Power System
Short-term Load Forecasting

Vom Fachbereich 18

Elektrotechnik und Informationstechnik

der Technischen Universität Darmstadt

zur Erlangung der Würde

einer Doktor-Ingenieurin (Dr.-Ing.)

vorgelegte Dissertation

von

M. Sc. Jingfei Yang

geboren am 12. Februar 1974

in Beijing, China

Referent:	Prof. Dr.-Ing Jürgen Stenzel
Korreferent:	Prof. dr hab. Inż. Tadeusz Łobos
Tag der Einreichung:	06-12-2005
Tag der mündlichen Prüfung:	17-02-2006

D17
Darmstädter Dissertation

Berichte aus der Energietechnik

Jingfei Yang

Power System
Short-term Load Forecasting

D 17 (Diss. TU Darmstadt)

Acknowledgements

This work was supported by the European-Chinese Link in Electrical Engineering project from European Commission (project number: ASI/B7-301/98/679-026). I got the financial support from this project to study in the institute of electrical power systems, Darmstadt University of Technology.

First of all I would like to show my special thanks to Prof. Dr. Jürgen Stenzel. He supervised the whole process of my research in Darmstadt, and has provided many technical suggestions. I benefit a lot from his knowledge and experience on power system.

I wish to thank Prof. Dr. Tadeusz Łobos for carefully reviewing my thesis and giving his advice, Prof. Glesner, Prof. Gershman and Prof. Meißner for agreeing to serve on my thesis committee.

Mr. Greg Dew has read through the thesis very carefully and corrected the writing. Here I want to show my gratitude to him. Dr. Jörg Becker, Mr. Wei Li and Dr. Chuanwen Jiang have kindly provided the original data of my experiments. I am thankful for their support.

I wish to thank Mr. Torben Dietermann and Mr. Nis Martensen for their help on the thesis summary in German language.

Many thanks to every member in the institute of electrical power systems. Without their help and encouragement during the two years I wouldn’t have finished the PhD research work.

Many thanks to Prof. Jürgen Stenzel, Prof. Haozhong Cheng, Prof. Ettore Bompard and Prof. Roberto Napoli who gave me the chance to study in Germany.

I would also like to thank my parents, my parents in law, my husband and my son for their support and understanding of my research in Darmstadt.

Darmstadt, February 2006

Jingfei Yang

To my father Yihan Yang, my mother Xiuyan Qu, and my husband Wei Li

Contents

1 INTRODUCTION 1
  1.1 MOTIVATION 1
  1.2 OBJECTIVES 2
  1.3 THESIS ORGANIZATION OUTLINE AND CONVENTIONS 2
2 BASIC CONCEPTS OF SHORT-TERM LOAD FORECASTING 5
  2.1 CHARACTERISTICS OF THE POWER SYSTEM LOAD 5
  2.2 CLASSIFICATION OF DEVELOPED STLF METHODS 8
  2.3 REQUIREMENTS OF THE STLF PROCESS 14
  2.4 DIFFICULTIES IN THE STLF 15
3 HISTORICAL DATA PRETREATMENT 19
  3.1 OVERVIEW OF LOAD BAD DATA 19
  3.2 BAD DATA DETECTION AND REPLACEMENT 21
    3.2.1 Basic idea of second order difference 21
    3.2.2 Consideration of the segment with both good and bad data 24
    3.2.3 Selection of interval V 25
  3.3 SMOOTHING THE LOAD CURVE 26
  3.4 CASE STUDY 28
4 REGRESSION TREE BASED STLF 35
  4.1 CART REGRESSION TREE ALGORITHM 35
  4.2 APPLICATION OF CART IN SHORT-TERM LOAD FORECASTING 37
    4.2.1 Non-increment method 37
    4.2.2 Increment regression tree 39
    4.2.3 Tree prediction result combination 42
    4.2.4 Finding the desert border variable 45
  4.3 HISTORICAL DATA SELECTION 47
  4.4 CASE STUDY 51
5 SHORT-TERM LOAD FORECASTING BASED ON SUPPORT VECTOR MACHINE 61
  5.1 SUPPORT VECTOR MACHINE THEORY 61
    5.1.1 Support vector regression 61
    5.1.2 Support vector clustering 63
  5.2 LIBSVM SOLUTION 65
    5.2.1 Basic solution 65
    5.2.2 Selection of working set 66
  5.3 APPLICATION OF SUPPORT VECTOR MACHINE TO STLF 66
    5.3.1 Clustering of historical load 68
    5.3.2 Repetitious clustering 68
    5.3.3 Slight revision of the algorithm to allow overlapping clusters 70
  5.4 REGRESSION OF THE CLUSTERED DATA 72
    5.4.1 Decision tree application 72
    5.4.2 Support vector regression for the clustered data 73

5.5 CALCULATION RESULTS 73
  5.5.1 Conditions 73
  5.5.2 Prediction results 74

6 INTEGRATIVE ALGORITHM 79

6.1 LOAD FORECASTING FOR HOLIDAY AND ANOMALOUS DAYS 79
6.2 INTEGRATION OF SVM, RT AND OTHER TRADITIONAL ALGORITHMS 81
  6.2.1 Integration of SVM and RT 81
  6.2.2 Extended dispersion calculation in RTSVM forecasted result 84
  6.2.3 Integration of different algorithms 85
  6.2.4 Smoothing of the forecasted load curve 87
6.3 GENERALIZED PROGRAMMING SYSTEM DESIGN 88
6.4 CASE STUDY 92

7 CONCLUSION AND OUTLOOK 97

7.1 CONCLUSION 97
7.2 EXPERIENCES AND OUTLOOK 100

APPENDIX A METHODOLOGY FOR BUILDING A CLASSIFICATION TREE 103

APPENDIX B FORECASTING RESULT FOR FRANKFURT SUBSTATION 109

APPENDIX C ZUSAMMENFASSUNG IN DEUTSCH 123

LIST OF SYMBOLS AND ABBREVIATIONS 125

LITERATURE 129

LEBENSLAUF 133

1 Introduction

1.1 Motivation

Load forecasting is an important component for power system energy management system. Precise load forecasting helps the electric utility to make unit commitment decisions, reduce spinning reserve capacity and schedule device maintenance plan properly. Besides playing a key role in reducing the generation cost, it is also essential to the reliability of power systems. The system operators use the load forecasting result as a basis of off-line network analysis to determine if the system might be vulnerable. If so, corrective actions should be prepared, such as load shedding, power purchases and bringing peaking units on line.

Since in power systems the next days' power generation must be scheduled everyday, day-ahead short-term load forecasting (STLF) is a necessary daily task for power dispatch. Its accuracy affects the economic operation and reliability of the system greatly. Underprediction of STLF leads to insufficient reserve capacity preparation and, in turn, increases the operating cost by using expensive peaking units. On the other hand, overprediction of STLF leads to the unnecessarily large reserve capacity, which is also related to high operating cost. It is estimated that in the British power system every 1% increase in the forecasting error is associated with an increase in operating costs of 10 million pounds per year [1].

In spite of the numerous literatures on STLF published since 1960s, the research work in this area is still a challenge to the electrical engineering scholars because of its high complexity. How to estimate the future load with the historical data has remained a difficulty up to now, especially for the load forecasting of holidays, days with extreme weather and other anomalous days. With the recent development of new mathematical, data mining and artificial intelligence tools, it is potentially possible to improve the forecasting result.

With the recent trend of deregulation of electricity markets, STLF has gained more importance and greater challenges. In the market environment, precise forecasting is the basis of electrical energy trade and spot price establishment for the system to gain the minimum electricity purchasing cost. In the real-time dispatch operation, forecasting error causes more purchasing electricity cost or breaking-contract penalty cost to keep the electricity supply and consumption balance. There are also some modifications of STLF models due to the implementation of the electricity market. For example, the demand-side management and volatility of spot markets causes the consumer's active response to the electricity price. This should be considered in the forecasting model in the market environment.

1.2 Objectives

Due to some data measurement and transmission problems, in the historical database there might be some bad data, which are far away from their real values. The existence of bad data in historical load curve affects the precision of load forecasting results. One of the objectives of this research work is to find a way to detect the bad data, eliminate them and evaluate the real data.

Since precise load forecasting remains a great challenge, another objective of this work is to develop some new and practical models and algorithms with some up-to-date techniques. The power system operators always have very good intuition in manual load forecasting with their long time working experience. Therefore it is an attempt to combine the operators' experience with the presented models in a convenient way.

As can be seen from the bibliography, many methods have been developed for STLF. From the experimental results the conclusion can be drawn that different methods might outperform the others in different situations, i.e. one method might gain the lowest prediction error for one time point, and another might for another time point. How to choose a good method or the combination of different methods for different situations becomes necessary. This research tries to develop a comprehensive method selection to fulfill this goal.

1.3 Thesis Organization Outline and Conventions

The following chapters of this thesis can be mainly divided into 3 parts: the pretreatment of the historical data, the load forecasting with some proposed methods, and the integrative algorithm to combine the various approaches. The thesis is organized as follows.

Chapter 2 gives an overview of the short-term load forecasting problem. The property of the system load, various forecasting methods, and the difficulty in forecasting are introduced. In chapter 3 the pretreatment of historical load data is discussed. This includes bad data detection and load curve smoothing. A regression tree algorithm is applied to short-term load forecasting, which is explained in detail in chapter 4. The experts' experience is combined with the algorithm to enhance its performance. In chapter 5 a support vector machine approach is proposed, which is composed of the cascaded modules of clustering, classification and fine regression. Chapter 6 describes the forecasting from a systematic point of view, including the integrative algorithm to combine different forecasting results and the generalized programming system device. The final chapter summarizes the research work and closes the thesis.

In this thesis the following conventions will be employed unless otherwise stated.

- The number of sample load points of per day is 96, i.e. the sampling interval is 15 minutes.
- The examples are mainly from the Shanghai Power Grid data. German data have also been employed for the generalization of the methods. But since Shanghai load is more difficult to predict, Shanghai data is the default data for the case study.
- Mean Absolute Percentage Error (MAPE) will be employed to measure the error of the methods.
- For simplicity's sake, term “target load”, “target day”, and “target time point” are used to represent respectively “the load which is to be forecasted”, “the day for which the load is to be forecasted”, and “the time point at which the load is to be forecasted”, and “point i” is used to represent the i^{th} point of the daily load curve.

2 Basic Concepts of Short-term Load Forecasting

2.1 Characteristics of the Power System Load

The system load is the sum of all the consumers' load at the same time. The objective of system STLF is to forecast the future system load. Good understanding of the system characteristics helps to design reasonable forecasting models and select appropriate models in different situations. Various factors influence the system load behavior, which can be mainly classified into the following categories

- weather
- time
- economy
- random disturbance.

The effects of all these factors are introduced as follows to provide a basic understanding of the load characteristics.

Weather

Weather factors include temperature, humidity, precipitation, wind speed, cloud cover, light intensity and so on. The change of the weather causes the change of consumers' comfort feeling and in turn the usage of some appliances such as space heater, water heater and air conditioner. Weather-sensitive load also includes appliance of agricultural irrigation due to the need of the cultivated plants. In the areas where summer and winter have great meteorological difference, the load patterns differ greatly. Fig. 2.1 shows the typical different seasonal weekday Shanghai load profiles of the year.

Normally the intraday temperatures are the most important weather variables in terms of their effects on the load; hence they are often selected as the independent variables in STLF. Temperatures of the previous days also affect the load profile. For example, continuous high temperature days might lead to heat buildup and in turn a new system peak. Humidity is also an important factor, because it affects the human being's comfort feeling greatly. People feel hotter in the environment of 35℃ and 70% relative humidity than in the environment of 37℃ and 50% relative humidity. That's why THI (temperature-humidity index) is sometimes employed as an affecting factor of load forecasting. Furthermore, WCI (wind chill index) is another factor that measures the cold feeling. It is a meaningful topic to select the appropriate weather variables as the inputs of STLF.

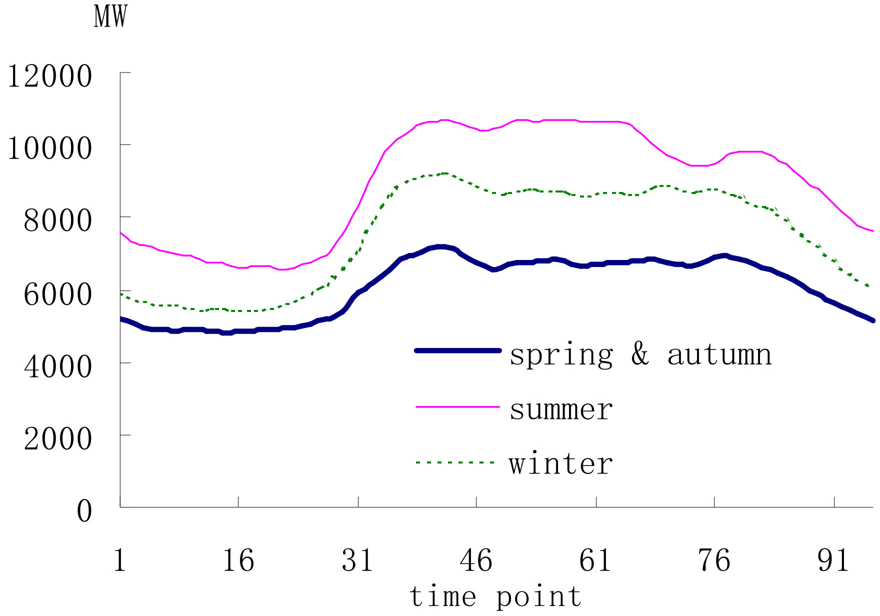


Fig. 2.1 Typical seasonal workday Shanghai load profiles

Time

Time factors influencing the load include time point of the day, holiday property, weekday/weekend property and season property. From the observation of the load curves it can be seen that there are certain rules of the load variation with the time point of the day. For example, the typical load curve of the normal winter weekdays (from Monday to Friday) of the E.ON power grid in Germany is shown in Fig. 2.2, with the sample interval of 15 minutes, i.e. there are altogether 96 sample points in one day. The load is low and stable from 0:00 to 6:00; it rises from around 6:00 to 9:00 and then becomes flat again until around 12:00; then it descends gradually until 17:00; thereafter it rises again until 19:00; it descends again until the end of the day, but in between there is a sudden jump at 22:00 because the electricity price becomes lower at this time. Actually this load variation with time reflects the arrangement of people’s daily life: working time, leisure time and sleeping time.

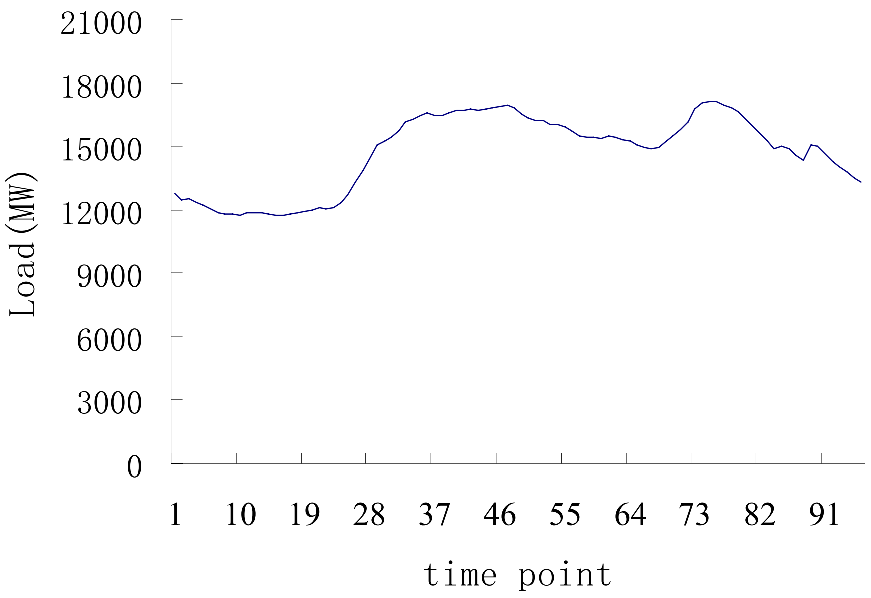


Fig. 2.2 Typical load curve of the normal winter weekdays of the E.ON power grid

There are also some other rules of load variation with time. The weekend or holiday load curve is lower than the weekday curve, due to the decrease of working load. Shifts to and from daylight savings time and start of the school year also contribute to the significant change of the previous load profiles.

