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 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, … 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)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

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?

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

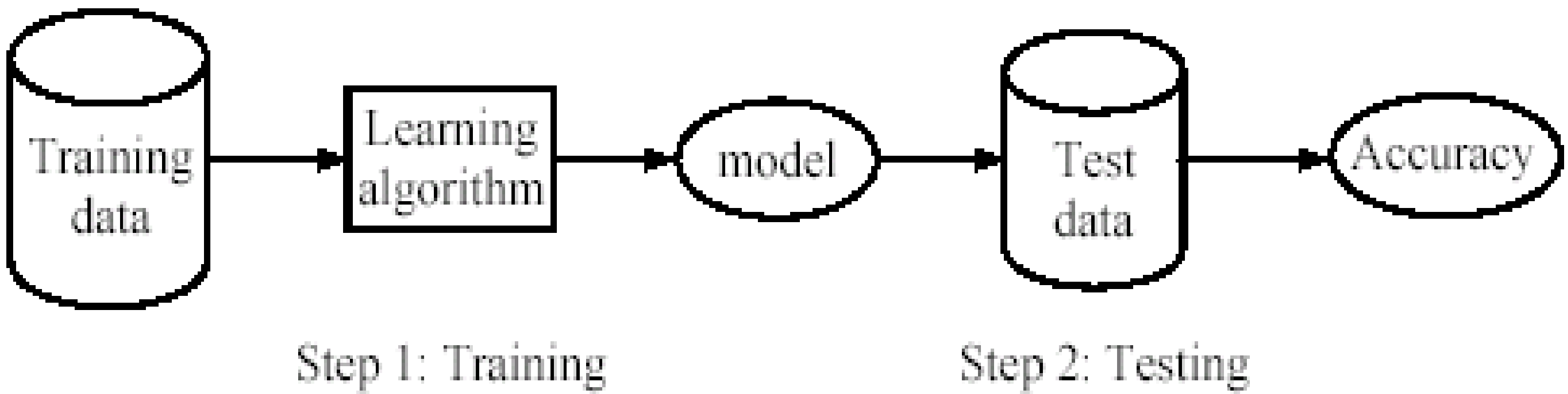
Supervised vs. unsupervised Learning

- 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

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



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):

Accuracy = 9/15 = 60%.

- We can do better than 60% with learning.

Fundamental assumption of 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 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.

