Data Mining Methods and Applications

38

Kwok-Leung Tsui, Victoria Chen, Wei Jiang, Fangfang Yang, and Chen Kan

Contents

Abstract

In this chapter, we provide a review of the knowledge discovery process, including data handling, data mining methods and software, and current research activities. The introduction defines and provides a general background to data mining knowledge discovery in databases, following by an outline of the entire process in the second part. The third part presents data handling issues, including databases and preparation of the data for analysis. The fourth part, as the core of the chapter, describes popular data mining methods, separated as supervised versus unsupervised learning. Supervised learning methods are described in the context of both regression and classification, beginning with the simplest case of linear models, then presenting more complex modeling with trees, neural networks, and support vector machines, and concluding with some methods only for classification. Unsupervised learning methods are described under two categories: association rules and clustering. The fifth part presents past and current research projects, involving both industrial and business applications. Finally, the last part provides a brief discussion on remaining problems and future trends.

Keywords

Data mining · Linear discriminant analysis · Association rule · Unsupervised learning · Statistical process control

K.-L. Tsui (✉)
Grado Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA
e-mail: kltsui@vt.edu

V. Chen · C. Kan
Department of Industrial, Manufacturing, & Systems Engineering, University of Texas at Arlington, Arlington, TX, USA
e-mail: vchen@uta.edu; chen.kan@uta.edu

W. Jiang
Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai, China
e-mail: jiangwei@sjtu.edu.cn

F. Yang
School of Intelligent Systems Engineering Sun Yat-sen University, Guangdong, China
e-mail: yangff7@mail.sysu.edu.cn

Data mining (DM) is the process of exploration and analysis, by automatic or semiautomatic means, of large quantities of data to discover meaningful patterns and rules [1]. Statistical DM is exploratory data analysis with little or no human interaction using computationally feasible techniques, i.e., the attempt to find *unknown interesting* structure [2]. Knowledge discovery in databases (KDD) is a multidisciplinary research field for nontrivial extraction of implicit, previously unknown, and potentially useful knowledge from data [3]. Although some treat DM and KDD equivalently, they can be distinguished as follows. The KDD process employs DM methods (algorithms) to extract knowledge according to the specifications of measures and thresholds, using a database

H. Pham (ed.), *Springer Handbook of Engineering Statistics*, Springer Handbooks,
https://doi.org/10.1007/978-1-4471-7503-2_38

along with any necessary preprocessing or transformations. DM is a step in the KDD process consisting of particular algorithms (methods) that, under some acceptable objective, produce particular patterns or knowledge over the data. The two primary fields that develop DM methods are statistics and computer science. Statisticians support DM by mathematical theory and statistical methods while computer scientists develop computational algorithms and relevant software [4]. Prerequisites for DM include: (1) advanced computer technology (large CPU, parallel architecture, etc.) to allow fast access to large quantities of data and enable computationally intensive algorithms and statistical methods and (2) knowledge of the business or subject matter to formulate the important business questions and interpret the discovered knowledge.

With competition increasing, DM and KDD have become critical for companies to retain customers and ensure profitable growth. Although most companies are able to collect vast amounts of business data, they are often unable to leverage this data effectively to gain new knowledge and insights. DM is the process of applying sophisticated analytical and computational techniques to discover exploitable patterns in complex data. In many cases, the process of DM results in actionable knowledge and insights. Examples of DM applications include fraud detection, risk assessment, customer relationship management, cross-selling, insurance, banking, retail, etc.

While many of these applications involve customer relationship management in the service industry, a potentially fruitful area is performance improvement and cost reduction through DM in industrial and manufacturing systems. For example, in the fast-growing and highly competitive electronics industry, total revenue worldwide in 2003 was estimated to be $900 billion, and the growth rate is estimated at 8% per year (www.selectron.com). However, economies of scale, purchasing power, and global competition are making the business such that one must either be a big player or serve a niche market. Today, extremely short life cycles and constantly declining prices are pressuring the electronics industry to manufacture their products with high quality, high yield, and low production cost.

To be successful, industry will require improvements at all phases of manufacturing. Figure 38.1 illustrates the three primary phases: design, ramp-up, and production. In the production phase, maintenance of a high-performance level via improved system diagnosis is needed. In the ramp-up phase, reduction in new product development time is sought by achieving the required performance as quickly as possible. Market demands have been forcing reduced development time for new product and production system design. For example, in the computer industry, a product's life cycle has been shortened to 2–3 years recently, compared to a life cycle of 3–5 years a few years ago. As a result, there are a number of new concepts in the area of production systems,

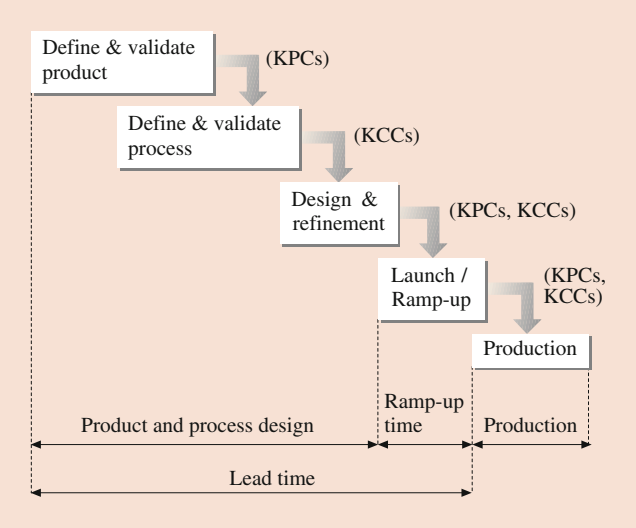

Fig. 38.1 Manufacturing system development phases. KPCs key product characteristics, KCCs key control characteristics

such as flexible and reconfigurable manufacturing systems. Thus, in the design phase, improved system performance integrated at both the ramp-up and production phases is desired. Some of the most critical factors and barriers in the competitive development of modern manufacturing systems lie in the largely uncharted area of predicting system performance during the design phase [5, 6]. Consequently, current systems necessitate that a large number of design/engineering changes be made after the system has been designed.

At all phases, system performance depends on many manufacturing process stages and hundreds or thousands of variables whose interactions are not well understood. For example, in the multistage printed circuit board (PCB) industry, the stages include process operations such as paste printing, chip placement, and wave soldering, and also include test operations such as optical inspection, vision inspection, and functional test. Due to advancements in information technology, sophisticated software and hardware technologies are available to record and process huge amounts of daily data in these process and testing stages. This makes it possible to extract important and useful information to improve process and product performance through DM and quality improvement technologies.

38.1 The KDD Process

The KDD process consists of four main steps:

1. Determination of business objectives
2. Data preparation
   (a) Create target datasets
   (b) Data quality, cleaning, and preprocessing
   (c) Data reduction and projection

3. Data mining
   (a) Identify DM tasks
   (b) Apply DM tools
4. Consolidation and application
   (a) Consolidate discovered knowledge
   (b) Implement in business decisions

As an example of formulating business objectives, consider a telecommunications company. It is critically important to identify those customer traits that retain profitable customers and predict fraudulent behavior, credit risks, and customer churn. This knowledge may be used to improve programs in target marketing, marketing channel management, micromarketing, and cross-selling. Finally, continually updating this knowledge will enable the company to meet the challenges of new product development effectively in the future. Steps 2–4 are illustrated in Figs. 38.2, 38.3, and 38.4. Approximately 20–25% of effort is spent on determining business objectives, 50–60% of effort is spent on data preparation, 10–15% is spent on DM, and about 10% is spent on consolidation/application.

38.2 Handling Data

The largest percentage effort of the KDD process is spent on processing and preparing the data. In this section, common forms of data storage and tools for accessing the data are described, and the important issues in data preparation are discussed.

38.2.1 Databases and Data Warehousing

A *relational database* system contains one or more objects called tables. The data or information for the database are stored in these tables. Tables are uniquely identified by their names and are comprised of columns and rows. Columns contain the column name, data type, and any other attributes for the column. Rows contain the records or data for the columns. The structured query language (SQL) is the communication tool for relational database management systems. SQL statements are used to perform tasks such as updating data in a database, or retrieving data from a database. Some

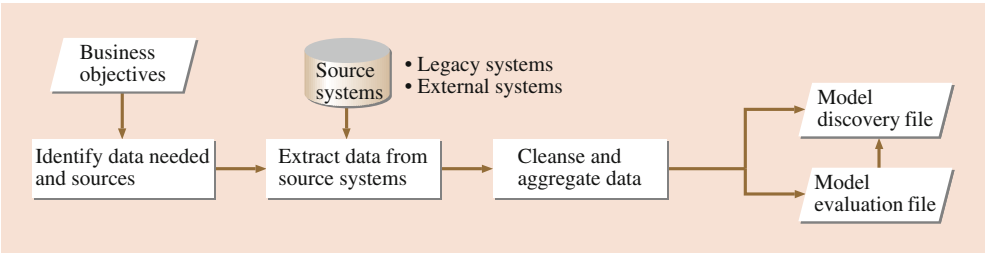

Fig. 38.2 Data preparation flow chart

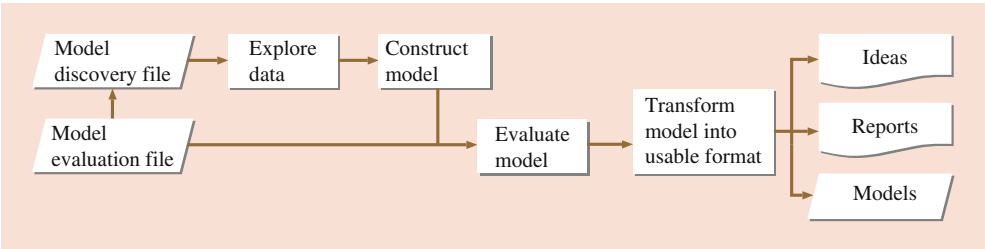

Fig. 38.3 Data mining flow chart

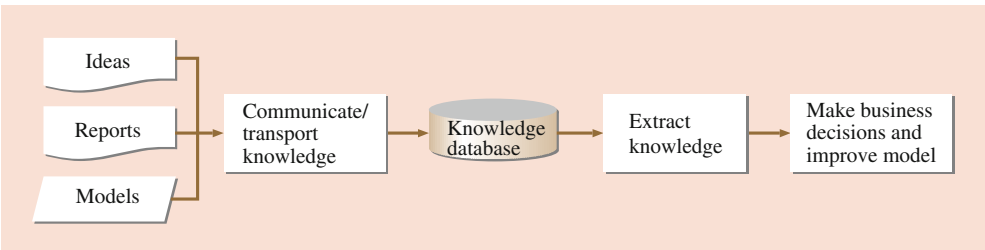

Fig. 38.4 Consolidation and application flow chart

common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, and Ingres. Standard SQL commands, such as *select*, *insert*, *update*, *delete*, *create*, and *drop*, can be used to accomplish almost everything that one needs to do with a database.

A *data warehouse* holds local databases assembled in a central facility. A *data cube* is a multidimensional array of data, where each dimension is a set of sets representing domain content, such as time or geography. The dimensions are scaled categorically, for example, region of country, state, quarter of year, and week of quarter. The cells of the cube contain aggregated measures (usually counts) of variables. To explore the data cube, one can *drill down*, *drill up*, and *drill through*. Drill down involves splitting an aggregation into subsets, e.g., splitting region of country into states. Drill up involves consolidation, i.e., aggregating subsets along a dimension. Drill through involves subsets crossing multiple sets, e.g., the user might investigate statistics within a state subset by time. Other databases and tools include object-oriented databases, transactional databases, time series and spatial databases, online analytical processing (OLAP), multidimensional OLAP (MOLAP), and relational OLAP using extended SQL (ROLAP). See Chap. 2 of *Han* and *Kamber* [7] for more details.