Periodicity is another property of the load curve. There is very strong daily, weekly, seasonal and yearly periodicity in the load data. Taking good use of this property can benefit the load forecasting result.

Economy

Electricity is a kind of commodity. The economic situation also influences the utilization of this commodity. Economic factors, such as the degree of industrialization, price of electricity and load management policy have significant impacts on the system load growth/decline trend. With the development of modern electricity markets, the relationship between electricity price and load profile is even stronger. Although time-of-use pricing and demand-side management had arrived before deregulation, the volatility of spot markets and incentives for consumers to adjust loads are potentially of a much greater magnitude. At low prices, elasticity is still negligible, but at times of extreme conditions, price-induced rationing is a much more likely scenario in a deregulated market compared to that under central planning.

Random Disturbance

The modern power system is composed of numerous electricity users. Although it is not possible to predict how each individual user consumes the energy, the amount of the total loads of all the small users shows good statistical rules and in turn, leads to smooth load curves. This is the groundwork of the load forecasting work. But the startup and shutdown of the large loads, such as steel mill, synchrotrons and wind tunnels, always lead to an obvious impulse to the load curve. This is a random disturbance, since for the dispatchers, the startup and shutdown time of these users is quite random, i.e. there is no obvious rule of when and how they get power from the grid. When the data from such a load curve are used in load forecasting training, the impulse component of the load adds to the difficulty of load forecasting. Special events, which are known in advance but whose effect on load is not quite certain, are another source of random disturbance. A typical special event is, for example, a world cup football match, which the dispatchers know for sure will cause increasing usage of television, but cannot best decide the amount of the usage. Other typical events include strikes and the government's compulsory demand-side management due to forecasted electricity shortage.

2.2 Classification of Developed STLF Methods

In terms of lead time, load forecasting is divided into four categories:

- Long-term forecasting with the lead time of more than one year
- Mid-term forecasting with the lead time of one week to one year
- Short-term load forecasting with the lead time of 24 to 168 hours
- Very short-term load forecasting with the lead time shorter than one day

Different categories of forecasting serve for different purposes. In this thesis short-term load forecasting which serves the next day(s) unit commitment and reliability analysis is focused on.

The research approaches of short-term load forecasting can be mainly divided into two categories: statistical methods and artificial intelligence methods. In statistical methods, equations can be obtained showing the relationship between load and its relative factors after training the historical data, while artificial intelligence methods try to imitate human beings' way of thinking and reasoning to get knowledge from the past experience and forecast the future load.

The statistical category includes multiple linear regression [2], stochastic time series [3], general exponential smoothing [4], state space [5], etc. Recently support vector regression (SVR) [6, 7], which is a very promising statistical learning method, has also been applied to short-term load forecasting and has shown good results. Usually statistical methods can predict the load curve of ordinary days very well, but they lack the ability to analyze the load property of holidays and other anomalous days, due to the inflexibility of their structure. Expert system [8], artificial neural network (ANN) [9] and fuzzy inference [10] belong to the artificial intelligence category. Expert systems try to get the knowledge of experienced operators and express it in an "if…then" rule, but the difficulty is sometimes the experts' knowledge is intuitive and could not easily be expressed. Artificial neural network doesn't need the expression of the human experience and aims to establish a network between the input data set and the observed outputs. It is good at dealing with the nonlinear relationship between the load and its relative factors, but the shortcoming lies in overfitting and long training time. Fuzzy inference is an extension of expert systems. It constructs an optimal structure of the simplified fuzzy inference that minimizes model errors and the number of the membership functions to grasp nonlinear behaviour of short-term loads, yet it still needs the experts' experience to generate the fuzzy rules. Generally artificial intelligence methods are flexible in finding the relationship between load and its relative factors, especially for the anomalous load forecasting.

Some main STLF methods are introduced as follows.

Regression Methods

Regression is one of most widely used statistical techniques. For load forecasting regression methods are usually employed to model the relationship of load consumption and other factors such as weather, day type and customer class.

Engle et al. [11] presented several regression models for the next day load forecasting. Their models incorporate deterministic influences such as holidays, stochastic influences such as average loads, and exogenous influences such as weather. [12], [13], [14] and [15] describe other applications of regression models applied to load forecasting.

Time Series

Time series methods are based on the assumption that the data have an internal structure, such as autocorrelation, trend or seasonal variation. The methods detect and explore such a structure.

Time series have been used for decades in such fields as economics, digital signal processing, as well as electric load forecasting. In particular, ARMA (autoregressive moving average), ARIMA (autoregressive integrated moving average) and ARIMAX (autoregressive integrated moving average with exogenous variables) are the most often used classical time series methods. ARMA models are usually used for stationary processes while ARIMA is an extension of ARMA to nonstationary processes. ARMA and ARIMA use the time and load as the only input parameters. Since load generally depends on the weather and time of the day, ARIMAX is the most natural tool for load forecasting among the classical time series models.

Fan and McDonald[16] and Cho et al. [17] described implementations of ARIMAX models for load forecasting. Yang et al. [18] used an evolutionary programming (EP) approach to identify the ARMAX model parameters for one day to one week ahead hourly-load-demand-forecasting. The evolutionary programming is a method for simulating evolution and constitutes a stochastic optimization algorithm. Yang and Huang [19] proposed a fuzzy autoregressive moving average with exogenous input variables (FARMAX) for one day ahead hourly load forecasting.

Neural Networks

The use of artificial neural networks (ANN or simply NN) has been a widely studied load forecasting technique since 1990 [20]. Neural networks are essentially non-linear circuits that have the demonstrated capability to do non-linear curve fitting.

The outputs of an artificial neural network are some linear or non-linear mathematical function of its inputs. The inputs may be the outputs of other network elements as well as actual network inputs. In practice network elements are arranged in a relatively small number of connected layers of elements between network inputs and outputs. Feedback paths are sometimes used.

In applying a neural network to load forecasting, one must select one of a number of architectures (e.g. Hopfield, back propagation, Boltzmann machine), the number and connectivity of layers and elements, use of bi-directional or uni-directional links and the number format (e.g. binary or continuous) to be used by inputs and outputs [19].

The most popular artificial neural network architecture for load forecasting is back propagation. This network uses continuously valued functions and supervised learning. That is, under supervised learning, the actual numerical weights assigned to element inputs are determined by matching historical data (such as time and weather) to desired outputs (such as historical loads) in a pre-operational "training session". Artificial neural networks with unsupervised learning do not require pre-operational training.

Bakirtzis et al. [62] developed an ANN based short-term load forecasting model for the Energy Control Center of the Greek Public Power Corporation. In the development they used a fully connected three-layer feed forward ANN and a back propagation algorithm was used for training. Input variables include historical hourly load data, temperature, and the day of week. The model can forecast load profiles from one to seven days. Also Papalexopoulos et al. [22] developed and implemented a multi-layered feed forward ANN for short-term system load forecasting. In the model three types of variables are used as inputs to the neural networks: seasonal related inputs, weather related inputs, and historical loads. Khotanzad et al [23] described a load forecasting system known as ANNSTLF. It is based on multiple ANN strategy that captures various trends in the data. In the development they used a multilayer perceptron trained with an error back propagation algorithm. ANNSTLF can consider the effect of temperature and relative humidity on the load. It also contains forecasters that can generate the hourly temperature and relative humidity forecasts needed by the system. An improvement of the above system was described in [24]. In the new generation, ANNSTLF includes two ANN forecasters: one predicts the base load and the other forecasts the change in load. The final forecast is computed by adaptive combination of these forecasts. The effect of humidity and wind speed are considered through a linear transformation of temperature. At the time it was reported in [23], ANNSTLF was being used by 35 utilities across the USA and Canada. Chen et al. [25] also developed a three layer fully connected feed forward neural network and a back propagation algorithm was used as the training method. Their ANN though considers electricity price as one of the main characteristics of the system load. Many published studies use artificial neural networks in conjunction with other forecasting techniques such as time series [26] and fuzzy logic [27].

Similar Day Approach

This approach [28] is based on searching historical data for days within one, two or three years with similar characteristics to the forecast day. Similar characteristics include weather, day of the week and the date. The load of a similar day is considered as a forecast. Instead of a single similar day load, the forecast can be a linear combination or regression procedure that can include several similar days. The trend coefficients can be used for similar days in the previous years.

Expert Systems

Rule-based forecasting makes use of rules, which are often heuristic in nature, to do accurate forecasting. Expert systems incorporate rules and procedures used by human experts in the field of interest into software that is then able to automatically make forecasts without human assistance.

Ho et al. [29] proposed a knowledge-based expert system for the short-term load forecasting of the Taiwan power system. Operators' knowledge and the hourly observation of system load over the past five years are employed to establish eleven day-types. Weather parameters were also considered. Rahman and Hazim [30] developed a site-independent technique for short-term load forecasting. Knowledge about the load and the factors affecting it is extracted and represented in a parameterized rule base. This rule-based system is complemented by a parameter database that varies from site to site. The technique is tested in different sites in the United States with low forecasting errors. The load model, the rules and the parameters presented in the paper have been designed using no specific knowledge about any particular site. Results improve if operators at a particular site are consulted.

Fuzzy Logic

Fuzzy logic is a generalization of the usual Boolean logic used for digital circuit design. An input under Boolean logic takes on a value of “True” or “False”. Under fuzzy logic an input is associated with certain qualitative ranges. For instance the temperature of a day may be “low”, “medium” or “high”. Fuzzy logic allows one to logically deduce outputs from fuzzy inputs. In this sense fuzzy logic is one of a number of techniques for mapping inputs to outputs.

Among the advantages of the use of fuzzy logic are the absence of a need for a mathematical model mapping inputs to outputs and the absence of a need for precise inputs. With such generic conditioning rules, properly designed fuzzy logic systems can be very robust when used for forecasting. Of course in many situations an exact output is needed. After the logical processing of fuzzy inputs, a “defuzzification” can be used to produce such precise outputs. [31], [32] and [33] describe applications of fuzzy logic to load forecasting.

Data mining

Data mining is the process that explores information data in a large database to discover rules, knowledge, etc [34, 35]. Hiroyuki Mori et al. proposed a data mining method for discovering STLF rules in [49]. The method is based on a hybrid technique of optimal regression tree and an artificial neural network. It classifies the load range into several classes, and decides which class the forecasted load belongs to according to the classification rules. Then multi layer preceptron (MLP) is used to train the sample in every class. The paper puts an emphasis on clarifying the nonlinear relationship between input and output variables in a prediction model.

Wavelets

A STLF model of wavelet-based networks is proposed [37] to model the highly nonlinear, dynamic behavior of the system loads and to improve the performance of traditional ANNs. The three-layer networks of the wavelet, the weighting, and the summing nodes are built by an evolutionary computing algorithm. Basically, the first layer of wavelet nodes decomposes the input signals into diverse scales of signals, to which different weighting values are given by the second layer of weighting nodes. Finally the third layer of summing nodes combines the weighted scales of signals into the output. In the evolutionary computing constructive algorithm, the parameters to be tuned in the networks are compiled into a population of vectors. The populations are evolved according to the stochastic procedure of the offspring creation, the competition of the individuals, and the mutation.

To investigate the performance of the proposed evolving wavelet-based networks on load forecasting, the practical load and weather data for the Taiwan power systems were employed. Used as a reference for determining the input variables of the networks, a statistical analysis of correlation functions between the historical load and weather variables was conducted a priori. For comparison, the existing ANNs approach for the STLF, using a back propagation training algorithm, was adopted. The comparison shows wavelet-based ANN forecasting has a more accurate forecasting result and faster speed.

Integration of Different Algorithms

As there are many presented methods for STLF, it is natural to combine the results of several methods [38]. One simple way is to get the average value of them, which can lower the risk of individual unsatisfactory prediction. A more complicated and reasonable way is to get the weight coefficient of every forecasting method by reviewing the historical prediction results. The comprehensive result is deduced by weighted average method.

2.3 Requirements of the STLF Process

In nearly all the energy management systems of the modern control centres, there is a short-term load forecasting module. A good STLF system should fulfill the requirement of accuracy, fast speed, automatic bad data detection, friendly interface, automatic data access and automatic forecasting result generation.

Accuracy

The most important requirement of STLF process is its prediction accuracy. As mentioned before, good accuracy is the basis of economic dispatch, system reliability and electricity markets. The main goal of most STLF literatures and also of this thesis is to make the forecasting result as accurate as possible.

Fast Speed

Employment of the latest historical data and weather forecast data helps to increase the accuracy. When the deadline of the forecasted result is fixed, the longer the runtime of the STLF program is, the earlier historical data and weather forecast data can be employed by the program. Therefore the speed of the forecasting is a basic requirement of the forecasting program. Programs with too long training time should be abandoned and new techniques shortening the training time should be employed. Normally the basic requirement of 24 hour (96 points) forecasting should be less than 20 minutes.

Automatic Bad Data Detection

In the modern power systems, the measurement devices are located over the system and the measured data are transferred to the control centre by communication lines. Due to the sporadic failure of measurement or communication, sometimes the load data that arrive in the dispatch centre are wrong, but they are still recorded in the historical database. In the early days, the STLF systems relied on the power system operators to identify and get rid of the bad data. The new trend is to let the system itself do this instead of the operators, to decrease their work burden and to increase the detection rate.

Friendly Interface

The interface of the load forecasting should be easy, convenient and practical. The users can easily define what they want to forecast, whether through graphics or tables. The output should also be with the graphical and numerical format, in order that the users can access it easily.

Automatic Data Access

The historical load, weather and other load-relevant data are stored in the database. The STLF system should be able to access it automatically and get the needed data. It should also be able to get the forecasted weather automatically on line, through Internet or through specific communication lines. This helps to decrease the burden of the dispatchers.

Automatic Forecasting Result Generation

To reduce the risk of individual imprecise forecasting, several models are often included in one STLF system. In the past such a system always needs the operators' interference. In other words, the operators have to decide a weight for every model to get the combinative outcome. To be more convenient, the system should generate the final forecasting result according to the forecasting behavior of the historical days.

Portability

Different power systems have different properties of load profiles. Therefore a normal STLF software application is only suitable for the area for which it has been developed. If a general STLF software application, which is portable from one grid to another, can be developed, the effort of developing different software for different areas can be greatly saved. This is a very high-level requirement for the load forecasting, which has not been well realized up utill today.

2.4 Difficulties in the STLF

Several difficulties exist in short-term load forecasting. This section introduces them separately.