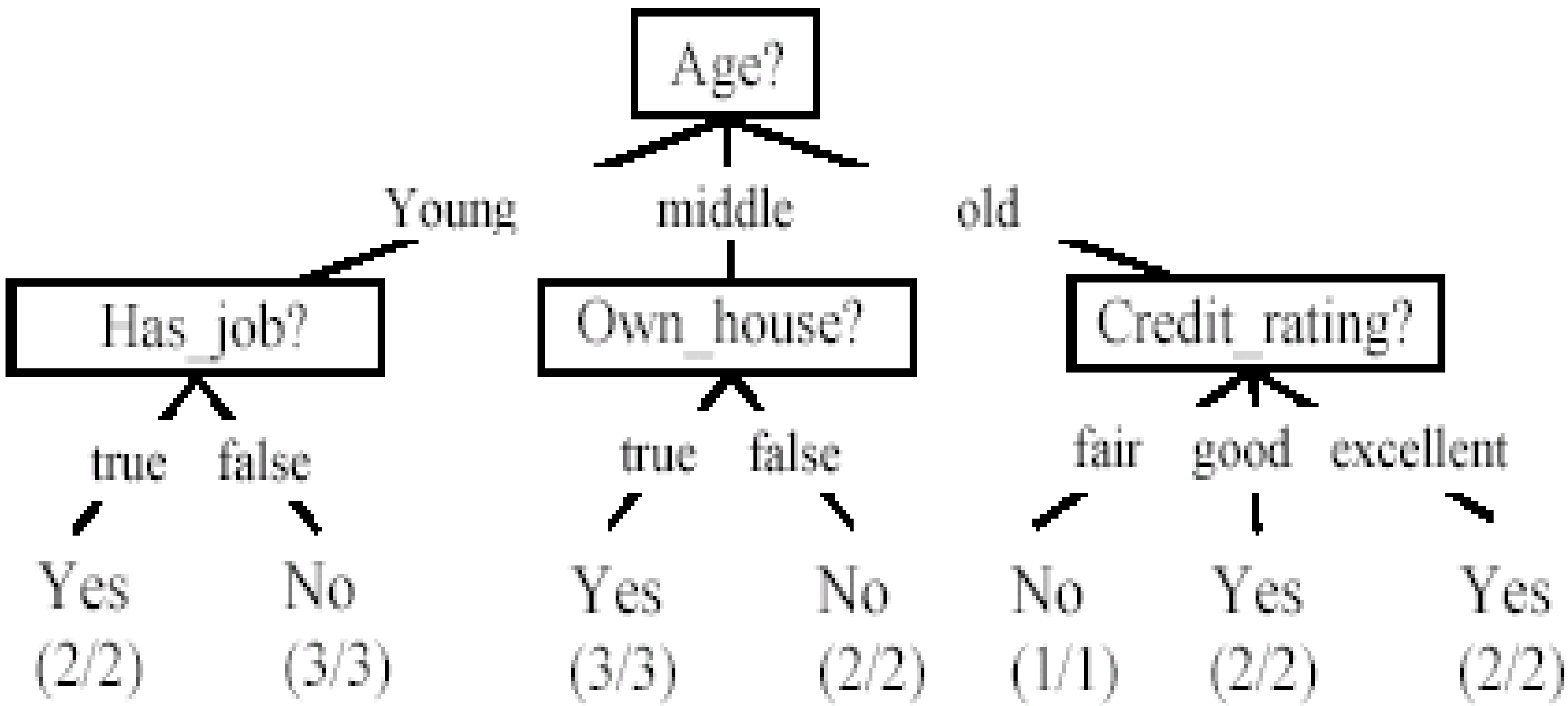
The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

