BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION SYSTEMS

Selected Topics of Information Systems: Data Classification

Semester project

2007 **Ing. Michael Kunc**

Abstract

This paper deals with techniques and algorithms of data classification. It describes the process of preparing the data and main ideas of four well-known algorithms – Decision tree induction, Bayesian classification, Backpropagation and Association rule mining.

Keywords

Data Classification, Decision Tree Induction, Bayesian Classification, Backpropagation, Association Rule Mining.

Acknowledgements

I thank to prof. Ing. Tomas Hruska, CSc., doc. Ing. Jaroslav Zendulka, CSc. and Ing. Radek Burget, Ph.D. for provided advices and expert assistance.

Contents

Chapter 1

Introduction

Data classification is a two-step process. In the first step, learning, training data are analyzed by a classification algorithm. In the second step, classification, test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data tuples.

In this work there will be described methods and algorithms of classification.

Chapter 2 describes the whole process of preparing the data. It concludes data cleaning, relevance analysis and data transformation. Problems connected with preparing the data are mentioned too. Next chapters of this work describe well-known algorithms and techniques of data classification: Decision tree induction, Bayesian classification, Backpropagation and Association rule mining.

Chapter 2

Preparing the Data

For improving the accuracy, efficiency and scalability of the classification process, there should be applied some preprocessing steps to the data: Data cleaning, relevance analysis and data transformation.

2.1 Data Cleaning

Data cleaning, also called scrubbing, deals with detecting and removing errors and inconsistencies from data in order to improve the quality of data. Problems with quality of data should be in single data collections, such as files and databases (e.g. missing information). The need for data cleaning increases significantly when we want to integrate multiple data sources, such as in data warehouses, global web-based information systems or federated databases. The sources often contain redundant data in different representations. In this cases, consolidation of different data representations and elimination of duplicate information is necessary.

Data warehouses load and continuously refresh huge amounts of data from variety of sources. In this case, there is very high probability of some sources containing dirty data. In data warehousing, data cleaning is one of the biggest problems. In Figure 2.1, there are shown steps of building a data warehouse – the ETL process (extraction, transformation, loading). Data cleaning is typically performed in a separate data staging area before loading the transformed data into the warehouse.

Web-based information systems and federated database systems have similar transformation steps to data warehouses. There is typically wrapper for extraction for each data source and mediator for integration. These systems provide only limited support for data cleaning. Data is not preintegrated as for data warehouses but needs to be extracted from multiple sources, transformed and combined during query runtime. It is difficult to achieve response times because of corresponding communication and processing delays.

2.2 Relevance Analysis

Some attributes in the data should be irrelevant or redundant to the classification task. Relevance analysis may be performed on the data with the aim of removing any irrelevant or redundant attributes from the learning process. This analysis can help improve classification efficiency and scalability.

