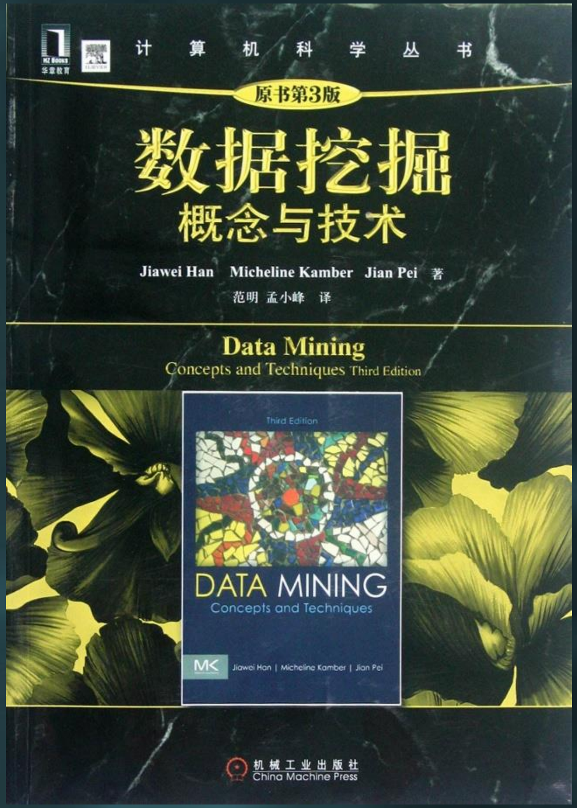
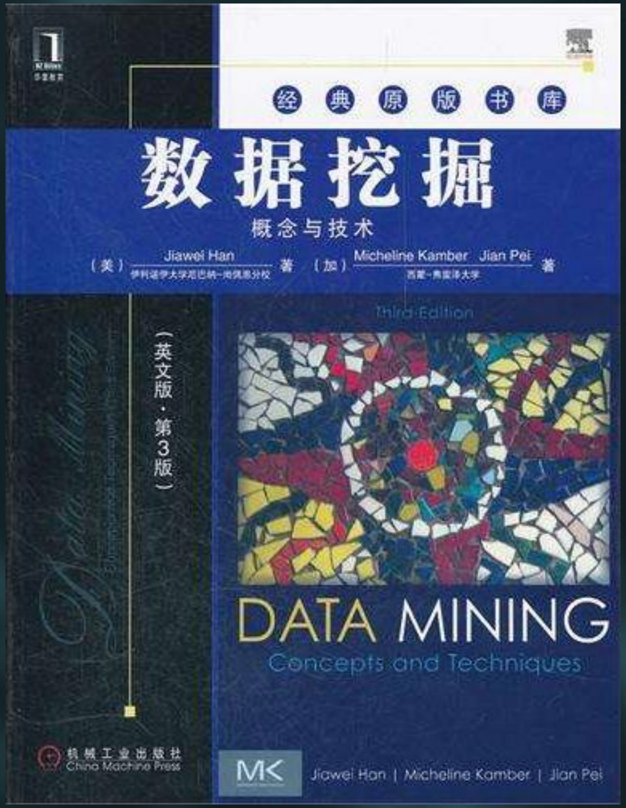
Technology
Science
Information
Networks
Computing

TSINC

Lecturer: Ting Wang (王挺)

利物浦大学计算机博士
清华大学计算机博士后
电子信息技术高级工程师
上海外国语大学网络与新媒体副教授
浙江清华长三角研究院海纳认知与智能研究中心主任





Chapter 8

Classification: Basic Concepts

Chapter 8: Classification

1. What is classification

a form of data analysis that extracts models describing data classes.

Steps：

1. *Learning*: Training data are analyzed by a classification algorithm.
2. *Classification*: Test data are used to estimate the accuracy of the classification rules

2. Supervise learning（监督学习） and unsupervised learning（无监督学习）

supervise learning：

- the learning of the classifier is “supervised” in that it is told to which class each training tuple belongs

unsupervised learning（clustering）：

- the class label of each training tuple is not known, and the number or set of classes to be learned may not be known in advance

Chapter 8

3. Decision Tree Induction

- Basic algorithm (a greedy algorithm)

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

Ref:

https://blog.csdn.net/neilgy/article/details/82746270

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset*.

Output: A decision tree.

