CS570 Introduction to Data Mining

Classification and Prediction

Partial slide credits:
Han and Kamber
Tan, Steinbach, Kumar
Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - kNN classification
  - Support Vector Machines (SVM)
  - Neural Networks
- Regression
- Evaluation and measures
- Ensemble methods
## Motivating Example – Fruit Identification

<table>
<thead>
<tr>
<th>Skin</th>
<th>Color</th>
<th>Size</th>
<th>Flesh</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hairy</td>
<td>Brown</td>
<td>Large</td>
<td>Hard</td>
<td>safe</td>
</tr>
<tr>
<td>Hairy</td>
<td>Green</td>
<td>Large</td>
<td>Hard</td>
<td>Safe</td>
</tr>
<tr>
<td>Smooth</td>
<td>Red</td>
<td>Large</td>
<td>Soft</td>
<td>Dangerous</td>
</tr>
<tr>
<td>Hairy</td>
<td>Green</td>
<td>Large</td>
<td>Soft</td>
<td>Safe</td>
</tr>
<tr>
<td>Smooth</td>
<td>Red</td>
<td>Small</td>
<td>Hard</td>
<td>Dangerous</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Classification vs. Prediction

- **Classification**
  - predicts categorical class labels
  - constructs a model based on the training set and uses it in classifying new data

- **Prediction (Regression)**
  - models continuous-valued functions, i.e., predicts unknown or missing values

- **Typical applications**
  - Credit approval
  - Target marketing
  - Medical diagnosis
  - Fraud detection
### Example – Credit Approval

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Income</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clark</td>
<td>35</td>
<td>High</td>
<td>Excellent</td>
</tr>
<tr>
<td>Milton</td>
<td>38</td>
<td>High</td>
<td>Excellent</td>
</tr>
<tr>
<td>Neo</td>
<td>25</td>
<td>Medium</td>
<td>Fair</td>
</tr>
</tbody>
</table>

**Classification rule:**
- If age = “31...40” and income = high then credit_rating = excellent

**Future customers**
- Paul: age = 35, income = high ⇒ excellent credit rating
- John: age = 20, income = medium ⇒ fair credit rating
Classification—A Two-Step Process

- **Model construction**: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae

- **Model usage**: for classifying future or unknown objects
  - **Estimate accuracy** of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known
Process (1): Model Construction

- Training Data
- Classification Algorithms
- Classifier (Model)
Process (2): Using the Model in Prediction
Supervised vs. Unsupervised Learning

- **Supervised learning (classification)**
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set

- **Unsupervised learning (clustering)**
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data
Issues: Evaluating Classification Methods

- Accuracy
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, decision tree size or compactness of classification rules
Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree
  - Bayesian classification
  - kNN classification
  - Support Vector Machines (SVM)
  - Others
- Evaluation and measures
- Ensemble methods
### Training Dataset

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>
A Decision Tree for “buys_computer”

- age?
  - <=30: student? (no)
  - 31..40: yes
  - >40: credit rating?
    - excellent: yes
    - fair: no

Data Mining: Concepts and Techniques
Algorithm for Decision Tree Induction

- ID3 (Iterative Dichotomiser), C4.5, by Quinlan
- CART (Classification and Regression Trees)
- Basic algorithm (a greedy algorithm) - tree is constructed with top-down recursive partitioning
  - At start, all the training examples are at the root
  - A test attribute is selected that “best” separate the data into partitions
  - Samples are partitioned recursively based on selected attributes
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left
Attribute Selection Measures

- Idea: select attribute that partition samples into homogeneous groups
- Measures
  - Information gain (ID3)
  - Gain ratio (C4.5)
  - Gini index (CART)
Attribute Selection Measure: Information Gain (ID3)

- Select the attribute with the highest information gain
- Let $p_i$ be the probability that an arbitrary tuple in $D$ belongs to class $C_i$, estimated by $|C_{i,D}|/|D|$
- Information (entropy) needed to classify a tuple in $D$ (before split):

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using $A$ to split $D$ into $v$ partitions) to classify $D$:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gain – difference between before and after splitting on attribute $A$

$$Gain(A) = Info(D) - Info_A(D)$$
Example: Information Gain

- Class P: buys_computer = “yes”,
- Class N: buys_computer = “no”

