) = 1 - \sum_{C_i \in \mathcal{U}} \left( \frac{|\{d \in D \mid d \in C_i, \ d[f] \in S\}|}{|D|} \right)^2
\]

The purity of a split of the value range \( S \) of an attribute \( f \) by some split-point into subranges \( S_1 \) and \( S_2 \) is then defined as

\[
gini_f^D(S_1, S_2) = \sum_{S \in \{S_1, S_2\}} \frac{|\{d \in D \mid d[f] \in S\}|}{|D|} * gini_f^D(S)
\]

we choose the feature \( f \) and the split-point \( p \) that minimizes \( gini_f^D(S_1, S_2) \) over all possible alternative features and split-points.
### Gini index of “Outlook”

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Windy</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>75</td>
<td>70</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>Sunny</td>
<td>80</td>
<td>90</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
<td>Don’t</td>
</tr>
<tr>
<td>Sunny</td>
<td>72</td>
<td>95</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>72</td>
<td>90</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>64</td>
<td>65</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>Overcast</td>
<td>81</td>
<td>75</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Rain</td>
<td>65</td>
<td>70</td>
<td>true</td>
<td>Don’t</td>
</tr>
<tr>
<td>Rain</td>
<td>75</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Rain</td>
<td>68</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>Rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
<td>Play</td>
</tr>
</tbody>
</table>

\[
gini_f^P(S) = 1 - \sum_{c_i \in U} \left( \frac{|\{d \in D | d \in C_i, \ d[f] \in S\}|}{|D|} \right)^2
\]

\[
gini_f^P(S_1, S_2) = \sum_{s \in \{S_1, S_2\}} \frac{|\{d \in D | d[f] \in S\}|}{|D|} \cdot gini_f^P(S)
\]

- \( \text{gini(Sunny)} = 1 - (2/5)^2 - (3/5)^2 = 0.48 \)
- \( \text{gini(Overcast)} = 1 - (4/4)^2 - (0/5)^2 = 0 \)
- \( \text{gini(Rain)} = 1 - (3/5)^2 - (2/5)^2 = 0.48 \)
- \( \text{gini(Outlook)} = 5/14 \cdot 0.48 + 4/14 \cdot 0 + 5/14 \cdot 0.48 = 0.34 \)
Characteristics of C4.5/CART trees

- Single coverage of training data (elegance)
- Divide-and-conquer splitting strategy
- Fragmentation problem ⇒ Locally reliable but globally insignificant rules
- Miss many globally significant rules; mislead system
Example Use of Decision Tree Methods:

Proteomics Approaches to Biomarker Discovery

• In prostate and bladder cancers (Adam et al. *Proteomics*, 2001)

• In serum samples to detect breast cancer (Zhang et al. *Clinical Chemistry*, 2002)

• In serum samples to detect ovarian cancer (Petricoin et al. *Lancet*; Li & Rao, *PAKDD* 2004)
Decision tree ensembles
Motivating example

• $h_1$, $h_2$, $h_3$ are independent classifiers with accuracy = 60%
• $C_1$, $C_2$ are the only classes
• $t$ is a test instance in $C_1$
• $h(t) = \text{argmax}_{c \in \{C_1, C_2\}} \{h_j \in \{h_1, h_2, h_3\} | h_j(t) = c\}$
• Then $\text{prob}(h(t) = C_1)$

\[
= \text{prob}(h_1(t) = C_1 \& h_2(t) = C_1 \& h_3(t) = C_1) + \\
\text{prob}(h_1(t) = C_1 \& h_2(t) = C_1 \& h_3(t) = C_2) + \\
\text{prob}(h_1(t) = C_1 \& h_2(t) = C_2 \& h_3(t) = C_1) + \\
\text{prob}(h_1(t) = C_2 \& h_2(t) = C_1 \& h_3(t) = C_1)
\]

\[
= 60\% \times 60\% \times 60\% + 60\% \times 60\% \times 40\% + \\
60\% \times 40\% \times 60\% + 40\% \times 60\% \times 60\% = 64.8\%
\]
Bagging

• Proposed by Breiman (1996)

• Also called Bootstrap aggregating

• Make use of randomness injected to training data
Main ideas

Original training set

50 p + 50 n

48 p + 52 n
49 p + 51 n
...
53 p + 47 n

A base inducer such as C4.5

A committee $H$ of classifiers:

$h_1$  $h_2$  ....  $h_k$

Draw 100 samples with replacement
Decision making by bagging

Given a new test sample $T$

$$\text{bagged}(T) = \arg\max_{C_j \in U} \left| \{ h_i \in \mathcal{H} \mid h_i(T) = C_j \} \right|$$

where $U = \{ C_1, \ldots, C_r \}$

• What does this formula mean?