Method:

(1) create a node *N*;
(2) **if** tuples in *D* are all of the same class, *C*, **then**
(3)   return *N* as a leaf node labeled with the class *C*;
(4) **if** *attribute_list* is empty **then**
(5)   return *N* as a leaf node labeled with the majority class in *D*; // majority voting
(6) apply **Attribute_selection_method**(*D*, *attribute_list*) to **find** the "best" *splitting_criterion*;
(7) label node *N* with *splitting_criterion*;
(8) **if** *splitting_attribute* is discrete-valued **and**
    multiway splits allowed **then** // not restricted to binary trees
(9)   *attribute_list* ← *attribute_list* − *splitting_attribute*; // remove *splitting_attribute*
(10) **for each** outcome *j* of *splitting_criterion*
    // partition the tuples and grow subtrees for each partition
(11)   let D_j be the set of data tuples in *D* satisfying outcome *j*; // a partition
(12)   **if** D_j is empty **then**
(13)   attach a leaf labeled with the majority class in *D* to node *N*;
(14)   **else** attach the node returned by **Generate_decision_tree**(D_j, *attribute_list*) to node *N*;
    endfor
(15) return *N*;

Basic algorithm for inducing a decision tree from training tuples.

Chapter 8: Classification

4. Information Gain

$$Gain(A) = Info(D) - Info_A(D). \quad (8.3)$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j). \quad (8.2)$$

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i), \quad (8.1)$$

Chapter 8: Classification

5. Naïve Bayes Classifier

① 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** = $(x_1, x_2, \ldots, x_n)$

② 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

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

③ 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

Chapter 8: Classification

6. Rule-based classification与decision tree

Ref:

https://blog.csdn.net/weixin_42555080/article/details/91401493

7. Rule Generation

sequential covering algorithm

Algorithm: Sequential covering. Learn a set of IF-THEN rules for classification.

Input:

- *D*, a data set of class-labeled tuples;
- *Att_vals*, the set of all attributes and their possible values.

Output: A set of IF-THEN rules.

Method:

```
Rule_set = {}; // initial set of rules learned is empty
for each class c do
    repeat
        Rule = Learn_One_Rule(D, Att_vals, c);
        remove tuples covered by Rule from D;
        Rule_set = Rule_set + Rule; // add new rule to rule set
    until terminating condition;
endfor
return Rule_Set;
```

Basic sequential covering algorithm.

Chapter 8: Classification

8. Classification model evaluation

Measure	*Formula*
accuracy, recognition rate	$\frac{TP+TN}{P+N}$
error rate, misclassification rate	$\frac{FP+FN}{P+N}$
sensitivity, true positive rate, recall	$\frac{TP}{P}$
specificity, true negative rate	$\frac{TN}{N}$
precision	$\frac{TP}{TP+FP}$
F, F_1, F-score, harmonic mean of precision and recall	$\frac{2 \times precision \times recall}{precision + recall}$
F_β, where β is a non-negative real number	$\frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$

Evaluation measures. Note that some measures are known by more than one name. TP, TN, FP, P, N refer to the number of true positive, true negative, false positive, positive, and negative samples, respectively (see text).

Chapter 8: Classification

9. Bagging与Boosting

- Bagging: averaging the prediction over a collection of classifiers
- Boosting: weighted vote with a collection of classifiers

区别:

1）样本选择上:

Bagging：训练集是在原始集中有放回选取的，从原始集中选出的各轮训练集之间是独立的。

Boosting：每一轮的训练集不变，只是训练集中每个样例在分类器中的权重发生变化。而权值是根据上一轮的分类结果进行调整。

2）样例权重:

Bagging：使用均匀取样，每个样例的权重相等

Boosting：根据错误率不断调整样例的权值，错误率越大则权重越大。

3）预测函数:

Bagging：所有预测函数的权重相等。

Boosting：每个弱分类器都有相应的权重，对于分类误差小的分类器会有更大的权重。

4）并行计算:

Bagging：各个预测函数可以并行生成

Boosting：各个预测函数只能顺序生成，因为后一个模型参数需要前一轮模型的结果。

5）bagging是减少variance，而boosting是减少bias

REF: https://www.cnblogs.com/earendil/p/8872001.html

Chapter 8: Classification

10. Adaboost

11. Random forest

Algorithm: AdaBoost. A boosting algorithm—create an ensemble of classifiers. Each one gives a weighted vote.

Input:

- D, a set of d class-labeled training tuples;
- k, the number of rounds (one classifier is generated per round);
- a classification learning scheme.

Output: A composite model.