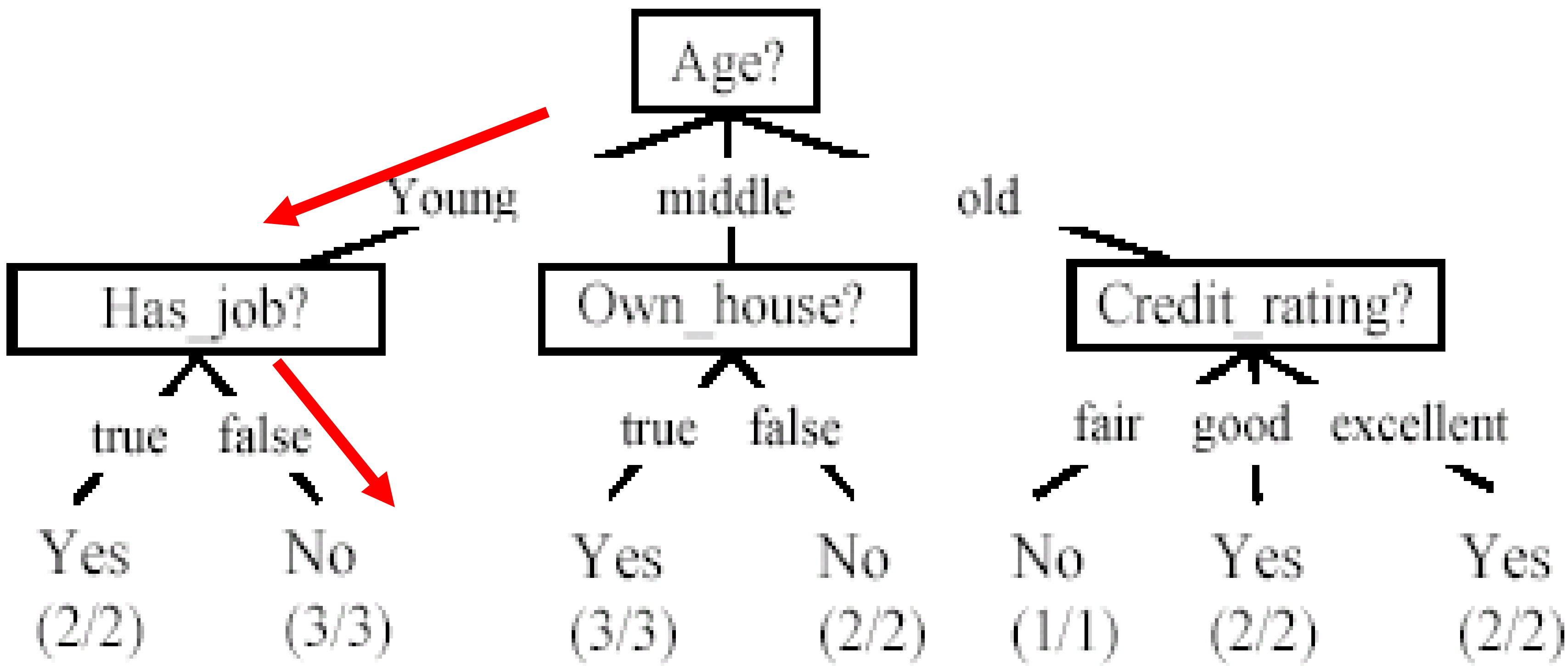
A decision tree from the loan data

- Decision nodes and leaf nodes (classes)



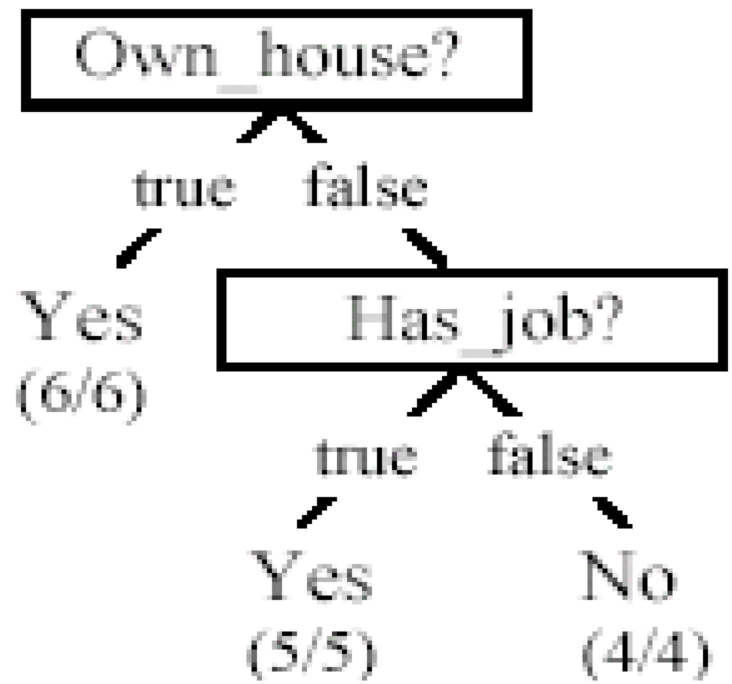
Use the decision tree

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	? No



Is the decision tree unique?

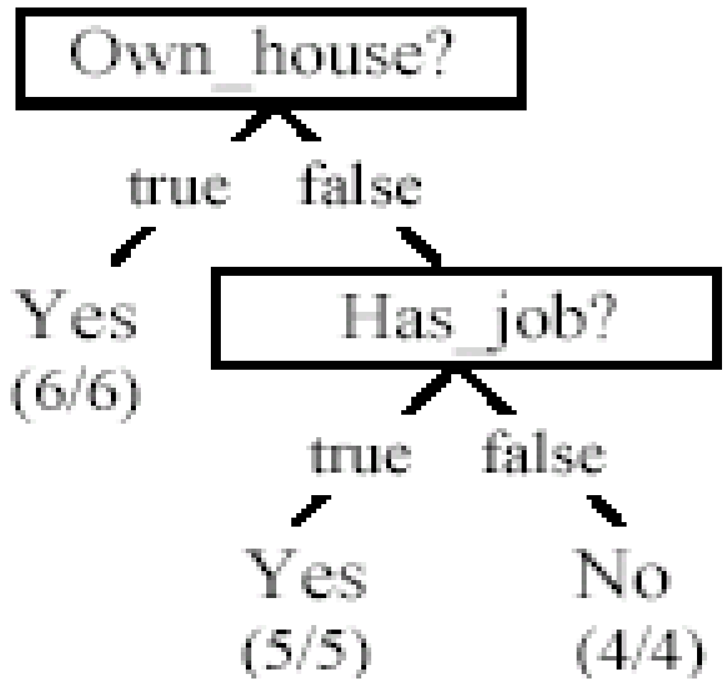
- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
  - Easy to understand and perform better.



