
Chapter 3: Supervised Learning

Most slides courtesy Bing Liu

Road Map

- **Basic concepts**
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- Ensemble methods: Bagging and Boosting
- Summary

An example 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 humans 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, \dots, 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, unseen test) cases/instances.

An example: data (loan appl.)

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	?

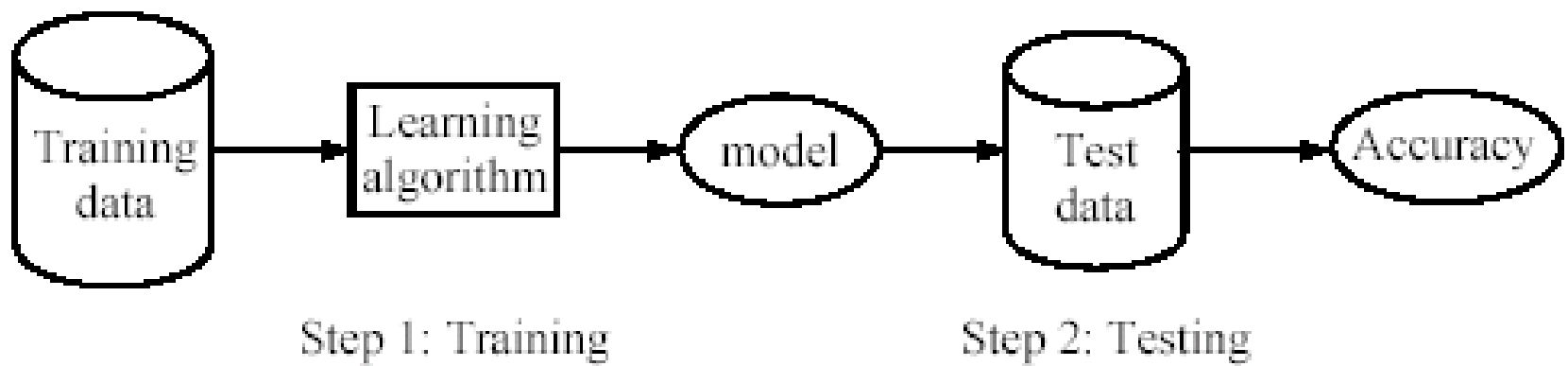
Supervised vs. unsup. learning

- **Supervised learning:** classification is seen as supervised learning from examples.
 - **Supervision:** The data (observations, measurements, etc.) are labeled with pre-defined classes. It is as if a “teacher” gave 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: two steps

- **Learning (training)**: Build 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**):

$$\text{Accuracy} = 9/15 = 60\%.$$

- **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 some 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**
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- Ensemble methods: Bagging and Boosting
- 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.

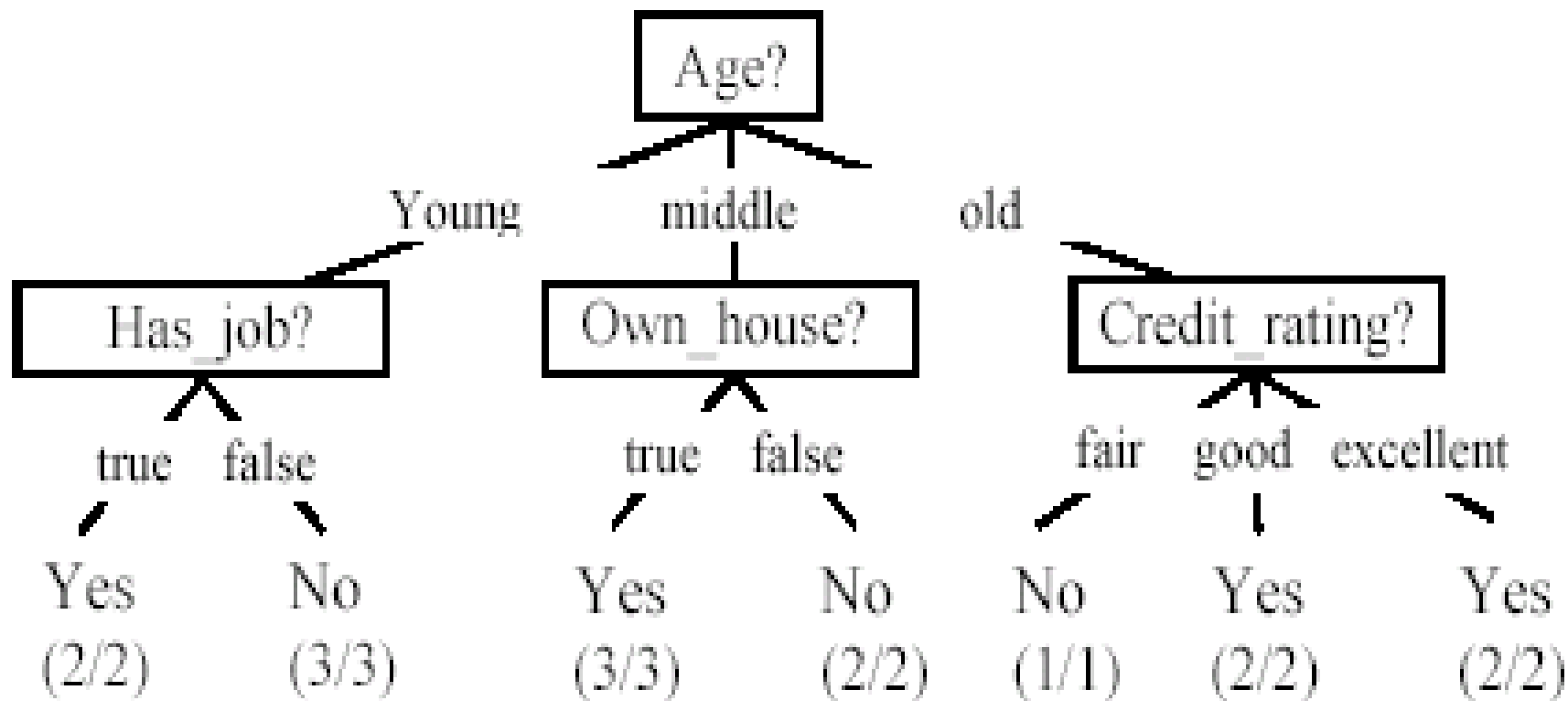
The loan data (reproduced)