Precise Hypothesis of the Input-output Relationship

Most of the STLF methods hypothesize a regression function (or a network structure, e.g. in ANN) to represent the relationship between the input and output variables. How to hypothesize the regression form or the network structure is a major difficulty because it needs detailed a prior knowledge of the problem. If the regression form or the network structure were improperly selected, the prediction result would be unsatisfactory. For example, when a problem itself is a quadratic, the prediction result will be very poor if a linear input-output relationship is supposed. Another similar problem is parameter selection: not only the form of the regression function (or the network structure), but also the parameters of it should be well selected to get a good prediction. Moreover, it is always difficult to select the input variables. Too many or too few input variables would decrease the accuracy of prediction. It should be decided which variables are influential and which are trivial for a certain situation. Trivial ones that do not affect the load behavior should be abandoned.

Because it is hard to represent the input-output relationship in one function, the mode recognition tool, clustering, has been introduced to STLF [54]. It divides the sample data into several clusters. Each cluster has a unique function or network structure to represent the input and output relationship. This method tends to have better forecasting results because it reveals the system property more precisely. But a prior knowledge is still required to do the clustering and determine the regression form (or network structure) for every cluster.

Generalization of Experts' Experience

Many experienced working staff in power grids are good at manual load forecasting. They are even always better than the computer forecasting. So it is very natural to use expert systems and fuzzy inference for load forecasting. But transforming the experts' experience to a rule database is a difficult task, since the experts' forecasting is often intuitive.

The Forecasting of Anomalous Days

Loads of anomalous days are also not easy to be predicted precisely, due to the dissimilar load behaviour compared with those of ordinary days during the year, as well as the lack of sufficient samples. These days include public holidays, consecutive holidays, days preceding and following the holidays, days with extreme weather or sudden weather change and special event days. Although the sample number can be greatly enhanced by including the days that are far away from the target day, e.g. the past 5 years historical data can be employed rather than only one or two years, the load growth through the years might lead to dissimilarity of two sample days. From the experimental results it is found that days with sudden weather change are extremely hard to forecast. This sort of day has two kinds of properties: the property of the previous neighbouring days and the property of the previous similar days. How to combine these two properties is a challenging task.

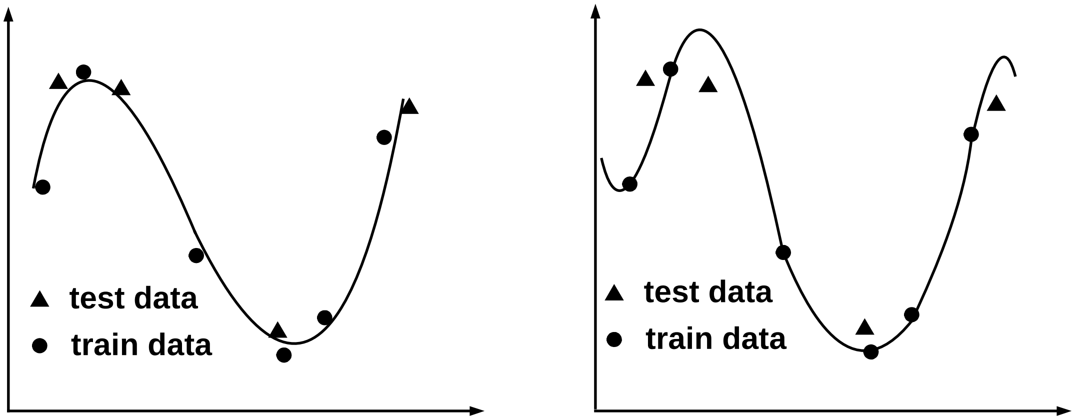
Inaccurate or Incomplete Forecasted Weather Data

As weather is a key factor that influences the forecasting result, it is employed in many models. Although the technique of weather forecasting, like the load forecasting, has been improved in the past several decades, sometimes it is still not accurate enough. The inaccurate weather report data employed in the STLF would cause large error.

Another problem is, sometimes the detailed forecasted weather data cannot be provided. The normal one day ahead weather report information includes highest temperature, lowest temperature, average humidity, precipitation probability, maximum wind speed of the day, weather condition of three period of the day (morning, afternoon and evening). Usually the number of the load forecasting points in a day is 96. If the forecasted weather data of these points can be known in advance, it would greatly increase the precision. However, normal weather reports do not provide such detailed information, especially when the lead time is long. This is a bottleneck of load forecasting.

Less Generalization Ability Caused By Overfitting

Overfitting is a technical problem that needs to be solved for load forecasting. Load forecasting is basically a “training and predicting” problem, which is related to two datasets: training data and testing data. Historical training data are trained in the proposed model and a basic representation can be obtained and in turn used to predict the testing data. For the outcoming training module, if the training error for the training data is low but the error for the testing data is high, “overfitting” is said to have occurred. Fig. 2.3 shows the regression curve of the 1-dimensional input to illustrates the effect of overfitting. The round dots represent the testing data and the triangle dots represent the training data. In (a) both the training error and the testing error are low. In (b) where overfitting exists, although the training error is almost zero, the testing error is quite high. A significant disadvantage of neural networks is overfitting; it shows perfect performance for training data prediction but much poorer performance for the future data prediction. Since the goal of STFL is to predict the future unknown data, technical solutions should be applied to avoid overfitting.



(a) Without overfitting (b) With overfitting

Fig. 2.3 Illustration of training result with/without overfitting

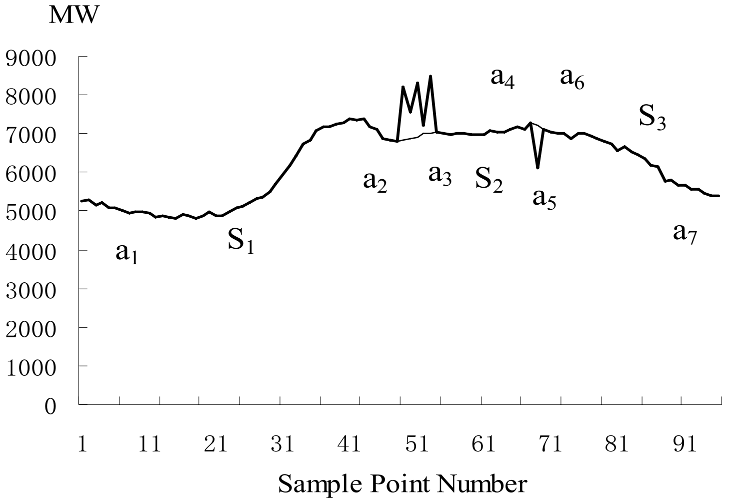
The Destroy of Load Curve Nature By Compulsory Demand-side Management

With the development of economical development and relative lag in power investment, energy shortage has appeared in many countries. To avoid reliability problem and assure the power supply of very important users, compulsory demand-side management is often executed. This compulsory command destroys the natural property of load curve. When this kind of load curve is included in training, it serves as noise and deteriorates the final results.

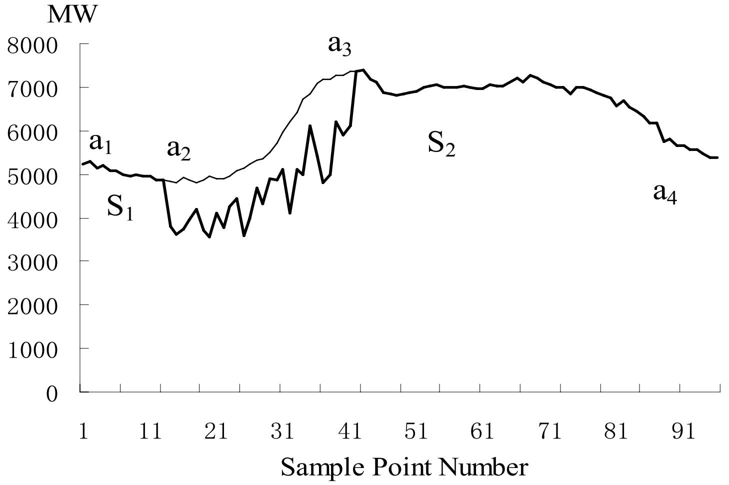
3 Historical Data Pretreatment

3.1 Overview of Load Bad Data

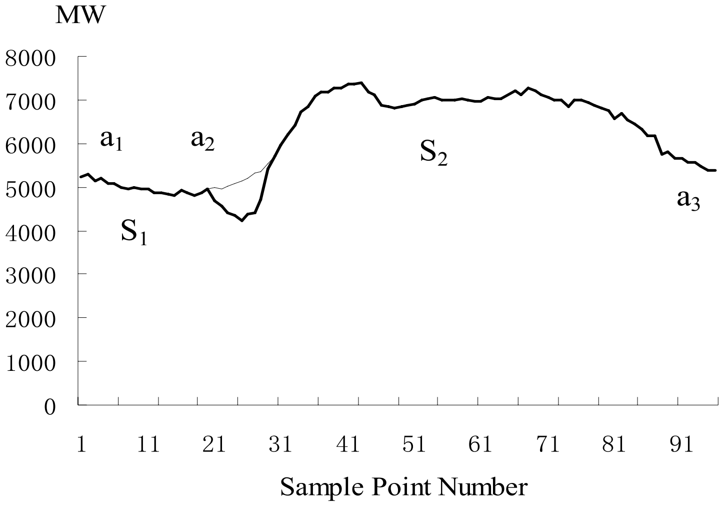
The existence of bad data in historical load curve affects the precision of load forecasting result. There are two kinds of bad data in the daily load curve: false channel bad data and abnormal event bad data. False channel bad data are due to the measurement and transmission mistakes, and they are far from their real physical values. Abnormal event bad data come from some unexpected sudden incidents, such as short circuit and equipment overhaul, which cause unnatural sudden changes of the load curve trend. According to the continuous time of the bad data appearance they can be put into two categories: long-last bad data and short-period bad data. Fig. 3.1 shows these two kinds of bad data.



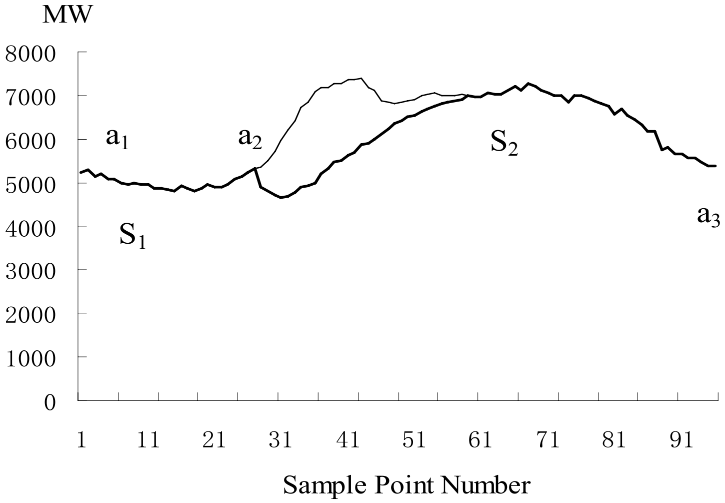
(a) Short-period false channel bad data



(b) Long-lasting false channel bad data



(c) Short-period abnormal event bad data



(d) Long-lasting abnormal event bad data

Fig. 3.1 Daily load curves with bad data

The thick lines of Fig. 3.1 (a), (b), (c) and (d) show respectively daily load curves with short-period false channel bad data, long-lasting false channel bad data, short-period abnormal event bad data, and long-lasting abnormal event bad data. In Fig. 3.1(a) and (b), where the bad data are caused by false channel, the thin lines correspond to the real physical values of bad data. In Fig. 3.1 (c) and (d), where the bad data are caused by abnormal events, the thin lines correspond to what the load values of bad data are supposed to be if the abnormal events didn’t take place.

Through observation and analysis of a large amount of historical load curves in different areas, it is discovered that most of the bad data, especially the false channel bad data do not last for a long time. For example, through the statistic research of Shanghai Power Grid of 2004, it is found out that more than 90% of the bad data lasted less than 30 minutes.

3.2 Bad Data Detection and Replacement

3.2.1 Basic idea of second order difference

To investigate the proposed second order difference for bad data detection, firstly two concepts of second order difference are introduced. Suppose $L(i)$ is the real load of the point i in the load curve, then its forward second order difference (FSOD) is defined as

$$\ddot{L}_{\mathrm{f}}(i) = (L(i) - L(i+1)) - (L(i+1) - L(i+2)) \tag{3.1}$$

and its backward second order difference (BSOD) as

$$\ddot{L}_{\mathrm{b}}(i) = \ddot{L}_{\mathrm{f}}(i-2) = (L(i-2) - L(i-1)) - (L(i-1) - L(i)) \tag{3.2}$$

The idea of second order difference for bad data detection is, for the continuously time-variant physical quantity in nature, in a short enough period of time, the second order difference of the continuous samples is close to zero, or located in a short interval $\boldsymbol{V} = [v_1, v_2]$, where v_1 is a small negative number, and v_2 is a small positive number. But the electrical power load bad data, whether they are caused by false channel or by abnormal event, usually lead to a sudden change in the load curve; thus their corresponding second order difference is far from zero and therefore doesn't belong to the interval $\boldsymbol{V}$. If FSOD of point i is within $\boldsymbol{V}$, points i, $i + 1$ and $i + 2$ are thought to be continuous and vise versa. If BSOD of point i is within $\boldsymbol{V}$, points i, $i - 1$ and $i - 2$ are thought to be continuous and vise versa.

The bad data separate a load curve into several segments. The points in every segment are continuous, e.g. S_1, S_2, S_3 in Fig. 3.1. By calculating the second order difference, the continuous segment(s) of a load curve can be detected. Suppose the indices of the starting and ending points of one segment are respectively m and n, they should satisfy the following two rules: $\ddot{L}_{\mathrm{f}}(i) \in V$, i = m,m + 1…,n - 2; and $\ddot{L}_{\mathrm{b}}(i) \in V$, i = m + 2, m + 2...n.

If a bad datum $n + 1$ appears next to a segment of normal data, it shows a sudden change in the curve and its backward second order difference absolute value is large: $\ddot{L}_{\mathrm{b}}(n+1) \notin V$.

