Chapter 3: Supervised Learning

Dr. Mehmet S. Aktaş

Acknowledgement: Thanks to Dr. Bing Liu for teaching materials.
Road Map

- Basic concepts
- Decision tree induction
- Classification using association rules
- Naïve Bayesian classification
- K-nearest neighbor
- Summary
An example application

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- A decision is needed: whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.
- **Problem:** to predict *high-risk patients* and discriminate them from *low-risk patients*.
Another application

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.

- **Problem**: to decide whether an application should be approved, or to classify applications into two categories, approved and not approved.
Machine learning and our focus

- Like human learning from past experiences.
- A computer does not have “experiences”.
- **A computer system learns from data**, which represent some “past experiences” of an application domain.
- **Our focus**: learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- The task is commonly called: **Supervised learning, classification**, or inductive learning.
The data and the goal

- **Data**: A set of data records (also called examples, instances or cases) described by
  - *k attributes*: $A_1, A_2, \ldots A_k$.
  - *a class*: Each example is labelled with a pre-defined class.

- **Goal**: To learn a **classification model** from the data that can be used to predict the classes of new (future, or test) cases/instances.
An example: data (loan application)

<table>
<thead>
<tr>
<th>ID</th>
<th>Age</th>
<th>Has_Job</th>
<th>Own_House</th>
<th>Credit_Rating</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>young</td>
<td>false</td>
<td>false</td>
<td>fair</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>young</td>
<td>false</td>
<td>false</td>
<td>good</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>young</td>
<td>true</td>
<td>false</td>
<td>good</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>young</td>
<td>true</td>
<td>true</td>
<td>fair</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>young</td>
<td>false</td>
<td>false</td>
<td>fair</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>middle</td>
<td>false</td>
<td>false</td>
<td>fair</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>middle</td>
<td>false</td>
<td>false</td>
<td>good</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>middle</td>
<td>true</td>
<td>true</td>
<td>good</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>middle</td>
<td>false</td>
<td>true</td>
<td>excellent</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>middle</td>
<td>false</td>
<td>true</td>
<td>excellent</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>old</td>
<td>false</td>
<td>true</td>
<td>excellent</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>old</td>
<td>false</td>
<td>true</td>
<td>good</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>old</td>
<td>true</td>
<td>false</td>
<td>good</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>old</td>
<td>true</td>
<td>false</td>
<td>excellent</td>
<td>Yes</td>
</tr>
<tr>
<td>15</td>
<td>old</td>
<td>false</td>
<td>false</td>
<td>fair</td>
<td>No</td>
</tr>
</tbody>
</table>
An example: the learning task

- Learn a classification model from the data
- Use the model to classify future loan applications into
  - Yes (approved) and
  - No (not approved)
- What is the class for following case/instance?

<table>
<thead>
<tr>
<th>Age</th>
<th>Has_Job</th>
<th>Own_house</th>
<th>Credit-Rating</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>young</td>
<td>false</td>
<td>false</td>
<td>good</td>
<td>?</td>
</tr>
</tbody>
</table>
Supervised vs. unsupervised Learning

- **Supervised learning**: classification is seen as supervised learning from examples.
  - **Supervision**: The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a “teacher” gives the classes (supervision).
  - Test data are classified into these classes too.

- **Unsupervised learning (clustering)**
  - Class labels of the data are unknown
  - Given a set of data, the task is to establish the existence of classes or clusters in the data
Supervised learning process: two steps

- **Learning (training):** Learn a model using the training data
- **Testing:** Test the model using **unseen test data** to assess the model accuracy

\[
\text{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}}
\]
What do we mean by learning?

- **Given**
  - a data set $D$,
  - a task $T$, and
  - a performance measure $M$,

  a computer system is said to **learn** from $D$ to perform the task $T$ if after learning the system’s performance on $T$ improves as measured by $M$.

- In other words, the learned model helps the system to perform $T$ better as compared to no learning.
An example

- **Data**: Loan application data
- **Task**: Predict whether a loan should be approved or not.
- **Performance measure**: accuracy.