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31...40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31...40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31...40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31...40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
\text{age} & \ p_i & \ n_i & l(p_i, n_i) \\
\hline
< = 30 & 2 & 3 & 0.971 \\
31...40 & 4 & 0 & 0 \\
40 & 3 & 2 & 0.971 \\
\hline
\end{array}
\]

\[
\text{Gain}\(\text{age}\) = \text{Info}\(D\) - \text{Info}_{\text{age}}\(D\) = 0.246
\]

\[
\text{Gain}\(\text{income}\) = 0.029
\]

\[
\text{Gain}\(\text{student}\) = 0.151
\]

\[
\text{Gain}\(\text{credit\_rating}\) = 0.048
\]

\[
\text{Info}\(D\) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940
\]

\[
\text{Info}_{\text{age}}\(D\) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694
\]
Information-Gain for Continuous-Value Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - \((a_i + a_{i+1})/2\) is the midpoint between the values of \(a_i\) and \(a_{i+1}\)
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying \(A \leq\) split-point, and
  - D2 is the set of tuples in D satisfying \(A >\) split-point
Attribute Selection Measure: Gain Ratio (C4.5)

- Information gain measure is biased towards attributes with a large # of values (# of splits)
- C4.5 uses gain ratio to overcome the problem (normalization to information gain)

\[
SplitInfo_A(D) = - \sum_{j=1}^{v} \left| \frac{D_j}{D} \right| \times \log_2 \left( \frac{\left| D_j \right|}{\left| D \right|} \right)
\]

- GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex. \( SplitInfo_A(D) = - \frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) - \frac{6}{14} \times \log_2 \left( \frac{6}{14} \right) - \frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) = 0.926 \)
  
  gain_ratio(income) = 0.029/0.926 = 0.031
- The attribute with the maximum gain ratio is selected as the splitting attribute
Attribute Selection Measure: Gini index (CART)

- If a data set $D$ contains examples from $n$ classes, gini index, $gini(D)$ is defined as
  \[
gini(D) = 1 - \sum_{j=1}^{n} p_j^2
\]
  where $p_j$ is the relative frequency of class $j$ in $D$

- If a data set $D$ is split on $A$ into two subsets $D_1$ and $D_2$, the gini index $gini(D)$ is defined as
  \[
gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)
\]

- Reduction in Impurity:
  \[
\Delta gini(A) = gini(D) - gini_A(D)
\]

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node
Example: Gini index

Ex. D has 9 tuples in buys_computer = “yes” and 5 in “no”

\[ gini(D) = 1 - \left( \frac{9}{14} \right)^2 - \left( \frac{5}{14} \right)^2 = 0.459 \]

Suppose the attribute income partitions D into 10 in \( D_1 \): \{low, medium\} and 4 in \( D_2 \)

\[ gini_{\text{income}\in\{\text{low,medium}\}}(D) = \left( \frac{10}{14} \right)Gini(D_1) + \left( \frac{4}{14} \right)Gini(D_1) \]
\[ = \frac{10}{14} \left( 1 - \left( \frac{6}{10} \right)^2 - \left( \frac{4}{10} \right)^2 \right) + \frac{4}{14} \left( 1 - \left( \frac{1}{4} \right)^2 - \left( \frac{3}{4} \right)^2 \right) \]
\[ = 0.450 \]
\[ = Gini_{\text{income}\in\{\text{high}\}}(D) \]

but \( gini_{\text{medium,high}} \) is 0.30 and thus the best since it is the lowest
Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - Information gain:
    - biased towards multivalued attributes
  - Gain ratio:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - Gini index:
    - biased to multivalued attributes
    - tends to favor tests that result in equal-sized partitions and purity in both partitions
Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on $\chi^2$ test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistics: has a close approximation to $\chi^2$ distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others
Overfitting:

- An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies and noises
Alaska Robotics

each choice
a limb

each decision
a branch

Decision Tree

no turning back

SNAP

this is your life.
Tree Pruning

Two approaches to avoid overfitting

- Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
  - Difficult to choose an appropriate threshold
- Postpruning: Remove branches from a “fully grown” tree
  - Use a set of data different from the training data to decide which is the “best pruned tree”
  - Occam's razor: prefers smaller decision trees (simpler theories) over larger ones
Enhancements to Basic Decision Tree Induction

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication
Scalable Decision Tree Induction Methods