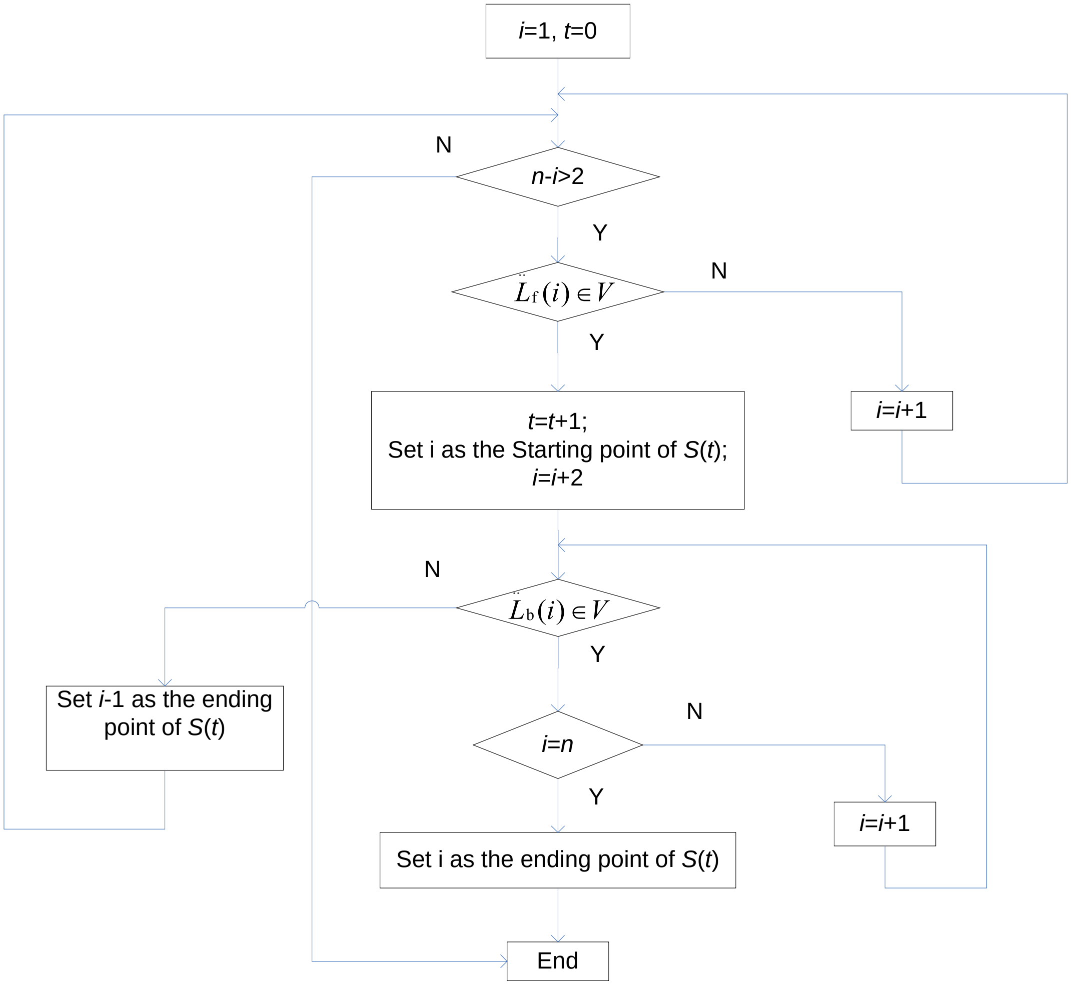
For a given load curve, the description of bad data and continuous segment detection is as follows. Note that the load curve doesn't need to be a daily load curve; it can be with arbitrary length.

1) First consider the leftmost point of a load curve, i.e. $i = 1$.

2) If $\ddot{L}_{\mathrm{f}}(i) \in V$, it is supposed to be the starting point of segment S_1; otherwise, consider the forward second order difference of the right-side neighboring point, i.e. $i = i + 1$, until the starting point is found.

3) Let $i = i + 2$; if $\ddot{L}_{\mathrm{b}}(i) \in V$, which means i is still in the continuous segment, consider its right-side neighbouring point, i.e. $i = i + 1$; if $\ddot{L}_{\mathrm{b}}(i) \notin V$, point i - 1 is regarded as the ending point of the continuous segment S_1.

4) Explore the remaining load curve with the above technique to find the other segments S_2, S_3, …until all the points of the daily load curve are covered.



i=1, t=0
N
n-i>2
Y
N
$\ddot{L}_{\mathrm{f}}(i) \in V$
Y
t=t+1;
Set i as the Starting point of S(t);
i=i+2
i=i+1
N
$\ddot{L}_{\mathrm{b}}(i) \in V$
Y
Set i-1 as the ending point of S(t)
N
i=n
Y
i=i+1
Set i as the ending point of S(t)
End

Fig. 3.2 Flowchart of finding continuous segment(s)

Fig. 3.2 illustrates how to find the continuous segment(s) for a series of sampling load data. In the figure n means the total sampling number in the curve, and $S(t)$ means the t^{th} segment the algorithm detects.

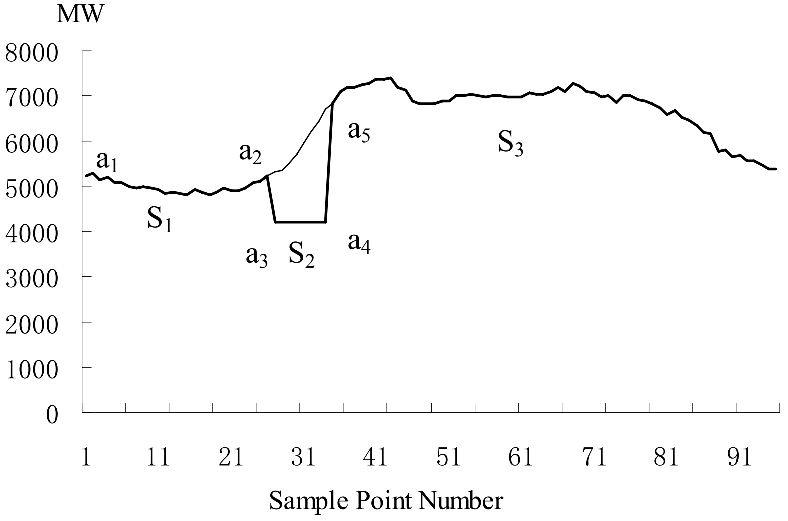


Fig. 3.3 Bad data in a continuous segment

When more than one continuous segment is obtained, the points between the neighbouring continuous segments are regressed in a quadratic form to revaluate the points between them. The continuous segments do not always represent the good data. Sometimes, bad data can also constitute a continuous segment. For example, segment S_2 in Fig. 3.3 contains false channel bad data, but it is still in a continuous segment. To detect the bad data that appear to be in a continuous segment, determine whether the bordering points (for example, a_2 and a_3 in Fig. 3.3) are still bad data according to the revalued points by calculating the related second order difference. The following are the procedures of revising the curve with bad data, Fig. 3.1(a) taken as an example:

1) Use data in the last n_1 points in S_1 and the first n_1 points in S_2 to form a least square quadratic regression formulation $L(t) = at^2 + bt + c$ and determine the parameter a, b and c, t being the time point.

2) With the regression result $L(t)$, replace the load data of the open interval between the ending point of $S_1(a_2)$ and the starting point of $S_2(a_3)$ (the thin line in Fig. 3.1(a)).

3) With data in S_1 and S_2 as well as the new load data derived in step 2, calculate $\ddot{L}_f(a_2 - 1)$ and $\ddot{L}_b(a_3 + 1)$. If both of them belong to **V**, the regression result is the acceptable substitution of bad data. Otherwise, S_2 is thought to be invalid and all the points in it are regarded as bad data. In this case, the above method is applied to segment S_1 and S_3.

4) Repeat the above procedure to replace all the bad data of the load curve.

If the interval between two segments is too long, it is considered a long-lasting bad data period, e.g. Fig. 3.1(b). Regarding such a long interval as a quadratic curve may cause unacceptable error. Fig. 3.4 shows an example of an unsuccessfully revised curve with long-lasting bad data, where the actual load and the revised curve are not close to each other. Since the lack of data makes it difficult to estimate these data, the corresponding load curve is given up and taken out of the database. In this thesis the upper limitation of the interval is set to be 75 minutes. Fortunately due to the property that most bad data don't last long, most of the bad data of the load curve can be revalued effectively. Due to a similar reason, the number of points ($2n_1$) that constitute the regression samples shouldn't be very large. In this thesis n_1 is set to occupy 45 minutes of the load curve.

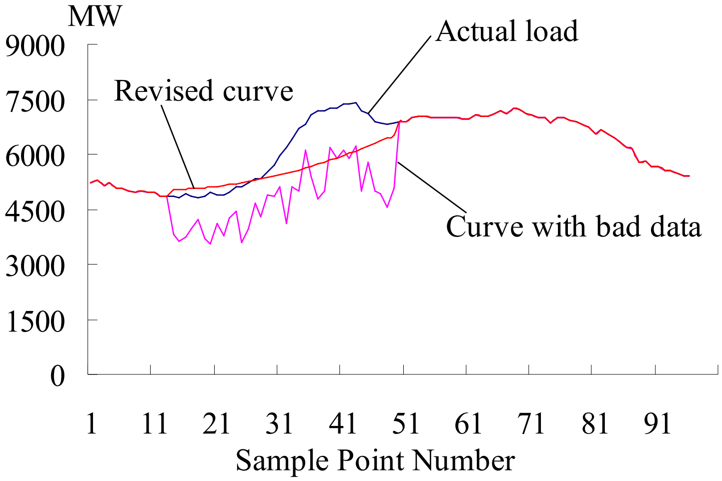


Fig. 3.4 Unsuccessfully revised curve with long-lasting bad data

3.2.2 Consideration of the segment with both good and bad data

In the case of Fig. 3.1(c) and Fig. 3.1(d), an abnormal event comes suddenly but recovers gradually. Thus, there might be the segment that contains both bad data and good data, and there is no obvious border that distinguishes bad data from good data. Here Fig. 3.5 is taken as an example to illustrate how to deal with it. It's not successful to make a smooth regression for S_1 and S_2 because a_2 is a sudden change point. Set a point b_1 which is to the right of the starting point of a_2 but still on the segment S_2. S_{21} is used to represent the segment between point b_1 and point a_3. Try to make smooth quadratic regression with S_1 and S_{21}. If it succeeds, the open interval between a_2 and b_1 is thought to be bad data and revalued by the regression. Otherwise, find in S_{21} a point b_2 to the right of b_1 and repeat the process. But for the case of Fig. 3.1(d), due to the long time for the bad data to recover, the regression result is not reliable, so it is given up.

If in this case S_2 is long enough (e.g. more than two days), it is considered that the forepart of it (e.g. four hours) has suspicious data and should be taken out of the database.

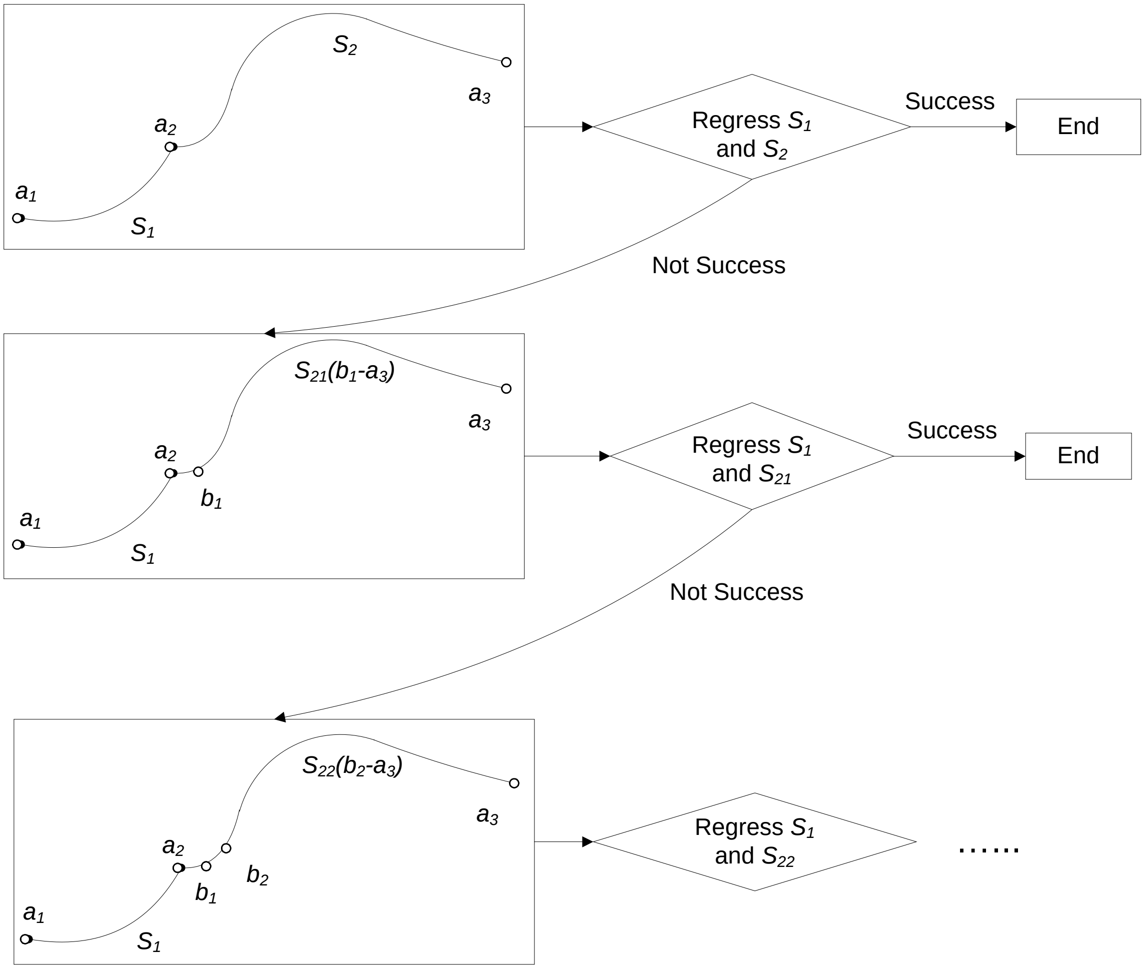


Fig. 3.5 Dealing segments with both good and bad data

3.2.3 Selection of interval *V*

The selection of ***V*** is very important. A too broad interval can cause the neglecting of some bad data, while a too narrow interval can cause misjudgment. In this thesis the statistics theory is applied. Consider n + 2 points of the load curve over a relatively long period of time (e.g. a month) and calculate the forward second order difference of every point:

$$FSOD(1), FSOD(2), ..., FSOD(n)$$

Define the average value of them

$$\overline{FSOD} = (FSOD(1) + FSOD(2) + ... + FSOD(n)) / n \tag{3.3}$$

The standard deviation of them is

$$DEV = \sqrt{\frac{1}{n-1}\left(\sum_{i=1}^{n}(FSOD(i) - \overline{FSOD})^2\right)} \tag{3.4}$$

Thus it can be derived

$$\boldsymbol{V} = [v_1,\ v_2] = [\overline{FSOD} - 3DEV, \overline{FSOD} + 3DEV] \tag{3.5}$$

According to the probability theory, the point whose FSOD value is outside $\boldsymbol{V}$ is considered to be a bad datum. Here n should be a large number to decrease the effect of bad data inside the second order difference sequence. In this thesis $n \geq 3 \times 10^4$ is required. It can be proved that $\boldsymbol{V}$ is also the acceptable interval for BSOD.

3.3 Smoothing the Load Curve

In some power systems, the daily load curve is not very smooth even without bad data, especially in the highly industrialized areas where there is a large amount of impulse load such as steel mill, synchrotrons and wind tunnels. The startup and shutdown time of these devices is quite random, i.e. there is no obvious regularity for them. When the data of such a load curve are used in load forecasting training, the impulse part of the load adds to the difficulty of load forecasting. After detecting the bad data and replacing them with reasonable ones, the load curve might still be not very smooth because of the impulse load, although the curve’s sudden change it causes is not as obvious as the bad data. In this research work the smoothing method is proposed.

It can be thought that a load curve is the sum of two load curves (Fig. 3.6), an essential load curve that represents the basic load requirement, and a vibrating curve that contains the information of sudden change of the large consumers’ state. With some experiments, it is found that the essential load curve has some regularity; however the regularity of vibrating curves is not so easy to get. Further more, the mean absolute value of the latter is much less than that of the former. Therefore the smoothed curve is used in training instead of the original curve, so that the ruleless vibration does not affect the prediction result. To prove this, two kinds of methods are employed in short-term load forecasting. The first one is to predict the load with the original historical load data, and the other one is to predict the load with the essential load curve. In the forecasting result shown in Chap. 6, it is found that the second method improves the forecasting accuracy of the first method by 18.6%. Actually this kind of prediction doesn’t take the vibrating load of the target load into account. Because of the lack of statistical significance, the prediction of vibrating load is not very predictable. Therefore it is better to only predict the essential load and regard it as the forecasting result.