Method:

```
(1)  initialize the weight of each tuple in D to 1/d;
(2)  for i = 1 to k do // for each round:
(3)      sample D with replacement according to the tuple weights to obtain D_i;
(4)      use training set D_i to derive a model, M_i;
(5)      compute error(M_i), the error rate of M_i (Eq. 8.34)
(6)      if error(M_i) > 0.5 then
(7)          go back to step 3 and try again;
(8)      endif
(9)      for each tuple in D_i that was correctly classified do
(10)         multiply the weight of the tuple by error(M_i)/(1 − error(M_i)); // update weights
(11)     normalize the weight of each tuple;
(12) endfor
```

To use the ensemble to classify tuple, X:

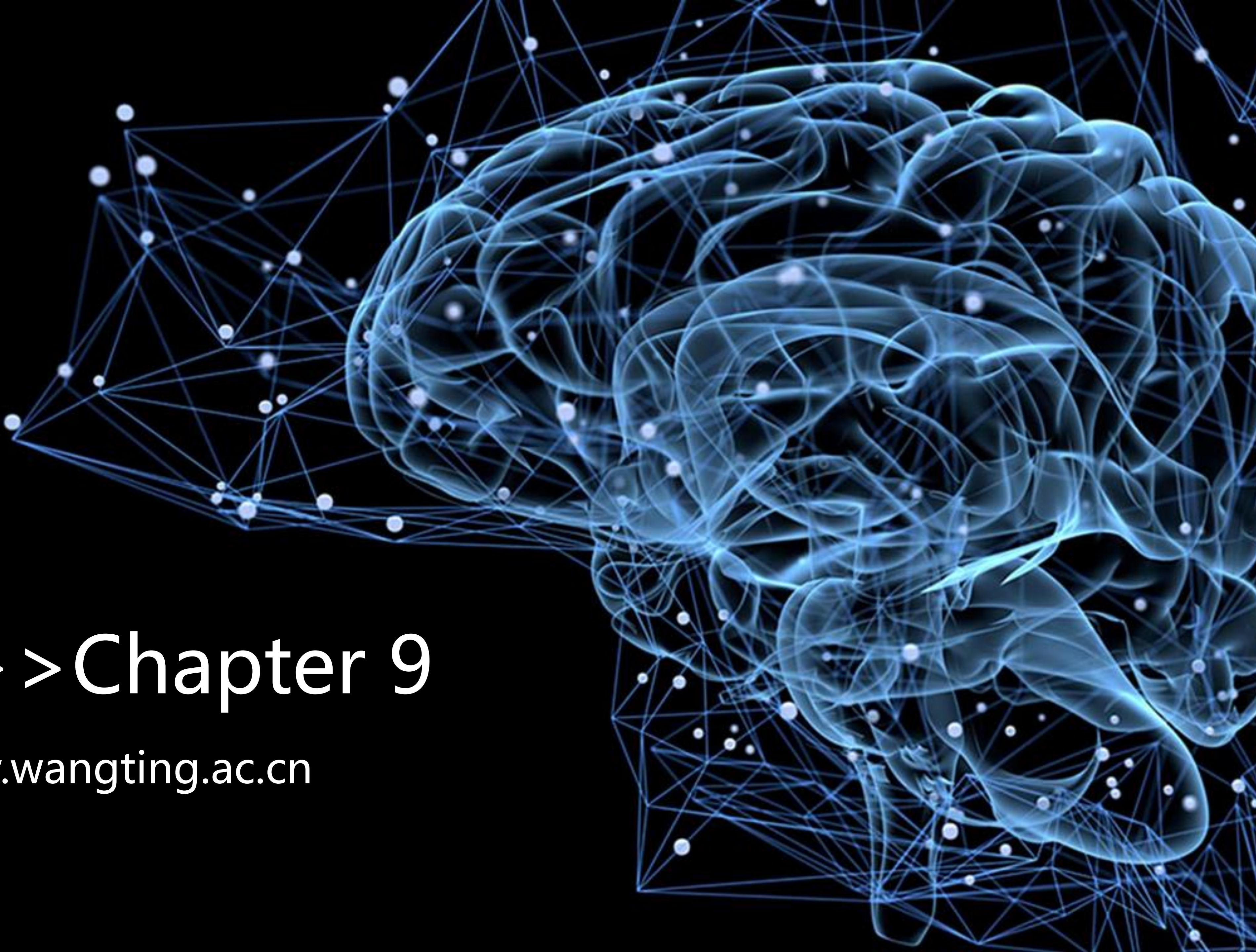
```
(1)  initialize weight of each class to 0;
(2)  for i = 1 to k do // for each classifier:
(3)      w_i = log((1 − error(M_i)) / error(M_i)); // weight of the classifier's vote
(4)      c = M_i(X); // get class prediction for X from M_i
(5)      add w_i to weight for class c
(6)  endfor
(7)  return the class with the largest weight;
```

AdaBoost, a boosting algorithm.

Chapter 8: Classification

12. Imbalanced Data

- (1) oversampling,
- (2) undersampling,
- (3) threshold moving,
- (4) ensemble techniques



Next>>Chapter 9

www.wangting.ac.cn