- **SLIQ** (EDBT’96 — Mehta et al.)
  - Builds an index for each attribute and only class list and the current attribute list reside in memory

- **SPRINT** (VLDB’96 — J. Shafer et al.)
  - Constructs an attribute list data structure

- **PUBLIC** (VLDB’98 — Rastogi & Shim)
  - Integrates tree splitting and tree pruning: stop growing the tree earlier

- **RainForest** (VLDB’98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)

- **BOAT** (PODS’99 — Gehrke, Ganti, Ramakrishnan & Loh)
  - Uses bootstrapping to create several small samples
RainForest

- Separates the scalability aspects from the criteria that determine the quality of the tree
- Builds an AVC-list: **AVC** (Attribute, Value, Class_label)
- **AVC-set** (of an attribute $X$)
  - Projection of training dataset onto the attribute $X$ and class label where counts of individual class label are aggregated
- **AVC-group** (of a node $n$)
  - Set of AVC-sets of all predictor attributes at the node $n$
### Training Examples

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31...40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31...40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31...40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31...40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

### AVC-set on \( \text{Age} \)

<table>
<thead>
<tr>
<th>Age</th>
<th>Buy_Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>3</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>2</td>
</tr>
<tr>
<td>31...40</td>
<td>4</td>
</tr>
<tr>
<td>31...40</td>
<td>0</td>
</tr>
<tr>
<td>&gt;40</td>
<td>3</td>
</tr>
<tr>
<td>&gt;40</td>
<td>2</td>
</tr>
</tbody>
</table>

### AVC-set on \( \text{Income} \)

<table>
<thead>
<tr>
<th>Income</th>
<th>Buy_Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>2</td>
</tr>
<tr>
<td>no</td>
<td>2</td>
</tr>
<tr>
<td>high</td>
<td>4</td>
</tr>
<tr>
<td>medium</td>
<td>2</td>
</tr>
<tr>
<td>low</td>
<td>1</td>
</tr>
</tbody>
</table>

### AVC-set on \( \text{Student} \)

<table>
<thead>
<tr>
<th>Student</th>
<th>Buy_Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>6</td>
</tr>
<tr>
<td>no</td>
<td>4</td>
</tr>
</tbody>
</table>

### AVC-set on \( \text{Credit rating} \)

<table>
<thead>
<tr>
<th>Credit Rating</th>
<th>Buy_Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>6</td>
</tr>
<tr>
<td>fair</td>
<td>2</td>
</tr>
<tr>
<td>excellent</td>
<td>3</td>
</tr>
<tr>
<td>no</td>
<td>4</td>
</tr>
</tbody>
</table>
BOAT (Bootstrapped Optimistic Algorithm for Tree Construction)

- Use a statistical technique called *bootstrapping* to create several smaller samples (subsets), each fits in memory.
- Each subset is used to create a tree, resulting in several trees.
- These trees are examined and used to construct a new tree $T'$.
  - It turns out that $T'$ is very close to the tree that would be generated using the whole data set together.
- Adv: requires only two scans of DB, an incremental alg.
Decision Tree: Comments

- Relatively faster learning speed (than other classification methods)
-Convertible to simple and easy to understand classification rules
-Comparable classification accuracy with other methods
Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - kNN classification
  - Support Vector Machines (SVM)
  - Others
- Evaluation and measures
- Ensemble methods
Bayesian Classification

- A statistical classifier: performs *probabilistic prediction*, i.e., predicts class membership probabilities
- **Foundation**: Based on Bayes’ Theorem.
- Naïve Bayesian
  - Independence assumption
- Bayesian network
  - Concept
  - Using Bayesian network
  - Training/learning Bayesian network
Bayes’ theorem

Bayes' theorem/rule/law relates the conditional and marginal probabilities of stochastic events

- $P(H)$ is the prior probability of $H$.
- $P(H|X)$ is the conditional probability (posteriori probability) of $H$ given $X$.
- $P(X|H)$ is the conditional probability of $X$ given $H$.
- $P(X)$ is the prior probability of $X$

\[
P(H | X) = \frac{P(X | H)P(H)}{P(X)}
\]

Cookie example:

- Bowl A: 10 chocolate + 30 plain; Bowl B: 20 chocolate + 20 plain
- Pick a bowl, and then pick a cookie
- If it’s a plain cookie, what’s the probability the cookie is picked out of bowl A?
Naïve Bayesian Classifier