In this stage, the essential load curve is achieved through weighted least square quadratic fitting. Consider a span for the point *t* to be fitted. Compute the regression weights for each data point in the span. The weights are given by the cube function shown in Eq. (3.6).

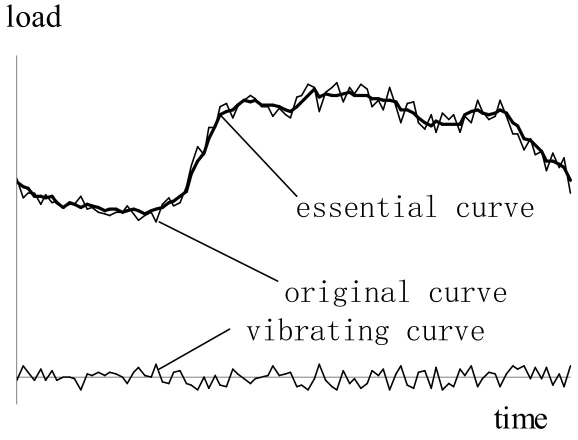


Fig. 3.6 Demonstration of curve decomposition

$$w_i = (1 - \left| \frac{t - t_i}{d(t)} \right|^3)^3 \tag{3.6}$$

t_i are the neighbors of *t* as defined by the span, and *d*(*t*) is the distance along the abscissa from *t* to the most distant predictor value within the span. The weights have the following characteristics. The data point to be smoothed has the largest weight and the most influence on the fitting. Data points outside the span have zero weight and no influence on the fitting. For the daily load curve the span is set to be 90 minutes long. A weighted quadratic least squares regression is performed according to the calculated weights of the points in the span, together with their corresponding load values. The smoothed value is given by the weighted regression at the predictor value of interest, namely, point *t*.

Fig. 3.7 shows the flowchart of the proposed historical data pretreatment system, including the segments search, bad data revaluation and curve smoothing. For a system where the load curve is very smooth, the smoothing module can be ignored. The bad data that couldn't be revalued will be regarded as absent data.

In some power systems, due to the electricity price policy, “natural” sudden change in the daily load curve exist. For example, the electricity price drops at the time of 22:00 everyday in the German E.ON power grid. This leads to the sudden jump at around this time in the daily load curve. In the proposed method of bad data detection and curve smoothing, these kind of points should be regarded as exceptions, which should not be considered bad data or smoothed.

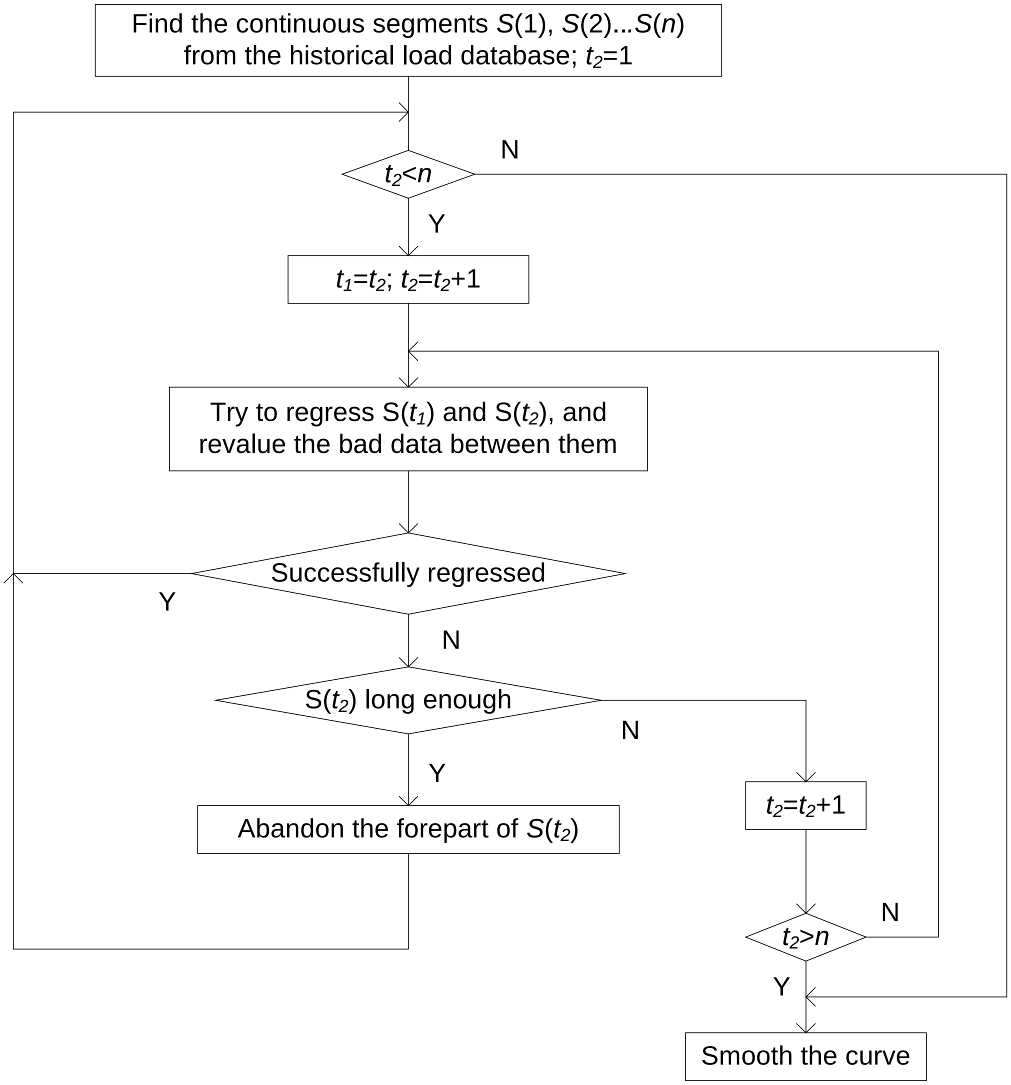

Find the continuous segments $S(1)$, $S(2)$...$S(n)$ from the historical load database; t_2=1
$t_2<n$
N
Y
$t_1=t_2$; $t_2=t_2+1$
Try to regress S(t_1) and S(t_2), and revalue the bad data between them
Successfully regressed
Y
N
S(t_2) long enough
N
Y
Abandon the forepart of $S(t_2)$
$t_2=t_2+1$
$t_2>n$
N
Y
Smooth the curve

Fig. 3.7 Historical data pretreatment system flowchart

3.4 Case Study

The proposed method of historical load data pretreatment is applied to the Shanghai Power Grid daily load curve. The historical database contains the load of 2001-2004, with the sampling interval being 15 minutes. Tab. 3.1 shows the statistics of bad data. Altogether 10.96% of all the daily load curves contain bad data, and altogether 86.04% of the bad data curves can be revalued. The statistics of bad data of the German E.ON power system in 2003 is shown in Tab. 3.2. Fig.

3.8 shows some daily load curves that contain some detected bad data. The examples are from the Shanghai Power Grid. The estimated actual values are also displayed on the figures, with the triangle symbol.

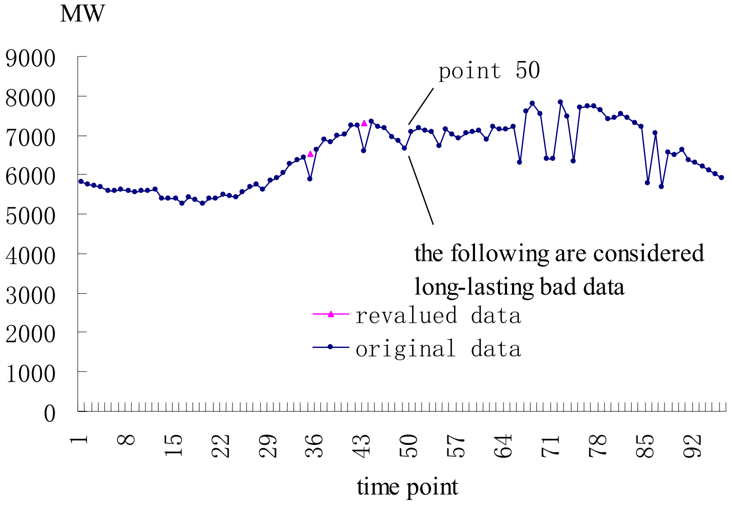
Tab. 3.1 Statistics of bad data from the Shanghai power system in 2001-2004

Daily load curves with bad data (%)	10.96
False channel curves (%)	5.21
Abnormal event curves (%)	5.75
Successfully revalued false channel curves (%)	5.02
Successfully revalued abnormal event curves (%)	4.41
Successfully revalued bad data curves (%)	9.43

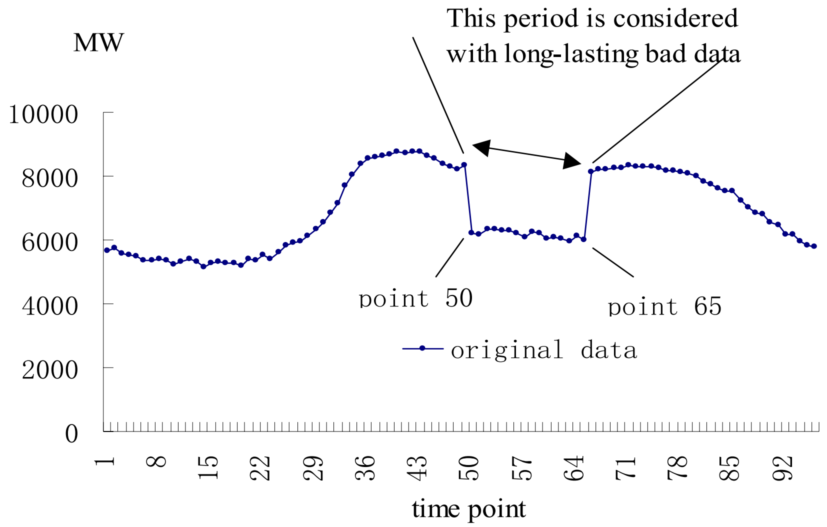
Tab. 3.2 Statistics of bad data from the German E.ON power system in 2003

Daily load curves with bad data (%)	1.37
False channel curves (%)	1.1
Abnormal event curves (%)	0.27
Successfully revalued false channel curves (%)	1.1
Successfully revalued abnormal event curves (%)	0
Successfully revalued bad data curves (%)	1.1

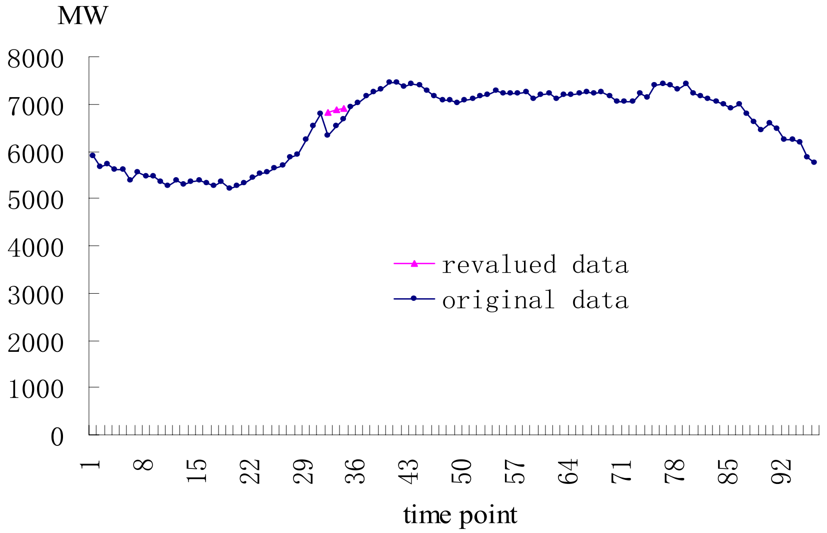
In Fig. 3.8(a) the first two occurrences of bad data are revalued to prevent the curve from sudden change. But points from point 50 on cannot be revalued, so this period is discarded. In (b) point 50-point 65 contains bad data. Since the period is very long, it is decided by the algorithm not to revalue them. The beginning part of the next segment from 66 on is thought suspicious and also discarded. In this research, it is supposed that the first four hours of this segment are not reliable, so the discarded period is from point 66 to point 82. In (c), (d) and (e) all the detected bad data are replaced with estimated values.