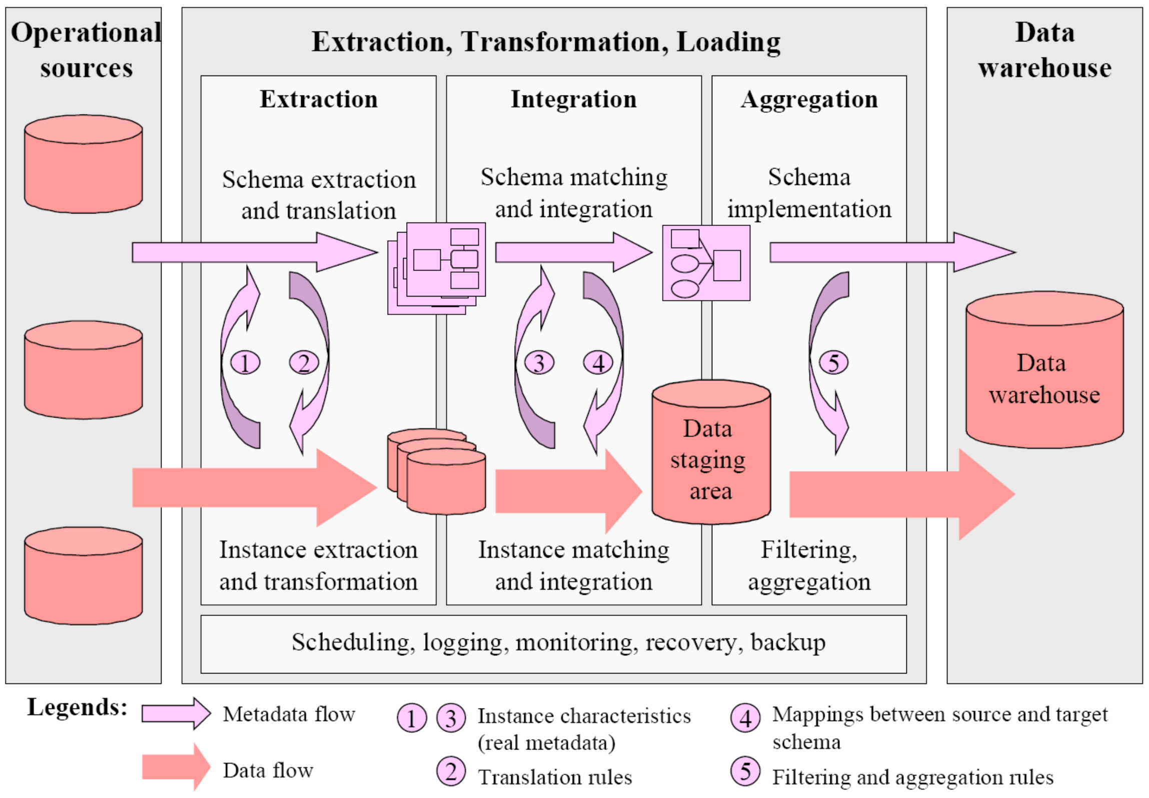
Operational sources
Extraction, Transformation, Loading
Extraction
Integration
Aggregation
Data warehouse
Schema extraction and translation
Schema matching and integration
Schema implementation
Data staging area
Data warehouse
Instance extraction and transformation
Instance matching and integration
Filtering, aggregation
Scheduling, logging, monitoring, recovery, backup
Legends:
Metadata flow
Data flow
1 3 Instance characteristics (real metadata)
2 Translation rules
4 Mappings between source and target schema
5 Filtering and aggregation rules

Figure 2.1: Steps of building a data warehouse: The ETL process (took over from [2])

2.3 Data Transformation

The data can be generalized to higher-level concepts. For this purpose, there should be used concept hierarchies. It is useful for continuous-valued attributes, e.g. numeric values for the attribute income may be generalized to discrete ranges such as low, medium and high. The data may be also normalized. Normalization involves scaling all values for a given attribute so that they fall within a small specific range, e.g. 0.0 to 1.0.

2.4 Problems connected with preparing the data

Raw data are represented by two common types: Numeric and categorical. A feature with numeric values as two important properties: Order relation (e.g. $2 < 5$ and $5 < 7$) and distance relation (e.g. $d(2.3, 4.2) = 1.9$. Categorical are often called symbolic. Categorical variables have neither of these two relations: The two values of a categorical variable can be either equal or not equal, and a categorical variable with N values can be converted into N binary numeric variables.

Another way of variable classification are continuous variables (quantitative or metric variables) and discrete variables (qualitative variables). Continuous variables are measured using either interval scale (the zero point in the interval scale is placed arbitrarily) and ratio scale (it has an absolute zero point). Discrete variables use one of two kinds of non-metric scales: Nominal scale (an orderless scale) and ordinal scale (consists of ordered discrete gradations).

Periodic variable is a special class of discrete variable. In this class exists the distance relation, but there is no order relation. It should be days of the week, month or year.