38.2.2 Data Preparation

The purpose of this step in the KDD process is to identify data quality problems, sources of noise, data redundancy, missing data, and outliers. Data quality problems can involve inconsistency with external datasets, uneven quality (e.g., if a respondent fakes an answer), and biased opportunistically collected data. Possible sources of noise include faulty data collection instruments (e.g., sensors), transmission errors (e.g., intermittent errors from satellite or Internet transmissions), data entry errors, technology limitations errors, misused naming conventions (e.g., using the same names for different meanings), and incorrect classification.

Redundant data exists when the same variables have different names in different databases, when a raw variable in one database is a derived variable in another, and when changes in a variable over time are not reflected in the database. These irrelevant variables impede the speed of the KDD process because dimension reduction is needed to eliminate them. Missing data may be irrelevant if we can extract useful knowledge without imputing the missing data. In addition, most statistical methods for handling missing data may fail for massive datasets, so new or modified methods still need to be developed. In detecting outliers, sophisticated methods like the Fisher information matrix or convex hull peeling are available, but are too complex for massive datasets. Although outliers may be easy to visualize in low dimensions, high-dimensional outliers may not show up in low-dimensional projections. Currently, clustering and other statistical modeling are used.

The data preparation process involves three steps: data cleaning, database sampling, and database reduction and transformation. Data cleaning includes removal of duplicate variables, imputation of missing values, identification and correction of data inconsistencies, identification and updating of stale data, and creating a unique record (case) identification (ID). Via database sampling, the KDD process selects appropriate parts of the databases to be examined. For this to work, the data must satisfy certain conditions (e.g., no systematic biases). The sampling process can be expensive if the data have been stored in a database system such that it is difficult to sample the data the way you want and many operations need to be executed to obtain the targeted data. One must balance a trade-off between the costs of the sampling process and the mining process. Finally, database reduction is used for data cube aggregation, dimension reduction, elimination of irrelevant and redundant attributes, data compression, and encoding mechanisms via quantizations, wavelet transformation, principle components, etc.

38.3 Data Mining (DM) Models and Algorithms

The DM process is illustrated in Fig. 38.5. In this process, one will start by choosing an appropriate class of models. To fit the best model, one needs to split the sample data

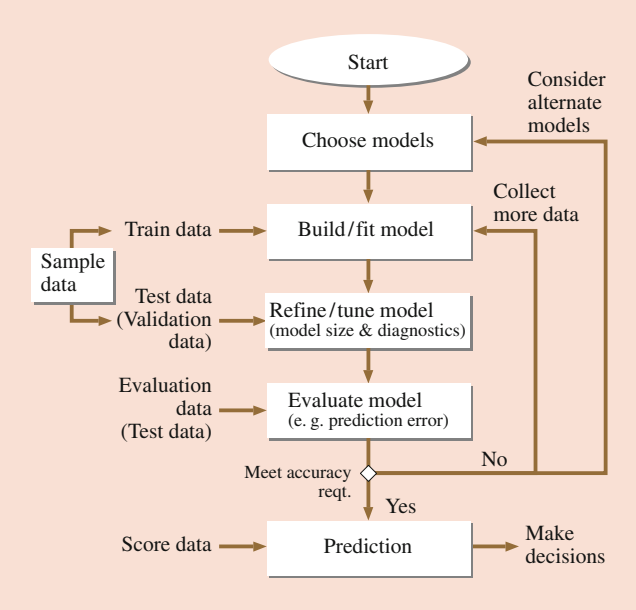

Fig. 38.5 Data mining process

into two parts: the training data and the testing data. The training data will be used to fit the model and the testing data is used to refine and tune the fitted model. After the final model is obtained, it is recommended to use an independent dataset to evaluate the goodness of the final model, such as comparing the prediction error to the accuracy requirement. (If independent data are not available, one can use the cross-validation method to compute prediction error.) If the accuracy requirement is not satisfied, then one must revisit earlier steps to reconsider other classes of models or collect additional data.

Before implementing any sophisticated DM methods, data description and visualization are used for initial exploration. Tools include descriptive statistical measures for central tendency/location, dispersion/spread, and distributional shape and symmetry; class characterizations and comparisons using analytical approaches, attribute relevance analysis, and class discrimination and comparisons; and data visualization using scatter-plot matrices, density plots, 3D stereoscopic scatter plots, and parallel coordinate plots. Following this initial step, DM methods take two forms: supervised versus unsupervised learning. Supervised learning is described as *learning with a teacher*, where the *teacher* provides data with correct answers. For example, if we want to classify online shoppers as buyers or nonbuyers using an available set of variables, our data would include actual instances of buyers and nonbuyers for training a DM method. Unsupervised learning is described as *learning without a teacher*. In this case, correct answers are not available, and DM methods would search for patterns or clusters of similarity that could later be linked to some explanation.

38.3.1 Supervised Learning

In supervised learning, we have a set of *input* variables (also known as predictors, independent variables, $\boldsymbol{x}$) that are measured or preset, and a set of *output* variables (also known as responses, dependent variables, $\boldsymbol{y}$) that are measured and assumed to be influenced by the inputs. If the outputs are continuous/quantitative, then we have a *regression* or *prediction* problem. If the outputs are categorical/qualitative, then we have a *classification* problem. First, a DM model/system is established based on the collected input and output data. Then, the established model is used to predict output values at new input values. The predicted values are denoted by $\hat{y}$.

The DM perspective of *learning with a teacher* follows these steps:

- Student presents an answer ($\hat{y}_i$ given $\boldsymbol{x}_i$)
- Teacher provides the correct answer $\boldsymbol{y}_i$ or an error $\boldsymbol{e}_i$ for the student's answer
- The result is characterized by some *loss function* or *lack-of-fit criterion*:LOF $\left(y, \hat{y}\right)$
- The objective is to minimize the expected loss

Supervised learning includes the common engineering task of function approximation, in which we assume that the output is related to the input via some function $f(\boldsymbol{x}, \epsilon)$, where ϵ represents a random error, and seek to approximate $f(\cdot)$.

Below, we describe several supervised learning methods. All can be applied to both the regression and classification cases, except for those presented below under "Other Classification Methods." We maintain the following notation. The j-th input variable is denoted by x_j (or random variable X_j) and the corresponding boldface $\boldsymbol{x}$ (or $\boldsymbol{X}$) denotes the vector of p input variables $(x_1, x_2, \ldots, x_p)^{\mathrm{T}}$, where boldface $\boldsymbol{x}_i$ denotes the i-th sample point; N is the number of sample points, which corresponds to the number of observations of the response variable; the response variable is denoted by y (or random variable Y), where y_i denotes the i-th response observation. For the regression case, the response y is quantitative, while for the classification case, the response values are indices for C classes ($c = 1, \ldots, C$). An excellent reference for these methods is *Hastie* et al. [8].

Linear and Additive Methods

In the regression case, the basic linear method is simply the *multiple linear* regression model form

$$\mu\left(\boldsymbol{x}; \beta\right) \;=\; E\left[Y \mid \boldsymbol{X} \;=\; \boldsymbol{x}\right] \;=\; \beta_0 + \sum_{m=1}^{M} \beta_m b_m\left(\boldsymbol{x}\right),$$

where the model terms $b_m(\boldsymbol{x})$ are prespecified functions of the input variables, for example, a simple linear term $b_m(\boldsymbol{x}) = x_j$ or a more complex interaction term $b_m\left(\boldsymbol{x}\right) = x_j x_k^2$. The key is that the model is *linear in the parameters* $\boldsymbol{\beta}$. Textbooks that cover linear regression are abundant (e.g., [9, 10]). In particular, *Neter* et al. [11] provides a good background on residual analysis, model diagnostics, and model selection using best subsets and stepwise methods. In model selection, insignificant model terms are eliminated, thus the final model may be a subset of the original prespecified model. An alternate approach is to use a shrinkage method that employs a penalty function to *shrink* estimated model parameters toward zero, essentially reducing the influence of less important terms. Two options are ridge regression [12], which uses the penalty form $\sum \beta_m^2$, and the lasso [13], which uses the penalty form $\sum \mid \beta_m \mid$.

In the classification case, linear methods generate *linear decision boundaries* to separate the C classes. Although a direct linear regression approach could be applied, it is known not to work well. A better method is *logistic regression* [14], which uses log-odds (or logit transformations) of the posterior probabilities $\mu_c(\boldsymbol{x}) = P(Y = c \mid \boldsymbol{X} = \boldsymbol{x})$ for classes $c = 1, \ldots, C - 1$ in the form

$$\log \frac{\mu_c(\boldsymbol{x})}{\mu_C(\boldsymbol{x})} = \log \frac{P(Y = c|\boldsymbol{X} = \boldsymbol{x})}{P(Y = C|\boldsymbol{X} = \boldsymbol{x})} = \beta_{c0} + \sum_{j=1}^{p} \beta_{cj} x_j,$$

where the C posterior probabilities $\mu_c(\boldsymbol{x})$ must sum to one. The decision boundary between class $c < C$ and class C is defined by the hyperplane $\{\boldsymbol{x} | \beta_{c0} + \sum \beta_{cj} x_j = 0\}$, where the log-odds are zero. Similarly, the decision boundary between classes $c \neq C$ and $d \neq C$, derived from the log-odds for classes c and d, is defined by $\{\boldsymbol{x} | \beta_{c0} + \sum \beta_{cj} x_j = \beta_{d0} + \sum \beta_{dj} x_j\}$. In the binary case ($C = 2$), if we define $\mu(\boldsymbol{x}) = P(Y = 1 | \boldsymbol{X} = \boldsymbol{x})$, then $1 - \mu(\boldsymbol{x}) = P(Y = 2 | \boldsymbol{X} = \boldsymbol{x})$. The logit transformation is then defined as $g(\mu) = \mu/(1 - \mu)$.

Closely related to logistic regression is *linear discriminant analysis* [15], which utilizes exactly the same linear form for the log-odds ratio, and defines linear *discriminant functions* $\delta_c(\boldsymbol{x})$, such that $\boldsymbol{x}$ is classified to class c if its maximum discriminant is $\delta_c(\boldsymbol{x})$. The difference between the two methods is how the parameters are estimated. Logistic regression maximizes the conditional likelihood involving the posterior probabilities $P(Y = c | \boldsymbol{X})$ while linear discriminant analysis maximizes the full log-likelihood involving the unconditional probabilities $P(Y = c, \boldsymbol{X})$. More general forms of discriminant analysis are discussed below under "Other Classification Methods."

Finally, it should be noted that the logistic regression model is one form of *generalized linear model* (GLM) [16]. GLM forms convert what appear to be nonlinear models into linear models, using tools such as transformations (e.g., logit) or conditioning on nonlinear parameters. This then enables the modeler to use traditional linear modeling analysis techniques. However, real data often do not satisfy the restrictive conditions of these models.

Rather than using prespecified model terms, as in a linear model, a *generalized additive model* (GAM) [17] provides a more flexible statistical method to enable modeling of nonlinear patterns in each input dimension. In the regression case, the basic GAM form is

$$\mu(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} f_j(x_j),$$

where the $f_j(\cdot)$ are unspecified (smooth) univariate functions, one for each input variable. The additive restriction prohibits inclusion of any interaction terms. Each function is fitted using a *nonparametric regression* modeling method, such as running-line smoothers (e.g., *lowess*, [18]), smoothing splines, or kernel smoothers [19–21]. In the classification case, an additive logistic regression model utilizes the logit transformation for classes $c = 1, \ldots, C - 1$ as above

$$\log \frac{\mu_c(\boldsymbol{x})}{\mu_C(\boldsymbol{x})} = \log \frac{P(Y = c|\boldsymbol{X} = \boldsymbol{x})}{P(Y = C|\boldsymbol{X} = \boldsymbol{x})} = \beta_0 + \sum_{j=1}^{p} f_j(x_j),$$

where an additive model is used in place of the linear model. However, even with the flexibility of nonparametric regression, GAM may still be too restrictive. The following sections describe methods that have essentially no assumptions on the underlying model form.