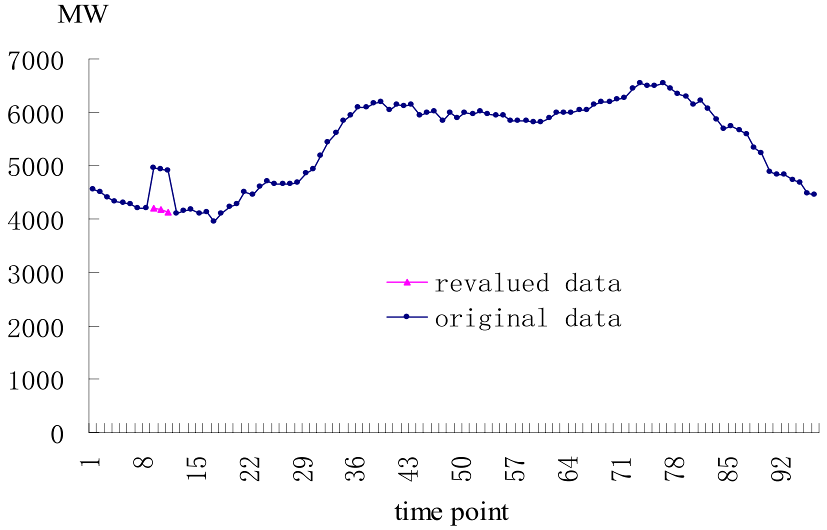
(a) Curve of 2002.11.24



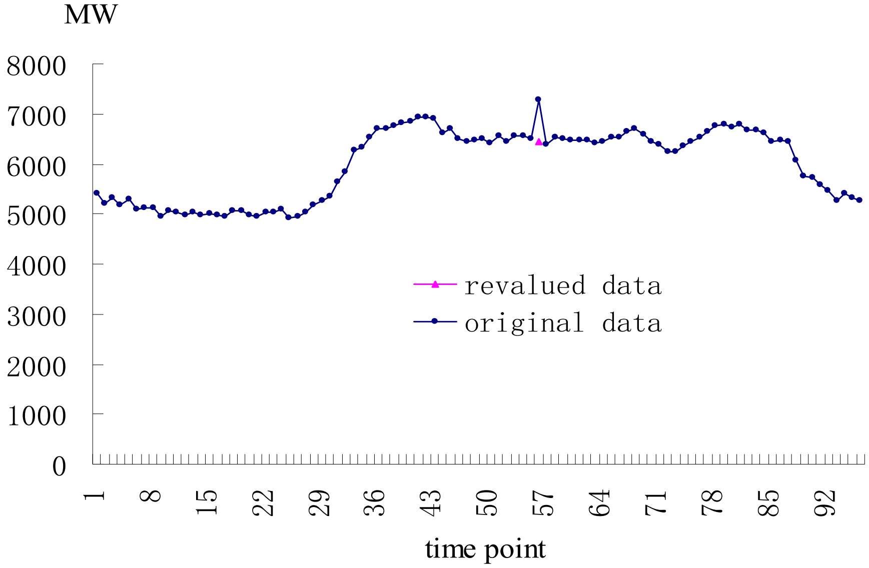
(b) Curve of 2002. 1.22



(c) Curve of 2001.9.1



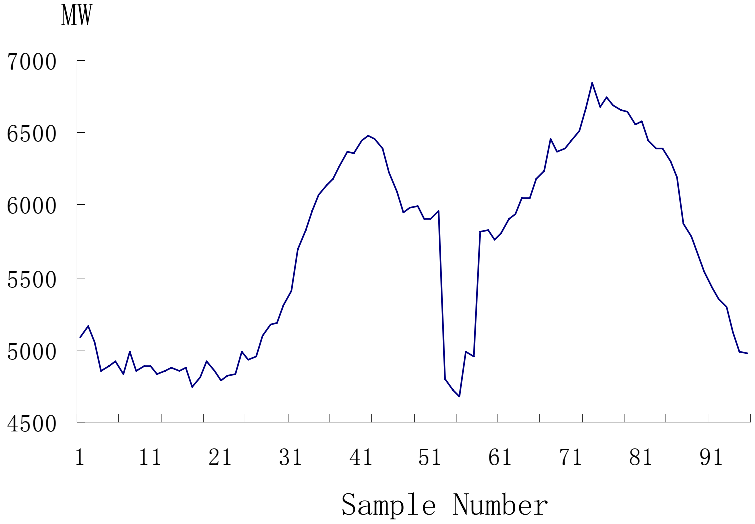
(d) Curve of 2001.3.18



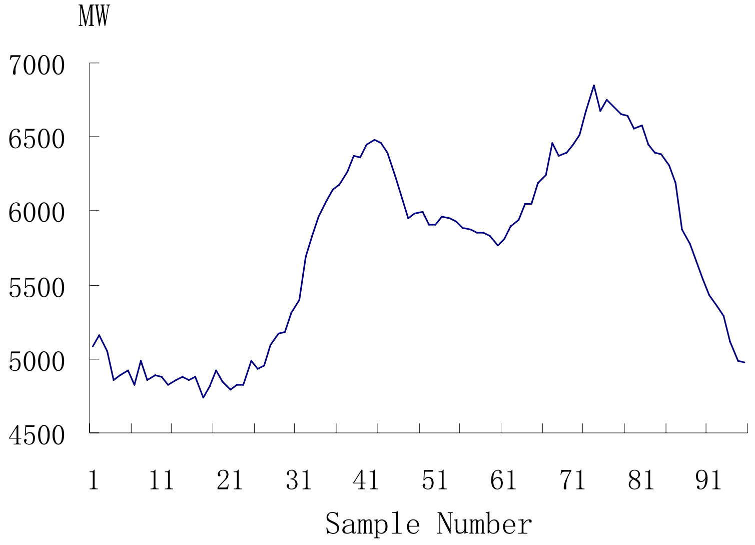
(e) Curve of 2001.6.17

Fig. 3.8 Daily load curves with bad data and the revaluation

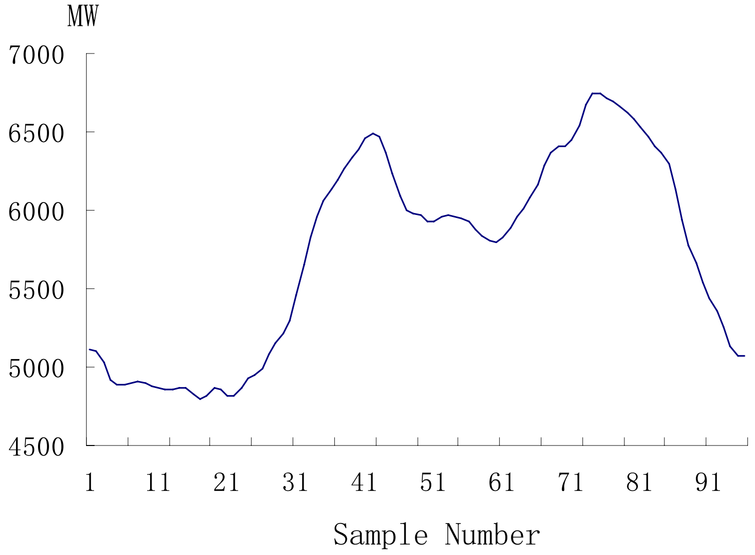
Fig. 3.9 shows the effect of bad data detection and fitting. Curve (a) is the load curve of 2001.04.15 with bad data and impulse load, curve (b) is bad data detection and substitution module output of curve (a), and curve (c) is the fitting module output of curve (b). It can be seen that curve (b) detected and revalued the bad data effectively, and curve (c) reflects the trend of the curve correctly and shows better smoothness than curve (b). The span corresponding to Eq. (3.6) is selected to include 7 points (90 minutes long).



(a) The Original Daily Load Curve



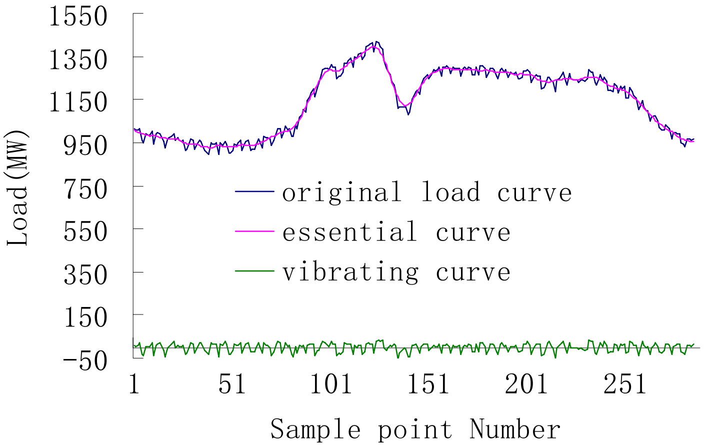
(b) Curve processed by bad data detection module



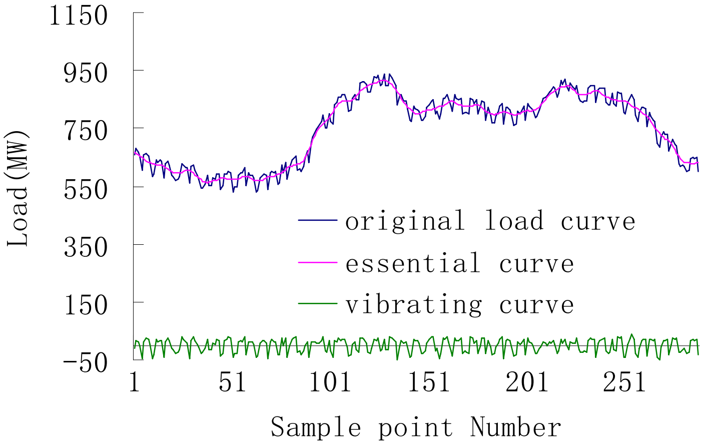
(c) Curve processed by fitting module

Fig. 3.9 The effect of bad data detection and fitting for a daily load curve

Shanghai is a metropolis with over a population of 16 million population and has a very advanced industry. Its numerous electricity users make the effect of any individual user behaviour in consuming electricity not so obvious. As a result, the impulse part of the Shanghai load curve is not very obvious. Therefore a relatively smaller utility, the Changzhou Power Grid in China, is taken here as an example of the smoothing effect. Fig. 3.10 shows the original load curve, essential load curve and vibrating curve of several days in Changzhou. In order to get detailed original curve information, the data with the 288 sample points are used, i.e. the sample interval is five minutes. The figures show that with the proposed smoothing algorithm the daily load curve can be smoothed very well.



(a) Changzhou load curves of 2003.01.29



(b) Changzhou load curves of 2003.03.23

Fig. 3.10 The effect of smoothing

In order to test the bad data detection ability of the proposed algorithm, some fictitious bad data are added to replace the actual load. The number of these fictitious data is 15% of the total load data. The proposed method is applied, and all the fictitious data are detected. The MAPE of the replaced data compared with the real data is 0.97%.

In this chapter the way of detecting bad data has been demonstrated, as well as the way of getting rid of the ruleless impulse component from the load curve. Traditional ways of bad data detection are to compare a load curve with the other curves. If it is abnormal, all of the data in the curve are regarded as useless. The application of second order difference bad data detection is an effective way to find out the bad data. It also revalues the bad data so that plenty of information is not lost. The load curve revised by fitting is much flatter than the original one, but it retains the basic changing trend of the original one very well. Later on this thesis will show that these methods help to decrease the prediction error. From the prediction results, it can be seen that the application of bad data detection and fitting can significantly increase the forecasting accuracy.

4 Regression Tree Based STLF

4.1 CART Regression Tree Algorithm

As a non-parameter algorithm, regression tree (RT) is an automatic classifier. For a learning sample consisting of n historical cases ($\boldsymbol{x}_1, y_1$), ($\boldsymbol{x}_2, y_2$), …($\boldsymbol{x}_n$, y_n), where $\boldsymbol{x}_t$ is the t^{th} independent variable with a form of m-dimensional vector, and y_t is the corresponding response variable with numerical value, RT forms a binary tree structure classifier. The tree is constructed by repeated splits of subsets into two descendant subsets according to sample input variables. Every split is an inquiry about the input variables, and the answers of "yes" and "no" lead respectively to the left and right descendant subsets. Fig. 4.1 is the regression tree of a predefined function $y = x_1 x_2 (x_1 - 1) + e^{-x_2} - 4x_2, x_1, x_2 \in [0,7]$.

Since a regression tree algorithm only deals with discrete values, firstly the function is discretized into 64 vectors of input and output variables in the domain. Part of the discretized data is shown in Tab. 4.1, on the basis of which the regression tree of Fig. 4.1 is constructed.

Tab. 4.1 Discretized inputs and outputs for $y = x_1 x_2 (x_1 - 1) + e^{-x_2} - 4x_2, x_1, x_2 \in [0,7]$

Vector ID	1	2	3	4	5	6	7	8	9	10	…	64
x_1	0	0	0	0	0	0	0	0	1	1	…	7
x_2	0	1	2	3	4	5	6	7	0	1	…	7
y	1.0	-3.6	-7.9	-12.0	-16.0	-20.0	-24.0	-28.0	1.0	-3.6	…	266.0

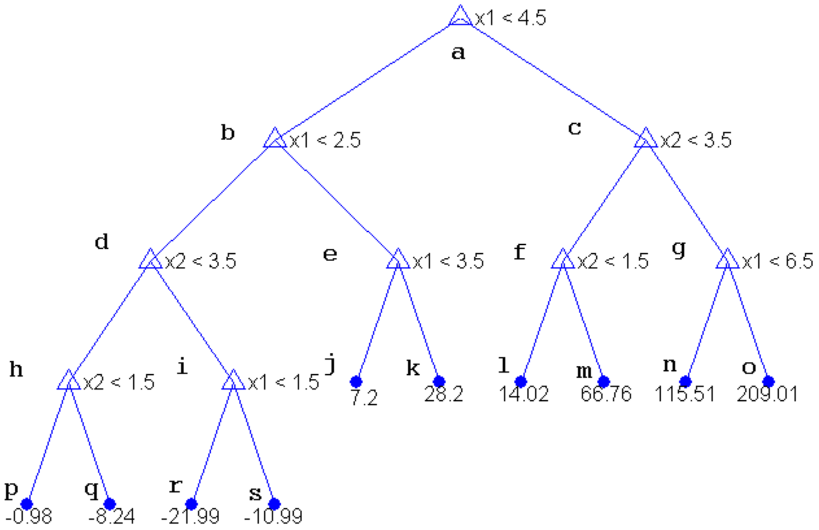


Fig. 4.1 Regression tree of $y = x_1 x_2 (x_1 - 1) + e^{-x_2} - 4x_2, x_1, x_2 \in [0,7]$

Every subset is called a “node”, e.g. node *a*, *b*,…,*s* in Fig. 4.1. If a split of node N_1 divides it into two nodes: N_2 and N_3, then N_1 is the parent of N_2 and N_3, and N_2 is N_3’s sibling. In Fig. 4.1 node *b* is the parent of *d* and *e*, and *d* and *e* are the siblings of one another. A leaf node is one without further splits, e.g node *j*, *k*…*s* in Fig. 4.1. The root node is the original sample set, e.g. node *a* in Fig. 4.1. Every leaf node has an output value and a rule which can be expressed in the form of “if…then…”. For instance the rule of node *m* in Fig. 4.1 is

“if $1.5 \leq x_2 < 3.5$ and $x_1 \geq 4.5$, then $y = 66.76$”.

In forming a regression tree, three elements are necessary to determine a tree predictor:

- A way to select a split at every intermediate node
- A rule for determining when a node is a leaf node
- A rule for assigning an output value to every leaf node

Breiman et al. proposed the classification and regression tree (CART) in 1984 [39]. The algorithm answers the above questions very well. To give an overview of these answers from CART, some concepts are introduced as follows.

For a node *k* that contains cases $(\mathbf{x}_{k1}, y_{k1})$, $(\mathbf{x}_{k2}, y_{k2})\ldots(\mathbf{x}_{kN}, y_{kN})$, its dispersion (or data dispersion) is measured as the total standard deviation (DEV) of y_{kt}, $t = 1, \ldots, N$:

$$DEV(k) = \sqrt{\left(\sum_{i=1}^{N}(y_{ki} - \bar{y}_k)^2\right)/(N-1)} \tag{4.1}$$

where $\bar{y}_k$ is the average value of *y* in node *k*, i.e.

$$\bar{y}_k = \left(\sum_{i=1}^{N} y_{ki}\right)/N \tag{4.2}$$

In order to find the best split variable and the best split value for this variable, the RT algorithm checks all possible splitting variables, as well as all possible values of every variable to be used to split the node. Suppose for any split *S* of node *k* into k_L and k_R, let

$$f = DEV(k) - DEV(k_L) - DEV(k_R) \tag{4.3}$$

The above 3 questions about forming a RT are answered as follows.

1) The best split of the node is the one that can maximize (4.3).

2) If the sample number N is too small, the statistical significance is not obvious. Therefore, the lower limitation of N is set: $Nmin$. The nodes are split until one of these conditions is satisfied.

- Condition 1: the sample number of the subset is less than N_{min}
- Condition 2: all the sample points of the subset have the same output value.

3) The output value of the leaf node k is the average output value of all the cases, i.e. $\bar{y}_k$ of the node k.

Compared with the other regression or network algorithms, RT has the following advantages. It is unnecessary to build a regression equation or network construction for the algorithm, because the algorithm itself can automatically classify the data and assign a value for every node without any a prior knowledge. The result of the algorithm is with the form of "if… then…", which can be easily understood. Both continuous and categorical independent variables are acceptable in forming a regression tree. It can handle the non-homogeneous relationship between input and output variables. It can estimate the error of the prediction values. It is robust with outliers. Given a redundant set of input variables, it is able to pick up the important input variables and ignore the redundant ones.

4.2 Application of CART in Short-term Load Forecasting

This thesis presents two kinds of RT application to STLF: non-increment RT method and increment RT method.