**No learning**: classify all future applications (test data) to the majority class (i.e., Yes):

\[
\text{Accuracy} = \frac{9}{15} = 60\%.
\]

- **We can do better than 60% with learning.**
Assumption: The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to a certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.
Road Map

- Basic concepts
- Decision tree induction
- Classification using association rules
- Naïve Bayesian classification
- K-nearest neighbor
- Summary
Introduction

- Decision tree learning is one of the most widely used techniques for classification.
  - Its classification accuracy is competitive with other methods, and
  - it is very efficient.
- The classification model is a tree, called decision tree.
- C4.5 by Ross Quinlan is perhaps the best known system. It can be downloaded from the Web.
The loan data (reproduced)

<table>
<thead>
<tr>
<th>ID</th>
<th>Age</th>
<th>Has_Job</th>
<th>Own_House</th>
<th>Credit_Rating</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>young</td>
<td>false</td>
<td>false</td>
<td>fair</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>young</td>
<td>false</td>
<td>false</td>
<td>good</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>young</td>
<td>true</td>
<td>false</td>
<td>good</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>young</td>
<td>true</td>
<td>true</td>
<td>fair</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>young</td>
<td>false</td>
<td>false</td>
<td>fair</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>middle</td>
<td>false</td>
<td>false</td>
<td>fair</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>middle</td>
<td>false</td>
<td>false</td>
<td>good</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>middle</td>
<td>true</td>
<td>true</td>
<td>good</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>middle</td>
<td>false</td>
<td>true</td>
<td>excellent</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>middle</td>
<td>false</td>
<td>true</td>
<td>excellent</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>old</td>
<td>false</td>
<td>true</td>
<td>excellent</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>old</td>
<td>false</td>
<td>true</td>
<td>good</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>old</td>
<td>true</td>
<td>false</td>
<td>good</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>old</td>
<td>true</td>
<td>false</td>
<td>excellent</td>
<td>Yes</td>
</tr>
<tr>
<td>15</td>
<td>old</td>
<td>false</td>
<td>false</td>
<td>fair</td>
<td>No</td>
</tr>
</tbody>
</table>
A decision tree from the loan data

- **Decision nodes** and **leaf nodes (classes)**
Use the decision tree

<table>
<thead>
<tr>
<th>Age</th>
<th>Has_Job</th>
<th>Own_house</th>
<th>Credit-Rating</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>young</td>
<td>false</td>
<td>false</td>
<td>good</td>
<td>?</td>
</tr>
</tbody>
</table>

No
Is the decision tree unique?

- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
  - Easy to understand and perform better.

- Finding the best tree is NP-hard.
- All current tree building algorithms are heuristic algorithms
From a decision tree to a set of rules

- A decision tree can be converted to a set of rules.
- Each path from the root to a leaf is a rule.

```
Own_house = true → Class = Yes  [sup=6/15, conf=6/6]
Own_house = false, Has_job = true → Class = Yes  [sup=5/15, conf=5/5]
Own_house = false, Has_job = false → Class = No  [sup=4/15, conf=4/4]
```
Algorithm for decision tree learning

- Basic algorithm (a greedy *divide-and-conquer* algorithm)
  - Assume attributes are categorical now (continuous attributes can be handled too)
  - Tree is constructed in a top-down recursive manner
  - At start, all the training examples are at the root
  - Examples are partitioned recursively based on selected attributes
  - Attributes are selected on the basis of an impurity function (e.g., information gain)

- Conditions for stopping partitioning
  - All examples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority class is the leaf
  - There are no examples left
Algorithm decisionTree(D, A, T)
1    if D contains only training examples of the same class c_f ∈ C then
2       make T a leaf node labeled with class c_f;
3    elseif A = ∅ then
4       make T a leaf node labeled with c_f, which is the most frequent class in D
5    else  // D contains examples belonging to a mixture of classes. We select a single
6       // attribute to partition D into subsets so that each subset is purer
7       p_0 = impurityEval-1(D);
8       for each attribute A_i ∈ {A_1, A_2, ..., A_k} do
9          p_i = impurityEval-2(A_i, D)
10       end
11      Select A_g ∈ {A_1, A_2, ..., A_k} that gives the biggest impurity reduction,
12         computed using p_0 − p_i;
13      if p_0 − p_g < threshold then  // A_g does not significantly reduce impurity p_0
14         make T a leaf node labeled with c_f, the most frequent class in D.
15      else  // A_g is able to reduce impurity p_0
16         Make T a decision node on A_g;
17         Let the possible values of A_g be v_1, v_2, ..., v_m. Partition D into m
18             disjoint subsets D_1, D_2, ..., D_m based on the m values of A_g.
19         for each D_j in {D_1, D_2, ..., D_m} do
20            if D_j ≠ ∅ then
21               create a branch (edge) node T_j for v_j as a child node of T;
22               decisionTree(D_j, A−{A_g}, T_j)// A_g is removed
23          end
24       end
25   end
Choose an attribute to partition data