Trees and Related Methods

One DM decision tree model is *chi-square automatic interaction detection* (CHAID) [22, 23], which builds nonbinary trees using a chi-square test for the classification case and an F-test for the regression case. The CHAID algorithm first creates categorical input variables out of any continuous inputs by dividing them into several categories with approximately the same number of observations. Next, input variable categories that are not statistically different are combined, while a Bonferroni p-value is calculated for those that are statistically different. The best split is determined by the smallest p-value. CHAID continues to select splits until the smallest p-value is greater than a prespecified significance level (α).

The popular *classification and regression trees* (CART) [24] utilize recursive partitioning (binary splits), which evolved from the work of *Morgan* and *Sonquist* [25] and *Fielding* [26] on analyzing survey data. CARTs have a forward stepwise procedure that adds model terms and backward procedure for pruning. The model terms partition the $\boldsymbol{x}$-space into disjoint hyper-rectangular regions via indicator functions: $b^+(x; t) = 1\{x > t\}$, $b^-(x; t) = 1\{x \leq t\}$, where the *split-point* t defines the borders between regions. The resulting model terms are:

$$f_m(\boldsymbol{x}) = \prod_{l=1}^{L_m} b^{S_{l,m}}\left(x_{v(l,m)}; t_{l,m}\right), \tag{38.1}$$

where, L_m is the number of univariate indicator functions multiplied in the m-th model term, $x_{v(l,m)}$ is the input variable corresponding to the l-th indicator function in the m-th model term, $t_{l,m}$ is the split-point corresponding to $x_{v(l,m)}$, and $s_{l,m}$ is $+1$ or -1 to indicate the direction of the partition. The CART model form is then

$$f(\boldsymbol{x}; \beta) = \beta_0 + \sum_{m=1}^{M} \beta_m f_m(\boldsymbol{x}). \tag{38.2}$$

The partitioning of the $\boldsymbol{x}$-space does not keep the parent model terms because they are redundant. For example, suppose the current set has the model term:

$$f_a(\boldsymbol{x}) = 1\{x_3 > 7\} \cdot 1\{x_4 \leq 10\},$$

and the forward stepwise algorithm chooses to add

$$f_b(\boldsymbol{x}) = f_a(\boldsymbol{x}) \cdot 1\{x_5 > 13\}$$
$$= 1\{x_3 > 7\} \cdot 1\{x_4 \leq 10\} \cdot 1\{x_5 > 13\}.$$

Then the model term $f_a(\boldsymbol{x})$ is dropped from the current set. Thus, the recursive partitioning algorithm follows a binary tree with the current set of model terms $f_m(\boldsymbol{x})$ consisting of the M leaves of the tree, each of which corresponds to a different region R_m.

In the regression case, CART minimizes the squared error loss function,

$$\text{LOF}\left(\hat{f}\right) = \sum_{i=1}^{N}\left[y_i - \hat{f}(\boldsymbol{x}_i)\right]^2,$$

and the approximation is a piecewise constant function. In the classification case, each region R_m is classified into one of the C classes. Specifically, define the proportion of class c observations in region R_m as

$$\hat{\delta}_{mc} = \frac{1}{N_m}\sum_{\boldsymbol{x}_i \in \boldsymbol{R}_m} 1\{y_i = c\},$$

where N_m is the number of observations in the region R_m. Then the observations in region R_m are classified into the class c corresponding to the maximum proportion $\hat{\delta}_{mc}$. The algorithm is exactly the same as for regression, but with a different loss function. Appropriate choices include minimizing the misclassification error (i.e., the number of misclassified observations), the Gini index, $\sum_{c=1}^{C}\hat{\delta}_{mc}\left(1-\hat{\delta}_{mc}\right)$, or the deviance $\sum_{c=1}^{C}\hat{\delta}_{mc}\log\left(\hat{\delta}_{mc}\right)$.

The exhaustive search algorithms for CART simultaneously conduct variable selection (x) and split-point selection (t). To reduce computational effort, the *fast algorithm for classification trees* [27] separates the two tasks. At each existing model term (leaf of the tree), F-statistics are calculated for variable selection. Then linear discriminant analysis is used to identify the split-point. A version for logistic and Poisson regression was presented by *Chaudhuri* et al. [28].

The primary drawback of CART and FACT is a bias toward selecting higher-order interaction terms due to the property of keeping only the leaves of the tree. As a consequence, these tree methods do not provide robust approximations and can have poor prediction accuracy. *Loh* and *Shih* [29] address this issue for FACT with a variant of their classification algorithm called QUEST that clusters classes into superclasses before applying linear discriminant analysis. For CART, *Friedman* et al. [30] introduced to the statistics literature the concepts of *boosting* [31] and *bagging* [32] from the machine learning literature. The bagging approach generates many bootstrap samples, fits a tree to each, and then uses their average prediction. In the framework of boosting, a model term, called a *base learner*, is a *small* tree with only L disjoint regions (L is selected by the user), call it $B(\boldsymbol{x}, \boldsymbol{a})$, where $\boldsymbol{a}$ is the vector of tree coefficients. The boosting algorithm begins by fitting a small tree $B(\boldsymbol{x}, \boldsymbol{a})$ to the data, and the first approximation, $\hat{f}_1(\boldsymbol{x})$, is then this first small tree. In the m-th iteration, residuals are calculated, then a small tree $B(\boldsymbol{x}, \boldsymbol{a})$ is fitted to the residuals and combined with the latest approximation to create the m-th approximation:

$$\hat{f}_m(\boldsymbol{x}; \beta_0, \beta_1, \ldots, \beta_m) = \hat{f}_{m-1}\Big(\boldsymbol{x}; \beta_0, \beta_1, \ldots, \beta_{m-1}\Big) + \beta_m B(\boldsymbol{x}, \boldsymbol{a}),$$

where a line search is used to solve for β_m. The resulting boosted tree, called a *multiple additive regression tree* (MART) [33], then consists of much lower-order interaction terms. *Friedman* [34] presents *stochastic gradient boosting*, with a variety of loss functions, in which a bootstrap-like bagging procedure is included in the boosting algorithm.

Finally, for the regression case only, *multivariate adaptive regression splines* (MARS) [35] evolved from CART as an alternative to its piecewise constant approximation. Like CART, MARS utilizes a forward stepwise algorithm to select model terms followed by a backward procedure to prune the model. A univariate version (appropriate for additive relationships) was presented by *Friedman* and *Silverman* [36]. The MARS approximation bends to model curvature at *knot* locations, and one of the objectives of the forward stepwise algorithm is to select appropriate knots. An important difference from CART is that MARS maintains the parent model terms, which are no longer redundant but are simply lower-order terms.

MARS model terms have the same form as (38.1), except the indicator functions are replaced with truncated linear functions,

$$\left[b^+(x;t) = [+(x-t)]+, b^-(x;t) = [-(x-t)]+,\right.$$

where $[q]_+ = \max(0, q)$ and t is an univariate knot. The search for new model terms can be restricted to interactions of a maximum order (e.g., $L_m \leq 2$ permits up through two-factor interactions). The resulting MARS approximation, following (38.2), is a continuous, piecewise linear function. After selection of the model terms is completed, smoothness to achieve a certain degree of continuity may be applied.

Hastie et al. [8] demonstrate significant improvements in accuracy using MART over CART. For the regression case, comparisons between MART and MARS yield comparable results [34]. Thus, the primary decision between these two

methods is whether a piecewise constant approximation is satisfactory or if a continuous, smooth approximation would be preferred.

Artificial Neural Networks and Convolutional Neural Networks

Artificial neural network (ANN) models have been very popular for modeling a variety of physical relationships (for a general introduction see *Lippmann* [37] or *Haykin* [38]; for statistical perspectives see *White* [39], *Baron* et al. [40], *Ripley* [23], or *Cheng* and *Titterington* [41]). The original motivation for ANNs comes from how *learning* strengthens connections along neurons in the brain. Commonly, an ANN model is represented by a diagram of nodes in various layers with weighted connections between nodes in different layers (Fig. 38.6). At the input layer, the nodes are the input variables and at the output layer, the nodes are the response variable(s). In between, there is usually at least one *hidden* layer which induces flexibility into the modeling. *Activation functions* define transformations between layers (e.g., input to hidden). Connections between nodes can *feed back* to previous layers, but for supervised learning the typical ANN is *feedforward* only with at least one hidden layer.

The general form of a feedforward ANN with one hidden layer and activation functions $b_1(\cdot)$ (input to hidden) and $b_2(\cdot)$ (hidden to output) is

$$f_c\left(\boldsymbol{x};\boldsymbol{w},\boldsymbol{v},\boldsymbol{\theta},\gamma_c\right) = b_2\left[\sum_{h=1}^{H} w_{hc}\cdot b_1\left(\sum_{j=1}^{p} v_{jh}x_j+\theta_h\right)+\gamma_c\right], \tag{38.3}$$

where $c = 1,\ldots,C$ and C is the number of output variables, p is the number of input variables, H is the number of hidden nodes, the weights v_{jh} link input nodes j to hidden nodes h and w_{hc} link hidden nodes h to output nodes c, and θ_h and γ_c are constant terms called bias nodes (like intercept terms). The number of coefficients to be estimated is $(p+1)H+(H+1)C$, which is often larger than N. The simplest activation function is a linear function $b(z) = z$, which reduces the ANN model in (38.3) with one response variable to a multiple linear regression equation. For more flexibility, the recommended activation functions between the input and hidden layer(s) are the S-shaped *sigmoidal* functions or the bell-shaped *radial basis functions*. Commonly used sigmoidal functions are the logistic function

$$b(z) = \frac{1}{1+\mathrm{e}^{-z}}$$

and the hyperbolic tangent

$$b(z) = \tanh(z) = \frac{1-\mathrm{e}^{-2x}}{1+\mathrm{e}^{-2x}}.$$

The most common radial basis function is the Gaussian probability density function.

In the regression case, each node in the output layer represents a quantitative response variable. The output activation function may be either a linear, sigmoidal, or radial basis function. Using a logistic activation function from input to hidden and from hidden to output, the ANN model in (38.3) becomes

$$f_c\left(\boldsymbol{x};\boldsymbol{w},\boldsymbol{v},\boldsymbol{\theta},\gamma_c\right) = \left[1+\exp\left(-\sum_{h=1}^{H} w_{hc}z_h+\gamma_c\right)\right]^{-1},$$

where for each hidden node h

$$z_h = \left[1+\exp\left(-\sum_{j=1}^{p} v_{jh}x_j+\theta_h\right)\right]^{-1}.$$

In the classification case with C classes, each class is represented by a different node in the output layer. The recommended output activation function is the *softmax* function. For output node c, this is defined as

$$b\left(z_1,\ldots,zc;c\right) = \frac{\mathrm{e}^{z_c}}{\sum_{d=1}^{C}\mathrm{e}^{z_d}}.$$

This produces output values between zero and one that sum to one and, consequently, permits the output values to be interpreted as posterior probabilities for a categorical response variable.