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

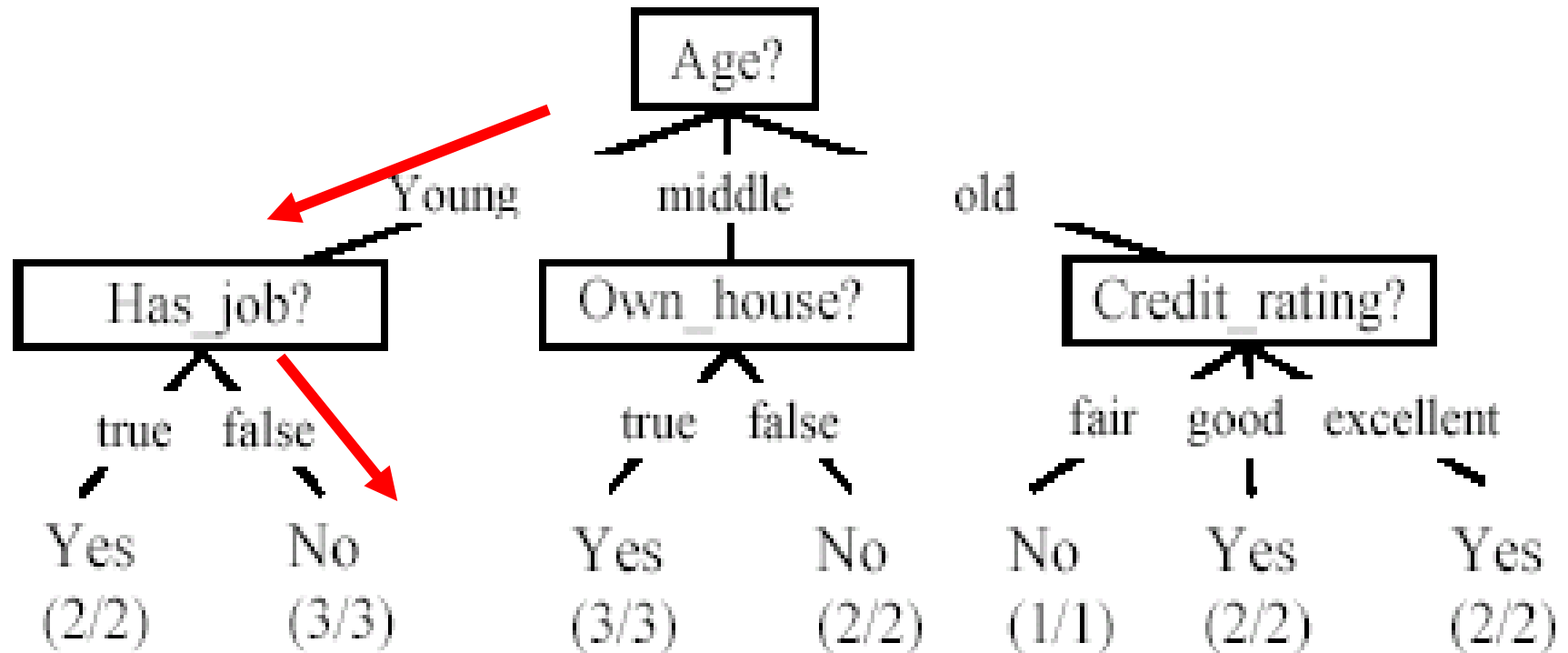
Decision tree from the loan data

- Decision nodes and leaf nodes (classes)