- The *key* to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.
  - A subset of data is *pure* if all instances belong to the same class.
- The *heuristic* in C4.5 is to choose the attribute with the maximum *Information Gain* or *Gain Ratio* based on information theory.
The loan data (reproduced)

<table>
<thead>
<tr>
<th>ID</th>
<th>Age</th>
<th>Has_Job</th>
<th>Own_House</th>
<th>Credit_Rating</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>young</td>
<td>false</td>
<td>false</td>
<td>fair</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>young</td>
<td>false</td>
<td>false</td>
<td>good</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>young</td>
<td>true</td>
<td>false</td>
<td>good</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>young</td>
<td>true</td>
<td>true</td>
<td>fair</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>young</td>
<td>false</td>
<td>false</td>
<td>fair</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>middle</td>
<td>false</td>
<td>false</td>
<td>fair</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>middle</td>
<td>false</td>
<td>false</td>
<td>good</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>middle</td>
<td>true</td>
<td>true</td>
<td>good</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>middle</td>
<td>false</td>
<td>true</td>
<td>excellent</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>middle</td>
<td>false</td>
<td>true</td>
<td>excellent</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>old</td>
<td>false</td>
<td>true</td>
<td>excellent</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>old</td>
<td>false</td>
<td>true</td>
<td>good</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>old</td>
<td>true</td>
<td>false</td>
<td>good</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>old</td>
<td>true</td>
<td>false</td>
<td>excellent</td>
<td>Yes</td>
</tr>
<tr>
<td>15</td>
<td>old</td>
<td>false</td>
<td>false</td>
<td>fair</td>
<td>No</td>
</tr>
</tbody>
</table>
Two possible roots, which is better?

- Fig. (B) seems to be better.
Road Map

- Basic concepts
- Decision tree induction
- **Classification using association rules**
- Naïve Bayesian classification
- K-nearest neighbor
- Summary
Using Class Association Rules

- **Classification**: mine a small set of rules existing in the data to form a classifier or predictor.
  - It has a target attribute: **Class attribute**

- **Association rules**: have no fixed target, but we can fix a target.

- **Class association rules (CAR)**: has a target class attribute. E.g.,
  
  \[
  \text{Own\_house = true } \rightarrow \text{Class =Yes } [\text{sup}=6/15, \text{conf}=6/6]
  \]

  - CARs can obviously be used for classification.
Decision tree vs. CARs

- The decision tree below generates the following 3 rules.
  Own_house = true → Class = Yes [sup=6/15, conf=6/6]
  Own_house = false, Has_job = true → Class = Yes [sup=5/15, conf=5/5]
  Own_house = false, Has_job = false → Class = No [sup=4/15, conf=4/4]

- But there are many other rules that are not found by the decision tree
There are many more rules

Age = young, Has_job = true → Class=Yes [sup=2/15, conf=2/2]
Age = young, Has_job = false → Class=No [sup=3/15, conf=3/3]
Credit_Rating = fair → Class=No [sup=4/15, conf=4/4]
Credit_Rating = good → Class=Yes [sup=5/15, conf=5/6]

and many more, if we use minsup = 2/15 = 13.3% and minconf = 80%.