Missing data, misrecorded data, data from the other population and distorted data are some possible characteristics of the raw data. Critical steps in data mining are preparation and transformation of the initial data set. There are two central tasks for the preparation of data: To organize data into some standard form and to prepare data sets that lead to the best data mining performances.

Techniques useful for raw data transformation are for example normalization and data smoothing. There are three main types of normalization:

- Decimal scaling
  - $v'(i) = \frac{v(i)}{10^k}$
  - for the smallest k such that $max(|v'(i)|) < 1$
- Min-max normalization
  - $v'(i) = \frac{v(i) - min(v(i))}{max(v(i)) - min(v(i))}$
  - for normalized interval $[0,1]$
- Standard deviation normalization
  - $v'(i) = \frac{v(i) - mean(v)}{sd(v)}$

Data smoothing will be demonstrated on this example: $F = 0.93, 1.01, 1.001, 3.02, 2.99, 5.03, 5.01, 4.98$ $F_{smoothed} = 1.0, 1.0, 1.0, 3.0, 3.0, 5.0, 5.0, 5.0$

When there are missing data, we can manually examine samples with missing data values or we can use automatic replacement:

- Replace all missing values with a single global constant (a selection of a global constant is highly application dependent).
- Replace a missing value with its feature mean.
- Replace a missing value with its feature mean for the given class (only a classification problems).

Chapter 3

Decision Tree Induction

3.1 The Main Idea

This chapter will demonstrate decision tree classifier. Simple example with data shown in table 3.1 will be used for demonstration. The vertebrates will be classified into two distinct groups: Mammals and non-mammals. A series of questions is one approach how to know if the animal is a mammal or a non-mammal. If it is cold-blooded, then it is not a mammal. Otherwise, it is either a bird or a mammal. Those that do give birth are mammals, while those that do not are likely to be non-mammals.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal

Table 3.1: Sample data (took over from [4])

The example shows how to solve a classification problem by asking a series of questions. Each question asks for one attribute of the test record. Each time we receive an answer, a follow-up question is asked until we reach a conclusion about the class label of the record. A decision tree (Figure 3.1) is a special form, where should be the series of question and their answers organized.

There are three types of nodes of the decision tree:

1. Root node
   - has no incoming edges and zero or more outgoing edges
2. Internal nodes
   - each of which has exactly one incoming edge and two or more outgoing edges
3. Leaf nodes

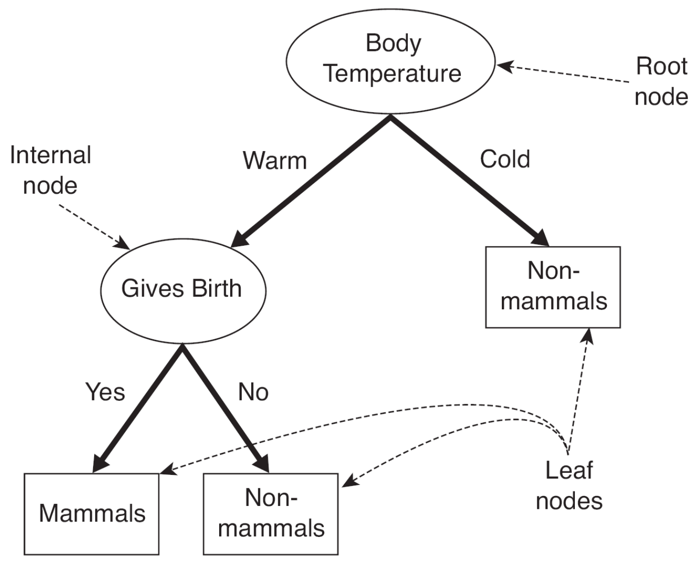

Figure 3.1: A decision tree for the mammal classification problem (took over from [4])

- each of which has exactly one incoming edge and no outgoing edges

Each leaf node is assigned a class label. The non-terminal nodes (root node or internal nodes) contain attribute test conditions to separate records that have different characteristics. In Figure 3.1, the root node uses the attribute *Body Temperature* to separate warm-blooded from cold-blooded vertebrates. A leaf node labeled *Non-mammals* is created as the right child of the root node, because all cold-blooded vertebrates are non-mammals. An attribute *Gives birth* is used if the vertebrate is warm-blooded to distinguish mammals.