Mathematically, an ANN model is a nonlinear statistical model, and a nonlinear method must be used to estimate the

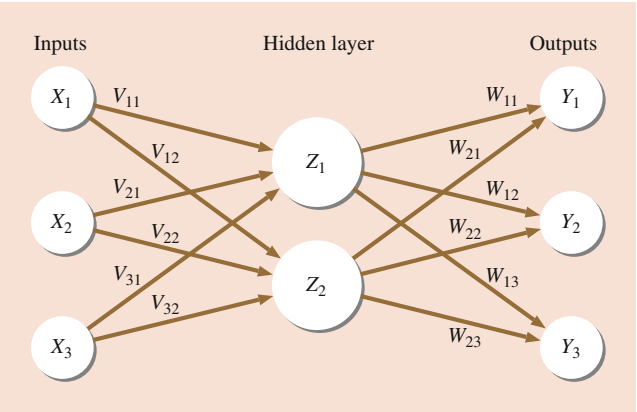

Fig. 38.6 Diagram of a typical artificial neural network for function approximation. The input nodes correspond to the input variables, and the output node(s) correspond to the output variable(s). The number of hidden nodes in the hidden layer must be specified by the user

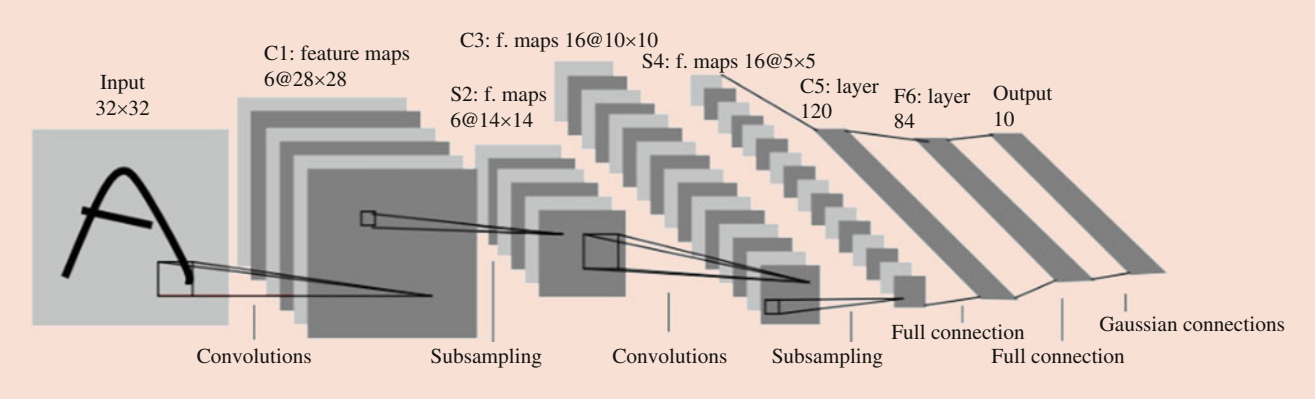

Fig. 38.7 Diagram of a typical convolutional neural network

coefficients (weights v_{jh} and w_{hc}, biases θ_h and γ_c) of the model. This estimation process is called network training. Typically, the objective is to minimize the squared error lack-of-fit criterion

$$\text{LOF}\left(\hat{f}\right) = \sum_{c=1}^{C}\sum_{i=1}^{N}\left[y_i - \hat{f}_c\left(\boldsymbol{x}_i\right)\right]^2.$$

The most common method for training is *backpropagation*, which is based on gradient descent. At each iteration, each coefficient (say w) is adjusted according to its contribution to the lack-of-fit

$$\Delta w = \alpha \frac{\partial\left(\text{LOF}\right)}{\partial w},$$

where the user-specified α controls the step size; see *Rumelhart* et al. [42] for more details. More efficient training procedures are a subject of current ANN research.

Another major issue is the network *architecture*, defined by the number of hidden nodes. If too many hidden nodes are permitted, the ANN model will overfit the data. Many model discrimination methods have been tested, but the most reliable is validation of the model on a testing dataset separate from the training dataset. Several ANN architectures are fitted to the training dataset and then prediction error is measured on the testing dataset. Although ANNs are generally flexible enough to model anything, they are computationally intensive, and a significant quantity of representative data is required to both fit and validate the model. From a statistical perspective, the primary drawback is the overly large set of coefficients, none of which provide any intuitive understanding for the underlying model structure. In addition, since the nonlinear model form is not motivated by the true model structure, too few training data points can result in ANN approximations with extraneous nonlinearity. However, given enough good data, ANNs can outperform other modeling methods.

Convolutional neural network (CNN) [43] is a feedforward ANN network effective for pattern recognition and feature extraction of image data. The CNN combines three architectural ideas: local receptive fields, shared weights, and spatial subsampling. As in Fig. 38.7, a typical CNN for character recognition usually consists of an input layer, a convolutional layer, a pooling layer, a fully connected layer, and an output layer. Before feeding into the input plane, images of characters are size normalized and centered. In the convolutional layer, each neuron receives inputs from a set of neurons located in a small neighborhood in the previous layer. By using the same convolutional filter, the neurons share the filter weights and the number of free parameters in the convolutional layer is greatly reduced. With local receptive fields, neurons can extract elementary visual features such as corners, end points, and oriented edges. These basic features are then combined in higher layers to form useful information. Each convolutional layer is followed by an additional pooling layer which performs a local averaging and a subsampling, reducing the resolution of the feature map, as well as the sensitivity of the outputs to shifts and distortions. The CNN extracts the topological and spatial features hidden inside the image data through layer-by-layer convolution and pooling operations. These features are finally fed into the fully connected layer for classification or regression.

Support Vector Machines

Referring to the linear methods for classification described earlier, the decision boundary between two classes is a hyperplane of the form $\{\boldsymbol{x}|\,\beta_0 + \sum \beta_j x_j = 0\}$. The *support vectors* are the points that are most critical to determining the optimal decision boundary because they lie close to the points belonging to the other class. With *support vector machines* (SVM) [44], the linear decision boundary is generalized to the more flexible form

$$f\left(\boldsymbol{x};\boldsymbol{\beta}\right) = \beta_0 + \sum_{m=1}^{M}\beta_m g_m\left(\boldsymbol{x}\right), \tag{38.4}$$

where the $g_m(\boldsymbol{x})$ are transformations of the input vector. The decision boundary is then defined by $\{\boldsymbol{x}|f(\boldsymbol{x};\boldsymbol{\beta}) = 0\}$. To solve for the optimal decision boundary, it turns out that we do not need to specify the transformations $g_m(\boldsymbol{x})$, but instead require only the *kernel function* [21, 45]:

$$K\left(\boldsymbol{x},\boldsymbol{x}'\right) = \left\langle \left[g_1\left(\boldsymbol{x}\right),\ldots,g_M\left(\boldsymbol{x}\right)\right], \left[g_1\left(\boldsymbol{x}'\right),\ldots,g_M\left(\boldsymbol{x}'\right)\right]\right\rangle.$$

Two popular kernel functions for SVM are polynomials of degree d, $K(\boldsymbol{x},\boldsymbol{x}') = (1 + \langle \boldsymbol{x},\boldsymbol{x}'\rangle)^d$, and radial basis functions, $K(\boldsymbol{x},\boldsymbol{x}') = \exp\left(-\|\boldsymbol{x}-\boldsymbol{x}'\|^2/c\right)$.

Given $K(\boldsymbol{x}, \boldsymbol{x}')$, we maximize the following Lagrangian dual-objective function:

$$\begin{aligned}&\max_{\alpha_1,\ldots\alpha_N} \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\alpha_i\alpha_{i'}y_iy_{i'}K\left(\boldsymbol{x}_i,\boldsymbol{x}_i'\right)\\ &\text{s.t.}\quad 0 \le \alpha_i \le \gamma, \quad \text{for } i = 1,\ldots,N \text{ and}\\ &\qquad \sum_{i=1}^{N}\alpha_i y_i = 0,\end{aligned}$$

where γ is an SVM tuning parameter. The optimal solution allows us to rewrite $f(\boldsymbol{x};\boldsymbol{\beta})$ as

$$f\left(\boldsymbol{x};\boldsymbol{\beta}\right) = \beta_0 + \sum_{i=1}^{N}\alpha_i y_i K\left(\boldsymbol{x},\boldsymbol{x}_i\right),$$

where β_0 and $\alpha_1,\ldots,\alpha_N$ are determined by solving $f(\boldsymbol{x};\boldsymbol{\beta}) = 0$. The support vectors are those $\boldsymbol{x}_i$ corresponding to nonzero α_i. A smaller SVM tuning parameter γ leads to more support vectors and a smoother decision boundary. A testing dataset may be used to determine the best value for γ.

The SVM extension to more than two classes solves multiple two-class problems. SVM for regression utilizes the model form in (38.4) and requires specification of a loss function appropriate for a quantitative response [8, 46]. Two possibilities are the ϵ-insensitive function

$$V_{\epsilon}(e) = \begin{cases} 0 & \text{if } |e| < \epsilon,\\ |e| - \epsilon & \text{otherwise,}\end{cases}$$

which ignores errors smaller than ϵ, and the *Huber* [47] function

$$V_{\mathrm{H}}(e) = \begin{cases} e^2/2 & \text{if } |e| \le 1.345,\\ 1.345\,|e| - e^2/2 & \text{otherwise,}\end{cases}$$

which is used in robust regression to reduce model sensitivity to outliers.

Other Classification Methods

In this section, we briefly discuss some other concepts that are applicable to DM classification problems. The basic intuition behind a good classification method is derived from the *Bayes classifier*, which utilizes the posterior distribution $P(Y = c|\ \boldsymbol{X} = \boldsymbol{x})$. Specifically, if $P(Y = c|\ \boldsymbol{X} = \boldsymbol{x})$ is the maximum over $c = 1,\ldots,C$, then $\boldsymbol{x}$ would be classified to class c.

Nearest neighbor (NN) [48] classifiers seek to estimate the Bayes classifier directly without specification of any model form. The k-NN classifier identifies the k closest points to $\boldsymbol{x}$ (using Euclidean distance) as the *neighborhood* about $\boldsymbol{x}$, then estimates $P(Y = c|\ \boldsymbol{X} = \boldsymbol{x})$ with the fraction of these k points that are of class c. As k increases, the decision boundaries become smoother; however, the *neighborhood* becomes less local (and less relevant) to $\boldsymbol{x}$. This problem of local representation is even worse in high dimensions, and modifications to the distance measure are needed to create a practical k-NN method for DM. For this purpose, *Hastie* and *Tibshirani* [49] proposed the discriminant adaptive NN distance measure to reshape the neighborhood adaptively at a given $\boldsymbol{x}$ to capture the critical points to distinguish between the classes.

As mentioned earlier, linear discriminant analysis may be too restrictive in practice. *Flexible discriminant analysis* replaces the linear decision boundaries with more flexible regression models, such as GAM or MARS. *Mixture discriminant analysis* relaxes the assumption that classes are more or less spherical in shape by allowing a class to be represented by multiple (spherical) clusters; see *Hastie* et al. [50] and *Ripley* [23] for more details.

K-means clustering classification applies the K-means clustering algorithm separately to the data for each of the C classes. Each class c will then be represented by K clusters of points. Consequently, nonspherical classes may be modeled. For a new input vector $\boldsymbol{x}$, determine the closest cluster, then assign $\boldsymbol{x}$ to the class associated with that cluster.

Genetic algorithms [51, 52] use processes such as genetic combination, mutation, and natural selection in an optimization based on the concepts of natural evolution. One generation of models competes to pass on characteristics to the next generation of models, until the best model is found. Genetic algorithms are useful in guiding DM algorithms, such as neural networks and decision trees [53].

38.3.2 Unsupervised Learning

Unsupervised learning, often called "learning without a teacher," has been widely used for exploratory analysis to identify hidden patterns or groups in data. Unlike supervised learning, it draws inferences from data *without* predefined label information (i.e., response variables are not available). Given a set of observations of a random variable X, the goal of unsupervised learning is to directly infer properties of the probability density $P(X)$ without the label information.

It may be noted that labeled samples are sometimes significantly more expensive to collect (e.g., by asking human experts to make judgments) than unlabeled samples. Therefore, unsupervised learning has gained increasing interest in a variety of applications when labels are difficult to obtain, including biology, medicine, social science, business marketing, etc. In this subsection, we introduce two commonly used unsupervised learning techniques: association rules and cluster analysis.