Use the decision tree

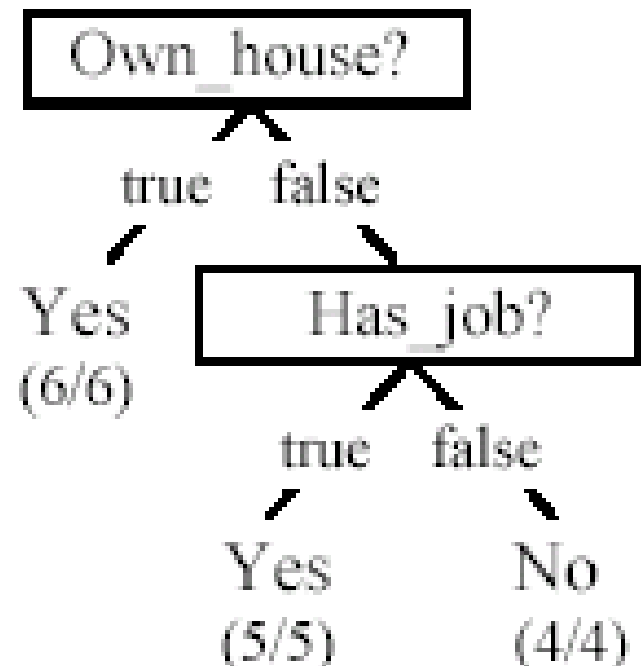
Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	? No



Is the decision tree unique?

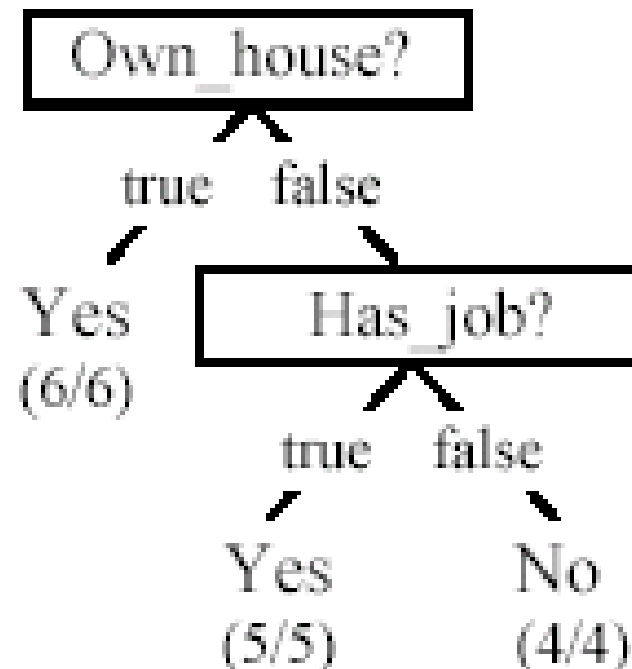
- **No.** Here is a simpler tree.
- We want **smaller tree** and **accurate tree**.
 - Easy to understand and often performs better.