Exercise #3
Summary of ensemble classifiers

Bagging
Random Forest

AdaBoost.M1

Randomization Trees

CS4

Rules may not be correct when applied to training data

Rules correct

Exercise: Describe the decision tree ensemble classifiers not explained in this ppt
Other machine learning approaches
Outline

• K-nearest neighbor (kNN)
• Support vector machines (SVM)
• Naïve Bayes
• Hidden Markov models (HMM)
K-nearest neighbours
How kNN works

- Given a new case
- Find k “nearest” neighbours, i.e., k most similar points in the training data set
- Assign new case to the same class to which most of these neighbours belong

- A common “distance” measure between samples x and y is

\[ \sqrt{\sum_f (x[f] - y[f])^2} \]

where f ranges over features of the samples

Exercise: What does the formula above mean?
Illustration of kNN (k=8)

Neighborhood

5 of class
3 of class

Image credit: Zaki
Some issues

• Simple to implement
• Must compare new case against all training cases
  ⇒ May be slow during prediction

• No need to train
• But need to design distance measure properly
  ⇒ May need expert for this

• Can’t explain prediction outcome
  ⇒ Can’t provide a model of the data
Example Use of kNN

Ovarian cancer diagnosis based on SELDI proteomic data

- Use kNN to diagnose ovarian cancers using proteomic spectra
- Data set is from Petricoin et al., *Lancet* 359:572-577, 2002

**Fig. 1.** Minimum, median and maximum of percentages of correct prediction as a function of the number of top-ranked m/z ratios in 50 independent partitions into learning and validation sets.
Support vector machines
Basic idea

(a) Linear separation not possible w/o errors
(b) Better separation by nonlinear surfaces in input space
(c) Nonlinear surface corr to linear surface in feature space. Map from input to feature space by “kernel” function $\Phi$
$\Rightarrow$ “Linear learning machine” + kernel function as classifier
Linear learning machines

- Hyperplane separating the x’s and o’s points is given by \((W \cdot X) + b = 0\), with \((W \cdot X) = \sum_j W[j] \cdot X[j]\)

\[\Rightarrow \text{Decision function is } \text{llm}(X) = \text{sign}((W \cdot X) + b)\]
Linear learning machines

- Solution is a linear combination of training points $X_k$ with labels $Y_k$
  \[ W = \sum_k \alpha_k Y_k X_k, \]
  with $\alpha_k > 0$, and $Y_k = \pm 1$

  \[ \Rightarrow \text{llm}(X) = \text{sign}(\sum_k \alpha_k Y_k (X_k \cdot X) + b) \]

  “data” appears only in dot product!
Kernel function

- $\text{llm}(X) = \text{sign}(\sum_k \alpha_k * Y_k * (X_k \cdot X) + b)$

- $\text{svm}(X) = \text{sign}(\sum_k \alpha_k * Y_k * (\Phi X_k \cdot \Phi X) + b)$

$\Rightarrow \text{svm}(X) = \text{sign}(\sum_k \alpha_k * Y_k * K(X_k, X) + b)$

where $K(X_k, X) = (\Phi X_k \cdot \Phi X)$
Kernel function

• $\text{svm}(X) = \text{sign}(\sum_k \alpha_k Y_k * K(X_k, X) + b)$

$\Rightarrow K(A, B) \text{ can be computed w/o computing } \Phi$

• In fact replace it w/ lots of more “powerful” kernels besides $(A \cdot B)$. E.g.,
  - $K(A, B) = (A \cdot B)^d$
  - $K(A, B) = \exp(-||A B||^2 / (2^*\sigma))$, …
How SVM works