- 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

Decision tree learning algorithm

```
. Algorithm decisionTree(D, A, T)
    if D contains only training examples of the same class c_j ∈ C then
        make T a leaf node labeled with class c_j;
    elseif A = ∅ then
        make T a leaf node labeled with c_j, which is the most frequent class in D
    else   // D contains examples belonging to a mixture of classes. We select a single
           // attribute to partition D into subsets so that each subset is purer
        p_0 = impurityEval-1(D);
        for each attribute A_i ∈ {A_1, A_2, …, A_k} do
            p_i = impurityEval-2(A_i, D)
        end
        Select A_g ∈ {A_1, A_2, …, A_k} that gives the biggest impurity reduction,
            computed using p_0 – p_i;
        if p_0 – p_g < threshold then     // A_g does not significantly reduce impurity p_0
           make T a leaf node labeled with c_j, the most frequent class in D.
        else                              // A_g is able to reduce impurity p_0
            Make T a decision node on A_g;
            Let the possible values of A_g be v_1, v_2, …, v_m. Partition D into m
                disjoint subsets D_1, D_2, …, D_m based on the m values of A_g.
            for each D_j in {D_1, D_2, …, D_m} do
                if D_j ≠ ∅ then
                   create a branch (edge) node T_j for v_j as a child node of T;
                   decisionTree(D_j, A-{A_g}, T_j)// A_g is removed
                end
            end
        end
    end
```

Choose an attribute to partition data

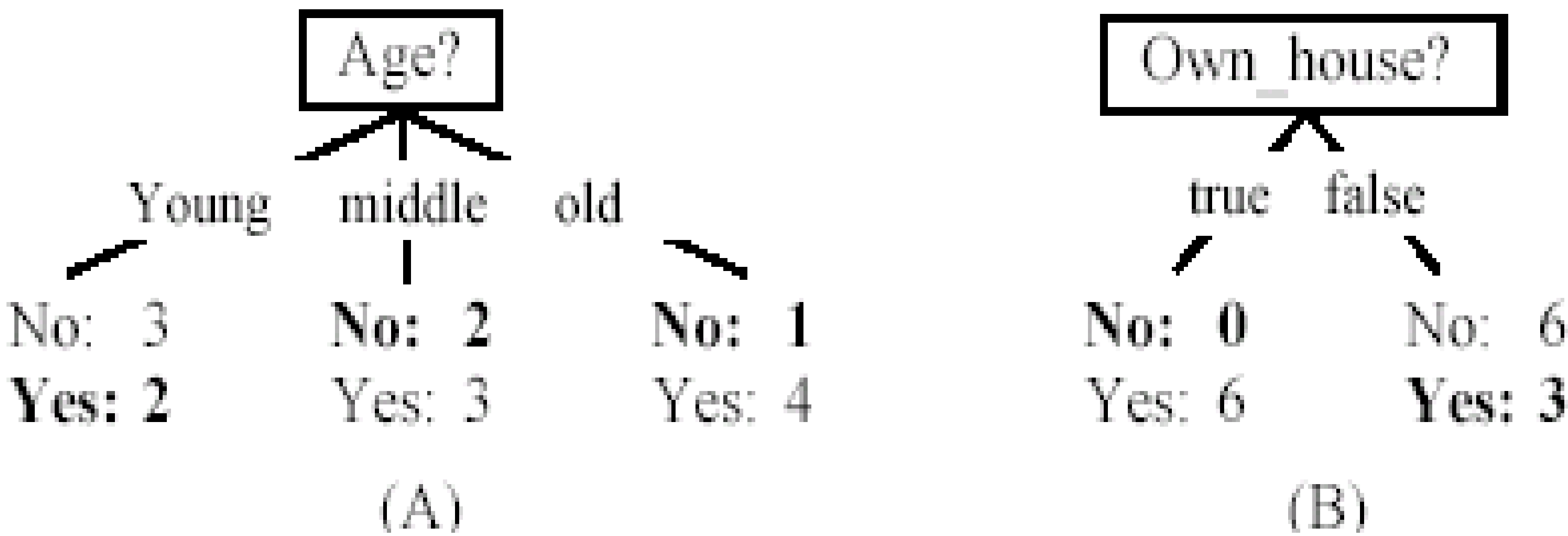
- The *key* to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.
  - A subset of data is pure if all instances belong to the same class.
- The *heuristic* in C4.5 is to choose the attribute with the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