Starting from the root node, we apply the test condition to the record and follow the appropriate branch based on the outcome of the test. This comes to an another internal node or to a leaf node. The associated class label is than assigned to the record. A Figure 3.2 shows the path in the decision tree that is used to predict the class label of a flamingo.

3.2 Building a Decision Tree

One of the well used algorithms is Hunt's algorithm, which is the basis of many existing decision tree induction algorithms including ID3, C4.5 and CART. In Hunt's algorithm, a decision tree is grown in a recursive way by partitioning the training records into subsets.

Let D_t be the set of training records associated with node t and $y = y_1, y_2, ..., y_c$ be the class labels. The algorithm should be divided into two steps:

1. If all records in D_t belong to the same class y_t, then t is leaf node labeled as y_t.
2. If D_t contains records that belong to more than one class, an attribute test condition is selected to partition the record into smaller subsets. A child node is created for each outcome of the test condition and the records in D_t are distributed to the children based on the outcomes. The algorithm is then recursively applied to each child node.

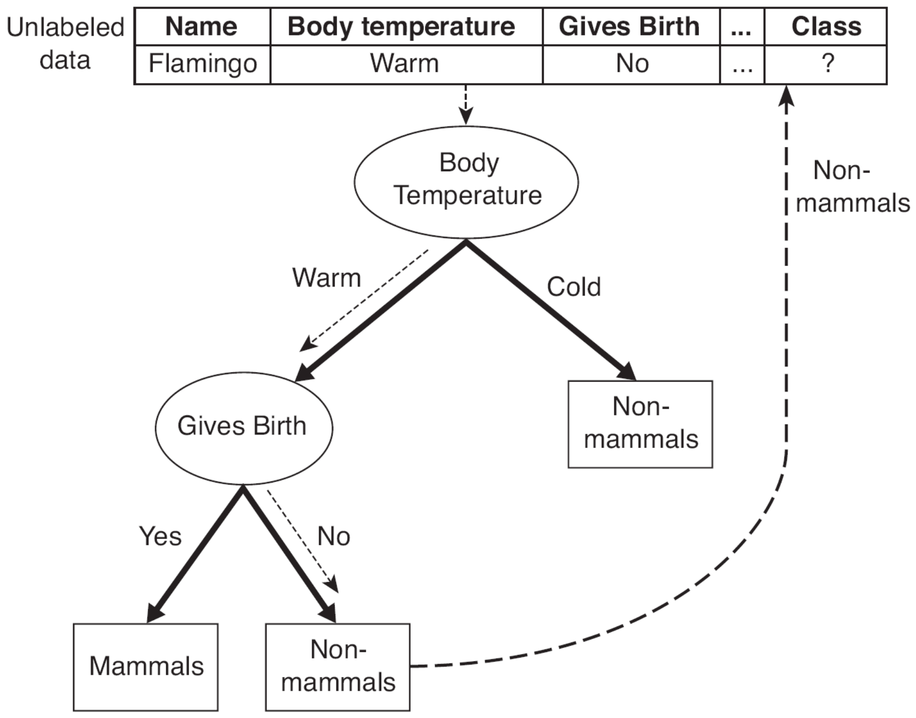

Figure 3.2: The path in the decision tree that is used to predict the class label of a flamingo (took over from [4])

3.3 Decision Tree Induction Summary

Decision tree induction is a nonparametric approach for building classification models. It doesn't require any aprior assumptions regarding the type of probability distributions satisfied by the class and other attributes. Finding an optimal decision tree is an NP-complete problem. Many decision tree algorithms employ a heuristic-based approach to guide their search in the vast hypothesis space. Constructing decision tree techniques are computationally inexpensive, making it possible to quickly construct models even when the training set size is very large. Once a decision tree has been built, classifying a test record is extremely fast, with a worst-case complexity of $O(w)$, where w is the maximum of the tree depth.