- CAR mining finds all of them.
- In many cases, rules not in the decision tree (or a rule list) may perform classification better.
- Such rules may also be actionable in practice
Decision tree vs. CARs (cont …)

- Association mining require discrete attributes. Decision tree learning uses both discrete and continuous attributes.
  - CAR mining requires continuous attributes discretized. There are several such algorithms.
- Decision tree is not constrained by minsup or minconf, and thus is able to find rules with very low support. Of course, such rules may be pruned due to the possible overfitting.
Building classifiers

- There are many ways to build classifiers using CARs. Several existing systems available.
- **Strongest rules:** After CARs are mined, do nothing.
  - For each test case, we simply choose the most confident rule that covers the test case to classify it. Microsoft SQL Server has a similar method.
  - Or, using a combination of rules.
- **Selecting a subset of Rules**
  - used in the CBA system.
  - similar to sequential covering.
Definition: Given two rules, \( r_i \) and \( r_j \), \( r_i \succ r_j \) (also called \( r_i \) precedes \( r_j \) or \( r_i \) has a higher precedence than \( r_j \)) if

- the confidence of \( r_i \) is greater than that of \( r_j \), or
- their confidences are the same, but the support of \( r_i \) is greater than that of \( r_j \), or
- both the confidences and supports of \( r_i \) and \( r_j \) are the same, but \( r_i \) is generated earlier than \( r_j \).

A CBA classifier \( L \) is of the form:

\[
L = \langle r_1, r_2, \ldots, r_k, \text{default-class} \rangle
\]
Classifier building using CARs

Algorithm CBA(S, D)
1. $S = \text{sort}(S)$;  // sorting is done according to the precedence >
2. $\text{RuleList} = \emptyset$;  // the rule list classifier
3. for each rule $r \in S$ in sequence do
4.    if $D \neq \emptyset$ AND $r$ classifies at least one example in $D$ correctly then
5.        delete from $D$ all training examples covered by $r$;
6.        add $r$ at the end of RuleList
7.    end
8. end
9. add the majority class as the default class at the end of RuleList

- This algorithm is very inefficient
- CBA has a very efficient algorithm (quite sophisticated) that scans the data at most two times.
Using rules as features

- Most classification methods do not fully explore multi-attribute correlations, e.g., naïve Bayesian, decision trees, etc.

- This method creates extra attributes to augment the original data by
  - Using the conditional parts of rules
  - Each rule forms an new attribute
  - If a data record satisfies the condition of a rule, the attribute value is 1, and 0 otherwise

- One can also use only rules as attributes
Using normal association rules for classification

- A widely used approach
- **Main approach**: strongest rules
- **Main application**
  - Recommendation systems in e-commerce Web site (e.g., amazon.com).
  - Each rule consequent is the recommended item.
- **Major advantage**: any item can be predicted.
- **Main issue**:
  - **Coverage**: rare item rules are not found using classic algo.
Road Map

- Basic concepts
- Decision tree induction
- Classification using association rules
- Naïve Bayesian classification
- K-nearest neighbor
- Summary
## Naïve Bayesian Classifier: 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>

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

### Data sample:
- X = (age <=30,
  Income = medium,
  Student = yes,
  Credit_rating = Fair)
Bayesian Theorem: Basics

- Let $X$ be a data sample ("evidence"): class label is unknown
- Let $H$ be a hypothesis that $X$ belongs to class $C$
- Classification is to determine $P(H|X)$, the probability that the hypothesis holds given the observed data sample $X$
  - $P(H)$ (prior probability), the initial probability
    - E.g., Any given tuple will buy computer, regardless of age, income, …
  - $P(X)$: probability that sample data is observed
  - $P(X|H)$ (posteriori probability), the probability of observing the sample $X$, given that the hypothesis holds
    - E.g., Given that $X$ will buy computer, the prob. that $X$ is 31..40, medium income
Bayesian Theorem

- Given training data \( X \), posteriori probability of a hypothesis \( H \), \( P(H|X) \), follows the Bayes theorem

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

- Informally, this can be written as
  \[
  \text{posteriori} = \text{likelihood} \times \frac{\text{prior}}{\text{evidence}}
  \]

- Predicts \( X \) belongs to \( C_i \) iff the probability \( P(C_i|X) \) is the highest among all the \( P(C_k|X) \) for all the \( k \) classes

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost
Towards Naïve Bayesian Classifier

- 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 $P(C_i|\mathbf{X})$
- This can be derived from Bayes’ theorem

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

- Since $P(\mathbf{X})$ is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized
A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

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

This greatly reduces the computation cost: Only counts the class distribution

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 continuous-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: 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>

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

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

- \( P(C_i) \):  
  \( P(\text{buys\_computer} = \text{“yes”}) = 9/14 = 0.643 \)  
  \( P(\text{buys\_computer} = \text{“no”}) = 5/14 = 0.357 \)