- $\text{svm}(X) = \text{sign}(\sum_k \alpha_k Y_k \* K(X_k, X) + b)$

- To find $\alpha_k$ is a quadratic programming problem
  
  \[
  \text{max: } \sum_k \alpha_k - 0.5 \* \sum_k \sum_h \alpha_k \alpha_h Y_k \* Y_h \* K(X_k, X_h)
  \]
  
  subject to: $\sum_k \alpha_k Y_k = 0$
  
  and for all $\alpha_k, C \geq \alpha_k \geq 0$

- To find $b$, estimate by averaging
  
  $Y_h - \sum_k \alpha_k Y_k \* K(X_h, X_k)$
  
  for all $\alpha_h \geq 0$
Example Use of SVM: Recognition of protein translation initiation sites

- Use SVM to recognize protein translation initiation sites from genomic sequences
- Raw data set is same as Liu & Wong, *JBCB* 1:139-168, 2003
Naïve Bayes
Bayes theorem

\[ P(h|d) = \frac{P(d|h) \times P(h)}{P(d)} \]

- \( P(h) \) = prior prob that hypothesis \( h \) holds
- \( P(d|h) \) = prob of observing data \( d \) given \( h \) holds
- \( P(h|d) \) = posterior prob that \( h \) holds given observed data \( d \)
Bayesian approach

• Let $H$ be all possible classes. Given a test instance w/ feature vector \( \{f_1 = v_1, \ldots, f_n = v_n\} \), the most probable classification is given by

\[
\arg\max_{h_j \in H} P(h_j | f_1 = v_1, \ldots, f_n = v_n)
\]

• Using Bayes Theorem, rewrites to

\[
\arg\max_{h_j \in H} \frac{P(f_1 = v_1, \ldots, f_n = v_n | h_j) \ast P(h_j)}{P(f_1 = v_1, \ldots, f_n = v_n)}
\]

• Since denominator is independent of $h_j$, this simplifies to

\[
\arg\max_{h_j \in H} P(f_1 = v_1, \ldots, f_n = v_n | h_j) \ast P(h_j)
\]
Naïve Bayes

But estimating \( P(f_1=v_1, \ldots, f_n=v_n|h_j) \) accurately may not be feasible unless training data set is large

“Solved” by assuming \( f_1, \ldots, f_n \) are conditionally independent of each other

Then

\[
\arg\max_{h_j \in H} P(f_1 = v_1, \ldots, f_n = v_n|h_j) * P(h_j)
\]

\[
= \arg\max_{h_j \in H} \prod_{i} P(f_i = v_i|h_j) * P(h_j)
\]

where \( P(h_j) \) and \( P(f_i=v_i|h_j) \) can often be estimated reliably from typical training data set

Exercise: How do you estimate \( P(h_j) \) and \( P(f_j=v_j|h_j) \)?
Abstractly, the probability model for a classifier is a conditional model
\[ p(C|F_1, \ldots, F_n) \]
over a dependent class variable \( C \) with a small number of outcomes or classes, conditional on several feature variables \( F_1 \) through \( F_n \). The problem is that if the number of features \( n \) is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.

Using Bayes' theorem, we write
\[ p(C|F_1, \ldots, F_n) = \frac{p(C) \ p(F_1, \ldots, F_n|C)}{p(F_1, \ldots, F_n)}. \]

In practice we are only interested in the numerator of that fraction, since the denominator does not depend on \( C \) and the values of the features \( F_i \) are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model
\[ p(C, F_1, \ldots, F_n) \]
which can be rewritten as follows, using repeated applications of the definition of conditional probability:
\[
p(C, F_1, \ldots, F_n) \\
= p(C) \ p(F_1, \ldots, F_n|C) \\
= p(C) \ p(F_1|C) \ p(F_2, \ldots, F_n|C, F_1) \\
= p(C) \ p(F_1|C) \ p(F_2|C, F_1) \ p(F_3, \ldots, F_n|C, F_1, F_2) \\
= p(C) \ p(F_1|C) \ p(F_2|C, F_1) \ p(F_3|C, F_1, F_2) \ p(F_4, \ldots, F_n|C, F_1, F_2, F_3) \\
\]
and so forth. Now the "naive" conditional independence assumptions come into play: assume that each feature \( F_i \) is conditionally independent of every other feature \( F_j \) for \( j \neq i \). This means that
\[ p(F_i|C, F_j) = p(F_i|C) \]
and so the joint model can be expressed as
\[
p(C, F_1, \ldots, F_n) = p(C) \ p(F_1|C) \ p(F_2|C) \ p(F_3|C) \ldots \\
= p(C) \prod_{i=1}^{n} p(F_i|C). \]