Chapter 4

Bayesian Classification

Bayesian classifiers are statistical classifiers that can predict class membership probabilities, such as the probability that some sample belongs to a particular class. This classification is based on Bayes theorem, it has high accuracy and speed when applied to large databases.

4.1 Bayes Theorem

Let X be a data sample whose class label is unknown and H some hypothesis, that the X belongs to class C. The probability that H holds given the X is $P(H|X)$. $P(H|X)$ is the posterior probability of H conditioned on X. $P(H)$ is the prior probability of H. $P(X|H)$ is the posterior probability of X conditioned on H. $P(X)$ is the prior probability of X.

Bayes theorem calculates the posterior probability $P(H|X)$ from $P(H)$, $P(X)$ and $P(X|H)$. Bayes theorem is:

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

4.2 Naive Bayesian Classification

Naive Bayessian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This is called *class conditional independence*. Simplifying the computations is the main reason why this technique is used.

The naive Bayessian classification algorithm work should be divided into five steps:

1. Each data sample is represented by an n-dimensional feature vector $X = (x_1, x_2, ..., x_n)$ where are n measurements made on sample from n attributes.
2. Suppose that there are m classes $C_1, C_2, ..., C_m$. Given an unknown data sample X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. The naive Bayessian classifier assigns an unknown sample X to the class C_i iff

$$P(C_i|X) > P(C_j|X) \text{ for } 1 \leq j \leq m, j \neq i.$$

Then we maximize $P(C_i|X)$:

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

3. As $P(X)$ is constant for all classes, only $P(X|C_i)P(C_i)$ need be maximized. The class prior probabilities may be estimated by $P(C_i) = \frac{s_i}{s}$, where s_i is the number of training samples of class C_i and s is the total number of training samples.

4. Because of computation reduction, the naive assumption of class condition independence is made. There are no dependence relationships among the attributes.

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i).$$

The probabilities $P(x_1|C_i), P(x_2|C_i), ..., P(x_n|C_i)$ are evaluated from the training set:

- If A_k is categorical, then:

$$P(x_k|C_i) = \frac{s_{ik}}{s_i},$$

where s_{ik} is the number of training samples of class C_i having the value x_k for A_k and s_i is the number of training samples belonging to C_i.

- If A_k is continuous-valued then the attribute is assumed to have a Gaussian distribution:

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-\frac{(x_k - \mu_{C_i})^2}{2\sigma_{C_i}^2}},$$

where $g(x_k, \mu_{C_i}, \sigma_{C_i})$ is the Gaussian density function for attribute A_k while μ_{C_i} and $\sigma_{C_i})$ are the mean and standard deviation.

5. $P(X|C_i)P(C_i)$ is evaluated for each class C_i. Sample X is then assigned to the class C_i iff

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \text{ for } 1 \leq j \leq m, j \neq i.$$

4.3 Bayesian Belief Networks

Bayesian belief networks are graphical models, which unlike naive Bayesian classifiers allow the representation of dependencies among subsets of attributes. Bayessian belief networks can also be used for classification. Bayesian belief networks specify joint conditional probability distributions. They allow class conditional independencies to be defined between subsets of variables.

A belief network is defined by two components:

- Directed acyclic graph

  - Each node represents a random variable and each arc represents a probabilistic dependence. Y is a parent of Z and Z is descendent of Y, if an arc is drawn from a node Y to node Z. Each variable is conditionally independent of its parents.

- Conditional probability table (CPT)

  - The CPT for variable Z specifies the conditional distribution $P(Z|Parents(Z))$.

The joint probability of any tuple $(z_1, z_2, ..., z_n)$ corresponding to the variables or attributes $Z_1, Z_2, ..., Z_n$ is computed by:

$$P(z_1, z_2, ..., z_n) = \prod_{i=1}^{n} P(z_i|Parents(Z_i)),$$

where the values for $P(z_i|Parents(Z_i))$ correspond to the entries in the CPT for Z_i.