- Compute \( P(X|C_i) \) for each class:
  
  \( P(\text{age} = \text{“\leq 30”} \mid \text{buys\_computer} = \text{“yes”}) = 2/9 = 0.222 \)
  
  \( P(\text{income} = \text{“medium”} \mid \text{buys\_computer} = \text{“yes”}) = 4/9 = 0.444 \)
  
  \( P(\text{student} = \text{“yes”} \mid \text{buys\_computer} = \text{“yes”}) = 6/9 = 0.667 \)
  
  \( P(\text{credit\_rating} = \text{“fair”} \mid \text{buys\_computer} = \text{“yes”}) = 6/9 = 0.667 \)
  
  \( P(\text{age} = \text{“\leq 30”} \mid \text{buys\_computer} = \text{“no”}) = 3/5 = 0.6 \)
  
  \( P(\text{income} = \text{“medium”} \mid \text{buys\_computer} = \text{“no”}) = 2/5 = 0.4 \)
  
  \( P(\text{student} = \text{“yes”} \mid \text{buys\_computer} = \text{“no”}) = 1/5 = 0.2 \)
  
  \( P(\text{credit\_rating} = \text{“fair”} \mid \text{buys\_computer} = \text{“no”}) = 2/5 = 0.4 \)

- \( X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair}) \)

\[
\begin{align*}
P(X|C_i) & : P(X|\text{buys\_computer} = \text{“yes”}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044 \\
P(X|\text{buys\_computer} = \text{“no”}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019 \\
P(X|C_i) \times P(C_i) & : P(X|\text{buys\_computer} = \text{“yes”}) \times P(\text{buys\_computer} = \text{“yes”}) = 0.028 \\
P(X|\text{buys\_computer} = \text{“no”}) \times P(\text{buys\_computer} = \text{“no”}) = 0.007
\end{align*}
\]

Therefore, \( X \) belongs to class (\( “\text{buys\_computer} = \text{yes}” \))
On naïve Bayesian classifier

- **Advantages:**
  - Easy to implement
  - Very efficient
  - Good results obtained in many applications

- **Disadvantages**
  - Assumption: class conditional independence, therefore loss of accuracy when the assumption is seriously violated (those highly correlated data sets)
Road Map

- Basic concepts
- Decision tree induction
- Classification using association rules
- Naïve Bayesian classification
- **K-nearest neighbor**
- Summary
k-Nearest Neighbor Classification (kNN)

- Unlike all the previous learning methods, kNN does not build model from the training data.
- To classify a test instance \( d \), define \( k \)-neighborhood \( P \) as \( k \) nearest neighbors of \( d \)
- Count number \( n \) of training instances in \( P \) that belong to class \( c_j \)
- Estimate \( \Pr(c_j|d) \) as \( n/k \)
- No training is needed. Classification time is linear in training set size for each test case.
**kNN Algorithm**

**Algorithm** kNN\((D, d, k)\)

1. Compute the distance between \(d\) and every example in \(D\);
2. Choose the \(k\) examples in \(D\) that are nearest to \(d\), denote the set by \(P(\subseteq D)\);
3. Assign \(d\) the class that is the most frequent class in \(P\) (or the majority class);

- **\(k\) is usually chosen empirically via a validation set or cross-validation by trying a range of \(k\) values.**
- **Distance function** is crucial, but depends on applications.
Example: $k=6$ (6NN)

A new point Pr(science|_n)?
Discussions

- kNN can deal with complex and arbitrary decision boundaries.
- Despite its simplicity, researchers have shown that the classification accuracy of kNN can be quite strong and in many cases as accurate as those elaborated methods.
- kNN is slow at the classification time
- kNN does not produce an understandable model
Road Map

- Basic concepts
- Decision tree induction
- Classification using association rules
- Naïve Bayesian classification
- K-nearest neighbor
- Summary
Summary

- Applications of supervised learning are in almost any field or domain.
- We studied some classification techniques.
- There are still many other methods, e.g.,
  - Bayesian networks
  - Neural networks
  - Genetic algorithms
  - Fuzzy classification
  This large number of methods also show the importance of classification and its wide applicability.
- It remains to be an active research area.