- Naïve Bayesian / idiot Bayesian / simple Bayesian
- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector \( \mathbf{X} = (x_1, x_2, \ldots, x_n) \)
- Suppose there are \( m \) classes \( C_1, C_2, \ldots, C_m \).
- Classification is to derive the maximum posteriori, i.e., the maximal \( P(C_i|\mathbf{X}) \)

\[
P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}
\]

- Since \( P(\mathbf{X}) \) is constant for all classes, maximal \( P(\mathbf{X}|C_i)P(C_i) \)
Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

\[
P(X | C_i) = \prod_{k=1}^{n} P(x_k | C_i) = P(x_1 | C_i) \times P(x_2 | C_i) \times \ldots \times P(x_n | C_i)
\]

- If \( A_k \) is categorical, \( P(x_k | C_i) \) is the # of tuples in \( C_i \) having value \( x_k \) for \( A_k \) divided by \( |C_i, D| \) (# of tuples of \( C_i \) in \( D \))

- If \( A_k \) is continous-valued, \( P(x_k | C_i) \) is usually computed based on Gaussian distribution with a mean \( \mu \) and standard deviation \( \sigma \)

\[
g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi \sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
\]

and \( P(x_k | C_i) \) is

\[
P(X | C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})
\]
Naïve Bayesian Classifier: Example

Class:
C1:buys_computer = ‘yes’
C2:buys_computer = ‘no’

Data sample
X = (age <=30, Income = medium, Student = yes Credit_rating = Fair)
Naïve Bayesian Classifier: Example

- **P(C_i):**
  - P(buys_computer = “yes”) = 9/14 = 0.643
  - P(buys_computer = “no”) = 5/14 = 0.357

- Compute P(X|C_i) for each class
  - P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222
  - P(age = “<=30” | buys_computer = “no”) = 3/5 = 0.6
  - P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
  - P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4
  - P(student = “yes” | buys_computer = “yes”) = 6/9 = 0.667
  - P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2
  - P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667
  - P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

- X = (age <= 30, income = medium, student = yes, credit_rating = fair)

  \[
P(X|C_i) : \begin{align*}
P(X|buys\_computer = \text{“yes”}) &= 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044 \\
P(X|buys\_computer = \text{“no”}) &= 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019
\end{align*}
\]

  \[
P(X|C_i) \times P(C_i) : \begin{align*}
P(X|buys\_computer = \text{“yes”}) \times P(buys\_computer = \text{“yes”}) &= 0.028 \\
P(X|buys\_computer = \text{“no”}) \times P(buys\_computer = \text{“no”}) &= 0.007
\end{align*}
\]

Therefore, X belongs to class (“buys\_computer = yes”)
Naïve Bayesian Classifier: Comments

- Advantages
  - Fast to train and use
  - Can be highly effective in most of the cases

- Disadvantages
  - Based on a false assumption: class conditional independence - practically, dependencies exist among variables

- How to deal with dependencies?
  - Bayesian Belief Networks
Bayesian Belief Networks – Motivating Example

- Symptoms: difficult to breath
- Patient profile: smoking? age?
- Family history?
- XRay?

Lung Cancer?
Bayesian Belief Networks

- Bayesian belief networks (belief networks, Bayesian networks, probabilistic networks) is a graphical model that represents a set of variables and their probabilistic independencies.
- One of the most significant contributions in AI.
- Trained Bayesian networks can be used for classification and reasoning.
- Many applications: spam filtering, speech recognition, diagnostic systems.
Bayesian Network: Definition

A Bayesian network is made up of:

1. A Directed Acyclic Graph

- A
  - B
  - C
  - D

2. A conditional probability table for each node in the graph

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>P(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>false</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td>0.4</td>
</tr>
</tbody>
</table>

|   | A   | B   | P(B | A)  |
|---|-----|-----|--------|
|   | false | false | 0.01   |
|   | false | true  | 0.99   |
|   | true  | false | 0.7    |
|   | true  | true  | 0.3    |

|   | B   | D   | P(D | B)  |
|---|-----|-----|--------|
|   | false | false | 0.02   |
|   | false | true  | 0.98   |
|   | true  | false | 0.05   |
|   | true  | true  | 0.95   |

|   | B   | C   | P(C | B)  |
|---|-----|-----|--------|
|   | false | false | 0.4    |
|   | false | true  | 0.6    |
|   | true  | false | 0.9    |
|   | true  | true  | 0.1    |
Each node in the graph is a random variable.