The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

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.,

  Own_house = true $\rightarrow$ Class =Yes [sup=6/15, conf=6/6]
  - CARs can obviously be used for classification.

Decision tree vs. CARs

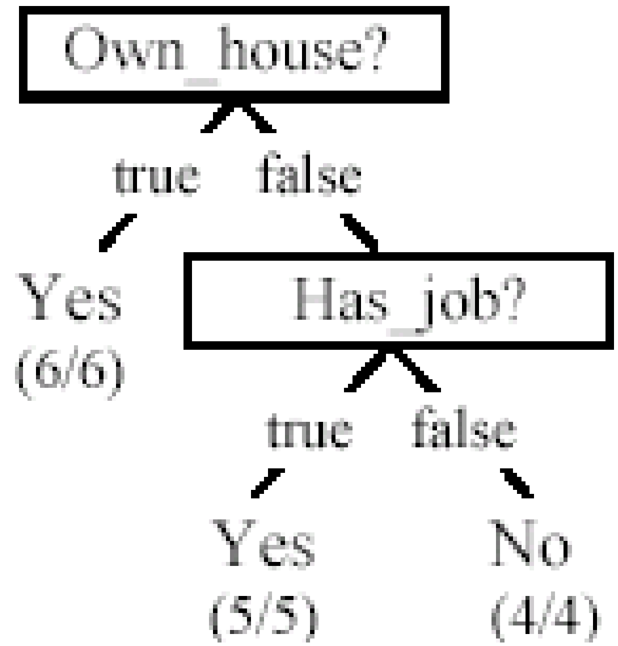
- **The decision tree below generates the following 3 rules.**

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

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	excellent	No
3	young	true	false	good	Yes
4	young	true	true	good	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Decision tree vs. CARs (cont …)

- Association mining require discrete attributes. Decision tree learning uses both discrete and continuous attributes.
- 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.

CBA: Rules are sorted first

Definition: Given two rules, r_i and r_j, $r_i \succ r_j$ (also called *ri* 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 = <r_1, r_2, …, r_k, *default-class*>