- 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( $D, A, T$ )
1   if  $D$  contains only training examples of the same class  $c_j \in C$  then
2     make  $T$  a leaf node labeled with class  $c_j$ ;
3   elseif  $A = \emptyset$  then
4     make  $T$  a leaf node labeled with  $c_j$ , 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      $p_0 = \text{impurityEval-1}(D)$ ;
8     for each attribute  $A_i \in \{A_1, A_2, \dots, A_k\}$  do
9        $p_i = \text{impurityEval-2}(A_i, D)$ 
10    end
11    Select  $A_g \in \{A_1, A_2, \dots, A_k\}$  that gives the biggest impurity reduction,
        computed using  $p_0 - p_i$ ;
12    if  $p_0 - p_g < \text{threshold}$  then //  $A_g$  does not significantly reduce impurity  $p_0$ 
13      make  $T$  a leaf node labeled with  $c_j$ , the most frequent class in  $D$ .
14    else //  $A_g$  is able to reduce impurity  $p_0$ 
15      Make  $T$  a decision node on  $A_g$ ;
16      Let the possible values of  $A_g$  be  $v_1, v_2, \dots, v_m$ . Partition  $D$  into  $m$ 
        disjoint subsets  $D_1, D_2, \dots, D_m$  based on the  $m$  values of  $A_g$ .
17      for each  $D_j$  in  $\{D_1, D_2, \dots, D_m\}$  do
18        if  $D_j \neq \emptyset$  then
19          create a branch (edge) node  $T_j$  for  $v_j$  as a child node of  $T$ ;
20          decisionTree( $D_j, A - \{A_g\}, T_j$ ) //  $A_g$  is removed
21        end
22      end
23    end
24  end
```

Choose attribute to partition data

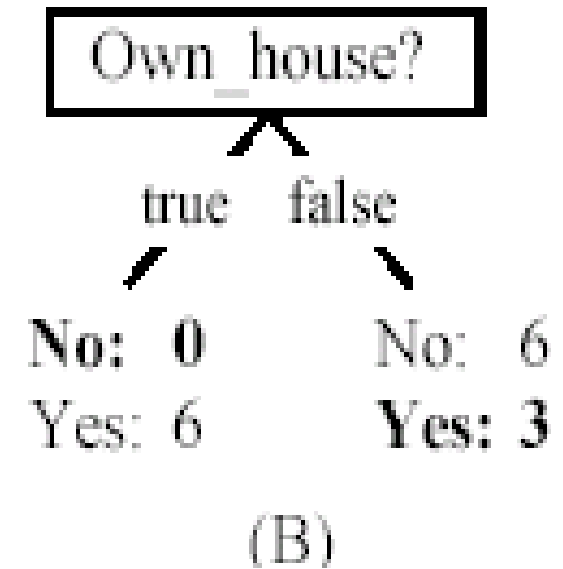
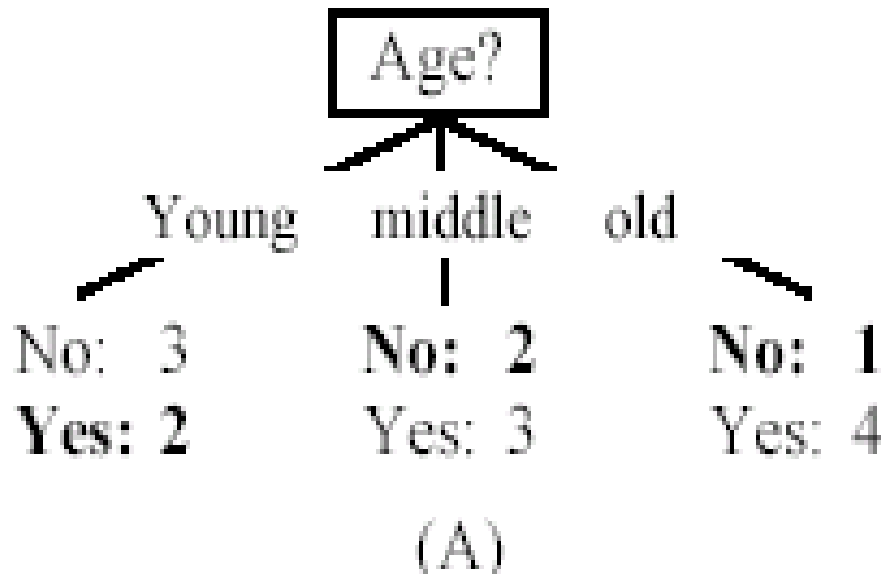
- The *key* to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.
 - A subset of data is *pure* if all instances belong to the same class.
- The *heuristic* in C4.5 is to choose the attribute with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

Information theory

- **Information theory** provides a mathematical basis for measuring the information content.
- To understand the notion of information, think about it as providing the answer to a question, for example, whether a coin will come up heads.
 - If one already has a good guess about the answer, then the actual answer is less informative.
 - If one already knows that the coin is rigged so that it will come with heads with probability 0.99, then a message (advanced information) about the actual outcome of a flip is worth less than it would be for a honest coin (50-50).

Information theory (cont ...)

- For a fair (honest) coin, you have no information, and you are willing to pay more (say in terms of \$) for advanced information - less you know, the more valuable the information.
- **Information theory** uses this same intuition, but instead of measuring the value for information in dollars, it measures information contents in **bits**.
- One bit of information is enough to answer a yes/no question about which one has no idea, such as the flip of a fair coin

Information theory: Entropy

- The entropy formula,

$$\text{entropy}(D) = - \sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

$$\sum_{j=1}^{|C|} \Pr(c_j) = 1,$$

- $\Pr(c_j)$ is the probability of class c_j in data set D
- We use entropy as a **measure of impurity or disorder** of data set D . (Or, a measure of information in a tree)

Entropy measure: some intuition

1. The data set D has 50% positive examples ($\Pr(\text{positive}) = 0.5$) and 50% negative examples ($\Pr(\text{negative}) = 0.5$).