A node $X$ is a parent of another node $Y$ if there is an arrow from node $X$ to node $Y$.

E.g. $A$ is a parent of $B$.

Informally, an arrow from node $X$ to node $Y$ means $X$ has a direct influence on $Y$. 
Conditional Probability Table

Each node $X_i$ has a conditional probability distribution $P(X_i | \text{Parents}(X_i))$ that quantifies the effect of the parents on the node.

For a Boolean variable with $k$ Boolean parents, how many probabilities need to be stored?
Bayesian Networks: Important Properties

1. Encodes the conditional independence relationships between the variables in the graph structure
2. Is a compact representation of the joint probability distribution over the variables
Conditional Independence

The Markov condition: given its parents \((P_1, P_2)\), a node \((X)\) is conditionally independent of its non-descendants \((ND_1, ND_2)\).
Joint Probability Distribution

Due to the Markov condition, we can compute the joint probability distribution over all the variables $X_1, \ldots, X_n$ in the Bayesian net using the formula:

$$P(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i \mid \text{Parents}(X_i))$$

Example:
$P(\text{A} = \text{true}, \text{B} = \text{true}, \text{C} = \text{true}, \text{D} = \text{true})$

$= P(\text{A} = \text{true}) \times P(\text{B} = \text{true} \mid \text{A} = \text{true}) \times$

$P(\text{C} = \text{true} \mid \text{B} = \text{true}) \times P(\text{D} = \text{true} \mid \text{B} = \text{true})$

$= (0.4) \times (0.3) \times (0.1) \times (0.95)$
Bayesian Networks: Example

The conditional probability table (CPT) for variable LungCancer:

<table>
<thead>
<tr>
<th></th>
<th>(FH, S)</th>
<th>(FH, ~S)</th>
<th>(~FH, S)</th>
<th>(~FH, ~S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LC</td>
<td>0.8</td>
<td>0.5</td>
<td>0.7</td>
<td>0.1</td>
</tr>
<tr>
<td>~LC</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Using the Bayesian Network:
P(LungCancer | Smoker, PXRay, Dyspnea)?
Using Bayesian Network for Inference

- Using a Bayesian network to compute probabilities is called inference.
- General form: $\Pr(X \mid E)$

  $E = \text{The evidence variable(s)}$

  $X = \text{The query variable(s)}$

- Exact inference is feasible in small to medium-sized networks.
- Exact inference in large networks takes a very long time.
  - Approximate inference techniques which are much faster and give pretty good results.
Inference Example

Joint probability:
\[ P(C, S, R, W) = P(C) \times P(S|C) \times P(R|C) \times P(W|S,R) \]

Suppose the grass is wet, which is more likely?

\[
\begin{align*}
\Pr(S = 1|W = 1) &= \frac{\Pr(S = 1, W = 1)}{\Pr(W = 1)} = \frac{\sum_{c,r} \Pr(C = c, S = 1, R = r, W = 1)}{\Pr(W = 1)} = 0.2781/0.6471 = 0.430 \\
\Pr(R = 1|W = 1) &= \frac{\Pr(R = 1, W = 1)}{\Pr(W = 1)} = \frac{\sum_{c,s} \Pr(C = c, S = s, R = 1, W = 1)}{\Pr(W = 1)} = 0.4581/0.6471 = 0.708
\end{align*}
\]

where

\[ \Pr(W = 1) = \sum_{c,r,s} \Pr(C = c, S = s, R = r, W = 1) = 0.6471 \]
Training Bayesian Networks

- Several scenarios:
  - Given both the network structure and all variables observable: *learn only the CPTs*
  - Network structure known, some hidden variables: *gradient descent* (greedy hill-climbing) method, analogous to neural network learning
  - Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
  - Unknown structure, all hidden variables: No good algorithms known for this purpose
  - Ref. D. Heckerman: Bayesian networks for data mining
Related Graphical Models

- Bayesian networks (directed graphical model)
- Markov networks (undirected graphical model)
  - Conditional random field
- Applications:
  - Sequential data
    - Natural language text
    - Protein sequences
Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - kNN classification
  - Support Vector Machines (SVM)
  - Neural Networks
- Regression
- Evaluation and measures
- Ensemble methods