4.2.1 Non-increment method

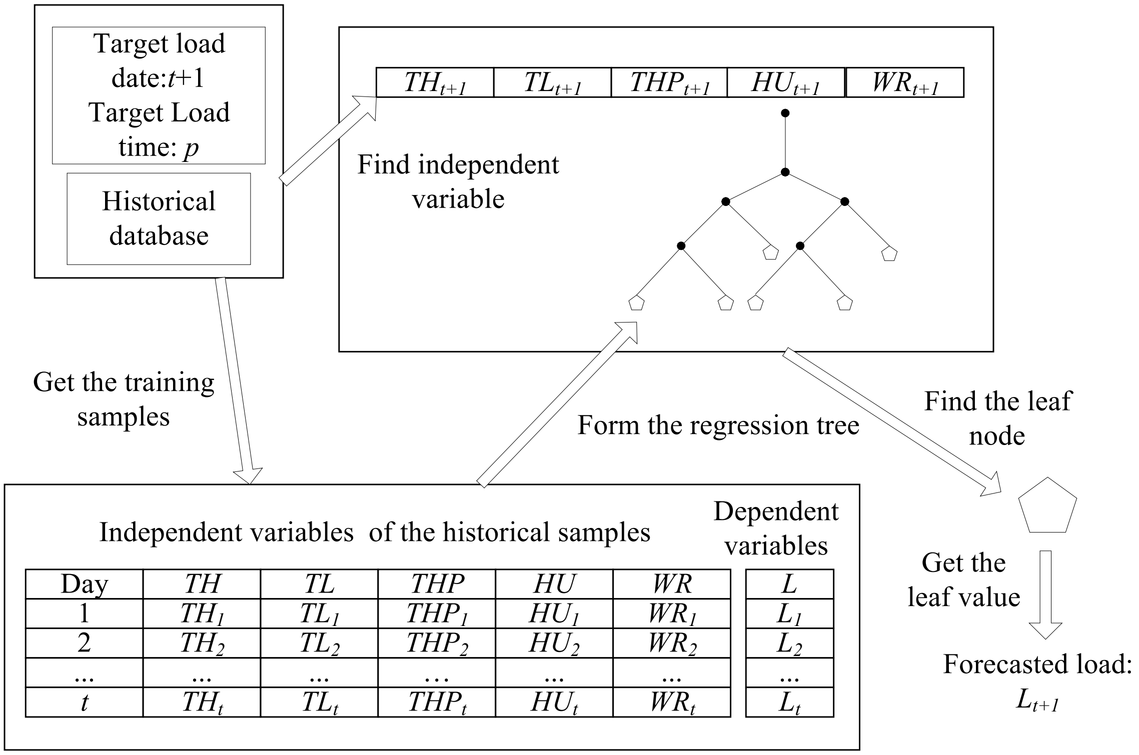
The non-increment RT method regards every day as a sample object of the tree. Suppose the historical day indices are $1,2,...,t$, and the p^{th} point of day $t + 1$ is to be forecasted. Then the p^{th} point load of day $1…t$ are regarded as the response value of the learning sample, with a great number of corresponding independent variables of the focus day: *TH*, *TL*, *THP*, *HU*, *WR*, whose meaning is shown in Tab. 4.2.

The regression tree is developed based on these data. For the target load to be forecasted, the related input variables are employed to find the leaf node, the output value of which is considered as the prediction value. The dispersion and the node sample number of the leaf node can also be obtained.

Tab. 4.2 Input variable definition for non-increment regression tree

Parameter	**Definition**
TH	The highest temperature of the sample day (℃)
TL	The lowest temperature of the sample day (℃)
THP	The highest temperature of the sample day's previous day (℃)
HU	Average humidity of the sample day (%)
WR	Weekday rank of the sample day, 1...7 means from Monday to Sunday

An experienced dispatcher usually compares only the days with the same weekday type to predict the future load because the data similarity of weekday and weekends is not very strong. For example, if he wants to forecast the load of Wednesday, he would use the data of Tuesday, Monday and last Friday, Thursday and avoid using the data of last Saturday and Sunday. Since the weekend and weekday daily load curves are quite different, in our research three different trees are constructed to decrease the dimension of the problem: the pure weekday tree, the pure Saturday tree and the pure Sunday tree. The pure weekday tree only deals with the data of weekdays, and the pure weekend tree only with the data of weekends. Holiday curves are usually quite different from the normal curves, so they are neglected in forming the historical data of a non-holiday. Later the holiday load forecasting will be surveyed specially. Fig. 4.2 shows the basic process of the non-increment regression tree forecasting.



Day	*TH*	*TL*	*THP*	*HU*	*WR*
1	TH_1	TL_1	THP_1	HU_1	WR_1
2	TH_2	TL_2	THP_2	HU_2	WR_2
...	...	...	…	...	...
t	TH_t	TL_t	THP_t	HU_t	WR_t

L
L_1
L_2
...
L_t

Fig. 4.2 Process of non-increment regression tree forecasting

In section 4.1 the CART rule for determining when a node is a leaf node was stated. In this research another stopping condition is added for it. This was proposed due to the following fact: for a forecasted result if the dispersion *DEV* is too large, the result is not believable because of the high historical data decentralization. As a result the upper limitation of DEV_{max} is set. Therefore besides condition 1 and condition 2, the third condition is added:

- Condition 3: The dispersion of the subset is less than DEV_{max}

In this case condition 2 can be ignored, because when all the sample points of the subset have the same output value, $DEV(k) = 0$, and this is just a special case for condition 3. Therefore once condition 1 or condition 3 is satisfied, the subset is thought to be a leaf node. For node *k* the algorithm executes the following procedure to decide if it is a leaf node:

```
If $N \geq N_{min}$
    If $DEV(k) < DEV_{max}$
        Node $k$ is regarded as a leaf node and not split any more
    Else
        Go on splitting
    End
Else
    Node $k$ is not split according to Eq. (4.3)
End
```

In this thesis $DEV_{max} = 175MW$ and $N_{min} = 5$. The forecasting result shows that this method often leads to a good result, especially for the leaf nodes that contain a large number of samples and small dispersion. But there are still some leaf nodes that either contain insufficient number of samples, or have large dispersion. For the target load to be forecasted, whose input values fall into these kind of nodes, there are insufficient similar samples, which often correspond to abnormal weather or special events. Although the sample numbers can be increased by including more historical days (e.g. five years' historical data can be applied rather than only one or two years), it would lead to another problem: because of the change of the economic situation and its corresponding change of consuming electricity, two different days with similar weather and weekday conditions may have totally different load curves if the time interval between them is too long.

4.2.2 Increment regression tree

The idea of increment regression tree comes from the experience of the power system dispatchers. Although they don't use any algorithm in STLF, their prediction result is usually more accurate than many complex algorithms. That’s why in some power companies manual STLF is employed instead of the computer prediction. One of their key ways for prediction is to compare the forecasting target day condition with several previous reference days and predict the increment with experience. Unlike the non-increment RT, which regards every historical day as a sample case, the presented approach focuses on the difference of two different days. Suppose the historical days are $1,2,\ldots,t$, and the p^{th} point of day $t + 1$ is to be forecasted, the following is the procedure of increment regression tree method for STLF.

Select two days t_1 and t_2 which are in the historical database. The comparison of day t_1 and t_2 is regarded as a sample object of the increment regression tree. The independent variable has the form

$$[TH, TL, THP, HU, DTH, DTL, DTHP, SR, DHU],$$

whose meaning is shown in Tab. 4.3.

Tab. 4.3 Input variable definition for increment regression tree

Parameter	**Definition**
TH	The highest temperature of day t_2 (℃)
TL	The lowest temperature of day t_2 (℃)
THP	The highest temperature of day t_2’s previous day (℃)
HU	The average humidity of day t_2 (%)
DTH	The highest temperature difference between day t_2 and t_1 (℃)
DTL	The lowest temperature difference between day t_2 and t_1 (℃)
DTHP	The highest temperature difference between day t_2 - 1 and t_1 - 1 (℃)
SR	Whether t_1 and t_2 have the same day rank in a week
DHU	The average humidity difference between day t_2 and t_1 (%)

The response variable is the relative increment of load of day t_1 and t_2 $DL_{t2-t1} = (L_{t2} - L_{t1}) / L_{t1}$ where L_{t1}, L_{t2} are respectively the p^{th} load of day t_1 and t_2.

Here the response value of the regression tree is a relative increment value, therefore this method is named “relative value increment regression tree”, to distinguish it from the “absolute value increment regression tree” method that will be introduced in the later part of this section.

Suppose day indices from 1 to d are in the historical database, theoretically there are $d - 1 + d - 2 + \ldots + 1 = (d - 1)d / 2$ samples; this might lead to an overlarge tree when d is very large. Based on the dispatchers’ experience, only the difference of adjacent days is meaningful in comparison, because the load difference between days with a long interval doesn’t show statistical significance. Therefore the upper limitation of the day difference $DDay_{max}$ is set, and only the difference of day t_i, t_j that satisfy $|t_i - t_j| < DDay_{max}$ are valid in forming an increment sample. All the qualified historical samples are selected, the independent and dependent variables of which are employed to form the regression tree.

In order to forecast the object load of day $t + 1$, first find its adjacent days: $l_1, l_2, \ldots, l_n$ as reference days. All the adjacent days should satisfy the requirement of $|l_i - (t + 1)| < DDay_{max}$. For every reference day l_i, the independent variables can be obtained:

$$[TH_{li}, TL_{li}, THP_{li}, HU_{li}, SR_{(t+1)-li}, DTH_{(t+1)-li}, DTL_{(t+1)-li}, DTHP_{(t+1)-li}, DHU_{(t+1)-li}].$$

Use these variables in the regression tree to reach a leaf node and the related load increment value $DL_{(t+1)-li}$. The corresponding dispersion and the node sample number of the leaf node can be obtained. Suppose the p^{th} load of reference day l_i is L_{li}, which is named in this research "base load", the prediction value based on it is L_{li} $(1 + DL_{(t+1)-li})$. Fig. 4.3 shows the process of the relative value increment regression tree forecasting.

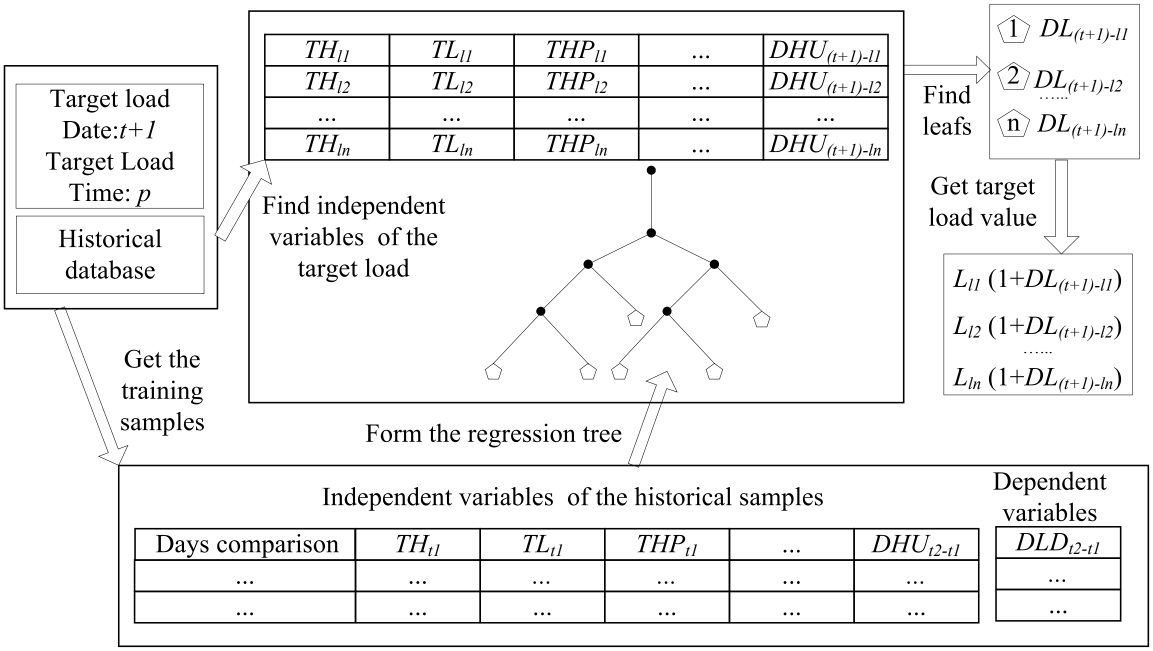

Target load
Date: $t+1$
Target Load
Time: p
Historical database
Find independent variables of the target load
TH_{l1} | TL_{l1} | THP_{l1} | … | $DHU_{(t+1)-l1}$
TH_{l2} | TL_{l2} | THP_{l2} | … | $DHU_{(t+1)-l2}$
… | … | … | … | …
TH_{ln} | TL_{ln} | THP_{ln} | … | $DHU_{(t+1)-ln}$
Find leafs
1 $DL_{(t+1)-l1}$
2 $DL_{(t+1)-l2}$
……
n $DL_{(t+1)-ln}$
Get target load value
L_{l1} $(1+DL_{(t+1)-l1})$
L_{l2} $(1+DL_{(t+1)-l2})$
……
L_{ln} $(1+DL_{(t+1)-ln})$
Get the training samples
Form the regression tree
Independent variables of the historical samples
Days comparison | TH_{t1} | TL_{t1} | THP_{t1} | … | DHU_{t2-t1}
… | … | … | … | … | …
… | … | … | … | … | …
Dependent variables
DLD_{t2-t1}
…
…

Fig. 4.3 Process of relative value increment regression tree forecasting

Similar to the relative value increment regression tree approach, another kind of regression tree method is proposed, which is named absolute value increment regression tree. Its input variables are the same as relative value increment regression tree. The only difference is in the output variable; here it employs the absolute value of the load increment: $DL_{t2-t1} = (L_{t2} - L_{t1})$. The way of forming the regression tree is the same as the relative value method. The corresponding dispersion and the node sample number of the leaf node can be obtained. Suppose the p^{th} load of reference day l_i is L_{li} and $DL_{(t+1)-li}$ is the leaf node value of the target input variable, then the predicted load value based on them is $L_{li} + DL_{(t+1)-li}$.

Whatever kind of regression tree is employed, the output leaf node has two properties: number of samples and standard deviation. Because it can never be known in advance the error of using the leaf node output value as the prediction of the actual value, the standard deviation is simply regarded as a measure of the real error. However, their values are not really the same, but statistically the standard deviation is in accordance with the error. It is just thought that the lower the standard deviation is, the more possible a low forecasted error might be obtained. Hence it is used as a measure of the error.

4.2.3 Tree prediction result combination

Similar to the non-increment tree, the upper limitation of dispersion and the lower limitation of leaf node sample number are also set for regression tree: DEV_{max}, N_{min}. In our research DEV_{max} = 2.75% and $N_{min} = 7$ for relative value regression tree, and DEV_{max} = 300MW and $N_{min} = 7$ for absolute value regression tree.