$$\text{entropy}(D) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$$

2. The data set D has 20% positive examples ($\Pr(\text{positive}) = 0.2$) and 80% negative examples ($\Pr(\text{negative}) = 0.8$).

$$\text{entropy}(D) = -0.2 \times \log_2 0.2 - 0.8 \times \log_2 0.8 = 0.722$$

3. The data set D has 100% positive examples ($\Pr(\text{positive}) = 1$) and no negative examples, ($\Pr(\text{negative}) = 0$).

$$\text{entropy}(D) = -1 \times \log_2 1 - 0 \times \log_2 0 = 0$$

- As the data become purer and purer, the entropy value becomes smaller and smaller. This is useful to us!

Information gain

- Given a set of examples D , we first compute its entropy:

$$\text{entropy}(D) = - \sum_{j=1}^{|C|} \text{Pr}(c_j) \log_2 \text{Pr}(c_j)$$

- If we make attribute A_i , with v values, the root of the current tree, this will partition D into v subsets D_1, D_2, \dots, D_v . The expected entropy if A_i is used as the current root:

$$\text{entropy}_{A_i}(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times \text{entropy}(D_j)$$

Information gain (cont ...)

- **Information gained** by selecting attribute A_i to branch or to partition the data is

$$\textit{gain}(D, A_i) = \textit{entropy}(D) - \textit{entropy}_{A_i}(D)$$

- We choose the attribute with the highest gain to branch/split the current tree.

An example

$$entropy(D) = -\frac{6}{15} \times \log_2 \frac{6}{15} - \frac{9}{15} \times \log_2 \frac{9}{15} = 0.971$$

$$\begin{aligned} entropy_{Own_house}(D) &= -\frac{6}{15} \times entropy(D_1) - \frac{9}{15} \times entropy(D_2) \\ &= \frac{6}{15} \times 0 + \frac{9}{15} \times 0.918 \\ &= 0.551 \end{aligned}$$

$$\begin{aligned} entropy_{Age}(D) &= -\frac{5}{15} \times entropy(D_1) - \frac{5}{15} \times entropy(D_2) - \frac{5}{15} \times entropy(D_3) \\ &= \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.722 \\ &= 0.888 \end{aligned}$$

- Own_house is the best choice for the root.

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

Age	Yes	No	entropy(D _i)
young	2	3	0.971
middle	3	2	0.971
old	4	1	0.722

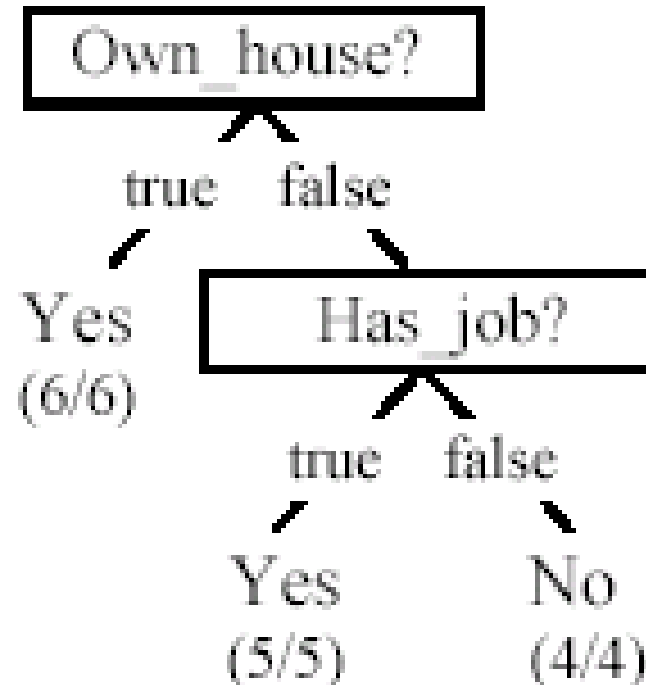
$$gain(D, Age) = 0.971 - 0.888 = 0.083$$

$$gain(D, Own_house) = 0.971 - 0.551 = 0.420$$

$$gain(D, Has_Job) = 0.971 - 0.647 = 0.324$$

$$gain(D, Credit_Rating) = 0.971 - 0.608 = 0.363$$

We build the final tree

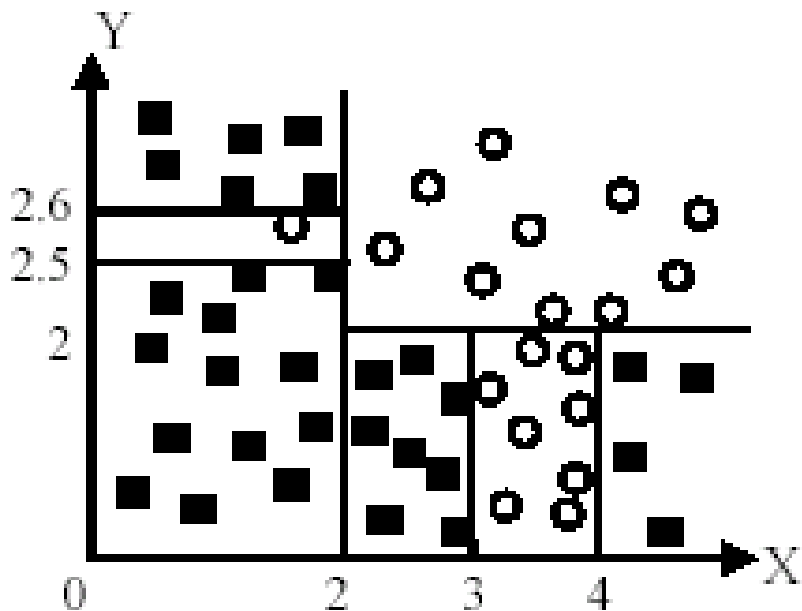


- We could use information gain ratio to evaluate the impurity as well (see book)

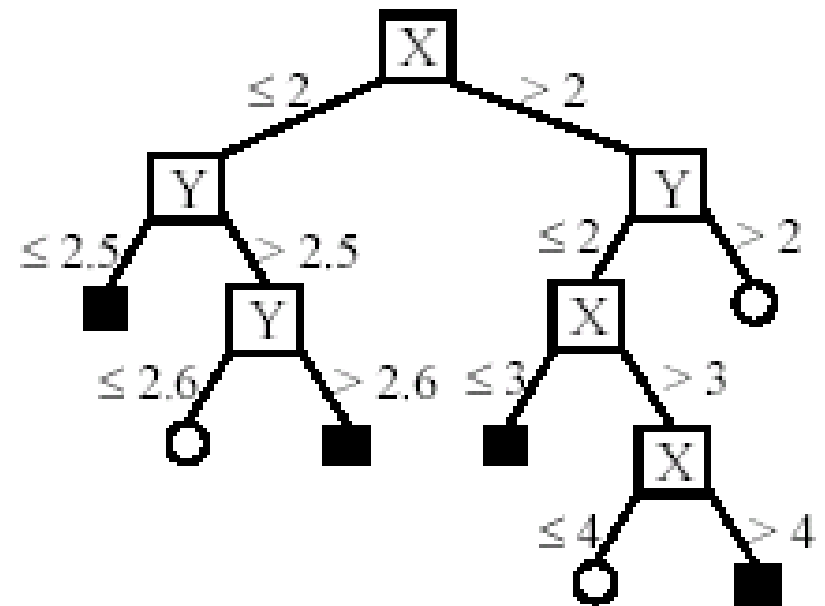
Handling continuous attributes

- Handle continuous attribute by splitting into two intervals (can be more) at each node.
- How to find the best threshold to divide?
 - Use information gain or gain ratio again
 - Sort all the values of an continuous attribute in increasing order $\{v_1, v_2, \dots, v_r\}$,
 - One possible threshold between two adjacent values v_i and v_{i+1} . Try all possible thresholds and find the one that maximizes the gain (or gain ratio).

A continuous space example



(A) A partition of the data space

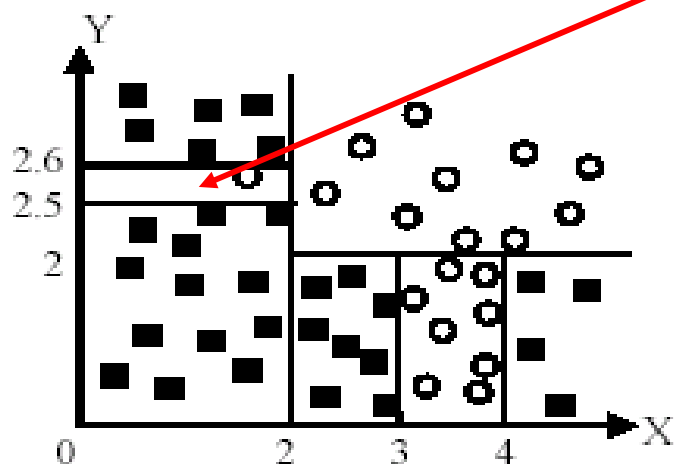


(B). The decision tree

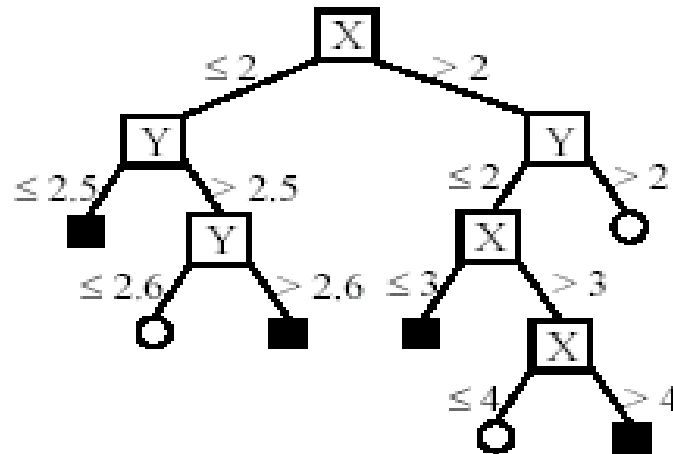
Avoid overfitting in classification

- **Overfitting:** A tree may overfit the training data
 - Good accuracy on training data but poor on test data
 - Symptoms: tree too deep and too many branches, some may reflect anomalies due to noise or outliers