4.3.1 Training Process

Training the network is straightforward only if the network structure is known and the variables are observable. Then it's computing CPT entries similarly as in naive Bayesian classification.

If some of the variables are hidden, a method of gradient descent can be used to train the belief network. The main idea is to learn the values for CPT entries. Let S be a set of s training samples, $X_1, X_2, ..., X_s$. Let w_{ijk} be a CPT entry for the variable $Y_i = y_{ij}$ having the parents $U_i = u_{ik}$. The w_{ijk} are viewed as weights. The set of weights is collectively referred to as w. The weights are initialized to random probability values. The method searches for the w_{ijk} values that best model the data. The goal is to maximize $P_w(S) = \prod_{d=1}^{s} P_w(X_d)$. It is done by following the gradient of $\ln P_w(S)$. There are three steps of the algorithm:

1. Compute the gradients

$$\frac{\delta \ln P_w(S)}{\delta w_{ijk}} = \sum_{d=1}^{s} \frac{P(Y_i = y_{ij}, U_i = u_{ik}|X_d)}{w_{ijk}}$$

2. Take a small step in the direction of the gradient

$$w_{ijk} \leftarrow w_{ijk} + (l)\frac{\delta \ln P_w(S)}{\delta w_{ijk}},$$

   where l is the learning rate representing the step size.

3. Renormalize the weights, because the weights w_{ijk} must be between 0.0 and 1.0 and $\sum_j w_{ijk}$ must equal 1 for all i, k.

Chapter 5

Backpropagation

Backpropagation is a neural network learning algorithm. A neural network is a set of connected input/output devices, where each connection has a weight. The network learns by adjusting the weights to be able to predict the correct class label of the input samples.

The biggest advantages of neural networks are high tolerance to noisy data and their ability to classify patterns on which they have not been trained. These networks but involve long training times and they require a number of parameters that are typically determined empirically.

5.1 A Multilayer Feed-Forward Neural Network

A multilayer feed-forward neural network is the type of neural network on which the backpropagation algorithm performs. The inputs correspond to the attributes measured for each training sample. The inputs are in the input layer. The weighted outputs of these units are fed simultaneously to a hidden layer. The hidden layer's weighted outputs can be input to another hidden layer etc. Usually is used only one hidden layer. The output layer, which emits the network's prediction for given samples, is made from last hidden layer output.

The network is feed-forward, because none of the weights cycles back to an input init or to an output unit of a previous layer. Fully connected mean, that each unit provides input to each unit in the next forward layer.

5.2 Defining a Network Topology

The user must specify the number of units in the input layer, the number of hidden layers, the number of units in each hidden layer, and the number of units in the output layer. Input values are normalized between 0.0 and 1.0 because of speed in learning phase. Once a network has been trained and its accuracy is not considered acceptable, it is common to repeat the training process with a different network topology or different set of initial weights.

5.3 Backpropagation

Backpropagation learns by iteratively processing a set of training samples. It compares the network's prediction for each sample with the actual known class label.

The weights are modified to minimize the mean squared error between the network's prediction and the actual class for each training sample. These modifications are made from the output layer through each hidden layer down to the first hidden layer. The algorithm will be described in four steps:

1. Initialize the weights

   The weights and the biases are initialized to small random numbers in interval -1.0 to 1.0.

2. Propagate the inputs forward

   The net input and output of each unit in the hidden and output layers are computed. The training sample is fed to the input layer of the network. For unit j in the input layer its output is equal to its input for input unit j, $O_j = I_j$. The net input to each unit in the hidden and output layers is computed as a linear combination of its inputs. To compute the net input to the unit, each input connected to the unit is multiplied by its corresponding weight, and this is summed:

   $$I_j = \sum_i w_{ij} O_i + \theta_j,$$

   where w_{ij} is the weight of the connection from init i in the previous layer to unit j. O_i is the output of unit i from the previous layer. θ_j is the bias of the unit.