Association Rules

Association rule analysis [8] seeks to discover co-occurrence between items in a collection and expresses such relationships as *association rules*. It is most often applied as *market basket analysis*, which deals with sales transactions to link specific products for the analysis of purchasing behaviors of customers. For example, the following rule:

$$\{\text{Diapers}\} \rightarrow \{\text{Beer}\}$$

suggests co-occurrence exists between the sale of diapers and beer. In other words, customers tend to purchase diapers and beer together. Such information is helpful for retailers to increase profit by optimizing their cross-promotion strategies, catalog design, stocking shelves, and customer relationship management.

In association analysis, a collection of one or more items is termed an *itemset*. If an itemset contains k items, it is called a k-itemset. For example, {Diapers, Beer, Eggs} is a 3-itemset. An important property of an itemset is its *support count* σ, which is defined as the number of transactions that contain this itemset. In the dataset below, the support count for {Diapers, Beer, Eggs} is two because only transactions #3 and #4 contain these three items.

TID	Items
#1	{Bread, Diapers}
#2	{Bread, Diapers, Milk, Eggs}
#3	{Diapers, Beer, Eggs}
#4	{Milk, Eggs, Beer, Diapers}
#5	{Bread, Milk, Diapers, Beer}

An *association rule* is expressed as $A \rightarrow B$, where A is called the *antecedent* and B is called the *consequent*. Given an itemset, association rule can be generated by assigning one or more items as the *antecedent* and one or more items as the *consequent*. Notably, A and B are disjoint sets, i.e., $A \cap B = \varnothing$. The importance of an association rule can be measured in terms of its *support* and *confidence*. *Support* (s) determines how frequently a rule appears among all transactions and it is defined as the support count of the rule over the total number of transactions N:

$$s\,(A \rightarrow B) = \frac{\sigma\,(A \cup B)}{N}$$

For example, support of rule {Diapers} $\rightarrow$ {Beer}is 3/5 and support of rule {Diapers, Eggs} $\rightarrow$ {Beer} is 2/5. Notably, a low support indicates the rule is uninteresting and customers seldom buy these items together. Minimum support (*minsup*) can be defined to eliminate uninteresting rules with $s <$ *minsup*.

Confidence (c), on the other hand, represents how frequently items in B appear in transactions that contain A. It is defined as:

$$c\,(A \rightarrow B) = \frac{\sigma\,(A \cup B)}{\sigma\,(A)}$$

For example, the confidence of rule {Diapers} $\rightarrow$ {Beer} is 3/5, and the confidence of rule {Diapers, Eggs} $\rightarrow$ {Beer} is 2/3. A high confidence suggests it is likely for B to be present in transactions that contain A. In other words, customers tend to buy these items together and it is profitable to promote them together.

Furthermore, *lift* can be calculated as the ratio of the confidence over the expected confidence,

$$L\,(A \rightarrow B) = \frac{c\,(A \rightarrow B)}{\sigma\,(B)/N}$$

which, if greater than one, can be interpreted as the increased prevalence of B when associated with A. For example, if $\sigma(B)/N = 5\%$, then B is estimated to occur unconditionally 5% of the time. If $c(A \rightarrow B) = 40\%$, then given A occurs, B is estimated to occur 40% of the time. This results in a lift of 8, implying that B is 8 times more likely to occur if A occurs.

Cluster Analysis

Cluster analysis, or clustering, seeks to segment a set of objects into clusters, such that objects within a cluster are more similar to each other than those assigned to different clusters [54]. It is an unsupervised learning method since no predefined label information is needed. The goal of cluster analysis is to discover the underlying structure of the data to obtain insight into the data distribution, arrange objects into a natural hierarchy, or serve as a preprocessing step for other algorithms. Notably, defining the similarity measure among objects is critical for cluster analysis. Each object i can be represented as a set of measurements, x_{ip} $(p = 1, 2, \ldots P)$, where P is the number of variables (also called attributes). Then, the pairwise distance between objects i and j is defined as $d\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \sum_{p=1}^{P} d_p\left(x_{ip}, x_{jp}\right)$, and $d_p(x_{ip}, x_{jp})$ is the dissimilarity between the values of the pth variable. For quantitative variables, the common choice for $d_p(x_{ip}, x_{jp})$ is the squared distance, i.e., $d_p(x_{ip}, x_{jp}) = (x_{ip} - x_{jp})^2$. For nonquantitative variables (e.g., ordinal or categorical variables), numerical

coding approaches [55] or Hamming distance [56] can be used. Below, we introduce some widely used algorithms for cluster analysis.

K-means [57] is one of the most popular clustering tools. It aims to partition the N objects into K clusters $\boldsymbol{C} = \{C_1, C_2, \ldots, C_K\}$, so that the within-cluster distance is minimized as follows:

$$\underset{\boldsymbol{C}}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{i \in C_k} \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_k \right\|^2$$

where $\boldsymbol{\mu}_k$ is the mean of objects in cluster C_k. The objective function is minimized using an iterative refinement approach: In the *assignment step*, the distance between each object to the means of K clusters are calculated and the object is assigned to the cluster with minimum distance. In the *update step*, the new means of the objects in new clusters are calculated. The initialization of cluster means can be randomly selected K objects. The *assignment step* and *update step* are repeated until assignments are no longer changed. Notably, *K-means* needs to specify the desired number of clusters K, which can be selected by using prior knowledge or by trying different values and looking for the one with the most interpretable solution. Also, it is sensitive to the initialization and may converge to a local optimum. Thus, it is crucial to run the algorithm many times from multiple random starting points. The best solution is selected with the smallest value of the objective function. In addition, it is difficult for *K-means* to handle noisy data and outliers. Many algorithms, such as *K-Medoids*, were developed to improve the robustness of *K-means* [58].

Density-based clustering (DBSCAN) [59] is rooted in the idea that clusters are dense regions with objects packed together, separated by regions of lower density. It is associated with two key parameters: the radius of a neighborhood with respect to an object (ε) and the minimum number of objects to form a dense region (*MinPts*). The ε-neighborhood of object w is defined as $N_\varepsilon(w) : \{v | \, d(w, v) \leq \varepsilon\}$. Then, objects can be segmented into three groups: if $N_\varepsilon(w)$ contains at least *MinPts* objects, then the object w is a *core object*; if object w has fewer than *MinPts* objects in its ε-neighborhood, but it is in the ε-neighborhood of a core object, it is called a *border object*; and if the object is neither a core object nor a border object, it is categorized as a *noise object*. An object v is called *directly density-reachable* from an object w w.r.t. ε and *MinPts* if w is a core object and $v \in N_\varepsilon(w)$. An object v is called *density-reachable* from w w.r.t. ε and *MinPts* if there is a chain of objects $o_1, o_2, \ldots, o_n$ with $o_1 = w$ and $o_n = v$, and o_{i+1} is directly reachable from o_i. Two objects v and w are called density-connected if there is an object o such that both v and w are density-reachable from o. Then, a cluster satisfies two properties: (i) all objects within the cluster are mutually density-connected and (ii) if an object is density-reachable from any object of the cluster, it should be included into the cluster. To find a cluster, DBSCAN starts with an object o that has not been visited. If o is a core object, then it collects all objects that are density-reachable from o and forms a cluster. Otherwise, the object is considered as noise. As opposed to *K-means*, DBSCAN is less sensitive to outliers and better handles data with arbitrary geometric shapes. However, it cannot handle data with varying density and is sensitive to parameters settings.

Hierarchical clustering [8] seeks to build a hierarchy of clusters based on pairwise dissimilarities among objects. Strategies for hierarchical clustering generally fall into two categories: agglomerative (bottom up) and divisive (top down). Agglomerative strategies start at the bottom with each object in its own cluster. It recursively merges pairs of clusters with smallest inter-cluster dissimilarity as one moves up the hierarchy. Divisive strategies start at the top with all objects in one cluster. It moves down the hierarchy by recursively splitting an existing cluster into two new clusters with the largest inter-cluster dissimilarity. Let W and V represent two clusters of objects. The dissimilarity of W and V is computed from the pairwise dissimilarities d_{wv}, where one member of the pair w is from W and the other one v is from V. *Average linkage*: $d_{\mathrm{AL}}(W, V) = \frac{1}{N_W N_V} \sum_{v \in V} \sum_{w \in W} d_{wv}$, *complete linkage*: $d_{\mathrm{CL}}(W, V) = \max_{v \in V, w \in W} d_{wv}$, and *single linkage*: $d_{\mathrm{SL}}(W, V) = \min_{v \in V, w \in W} d_{wv}$ are commonly used to measure the inter-cluster dissimilarity. It is up to the user to decide when (i.e., at which level) to stop to obtain a "natural" result: objects within each cluster are sufficiently more similar to each other than to objects assigned to different clusters. The resulting hierarchical structure can be graphically represented as a *dendrogram*.

Affinity propagation (AP) [60] is based on neighbor information propagation. It finds the optimal set of class representative objects (i.e., exemplars), which make the sum of the similarities of all objects to their nearest exemplars as large as possible. For objects i, j, and k, a similarity matrix s is defined such that $s(i,j) > s(i,k)$, if object i is more similar to object j than to k. A commonly used function is $s(i,j) = -\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$. The AP algorithm proceeds by letting all objects send messages to all other objects to determine exemplars with two matrices: (1) a "responsibility matrix" with element $r(i,j)$ that quantifies how well-suited object j is to be an exemplar to object i and (2) an "availability matrix" with element $a(i,j)$ that represents how appropriate it would be for object i to choose object j as its exemplar. As such, responsibility messages are sent around as: $r(i,k) \leftarrow s(i,k) - \max_{k_1 \neq k} \{a(i,k_1) + s(i,k_1)\}$. The availability messages are updated as: $a(i,k) \leftarrow$

$\min\left\{0, r(k,k) + \sum_{i_1 \neq i \& i_1 \neq k} \max\{0, r(i_1,k)\}\right\}$ for $i \neq k$ and $a(k,k) \leftarrow \sum_{i_1 \neq k} \max\{0, r(i_1,k)\}$. The algorithm iterates until cluster assignments are not changed. The final exemplars are chosen as those with $r(i,i) + a(i,i) > 0$. Although exemplars are similar to "centroids" generated by the *K-means*, the AP algorithm does not require the number of clusters to be predefined.

A *self-organizing map (SOM)* [61] is a type of unsupervised ANN model that represents high-dimensional input data in a low-dimensional map, preserves the topological relationship of the original data, and organizes the data according to inherent structures. Given an input $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$, the distance to each neuron in the SOM is calculated as $d_i = \|\boldsymbol{w}_i - \boldsymbol{x}\|$, where $\boldsymbol{w}_i = (w_{i1}, w_{i2}, \ldots, w_{id})$ is the weight of the ith neuron in the SOM (in total M neurons). The resulting distance vector $\boldsymbol{d} = (d_1, d_2, \ldots, d_M)$ is obtained to determine the best matching neuron (BMN) by finding the smallest d_i. Then, d_i is assigned as 1 and all other elements in $\boldsymbol{d}$ are set as 0. Further, the weights of the BMN and its neighbors are updated toward the input according to the rule of the Kohonen update as: $\boldsymbol{w}_i^{t+1} \leftarrow \boldsymbol{w}_i^t + h^t \cdot \left(\boldsymbol{x}^t - \boldsymbol{w}_i^t\right)$, where t is the iteration index and h^t is a neighborhood function to characterize the closeness of the BMN to other neurons in the map. In this way, a SOM arranges high-dimensional input in a two-dimensional map such that similar inputs are mapped onto neighboring regions. Thus, similar patterns of the input data are preserved. Outputs of a SOM can be characterized using a U-matrix or a Hits Map to visualize the clustering results. Conventional SOMs are designed for unsupervised learning, whereas supervised SOMs are also investigated in the literature that integrate label information as an additional element in the input vector $\boldsymbol{x}$ during the training phase [62].