- Two approaches to avoid overfitting
 - **Pre-pruning:** Halt tree construction early
 - Difficult to decide because we do not know what may happen subsequently if we keep growing the tree.
 - **Post-pruning:** Remove branches or sub-trees from a “fully grown” tree.
 - This method is commonly used. C4.5 uses a statistical method to estimate the errors at each node for pruning.
 - A validation set may be used for pruning as well.

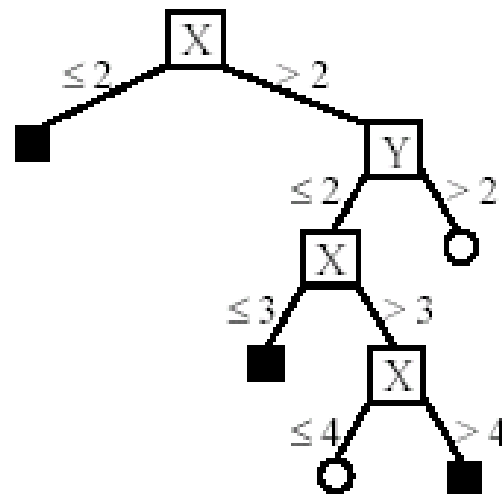
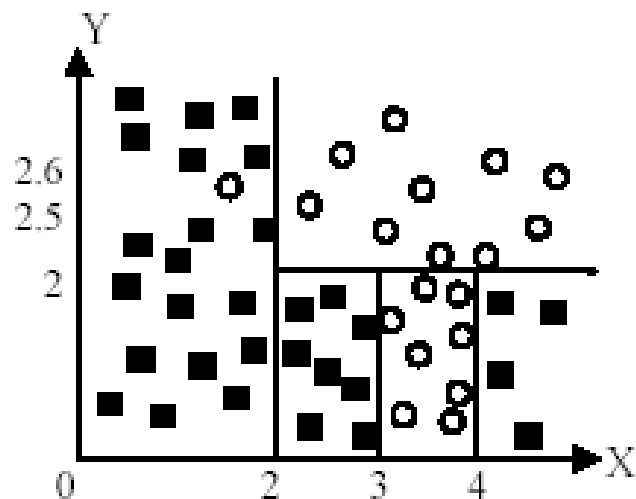
An example Likely to overfit the data



(A) A partition of the data space



(B). The decision tree



Road Map

- Basic concepts
- Decision tree induction
- **Evaluation of classifiers**
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- Ensemble methods: Bagging and Boosting
- Summary

Evaluating classification methods

- **Predictive accuracy**

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

- **Efficiency**

- time to construct the model
- time to use the model

- **Robustness**: handling noise and missing values

- **Scalability**: efficiency in disk-resident databases

- **Interpretability**:

- understandability and insight provided by the model

- **Compactness of the model**: size of the tree, or the number of rules.

Evaluation methods

- **Holdout set:** The available data set D is divided into two disjoint subsets,
 - the *training set* D_{train} (for learning a model)
 - the *test set* D_{test} (for testing the model)
- **Important:** training set should not be used in testing and the test set should not be used in learning.
 - Unseen test set provides a unbiased estimate of accuracy.
- The test set is also called the **holdout set**. (the examples in the original data set D are all labeled with classes.)
- This method is mainly used when the data set D is large.

Evaluation methods (cont...)

- **n-fold cross-validation**: The available data is partitioned into n equal-size disjoint subsets.
- Use each subset as the test set and combine the rest $n-1$ subsets as the training set to learn a classifier.
- The procedure is run n times, which give n accuracies.
- The final estimated accuracy of learning is the average of the n accuracies.
- 10-fold and 5-fold cross-validations are commonly used.
- This method is used when the available data is not large.

Evaluation methods (cont...)

- **Leave-one-out cross-validation**: This method is used when the data set is very small.
- It is a special case of cross-validation
- Each fold of the cross validation has only **a single test example** and all the rest of the data is used in training.
- If the original data has m examples, this is **m -fold cross-validation**

Evaluation methods (cont...)

- **Validation set:** the available data is divided into three subsets,
 - a training set,
 - a validation set and
 - a test set.
- A validation set is used frequently for estimating / tuning parameters in learning algorithms.
- In such cases, the values that give the best accuracy on the validation set are used as the final parameter values.

Classification measures

- Accuracy is only one measure (error = 1-accuracy).
- **Accuracy is not suitable in some applications.**
- In text mining, we may only be interested in the documents of a particular topic, which are only a small portion of a big document collection.