   Each unit in the hidden and output layers takes its net input and then applies an activation function to it. The function symbolizes the activation of the neuron represented by the unit.

   $$O_j = \frac{1}{1 + e^{-I_j}}.$$

3. Backpropagate the error

   The error is propagated backwards by updating the weights and biases to reflect the error of the network's prediction. For a unit j in the output layer:

   $$Err_j = O_j(1 - O_j)(T_j - O_j),$$

   where O_j is the actual output of unit j, T_j is the true output and $O_j(1 - O_j)$ is the derivative of the logistic function. The error of a hidden layer unit j is:

   $$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk},$$

   where w_{jk} is the weight of the connection from unit j to a unit k in the next higher layer, Err_k is the error of unit k.

   The weights and biases are updated to reflect the propagated errors:

   $$\Delta w_{ij} = (l) Err_j O_i$$

$$w_{ij} = w_{ij} + \Delta w_{ij},$$

where Δw_{ij} is the change in weight w_{ij} and l is the learning rate. A constant l has typically a value between 0.0 and 1.0. A rule of thumb is to set the learning rate to $1/t$, where t is the number of iterations through the training set so far. Biases are updated by following equations:

$$\Delta\theta_j = (l)Err_j$$

$$\theta_j = \theta_j + \Delta\theta_j,$$

where $\Delta\theta_j$ is the change in bias θ_j.

4. Terminating condition

   Training stops when:

   - all Δw_{ij} in the previous epoch were so small as to below some specified treshold, or
   - the percentage of samples misclassified in the previous epoch is below some treshold, or
   - a prespecified number of epochs has expired.

Chapter 6

Association Rule Mining

6.1 Association Rule Problem

Let $I = I_1, I_2, ..., I_m$ be a set of m distinct attributes, T be transaction that contains a set of items such that $T \subseteq I$, D be a database with different transaction records T_s. An association rule is an implication in the form of $X \Rightarrow Y$, where $X, Y \subset I$ are sets of items called itemsets, and $X \cap Y = \emptyset$. X is called antecedent while Y is called consequent, the rule means X implies Y. The two basic parameters of association rule mining (ARM) are: Support and conficence.

Support(s) of an association rule is defined as the percentage/fraction of records that contain $X \cup Y$ to the total number of records in the database.

$$Support(XY) = \frac{Support\ count\ of\ XY}{Total\ number\ of\ transactions\ in\ D}$$

Confidence of an association rule is defined as the percentage/fraction of the number of transactions that contain $X \cup Y$ to the total number of records that contain X, where if the percentage exceeds the treshold of confidence an interesting association rule $X \Rightarrow Y$ can be generated.

$$Confidence(X|Y) = \frac{Support(XY)}{Support(X)}$$

ARM find out association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two subproblems. One is to find those itemsets whose occurrences exceed a predefined treshold in the database, those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with constraints of minimal confidence.

6.2 Association Rule Mining Approaches

6.2.1 AIS Algorithm

The AIS (Agrawal, Imienlinski, Swami) algorithm was the first algorithm proposed for mining association rule. It focus on improving the quality of databases together with necessary functionality to process decision support queries. In this algorithm, only one item consequent association rules are generated. This means that the consequent of those rules contain one item only.

The main drawback of the AIS algorithm is too many candidate itemsets that finally turned out to be small are generated, which requires more space and wastes much effort that turned out to be useless. At the same time this algorithm requires too many passes over the whole database.

6.2.2 Apriori Algorithm

Apriori is a great improvement in the history of association rule mining. The AIS is just a straightforward approach that requires many passes over the database, generating many candidate itemsets and storing counters of each candidate while most of them turn out to be not frequent. Apriori employs a different candidates generation method and a new pruning technique, so this algorithm is more effective.

Apriori algorithm still inherits the drawback of scanning the whole databases many times. Many new algorithms were developed with some modifications or improvements of apriori algorithm.