Independence vs Conditional independence

• Independence: \( P(A, B) = P(A) \times P(B) \)
• Conditional Independence: \( P(A, B|C) = P(A|C) \times P(B|C) \)
• Indep does not imply conditional indep
  – Consider tossing a fair coin twice
    • A is event of getting head in 1st toss
    • B is event of getting head in 2nd toss
    • C is event of getting exactly one head
  – Then \( A = \{\text{HT, HH}\} \), \( B = \{\text{HH, TH}\} \) and \( C = \{\text{HT, TH}\} \)
  – \( P(A, B|C) = P(\{\text{HH}\}|C) = 0 \)
  – \( P(A|C) = P(A,C)/P(C) = P(\{\text{HT}\})/P(C) = (1/4)/(1/2) = 1/2 \)
  – Similarly, \( P(B|C) = 1/2 \)

Source: Choi Kwok Pui
Example Use of Bayesian: Design of screens for macromolecular crystallization


- Xtallization of proteins requires search of expt settings to find right conditions for diffraction-quality xtals

- BMCD is a db of known xttallization conditions

- Use Bayes to determine prob of success of a set of expt conditions based on BMCD
Hidden Markov models
What is a HMM

- HMM is a stochastic generative model for seqs
  - Defined by model parameters
    - finite set of states $S$
    - finite alphabet $A$
    - transition prob matrix $T$
    - emission prob matrix $E$
  - Move from state to state as per $T$
    while emitting symbols as per $E$
Order of a HMM

• In $n$th order HMM, $T$ & $E$ depend on all $n$ previous states

• E.g., for 1st order HMM, given emissions $X = x_1, x_2, ..., \text{ & states } S = s_1, s_2, ..., \text{ the prob of this seq is}$

$$\text{Prob}(X, S) = \prod_i \text{Prob}(x_i|s_i) = \prod_i E(x_i|s_i) \ast T(s_{i-1}, s_i)$$
Using HMM

- Given the model parameters, compute the probability of a particular output sequence. Solved by the **forward algorithm**

- Given the model parameters, find the most likely sequence of (hidden) states which could have generated a given output sequence. Solved by the **Viterbi algorithm**

- Given an output sequence, find the most likely set of state transition and output probabilities. Solved by the **Baum-Welch algorithm**

**Exercise:** Describe these algorithms
Example: Dishonest casino

- **Casino has two dices:**
  - Fair dice
    - \( P(i) = \frac{1}{6}, \ i = 1..6 \)
  - Loaded dice
    - \( P(i) = \frac{1}{10}, \ i = 1..5 \)
    - \( P(i) = \frac{1}{2}, \ i = 6 \)

- Casino switches betw fair & loaded die with prob 1/2. Initially, dice is always fair

- **Game:**
  - You bet $1
  - You roll
  - Casino rolls
  - Highest number wins $2

- **Question:** Suppose we played 2 games, and the sequence of rolls was 1, 6, 2, 6. Were we likely to have been cheated?
"Visualization" of dishonest casino

**Emission Matrix**

<table>
<thead>
<tr>
<th>Event</th>
<th>Fair</th>
<th>Loaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>E(1</td>
<td>Fair) = 1/6</td>
<td>E(1</td>
</tr>
<tr>
<td>E(2</td>
<td>Fair) = 1/6</td>
<td>E(2</td>
</tr>
<tr>
<td>E(3</td>
<td>Fair) = 1/6</td>
<td>E(3</td>
</tr>
<tr>
<td>E(4</td>
<td>Fair) = 1/6</td>
<td>E(4</td>
</tr>
<tr>
<td>E(5</td>
<td>Fair) = 1/6</td>
<td>E(5</td>
</tr>
<tr>
<td>E(6</td>
<td>Fair) = 1/6</td>
<td>E(6</td>
</tr>
</tbody>
</table>