- In classification involving skewed or highly imbalanced data, e.g., network intrusion and financial fraud detections, **we are interested only in the minority class.**
 - High accuracy does not mean any intrusion is detected.
 - E.g., 1% intrusion. Achieve 99% accuracy by doing nothing.
- The class of interest is commonly called the **positive class**, and the rest **negative classes**.

Precision and recall measures

- Used in information retrieval and text classification.
- We use a confusion matrix to introduce them.

	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

where

TP: the number of correct classifications of the positive examples (**true positive**),

FN: the number of incorrect classifications of positive examples (**false negative**),

FP: the number of incorrect classifications of negative examples (**false positive**), and

TN: the number of correct classifications of negative examples (**true negative**).

Precision and recall measures

(cont...)

	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

$$p = \frac{TP}{TP + FP} \quad r = \frac{TP}{TP + FN}$$

- **Precision** p is the number of **correctly classified positive examples** divided by the total number of examples that are classified as positive.
- **Recall** r is the number of **correctly classified positive examples** divided by the total number of actual positive examples in the test set.

An example

	Classified Positive	Classified Negative
Actual Positive	1	99
Actual Negative	0	1000

- **This confusion matrix gives**

- precision $p = 100\%$ and
- recall $r = 1\%$

because we only classified one positive example correctly and no negative examples wrongly.

- **Note:** precision and recall only measure classification on the positive class.

F_1 -value (also called F_1 -score)

- It is hard to compare two classifiers using two measures. F_1 score combines precision and recall into one measure

$$F_1 = \frac{2pr}{p+r}$$

F_1 -score is the harmonic mean of precision and recall.

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

- The harmonic mean of two numbers tends to be closer to the smaller of the two.
- For F_1 -value to be large, both p and r must be large.

Road Map

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- **Rule induction**
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- Summary

Road Map

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- **Classification using association rules**
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- Ensemble methods: Bagging and Boosting
- Summary

Road Map

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- **Naïve Bayesian classification**
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- Ensemble methods: Bagging and Boosting
- Summary

Bayesian classification

- **Probabilistic view:** Supervised learning can naturally be studied from a probabilistic point of view.
- Let A_1 through A_k be attributes with discrete values. The class is C .
- Given a test example d with observed attribute values a_1 through a_k .
- Classification is basically to compute the following posteriori probability. The prediction is the class c_j such that

$$\Pr(C = c_j \mid A_1 = a_1, \dots, A_{|A|} = a_{|A|})$$

is maximal

Apply Bayes' Rule