6.2.3 Frequent Pattern Tree (FP-Tree) Algorithm

FP-Tree breaks the two bottlenecks of the apriori algorithm. The frequent itemsets are generated with only two passes over the database and without any candidate generation process. By avoiding the candidate generation process and less passes over the database, FP-Tree is an order of magnitude faster than the apriori algorithm. The frequent patterns generation process includes two subprocesses: Constructing the FP-Tree and generating frequent patterns from the FP-Tree.

FP-Tree is difficult to use in an interactive mining system. During the interactive mining process, users may change the treshold of support according to the rules. However for FP-Tree the changing of support may lead to repetition of the whole mining process. Another limitation is that FP-Tree is that it is not suitable for incremental mining. Since as time goes on databases keep changing, new datasets may be inserted into the database, those insertions may also lead to a repetition of the whole process if we employ FP-Tree algorithm.

6.2.4 Rapid Association Rule Mining (RARM)

RARM is another association rule mining method that uses the tree structure to represent the original database and avoids candidate generation process. RARM is claimed to be much faster than FP-Tree algorithm. By using SOTrieIT structure, RARM can generate large 1-itemsets and 2-itemsets quickly without scanning the database for the second time and candidates generation. Similarly to FP-Tree, every node of the SOTrieIT contains one item and the corresponding support count.

The efficiency of generating large 1-itemsets and 2-itemsets in the SOTrieIT algorithm improves the performance dramatically, SOTrieIT is much faster than FP-Tree, but has similar problems as FP-Tree.

Chapter 7

Conclusion

Classification is the form of data analysis that can be used to extract models describing important data classes. Preprocessing of the data in preparation for classification can involve data cleaning to reduce noise, relevance analysis to remove irrelevant attributes and data transformation like generalizing the data to higher level concepts.

Decision tree induction is nonparametric approach for building classification models. Naive Bayesian classification and Bayesian belief networks are based on Bayes theorem of posterior probability. Backpropagation is a neural network algorithm for classification that employs a method of gradient descent. Association mining techniques search for frequently occuring patterns in large databases.

My next steps will be in finding and testing tools for data classification. My main goal is to find the best classification technique for classifying segmented web-pages.

Bibliography

[1] Han, J., Kamber, M.: Data Mining: Concepts and Techniques, Second edition. Morgan Kaufmann Publishers, 2006.

[2] Rahm, E., Do, E. R.: Data Cleaning: Problems and Current Approaches. Available on WWW: http://homepages.inf.ed.ac.uk/wenfei/tdd/reading/cleaning.pdf (July 2007).

[3] Liou D.-M.: Preparing the Data. Lectures to Data Mining Technique course, National Yang-Ming University. Available on WWW: http://www.ym.edu.tw/~dmliou/course/dm05/data_ming_ch2.pdf (July 2007).

[4] Kumar, V.: Classification: Basic Concepts, Decision Trees, and Model Evaluation. Available on WWW: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf (July 2007).

[5] Mitchell, T. M.: Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression. Available on WWW: http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf (July 2007).

[6] Kamber, M., Winstone, L., Gong, W., Cheng, S., Han, J.: Generalization and Decision Tree Induction: Efficient Classification in Data Mining. Available on WWW: http://www-sal.cs.uiuc.edu/~hanj/pdf/ride97.pdf (July 2007).

[7] Zhao, Q., Bhowmick, S. S.: Association Rule Mining: A Survey. Available on WWW: http://www.cse.unsw.edu.au/~cs9318/readings/ARMining-Survey2.pdf (July 2007).

[8] Liu, B., Hsu, W., Ma, Y.: Integrating Classification and Association Rule Mining. Available on WWW: http://www.dsi.unive.it/~dm/liu98integrating.pdf (July 2007).

[9] Tan, Steibach, Kumar: Data Mining – Classification: Alternative Techniques. Available on WWW: http://www.gersteinlab.org/courses/545/07-spr/reading/tan-2005-chpt05.pdf (July 2007).