38.3.3 Software

Several DM software packages and libraries are available:

- SAS Enterprise Miner (www.sas.com/technologies/analytics/datamining/miner/)
- SPSS Clementine (www.spss.com/clementine/)
- XLMiner in Excel (www.xlminer.net)
- Ghostminer (www.fqspl.com.pl/ghostminer/)
- Quadstone (www.quadstone.com/)
- Insightful Miner (www.splus.com/products/iminer/)
- Statsmodels (www.statsmodels.org)
- SciKit-Learn (www.scikit-learn.org)
- Keras (www.keras.io)
- Tensorflow (www.tensorflow.org)
- Deep Learning Toolbox (www.mathworks.com/solutions/deep-learning/)
- Darknet (www.pjreddie.com/darknet/)

Haughton et al. [63] present a review of the first five listed above. The SAS and SPSS packages have the most complete set of KDD/DM tools (data handling, DM modeling, and graphics), while Quadstone is the most limited. Insightful Miner was developed by *S+* [www.splus.com], but does not require knowledge of the *S+* language, which is only recommended for users that are familiar with statistical modeling. For statisticians, the advantage is that Insightful Miner can be integrated with more sophisticated DM methods available with *S+*, such as flexible and mixture discriminant analysis. All six packages include trees and clustering, and all except Quadstone include ANN modeling. The SAS, SPSS, and XLMiner packages include discriminant analysis and association rules. Ghostminer is the only one that offers SVM tools.

SAS, SPSS, and Quadstone are the most expensive (over $ 40 000) while XLMiner is a good deal for the price (under $ 2 000). The disadvantage of XLMiner is that it cannot handle very large datasets. Each package has certain specializations, and potential users must carefully investigate these choices to find the package that best fits their KDD/DM needs. Below we describe some other software options for the DM modeling methods presented.

GLM or linear models are the simplest of DM tools and most statistical software can fit them, such as SAS, SPSS, *S+*, and Statistica [www.statsoftinc.com/]. However, it should be noted that Quadstone only offers a regression tool via scorecards, which is not the same as statistical linear models. GAM requires access to more sophisticated statistical software, such as *S+*.

Software for CART, MART, and MARS is available from Salford Systems [www.salford-systems.com]. SAS Enterprise Miner includes CHAID, CART, and the machine learning program C4.5 [www.rulequest.com], which uses classifiers to generate decision trees and if–then rules. SPSS Clementine and Insightful Miner also include CART, but Ghostminer and XLMiner utilize different variants of decision trees. QUEST [www.stat.wisc.edu/loh/quest.html] is available in SPSS's AnswerTree software and Statistica.

Python provides many open-source libraries for machine learning and deep learning. Statsmodels is a library that enables its users to conduct data exploration via the use of various methods of estimation of statistical models and performing statistical assertions and analysis. Scikits are additional packages of SciPy Stack designed for specific functionalities like image processing and machine learning facilitation. Keras and Tensorflow are two most prominent and convenient open-source libraries for deep learning. Keras builds neural networks at a high level of the interface. Tensorflow is developed by Google and sharpened for machine learning. It was designed to meet the high-demand requirements of Google environment for training neural networks and is a successor of DistBelief, a machine learning system based on neural networks.

Other softwares for ANN include Matlab's [http://www.mathworks.com] Neural Network Toolbox, Matlab's Deep Learning Toolbox, and Darknet [http://www.pjreddie.com/darknet/]. The Neural Network Toolbox provides a complete package for ANN modeling. The Deep Learning Toolbox supports CNN networks for classification and regression on image data. The Darknet is an open-source deep learning framework written in C and CUDA, which is fast, easy to install, and provides state-of-art methods for real-time object detection.

38.4 DM Research and Applications

Many industrial and business applications require modeling and monitoring processes with real-time data of different types: real values, categorical, and even text and image. DM is an effective tool for extracting process knowledge and discovering data patterns to provide a control aid for these processes. Advanced DM research involves complex system modeling of heterogeneous objects, where adaptive algorithms are necessary to capture dynamic system behavior. Various data mining algorithms [63], such as logistic regression, support vector machines, convolutional neural networks, decision trees, and combinations of these, have been widely adopted in practical applications. Some application examples include activity monitoring, manufacturing process modeling, object detection, health assessment, fault diagnosis, and remaining useful life prediction. DM algorithms serve as solutions to these tasks.

38.4.1 Activity Monitoring

One important DM application is the development of an effective data modeling and monitoring system for understanding customer profiles and detecting fraudulent behavior. This is generally referred to as *activity monitoring for interesting events requiring action* [65]. Other activity monitoring examples include credit card or insurance fraud detection, computer intrusion detection, some forms of fault detection, network performance monitoring, and news story monitoring.

Although activity monitoring has only recently received attention in the information industries, solutions to similar problems were developed long ago in the manufacturing industries, under the moniker *statistical process control* (SPC). SPC techniques have been used routinely for online process control and monitoring to achieve process stability and to improve process capability through variation reduction. In general, all processes are subject to some natural variability regardless of their state. This natural variability is usually small and unavoidable and is referred to as *common cause variation*. At the same time, processes may be subject to other variability caused by improper machine adjustment, operator errors, or low-quality raw material. This variability is usually large, but avoidable, and is referred to as *special cause variation*. The basic objective of SPC is to detect the occurrence of special cause variation (or process shifts) quickly, so that the process can be investigated and corrective action may be taken before quality deteriorates and defective units are produced. The main ideas and methods of SPC were developed in the 1920s by Walter Shewhart of Bell Telephone Laboratories and have had tremendous success in manufacturing applications [66]. *Montgomery* and *Woodall* [67] provide a comprehensive panel discussion on SPC, and multivariate methods are reviewed by *Hayter* and *Tsui* [68] and *Mason* et al. [69].

Although the principle of SPC can be applied to service industries, such as business process monitoring, fewer applications exist for two basic reasons that *Montgomery* identified. First, the system that needs to be monitored and improved is obvious in manufacturing applications, while it is often difficult to define and observe in service industries. Second, even if the system can be clearly specified, most nonmanufacturing operations do not have natural measurement systems that reflect the performance of the system. However, these obstacles no longer exist, due to the many natural and advanced measurement systems that have been developed. In the telecommunications industry, for example, advanced software and hardware technologies make it possible to record and process huge amounts of daily data in business transactions and service activities. These databases contain potentially useful information to the company that may not be discovered without knowledge extraction or DM tools.

While SPC ideas can be applied to business data, SPC methods are not directly applicable. Existing SPC theories are based on small- or medium-sized samples, and the basic hypothesis testing approach is intended to detect only simple shifts in a process mean or variance. Recently, *Jiang* et al. [70] successfully generalized the SPC framework to model and track thousands of diversified customer behaviors in the telecommunication industry. The challenge is to develop an integrated strategy to monitor the performance of an entire multistage system and to develop effective and efficient techniques for detecting the systematic changes that require action.

A dynamic business process can be described by the dynamic linear models introduced by *West* [71],

$$\begin{aligned}&\text{Observation equation:}\quad X_t = A_t\theta_t + \Delta_t,\\ &\text{System evolution equation:}\quad \theta_t = B_t\theta_{t-1} + \Lambda_t,\\ &\text{Initial information:}\quad \pi\,(S_0)\,,\end{aligned}$$

where A_t and B_t represent observation and state transition matrices, respectively, and Δ_t and Λ_t represent observation and system transition errors, respectively. Based on the dynamic system model, a model-based process monitoring and root-cause identification method can be developed. Monitoring and diagnosis include fault pattern generation and feature extraction, isolation of the critical processes, and root-cause identification. *Jiang* et al. [70] utilize this for individual customer prediction and monitoring. In general, individual modeling is computationally intractable and cluster models should be developed with mixture distributions [72].

One particularly competitive industry is telecommunications. Since divestiture and government deregulation, various telephone services, such as cellular, local and long distance, domestic, and commercial, have become battle grounds for telecommunication service providers. Because of the data and information-oriented nature of the industry, DM methods for knowledge extraction are critical. To remain competitive, it is important for companies to develop business planning systems that help managers make good decisions. In particular, these systems will allow sales and marketing people to establish successful customer loyalty programs for churn prevention and to develop fraud detection modules for reducing revenue loss through market segmentation and customer profiling.

A major task in this research is to develop and implement DM tools within the business planning system. The objectives are to provide guidance for targeting business growth, to forecast year-end usage volume and revenue growth, and to value risks associated with the business plan periodically. Telecommunication business services include voice and non-voice services, which can be further categorized to include domestic, local, international, products, toll-free calls, and calling cards. For usage forecasting, a minutes growth model is utilized to forecast domestic voice usage. For revenue forecasting, the average revenue per minute on a log scale is used as a performance measure and is forecasted by a double exponential smoothing growth function. A structural model is designed to decompose the business growth process into three major subprocesses: add, disconnect, and base. To improve explanatory power, the revenue unit is further divided into different customer groups. To compute confidence and prediction intervals, bootstrapping and simulation methods are used.

To understand the day effect and seasonal effect, the concept of bill-month equivalent business days (EBD) is defined and estimated. To estimate EBD, the factor characteristics of holidays (non-EBD) are identified and eliminated and the day effect is estimated. For seasonality, the US Bureau of the Census X-11 seasonal adjustment procedure is used.

38.4.2 Mahalanobis–Taguchi System

Genichi Taguchi is best known for his work on robust design and design of experiments. The Taguchi robust design methods have generated a considerable amount of discussion and controversy and are widely used in manufacturing [73–76]. The general consensus among statisticians seems to be that, while many of Taguchi's overall ideas on experimental design are very important and influential, the techniques he proposed are not necessarily the most effective statistical methods. Nevertheless, Taguchi has made significant contributions in the area of quality control and quality engineering. For DM, Taguchi has recently popularized the *Mahalanobis–Taguchi System* (MTS), a new set of tools for diagnosis, classification, and variable selection. The method is based on a Mahalanobis distance scale that is utilized to measure the level of abnormality in *abnormal* items as compared to a group of *normal* items. First, it must be demonstrated that a Mahalanobis distance measure based on all available variables is able to separate the abnormal from the normal items. Should this be successfully achieved, orthogonal arrays and signal-to-noise ratios are used to select an *optimal* combination of variables for calculating the Mahalanobis distances.

The MTS method has been claimed to be very powerful for solving a wide range of problems, including manufacturing inspection and sensing, medical diagnosis, face and voice recognition, weather forecasting, credit scoring, fire detection, earthquake forecasting, and university admissions. Two recent books have been published on the MTS method by *Taguchi* et al. [77] and *Taguchi* and *Jugulum* [78]. Many successful case studies in MTS have been reported in engineering and science applications in many large companies, such as Nissan Motor Co., Mitsubishi Space Software Co., Xerox, Delphi Automotive Systems, ITT Industries, Ford Motor Company, Fuji Photo Film Company, and others. While the method is getting a lot of attention in many industries, very little research [79] has been conducted to investigate how and when the method is appropriate.

38.4.3 Manufacturing Process Modeling

One area of DM research in manufacturing industries is quality and productivity improvement through DM and knowledge discovery. Manufacturing systems nowadays are often very complicated and involve many manufacturing process stages where hundreds or thousands of in-process measurements are taken to indicate or initiate process control of the system. For example, a modern semiconductor manufacturing process typically consists of over 300 steps, and in each

step multiple pieces of equipment are used to process the wafer. Inappropriate understanding of interactions among in-process variables will create inefficiencies at all phases of manufacturing, leading to long product/process realization cycle times and long development times, resulting in excessive system costs.