$$\begin{aligned} & \Pr(C = c_j \mid A_1 = a_1, \dots, A_{|A|} = a_{|A|}) \\ &= \frac{\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_j) \Pr(C = c_j)}{\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|})} \\ &= \frac{\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_j) \Pr(C = c_j)}{\sum_{r=1}^{|C|} \Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_r) \Pr(C = c_r)} \end{aligned}$$

- $\Pr(C=c_j)$ is the class *prior* probability: easy to estimate from the training data.

Computing probabilities

- The denominator $P(A_1=a_1, \dots, A_k=a_k)$ is irrelevant for decision making since it is the same for every class.
- We only need $P(A_1=a_1, \dots, A_k=a_k \mid C=c_j)$, which can be written as
$$\Pr(A_1=a_1 \mid A_2=a_2, \dots, A_k=a_k, C=c_j) * \Pr(A_2=a_2, \dots, A_k=a_k \mid C=c_j)$$
- Recursively, the second factor above can be written in the same way, and so on.
- Now an assumption is needed.

Important: Assumption of conditional independence

- All attributes are conditionally independent given the class $C = c_j$.
- Formally, we assume,

$$\Pr(A_1=a_1 \mid A_2=a_2, \dots, A_{|A|}=a_{|A|}, C=c_j) = \Pr(A_1=a_1 \mid C=c_j)$$

and so on for A_2 through $A_{|A|}$. I.e.,

$$\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_j) = \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

Final naïve Bayesian classifier

$$\begin{aligned} & \Pr(C = c_j \mid A_1 = a_1, \dots, A_{|A|} = a_{|A|}) \\ &= \frac{\Pr(C = c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)}{\sum_{r=1}^{|C|} \Pr(C = c_r) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_r)} \end{aligned}$$

- We are done!
- How do we estimate $P(A_i = a_i \mid C = c_j)$? Easy!.

Classify a test instance

- If we only need a decision on the most probable class for the test instance, we only need the numerator as its denominator is the same for every class.
- Thus, given a test example, we compute the following to decide the most probable class for the test instance

$$c = \arg \max_{c_j} \Pr(c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

An example

- Compute all probabilities required for classification

A	B	C
m	b	t
m	s	t
g	q	t
h	s	t
g	q	t
g	q	f
g	s	f
h	b	f
h	q	f
m	b	f

$$\Pr(C = t) = 1/2,$$

$$\Pr(C = f) = 1/2$$

$$\Pr(A = m \mid C = t) = 2/5$$

$$\Pr(A = g \mid C = t) = 2/5$$

$$\Pr(A = h \mid C = t) = 1/5$$

$$\Pr(A = m \mid C = f) = 1/5$$

$$\Pr(A = g \mid C = f) = 2/5$$

$$\Pr(A = h \mid C = f) = 2/5$$

$$\Pr(B = b \mid C = t) = 1/5$$

$$\Pr(B = s \mid C = t) = 2/5$$

$$\Pr(B = q \mid C = t) = 2/5$$

$$\Pr(B = b \mid C = f) = 2/5$$

$$\Pr(B = s \mid C = f) = 1/5$$

$$\Pr(B = q \mid C = f) = 2/5$$

Now we have a test example:

$$A = m \quad B = q \quad C = ?$$

An Example (cont ...)

- For $C = t$, we have

$$\Pr(C = t) \prod_{j=1}^2 \Pr(A_j = a_j | C = t) = \frac{1}{2} \times \frac{2}{5} \times \frac{2}{5} = \frac{2}{25}$$

- For class $C = f$, we have

$$\Pr(C = f) \prod_{j=1}^2 \Pr(A_j = a_j | C = f) = \frac{1}{2} \times \frac{1}{5} \times \frac{2}{5} = \frac{1}{25}$$

- $C = t$ is more probable. t is the final class.

Additional issues

- **Numeric attributes:** Naïve Bayesian learning assumes that all attributes are categorical. Numeric attributes need to be discretized.
- **Zero counts:** An particular attribute value never occurs together with a class in the training set. We need smoothing.

$$\Pr(A_i = a_i | C = c_j) = \frac{n_{ij} + \lambda}{n_j + \lambda n_i}$$

- **Missing values:** Ignored

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 (e.g., in highly correlated data sets)