**Transition Matrix**

<table>
<thead>
<tr>
<th>Transition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T(Loaded,Loaded) = 1/2</td>
<td></td>
</tr>
<tr>
<td>T(Loaded,Fair) = 1/2</td>
<td></td>
</tr>
<tr>
<td>T(Fair,Fair) = 1/2</td>
<td></td>
</tr>
<tr>
<td>T(Fair,Loaded) = 1/2</td>
<td></td>
</tr>
<tr>
<td>T(?,Fair) = 1.0</td>
<td></td>
</tr>
<tr>
<td>T(?,Loaded) = 0.0</td>
<td></td>
</tr>
</tbody>
</table>
1, 6, 2, 6?
We were probably cheated...

\[
\begin{align*}
\text{Prob}(X, S = \text{Fair, Fair, Fair, Fair}) &= \ E(1|\text{Fair}) \times T(?, \text{Fair}) \times \\
&\quad E(6|\text{Fair}) \times T(\text{Fair, Fair}) \times \\
&\quad E(2|\text{Fair}) \times T(\text{Fair, Fair}) \times \\
&\quad E(6|\text{Fair}) \times T(\text{Fair, Fair}) \\
&= \frac{1}{6} \times \frac{1}{6} \times \frac{1}{2} \times \frac{1}{6} \times \frac{1}{6} \times \frac{1}{2} \times \frac{1}{6} \times \frac{1}{2} \\
&= 9.6451 \times 10^{-5}
\end{align*}
\]

\[
\begin{align*}
\text{Prob}(X, S = \text{Fair, Loaded, Fair, Loaded}) &= \ E(1|\text{Fair}) \times T(?, \text{Fair}) \times \\
&\quad E(6|\text{Loaded}) \times T(\text{Fair, Loaded}) \times \\
&\quad E(2|\text{Fair}) \times T(\text{Loaded, Fair}) \times \\
&\quad E(6|\text{Loaded}) \times T(\text{Fair, Loaded}) \\
&= \frac{1}{6} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{6} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{6} \times \frac{1}{2} \\
&= 8.6806 \times 10^{-4}
\end{align*}
\]
Example Use of HMM: Protein families modelling

- Baldi et al., *PNAS* 91:1059-1063, 1994
- HMM is used to model families of biological sequences, such as kinases, globins, & immunoglobulins

- Bateman et al., *NAR* 32:D138-D141, 2004
- HMM is used to model 6190 families of protein domains in Pfam
Concluding remarks...
What have we learned?

• Decision trees

• Decision trees ensembles
  – Bagging

• Other methods
  – K-nearest neighbour
  – Support vector machines
  – Naïve Bayes
  – Hidden Markov models
Any question?

• Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

Exercise: Download a copy of WEKA. What are the names of classifiers in WEKA that correspond to C4.5 and SVM?
Acknowledgements

• Most of the slides used in this ppt came from a tutorial that WLS gave with Jinyan Li at the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa, Italy, 20-24 September 2004

• The dishonest casino example came from slides inherited from Ken Sung

• The “indep vs conditional indep” example came from Choi Kwok Pui
References

• L. Breiman, Bagging predictors, Machine Learning, 24:123-140, 1996
• L. Breiman, Random forests, Machine Learning, 45:5-32, 2001
• J. R. Quinlan, Induction of decision trees, Machine Learning, 1:81-106, 1986
• Jinyan Li et al., Data Mining Techniques for the Practical Bioinformatician, The Practical Bioinformatician, Chapter 3, pages 35-70, WSPC, 2004
References

• T. G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization, Machine Learning, 40:139-157, 2000
• Naïve Bayesian Classification, *Wikipedia*,
  http://en.wikipedia.org/wiki/Naive_Bayesian_classification
• Hidden Markov Model, Wikipedia,