Classifier building using CARs

```
Algorithm CBA(S, D)
S = sort(S);                // sorting is done according to the precedence ≻
RuleList = ∅;               // the rule list classifier
for each rule r ∈ S in sequence do
    if D ≠ ∅ AND r classifies at least one example in D correctly then
        delete from D all training examples covered by r;
        add r at the end of RuleList
    end
end
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 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

- Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

- Data sample:

X = (age <=30,

Income = medium,

Student = yes,

Credit_rating = Fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

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

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

$$P(C_1|\mathbf{X}) = \frac{P(\mathbf{X}|C_1)P(C_1)}{P(\mathbf{X})}$$

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

- Informally, this can be written as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_i iff the probability $P(C_i|\mathbf{X})$ 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 **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(X) is constant for all classes, only

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

  needs to be maximized

Derivation of Naïve Bayes Classifier

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

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

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

Naïve Bayesian Classifier: Training Dataset

- Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

- Data sample

X = (age <=30,

Income = medium,

Student = yes,

Credit_rating = Fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayesian Classifier: An 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(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667

  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

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

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = 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.

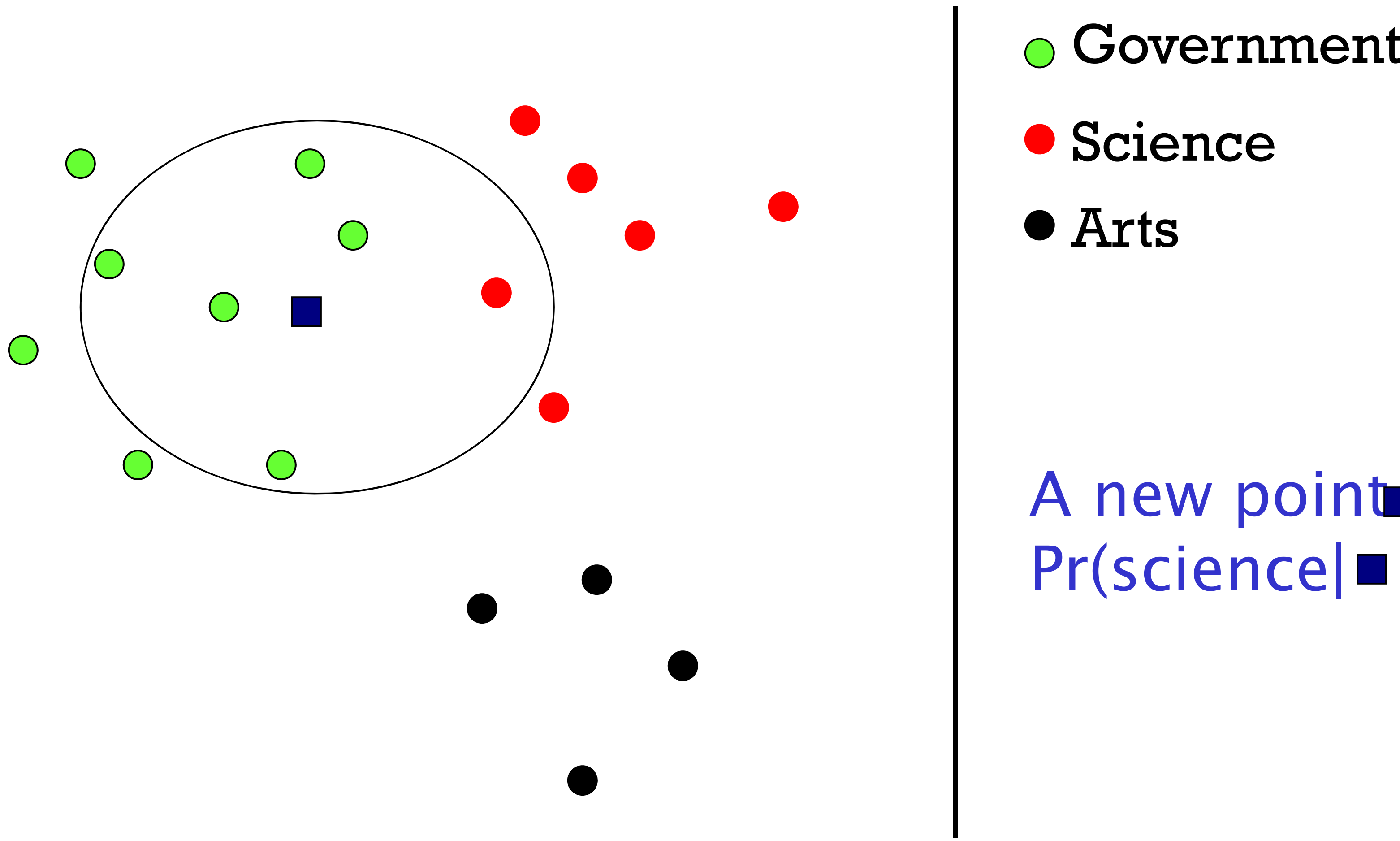
kNNAlgorithm

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|)?

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.