Current approaches to DM in electronics manufacturing include neural networks, decision trees, Bayesian models, and rough set theory [80, 81]. Each of these approaches carries certain advantages and disadvantages. Decision trees, for instance, produce intelligible rules and hence are very appropriate for generating process control or design of experiments strategies. They are, however, generally prone to outlier and imperfect data influences. Neural networks, on the other hand, are robust against data abnormalities but do not produce readily intelligible knowledge. These methods also differ in their ability to handle high-dimensional data, to discover arbitrarily shaped clusters [57] and to provide a basis for intuitive visualization [82]. They can also be sensitive to training and model building parameters [59]. Finally, the existing approaches do not take into consideration the localization of process parameters. The patterns or clusters identified by existing approaches may include parameters from a diverse set of components in the system. Therefore, a combination of methods that complement each other to provide a complete set of desirable features is necessary.

It is crucial to understand process structure and yield components in manufacturing, so that problem localization can permit reduced production costs. For example, semiconductor manufacturing practice shows that over 70% of all fatal detects and close to 90% of yield excursions are caused by problems related to process equipment [83]. Systematic defects can be attributed to many categories that are generally associated with technologies and combinations of different process operations. To implement DM methods successfully for knowledge discovery, some future research for manufacturing process control must include yield modeling, defect modeling, and variation propagation.

Yield Modeling

In electronics manufacturing, the ANSI standards [84] and practice gene rally assume that the number of defects on an electronics product follows a Poisson distribution with mean λ. The Poisson random variable is an approximation of the sum of independent Bernoulli trials, but defects on different components may be correlated since process yield critically depends on product groups, process steps, and types of defects [85]. Unlike traditional defect models, an appropriate logit model can be developed as follows. Let the number of defects of category X on an electronics product be

$$U_X = \sum Y_X$$

and

$$\text{logit}\,[E\,(Y_X)] = \alpha_X^0 + \alpha_X^O \cdot O_X + \alpha_X^C \cdot C_X + \alpha_X^{OC} \cdot O_X \cdot C_X,$$

where $\text{logit}(z) = \log[z/(1-z)]$ is the link function for Bernoulli distributions, and Y_X is a Bernoulli random variable representing a defect from defect category X. The default logit of the failure probability is α_X^O, and α_X^O and α_X^C are the main effects of operations (O_X) and components (C_X). Since the Y_Xs are correlated, this model will provide more detailed information about defects.

Multivariate Defect Modeling

Since different types of defects may be caused by the same operations, multivariate Poisson models are necessary to account for correlations among different types of defects. The *trivariate reduction method* suggests an additive Poisson model for the vector of Poisson counts $\boldsymbol{U} = (U_1, U_2, \cdots, U_k)'$,

$$\boldsymbol{U} = \boldsymbol{AV},$$

where $\boldsymbol{A}$ is a matrix of zeros and ones, and $\boldsymbol{V} = (v_1, v_2, \ldots, v_p)'$ consists of independent Poisson variables v_i. The variance–covariance matrix takes the form $\text{Var}(U) = A\Sigma A' = \boldsymbol{\Phi} + \nu\nu'$, where $\boldsymbol{\Phi} = \text{diag}(\mu_i)$ is a diagonal matrix with the mean of the individual series, and ν is the common covariance term. Note that the v_i are essentially latent variables, and a factor analysis model can be developed for analyzing multivariate discrete Poisson variables such that

$$\log\,[E(U)] = \mu + \boldsymbol{L} \cdot \boldsymbol{F},$$

where $\boldsymbol{U}$ is the vector of defects, $\boldsymbol{L}$ is the matrix of factor loadings, and $\boldsymbol{F}$ contains common factors representing effects of specific operations. By using factor analysis, it is possible to relate product defects to the associated packages and operations.

Multistage Variation Propagation

Inspection tests in an assembly line usually have functional overlap, and defects from successive inspection stations exhibit strong correlations. Modeling serially correlated defect counts is an important task for defect localization and yield prediction. Poisson regression models, such as the generalized event-count method [86] and its alternatives, can be utilized to account for serial correlations of defects in different inspection stations. Factor analysis methods based on hidden Markov models [87] can also be constructed to investigate how variations are propagated through assembly lines.

38.4.4 Object Detection

One area of DM image research is object detection, a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class in digital images and videos. An object-class detection is to localize and extract information of all objects in an image that belongs to a given class. Well-studied domains of object detection include face detection and pedestrian detection.

Face detection, as a specific case of object-class detection, focuses on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process. The main difficulties in face detection includes severe occlusion and variation of head poses.

Pedestrian detection provides fundamental information for semantic understanding of the video footages and considered an essential and significant in intelligent video surveillance system. It has an obvious extension to automotive applications due to enhance road safety and is offered as an advanced driver assistant system option by many car manufacturers in 2017. The major challenges of pedestrian detection arise from different possible posture, various appearance styles, the presence of occluding accessories, and frequent occlusions among pedestrians.

Existing methods for object detection generally fall into machine learning approaches and deep learning approaches. For machine learning approaches, it first extracts handcrafted features such as edges, corners, colors, etc. of the region of interest cropped by sliding window, then uses classifiers such as SVM to do the classification. On the other hand, deep learning approaches use CNN to extract the image features and thus able to do end-to-end object detection without specifically defining features. *Ross* et al. [88] prompted *regions with CNN* (R-CNN) to achieve dramatic improvements in accuracy of objects detection, which can be seen as a major breakthrough in the field of object detection. Subsequently, a series of R-CNN-based detection methods such as Fast R-CNN [89] and Faster R-CNN [90] were proposed. Those are classic two-stage object detection approaches that usually include a region proposal localization stage and a network classification stage. Then single shot detection methods, such as *single shot detector* (SSD) [91], RetinaNet [92], *you only look once* (YOLO) [93–95], etc., are developed. They skip the region proposal stage and run detection directly over a dense sampling of possible locations through a single CNN. The "one-stage" detection methods treat the detection as a single regression problem and are faster and simpler.

38.4.5 Surveillance of Public Health Systems

Public health surveillance is another important DM application. The objective of public health surveillance is to examine health trends, detect changes in disease incidence and death rates, and to plan, implement, and evaluate public health practice by systematically collecting, analyzing, and interpreting public health data (chronic or infectious diseases). Understanding challenges to nations' public health systems and how those challenges shift over time is of crucial importance for policymakers to establish effective strategies. In public health surveillance, the volume and velocity of data streams have dramatically grown in recent decades. In spite of the growing data volume, advances in information technology have enabled collection of cause-of-death data in a more timely manner.

The availability of public health big data provides a comprehensive picture of health system status in terms of the causes of significant population-wide changes, the underlying risks, the changes in the pattern of health-related losses, etc. Numerous efforts have been made to monitor and evaluate the health of populations by taking advantage of public health big data and data mining techniques. For example, Google flu trend (GFT) is a data analytics model developed by Google for predicting weekly reported influenza-like illness (ILI) rates using instant query data [96]. However, as reported in Refs. [96, 97], GFT failed to provide accurate predictions, and predicted more than double the actual rate of doctor visits for ILI reported by the Centers for Disease Control and Prevention during the 2012–2013 season. The model's failure has led to a large number of research papers aiming to improve its predictive accuracy [98–100]. One representative method was ARGO, proposed in Ref. [98], which not only incorporated seasonality in historical ILI rates, but also captured changes in the public's online searching behaviors over time.

38.5 Concluding Remarks

While DM and KDD methods are gaining recognition and have become very popular in many companies and enterprises, the success of these methods is still somewhat limited. Below, we discuss a few obstacles.

First, the success of DM depends on a close collaboration of subject-matter experts and data modelers. In practice, it is often easy to identify the right subject-matter expert, but difficult to find the qualified data modeler. While the data modeler must be knowledgeable and familiar with DM methods, it is more important to be able to formulate real problems such that the existing methods can be applied. In reality, traditional academic training mainly focuses on knowledge of modeling

algorithms and lacks training in problem formulation and interpretation of results. Consequently, many modelers are very efficient in fitting models and algorithms to data, but have a hard time determining when and why they should use certain algorithms. Similarly, the existing commercial DM software systems include many sophisticated algorithms, but there is a lack of guidance on which algorithms to use.

Second, implementation of DM is difficult to apply effectively across an industry. Although it is clear that extracting hidden knowledge and trends across an industry would be useful and beneficial to all companies in the industry, it is typically impossible to integrate the detailed data from competing companies due to confidentiality and proprietary issues. Currently, the industry practice is that each company will integrate their own detailed data with the more general, aggregated industry-wide data for knowledge extraction. It is obvious that this approach will be significantly less effective than the approach of integrating the detailed data from all competing companies. It is expected that, if these obstacles can be overcome, the impact of the DM and KDD methods will be much more prominent in industrial and commercial applications.

References

1. Berry, M.J.A., Linoff, G.: Mastering Data Mining: The Art and Science of Customer Relationship Management. Wiley, New York (2000)
2. Wegman, E.: Data Mining Tutorial, Short Course Notes, Interface 2001 Symposium. Cosa Mesa, Californien (2001)
3. Adriaans, P., Zantinge, D.: Data Mining. Addison-Wesley, New York (1996)
4. Friedman, J.H.: Data Mining and Statistics: What Is the Connection? Technical Report. Stat. Dep., Stanford University (1997)
5. Clark, K.B., Fujimoto, T.: Product development and competitiveness. J. Jpn. Int. Econ. **6**(2), 101–143 (1992)
6. LaBahn, D.W., Ali, A., Krapfel, R.: New product development cycle time. The influence of project and process factors in small manufacturing companies. J. Bus. Res. **36**(2), 179–188 (1996)
7. Han, J., Kamber, M.: Data Mining: Concept and Techniques. Morgan Kaufmann, San Francisco (2001)
8. Hastie, T., Friedman, J.H., Tibshirani, R.: Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, Berlin/Heidelberg/New York (2001)
9. Weisberg, S.: Applied Linear Regression. Wiley, New York (1980)
10. Seber, G.: Multivariate Observations. Wiley, New York (1984)
11. Neter, J., Kutner, M.H., Nachtsheim, C.J., Wasserman, W.: Applied Linear Statistical Models, 4th edn. Irwin, Chicago (1996)
12. Hoerl, A.E., Kennard, R.: Ridge regression: biased estimation of nonorthogonal problems. Technometrics. **12**, 55–67 (1970)
13. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B. **58**, 267–288 (1996)
14. Agresti, A.: An Introduction to Categorical Data Analysis. Wiley, New York (1996)
15. Hand, D.: Discrimination and Classification. Wiley, Chichester (1981)
16. McCullagh, P., Nelder, J.A.: Generalized Linear Models, 2nd edn. Chapman Hall, New York (1989)
17. Hastie, T., Tibshirani, R.: Generalized Additive Models. Chapman Hall, New York (1990)
18. Cleveland, W.S.: Robust locally-weighted regression and smoothing scatterplots. J. Am. Stat. Assoc. **74**, 829–836 (1979)
19. Eubank, R.L.: Spline Smoothing and Nonparametric Regression. Marcel Dekker, New York (1988)
20. Wahba, G.: Spline Models for Observational Data, Applied Mathematics, vol. 59. SIAM, Philadelphia (1990)
21. Härdle, W.: Applied Non-parametric Regression. Cambridge University Press, Cambridge (1990)
22. Biggs, D., deVille, B., Suen, E.: A method of choosing multiway partitions for classification and decision trees. J. Appl. Stat. **18**(1), 49–62 (1991)
23. Ripley, B.D.: Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge (1996)
24. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth, Belmont (1984)
25. Morgan, J.N., Sonquist, J.A.: Problems in the analysis of survey data, and a proposal. J. Am. Stat. Assoc. **58**, 415–434 (1963)
26. Fielding, A.: Binary segmentation: the automatic interaction detector and related techniques for exploring data structure. In: O'Muircheartaigh, C.A., Payne, C. (eds.) The Analysis of Survey Data, Volume I: Exploring Data Structures, pp. 221–258. Wiley, New York (1977)
27. Loh, W.Y., Vanichsetakul, N.: Tree-structured classification via generalized discriminant analysis. J. Am. Stat. Assoc. **83**, 715–728 (1988)
28. Chaudhuri, W.D.L., Loh, W.Y., Yang, C.C., Generalized Regression Trees: Stat. Sin. **5**, 643–666 (1995)
29. Loh, W.Y., Shih, Y.S.: Split-selection methods for classification trees. Stat. Sin. **7**, 815–840 (1997)
30. Friedman, J.H., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Ann. Stat. **28**, 337–407 (2000)
31. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm, machine learning. In: Kaufmann, M. (ed.) Proceedings of the Thirteenth International Conference, Bari, Italy, pp. 148–156 (1996)
32. Breiman, L.: Bagging predictors. Mach. Learn. **26**, 123–140 (1996)
33. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**, 1189–1232 (2001)
34. Friedman, J.H.: Stochastic gradient boosting. Comput. Stat. Data Anal. **38**(4), 367–378 (2002)
35. Friedman, J.H.: Multivariate adaptive regression splines (with discussion). Ann. Stat. **19**, 1–141 (1991)
36. Friedman, J.H., Silverman, B.W.: Flexible parsimonious smoothing and additive modeling. Technometrics. **31**, 3–39 (1989)
37. Lippmann, R.P.: An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, 4–22 April (1987)
38. Haykin, S.S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice Hall, Upper Saddle River (1999)
39. White, H.: Learning in neural networks: a statistical perspective. Neural Comput. **1**, 425–464 (1989)
40. Barron, A.R., Barron, R.L., Wegman, E.J.: Statistical learning networks: a unifying view, computer science and statistics. In: Wegman, E.J., Gantz, D.T., Miller, J.J. (eds.) Proceedings of the 20th Symposium on the Interface 1992, pp. 192–203. American Statistical Association, Alexandria (1992)
41. Cheng, B., Titterington, D.M.: Neural networks: a review from a statistical perspective (with discussion). Stat. Sci. **9**, 2–54 (1994)
42. Rumelhart, D., Hinton, G., Williams, R.: Learning internal representations by error propagation. In: Rumelhart, D.E., McClelland, J.L. (eds.) Parallel Distributed Processing: Explorations in the Microstructures of Cognition, vol. 1: Foundations, pp. 318–362. MIT, Cambridge (1986)

43. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE. **86**(11), 2278–2324 (1998)
44. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Knowledge Discovery and Data Mining. **2**(2), 121–167 (1998)
45. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
46. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press, Cambridge (2000)
47. Huber, P.: Robust estimation of a location parameter. Ann. Math. Stat. **53**, 73–101 (1964)
48. Dasarathy, B.V.: Nearest Neighbor Pattern Classification Techniques. IEEE Comput. Soc., New York (1991)
49. Hastie, T., Tibshirani, R.: Discriminant adaptive nearest-neighbor classification. IEEE Trans. Pattern Mach. Intell. **18**, 607–616 (1996)
50. Hastie, T., Tibshirani, R., Buja, A.: Flexible discriminant and mixture models. In: Kay, J., Titterington, M. (eds.) Statistics and Artificial Neural Networks. Oxford University Press, Oxford (1998)
51. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT, Cambridge (1992)
52. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming: An Introduction. Morgan Kaufmann, San Francisco (1998)
53. Smith, P.W.H.: Genetic programming as a data-mining tool. In: Abbass, H.A., Sarker, R.A., Newton, C.S. (eds.) Data Mining: a Heuristic Approach, pp. 157–173. Idea Group Publishing, London (2002)
54. Gordon, A.: Classification, 2nd edn. Chapman Hall, New York (1999)
55. Ralambondrainy, H.: A conceptual version of the *K*-means algorithm. Pattern Recogn. Lett. **16**, 1147–1157 (1995)
56. Zhang, P., Wang, X., Song, P.: Clustering categorical data based on distance vectors. J. Am. Stat. Assoc. **101**, 355–367 (2006)
57. Hartigan, J.A., Wong, M.A.: A *K*-means clustering algorithm. Appl. Stat. **28**, 100–108 (1979)
58. Park, H., Jun, C.: A simple and fast algorithm for K-Medoids clustering. Expert Syst. Appl. **36**, 3336–3341 (2009)
59. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering cluster in large spatial databases. In: Proceedings of 1996 International Conference on Knowledge Discovery and Data Mining (KDD96), Portland, 226–231 (1996)
60. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science. **315**(5814), 972–976 (2007)
61. Kohonen, T.: Self-Organization and Associative Memory, 3rd edn. Springer, Berlin Heidelberg New York (1989)
62. Chen, Y., Yang, H.: Self-organized neural network for the quality control of 12-lead ECG signals. Physiol. Meas. **33**, 1399–1418 (2012)
63. Haughton, D., Deichmann, J., Eshghi, A., Sayek, S., Teebagy, N., Topi, H.: A review of software packages for data mining. Am. Stat. **57**(4), 290–309 (2003)
64. Friedman, J., Hastie, T., Tibshirani, R.: The Elements of Statistical Learning, vol. 1: Springer Series in Statistics, New York (2001)
65. Fawcett, T., Provost, F.: Activity monitoring: noticing interesting changes in behavior. In: Proceedings of KDD-99, San Diego 1999, pp. 53–62, San Diego (1999)
66. Woodall, W.H., Tsui, K.-L., Tucker, G.R.: A review of statistical and fuzzy quality control based on categorical data. Front. Stat. Qual. Control. **5**, 83–89 (1997)
67. Montgomery, D.C., Woodall, W.H.: A discussion on statistically-based process monitoring and control. J. Qual. Technol. **29**, 121–162 (1997)
68. Hayter, A.J., Tsui, K.-L.: Identification and qualification in multivariate quality control problems. J. Qual. Technol. **26**(3), 197–208 (1994)
69. Mason, R.L., Champ, C.W., Tracy, N.D., Wierda, S.J., Young, J.C.: Assessment of multivariate process control techniques. J. Qual. Technol. **29**, 140–143 (1997)
70. Jiang, W., Au, S.-T., Tsui, K.-L.: A statistical process control approach for customer activity monitoring, Technical Report, AT&T Labs (2004)
71. West, M., Harrison, J.: Bayesian Forecasting and Dynamic Models, 2nd edn. Springer, New York (1997)
72. Fraley, C., Raftery, A.E.: Model-based clustering, discriminant analysis, and density estimation. J. Am. Stat. Assoc. **97**, 611–631 (2002)
73. Taguchi, G.: Introduction to Quality Engineering: Designing Quality into Products and Processes. Asian Productivity Organization, Tokyo (1986)
74. Nair, V.N.: Taguchi's parameter design: a panel discussion. Technometrics. **34**, 127–161 (1992)
75. Tsui, K.-L.: An overview of Taguchi method and newly developed statistical methods for robust design. IIE Trans. **24**, 44–57 (1992)
76. Tsui, K.-L.: A critical look at Taguchi's modeling approach for robust design. J. Appl. Stat. **23**, 81–95 (1996)
77. Taguchi, G., Chowdhury, S., Wu, Y.: The Mahalanobis–Taguchi System. McGraw-Hill, New York (2001)
78. Taguchi, G., Jugulum, R.: The Mahalanobis–Taguchi Strategy: A Pattern Technology System. Wiley, New York (2002)
79. Woodall, W.H., Koudelik, R., Tsui, K.-L., Kim, S.B., Stoumbos, Z.G., Carvounis, C.P.: A review and analysis of the Mahalanobis–Taguchi system. Technometrics. **45**(1), 1–15 (2003)
80. Kusiak, A., Kurasek, C.: Data mining of printed–circuit board defects. IEEE Trans. Robot. Autom. **17**(2), 191–196 (2001)
81. Kusiak, A.: Rough set theory: a data mining tool for semiconductor manufacturing. IEEE Trans. Electron. Packag. Manuf. **24**(1), 44–50 (2001)
82. Ultsch, A.: Information and Classification: Concepts, Methods and Applications. Springer, Berlin Heidelberg New York (1993)
83. Wong, A.Y.: A statistical approach to identify semiconductor process equipment related yield problems. In: IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, Paris 1997, pp. 20–22. IEEE Computer Society, Paris (1997)
84. ANSI: Am. Nat. Standards Institute, IPC-9261, In-Process DPMO and Estimated Yield for PWB (2002)
85. Baron, M., Lakshminarayan, C.K., Chen, Z.: Markov random fields in pattern recognition for semiconductor manufacturing. Technometrics. **43**, 66–72 (2001)
86. King, G.: Event count models for international relations: generalizations and applications. Int. Stud. Q. **33**(2), 123–147 (1989)
87. Smyth, P.: Hidden Markov models for fault detection in dynamic systems. Pattern Recogn. **27**(1), 149–164 (1994)
88. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 580–587 (2014)
89. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE international conference on computer vision, 1440–1448 (2015)
90. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NIPS), pp. 91–99 (2015)
91. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: SSD: single shot MultiBox detector. Lect. Notes Comput. Sci, 21–37 (2016)

92. Ye, T., Wang, B., Song, P., Li, J.: Automatic railway traffic object detection system using feature fusion refine neural network under shunting mode. Sensors. **18**, 1916 (2018)
93. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
94. Redmon, J., Farhadi, A.: YOLO9000: Better, Faster, Stronger. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
95. Redmon, J., Farhadi, A.: YOLOv3: an incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
96. Lazer, D., Kennedy, R., King, G., Vespignani, A.: The parable of Google Flu: traps in big data analysis. Science. **343**, 1203–1205 (2014)
97. Butler, D.: When Google got flu wrong. Nature. **494**, 155 (2013)
98. Yang, S., Santillana, M., Kou, S.: ARGO: a model for accurate estimation of influenza epidemics using Google search data. Proc. Natl. Acad. Sci. (2015)
99. Copeland, P., Romano, R., Zhang, T., Hecht, G., Zigmond, D., Stefansen, C.: Google disease trends: an update. Nature. **457**, 1012–1014 (2013)
100. Santillana, M., Zhang, D.W., Althouse, B.M., Ayers, J.W.: What can digital disease detection learn from (an external revision to) Google Flu Trends? Am. J. Prev. Med. **47**, 341–347 (2014)

Kwok-Leung Tsui is a chair professor in the School of Data Science at the City University of Hong Kong. He has a Ph.D. in statistics from the University of Wisconsin at Madison. Dr. Tsui is an (elected) fellow of American Statistical Association and American Society for Quality, and was a recipient of the NSF Young Investigator Award. He is currently the departmental editor of the IISE Transactions and a member of the Management Committee for Technometrics.

Victoria Chen is a professor in the Department of Industrial, Manufacturing, and Systems Engineering and director of the Center on Stochastic Modeling, Optimization, and Statistics at the University of Texas at Arlington. She is currently serving as an executive secretary on the INFORMS Board. She is a guest coeditor for the Annals of Operations Research.

Wei Jiang is a distinguished professor in Antai College of Economics and Management at Shanghai Jiao Tong University. He obtained his Ph.D. degree in industrial engineering and engineering management from Hong Kong University of Science and Technology in 2000. Prior to joining Shanghai Jiao Tong University, he worked as a statistical consultant at AT&T Labs, Morristown. His current research activities include statistical methods for quality control, data mining, and enterprise intelligence.

Fangfang Yang is a research fellow in School of Data Science at the City University of Hong Kong. She earned her Ph.D. in system engineering and engineering management from the City University of Hong Kong in 2017. Current research areas of Dr. Yang include prognostics and health management, degradation modeling, and deep learning.

Chen Kan is an assistant professor in the Department of Industrial, Manufacturing, and Systems Engineering, the University of Texas at Arlington. He earned his Ph.D. in industrial and manufacturing engineering from the Pennsylvania State University in 2018. Current research areas of Dr. Kan include complex systems modeling and monitoring, smart manufacturing and IoT, and health informatics. He is a member of IEEE, IISE, and INFORMS.