Suppose in the relative value RT for the n reference days, k of them are within valid leaf node ($DEV_{li} \le DEV_{max}$ and $N_{li} \ge N_{min}$), then k prediction values of the future load can be obtained. In addition, suppose in the absolute value RT for the n reference days, m of them are within valid leaf node, then m prediction values of the future load can also be obtained. For convenience here the indices of the qualified relative value RT leaf node are named $q_1, q_2 \ldots q_k$. and the indices of the qualified absolute value RT leaf node are named $q_{k+1}, q_{k+2} \ldots q_{k+m}$. DEV_{qi} means the standard deviation of the node q_i, and L_{qi} means the base load of node q_i. Furthermore there is the non-increment RT prediction result of leaf node value and dispersion. For convenience the dispersion is labeled as $DEV_{q(k+m+1)}$, and the node value as $L_{q(k+m+1)}$. To combine the leaf node values in a reasonable way, the weighted average method is applied to calculate the wanted load, introduced as follows.

$$CONF_i = 1/(DEV_{qi} \cdot L_{qi}), \qquad i = 1, \ldots k \tag{4.4}$$

L_{qi} means the p^{th} load of day q_i, and DL_{qi} is the increment RT leaf node output of day q_i. $CONF_i$ is defined as the confidence of the q_i^{th} forecasted result. Since the standard deviation of the q_i^{th}

forecast result is DEV_{qi}, approximately the upper and lower value of the forecasted target load are regarded to be respectively $L_{qi}(1 + DL_{qi} + DEV_{qi})$ and $L_{qi}(1 + DL_{qi} - DEV_{qi})$ for relative value RT. Both of these two have an absolute difference from the forecasted target load of $DEV_{qi}L_{qi}$. Consequently (4.4) is employed as the accuracy measurement with unit for the relative value increment tree. $CONF_i$ is referred to as the measurement of the precision of the result i^{th} result, so it represents the confidence of the i^{th} forecasted result.

Similarly the confidence of the absolute value increment RT and non-increment RT can be acquired:

$$CONF_i = 1 / DEV_{qi}, \quad i = k+1, \ldots, k+m+1 \tag{4.5}$$

Summing up all confidence of increment RT and non-increment RT results, the total confidence *TOTAL_CONF* is obtained:

$$TOTAL\_CONF = \sum_{i=1}^{k+m+1} CONF_i \tag{4.6}$$

Define W_i as the weight of the i^{th} forecasting result in the final:

$$W_i = CONF_i / TOTAL\_CONF \qquad i = 1, \ldots k+m+1 \tag{4.7}$$

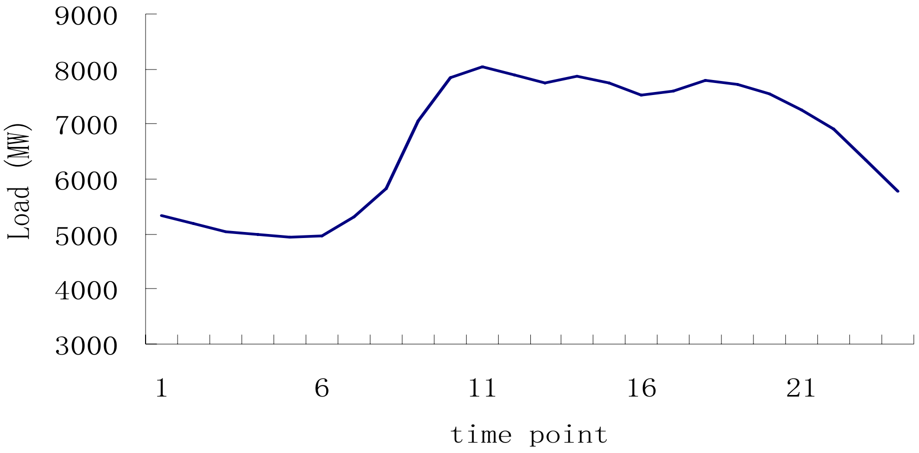
Equation (4.7) shows that the larger $CONF_i$, the larger W_i. This follows the rule “the more data density in the leaf node, the more reliable the result is”. For the q_i^{th} forecast result of relative value increment RT, L_{qi} is the base load of node q_i, the forecasted target load is $L_{qi}(1 + DL_{qi})$. In absolute value increment RT, the forecasted target load is $L_{qi} + DL_{qi}$. $D_{q(k+m+1)}$ is the forecasted value of the non-increment RT. All the $k + m + 1$ predicted results are be averaged according to their weights:

$$L = \sum_{i=1}^{k} W_i L_{qi}(1 + DL_{qi}) + \sum_{i=k+1}^{k+m} W_i (L_{qi} + DL_{qi}) + L_{q(k+m+1)} W_{k+m+1} \tag{4.8}$$

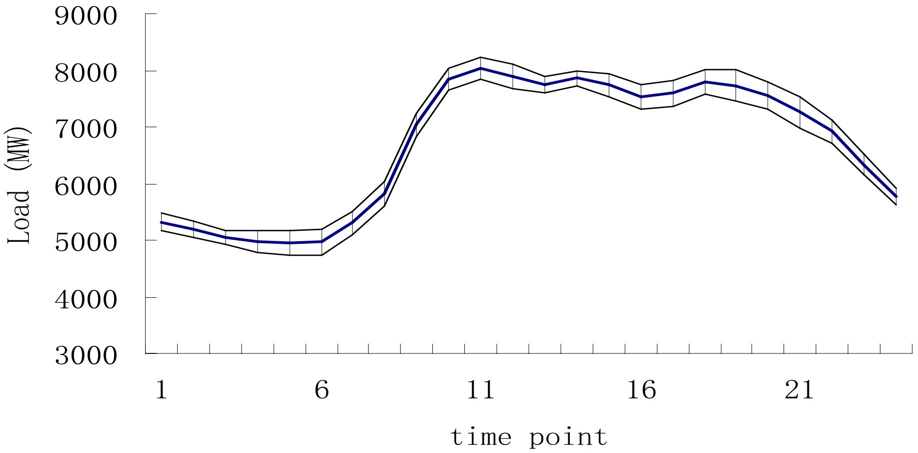
If sample point q_i’s relative value increment RT forecasted load has an error of ER_i, and it contributes to the integrated result W_i, so the error it contributes is $W_i \cdot LD_{qi} \cdot ER_i$. Similarly the errors contributed by the non-increment RT and absolute value RT can also be obtained. In forecasting people certainly do not know the actual error of every forecasting result, so the dispersion is just regarded as the possible error, and the total error indicator (*TEI*) is defined as

$$TEI = \sum_{i=1}^{k} W_i L_{qi} DEV_{qi} + \sum_{i=k+1}^{k+m+1} W_i DEV_{qi} \tag{4.9}$$

This is not equal to the forecasting error due to two facts: 1) the dispersion is not the actual error; 2) the absolute value of the estimated error DEV_{qi} is utilized since the sign of the error is actually unknown. But it gives the users an indicator of the probable error, which can be used to estimate the forecasting error. In normal ways of load forecasting, the result is a single load curve, but the introduction of TEI enables people to get the possible area of the future load, shown in Fig. 4.4. Fig. 4.4(a) shows a normal forecasted load curve, and (b) shows the forecasted area. This area is bordered by two curves: the higher curve corresponds to $L + TEI$, and the lower one $L - TEI$, and the curve between these two is the forecasted load curve. This helps the power system staff to make economical and reliable decisions on reserve capacity.



(a) Normal forecasted load curve



(b) Forecasted possible load area

Fig. 4.4 Comparison of two forecasted results

The experienced dispatchers predict the target load in many different ways and combine the forecasted results. The weighted average method takes advantage of the human's experience of predicting, combining several individual forecasted results according to their weights. This "average" result prevents the error from being overlarge. Especially when there is an undetected historical bad datum or abnormal datum, which is regarded as base load. Although this affects the result, its contribution is only a small fraction, due to the participation of the other forecasted results based on the correct base load. This adds to the robustness of the algorithm.

From the observation and analysis of every day load, in a week with the same time point, the conclusion can be drawn that from Monday to Friday the load values are similar and those of Saturday and Sunday are lower. Due to this, our research constructs six different trees, the explanation of which is shown in Tab. 4.4. Note that every kind of increment tree can be further divided into two types: relative value and absolute value. This helps to decrease the dimension of the problem.

Tab. 4.4 Explanation of different increment trees

Tree Name	Day 1	Day 2
Pure weekday increment tree	Mo ,Tu, We ,Th ,Fr	Mo ,Tu, We ,Th ,Fr
Pure Saturday increment tree	Sa	Sa
Pure Sunday increment tree	Su	Su
Pure weekday-Saturday increment tree	Mo ,Tu, We ,Th ,Fr	Sa
Pure weekday-Sunday increment tree	Mo ,Tu, We ,Th ,Fr	Su
Pure Saturday-Sunday increment tree	Sa	Su

4.2.4 Finding the desert border variable

According to the generated RT, every input variable can reach its leaf node. As mentioned before, the leaf node with small dispersion and large sample number is thought to be valid. But after a lot of simulation experiments a special case has been found, which would affect the forecasting result seriously. It is named the "desert border" case. In this case, although the independent variable can correspond to a seemingly good leaf node with small dispersion and large sample number, one (or more) of the independent variable components is far different from this component of the samples in this leaf node. Such an independent variable is mentioned in this thesis as the "desert border variable". In such a case, since the independent variable value is not very similar to the samples in the leaf node, the corresponding real value might also be far from the output value in the leaf node. Therefore, the prediction of the desert border variable should be discarded. This can be illustrated in Fig. 4.5, where the input variables (x_1, x_2) are 2-

dimensional. All the round points are sample points, and the area inside the contour surrounding the sample points suggests the feature of these points. The leaf node sample points correspond to the rule “$76 \le x_2 < 94$”. Note in the rule variable x_1 is not mentioned. Now let’s have a look at the target points: point A($x_1 = 16$, $x_2 = 87$) and B($x_1 = 7$, $x_2 = 90$). They all satisfy the leaf node rule, but it can be clearly seen in the figure that A is within the contour and B isn’t. In this case A is a desert border point. Desert border phenomena often appear in STLF when the weather of the target day shows a sudden change compared with the previous days. Some desert border point examples will be shown in the example section.

To make sure the target independent variable $\mathbf{x}_t$ in not a desert border point, find all the $\mathbf{x}_1,\mathbf{x}_2 \ldots \mathbf{x}_n$ which are the training samples that form the leaf node. Suppose the independent variable in m-dimensional, e.g.

$$\mathbf{x}_1 = [x_{11}, x_{12}, \ldots x_{1m}],\ \mathbf{x}_2 = [x_{21}, x_{22} \ldots x_{2m}] \ldots \mathbf{x}_n = [x_{n1}, x_{n2} \ldots x_{nm}],$$

and the independent variable of the target load $\mathbf{x}_t = [x_{t1}, x_{t2}, \ldots, x_{tm}]$. Examine if every component of $\mathbf{x}_t$ is a desert border component or not. Take the first component as an example: form the series $[x_{11}, x_{21}, \ldots, x_{n1}]$, and calculate the average value $\overline{x_1}$ as well as dispersion DEV_1. Set $\lambda \in [1,3]$ as a deviation coefficient. If $\overline{x_{t1}} \notin [\overline{x_1} - \lambda DEV_1, \overline{x_1} + \lambda DEV_1]$, the first component is regarded as a desert border component. Check the other m - 1 components in this way. If any of the components is a desert border component, the input variable is regarded as a desert border point and the leaf node output value cannot be taken as the forecasted value. Now look again at Fig. 4.5. Employing this method of detection it can be seen that the input variable x_1 of B is a desert border component according to the x_1 values of the sample points, although B satisfies the leaf node rule. Therefore B will not be considered as being in the same cluster of the leaf node.

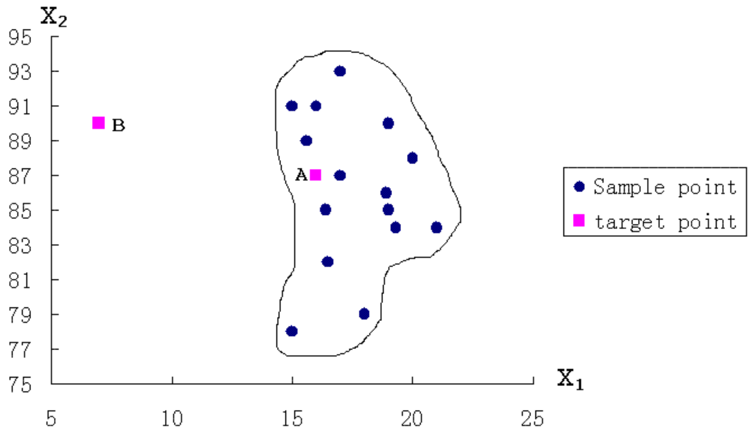


Fig. 4.5 Illustration of desert border point

The following steps are the procedure of desert border variable detection

```
1) For $i$ = 1:$m$, execute the following loop.
2) Calculate the average value $\overline{x}_i$ and dispersion $DEV_i$ of the $i^{th}$ component of the leaf
   node input variable samples $[x_{1i}, x_{2i}, \ldots, x_{ni}]$
    If $\overline{x}_{ti} \notin [\overline{x}_i - \lambda dev_i, \overline{x}_i + \lambda dev_i]$
        It is regarded as a desert border component, and the leaf node is regarded as a
        desert border point and break the "for" loop.
    Else
        It is not regarded as a desert border component.
    End
```

4.3 Historical Data Selection

For a good load prediction, not only the input variables, but also the historical data need to be selected. Usually the data near the forecasting point have more similar property to the target load than the distant ones. But if only a few adjacent data are selected, the sample number might not be enough. In this thesis, RT can be used together with the dispatchers' experience to solve this contradiction.

In selecting samples, normally the near date samples have more similarity to the target load than the distant date. Therefore, our principle of using historical date is that, if in the near date there are enough similar days, they are used as sample days instead of taking the distant date into consideration. But sometimes there might be the "input variable sudden change date", especially the sudden change of the temperature. For example, there are a hundred continuous days with the highest temperature lower than 27 degrees, but the following day's highest temperature is 32 degrees. Only using the nearby samples might lead to the "desert border problem" and, in turn, no prediction result can be obtained. In this case, it is better to turn to around this time last year for more close samples.

Suppose the target day is $t + 1$, and day $1^{st} \ldots d^{th}$ historical days' load, weather and date information is accessible. The load curve has very strong yearly periodicity. From the dispatchers' experience, normally the load curves are similar only when they are located in the adjacent relative position of the year (although they might be located in different years). So in this research the maximum relative position difference of two days, Day_Day_{max}, is also set. Suppose the historical day i is the T_i^{th} day of the year it belongs to, and the target day $(t + 1)$ is the T_{t+1}^{th} day of its year. It is required that only the days that satisfy

$$\left|T_i - T_{t+1}\right| \le Day\_Day_{max} \tag{4.10}$$

are regarded as qualified training data.

The selection of the historical date should follow the rule of "the nearer the historical time to the target, the more priority it has for training". Fig. 4.6 shows the diagram of the historical data selection. Day $t + 1$ is the target date. The days with triangle satisfy the requirement of (4.10). The rows correspond to different years: $Y_1, Y_2 \ldots Y_m$, where Y_m is the target year.

Divide the qualified days of every year into $2n$ equal (or nearly equal) columns. Therefore for every historical year Y_1 to Y_{m-1} there are the historical training columns D_1, $D_2, \ldots D_{2n}$. For the target year Y_m, since D_{n+1} to D_{2n} are future dates, only the columns D_1, $D_2, \ldots D_n$ contain historical training data. Now the potential historical sample days have been divided into $(2m - 1)n$ segments. Every individual segment is uniquely identified by its year ID (identity) and column ID. For simplicity the segment with year ID i and column ID j is named as Segment Y_iD_j. To satisfy the rule of "the nearer the historical date to the target, the more priority it has for training", arrange the segments according to their distance to the target day:

Y_mD_n, Y_mD_{n-1}, $Y_mD_{n-2}, \ldots, Y_mD_1$, $Y_{m-1}D_{n+1}$, $Y_{m-1}D_n$, $Y_{m-1}D_{n+2}$, $Y_{m-1}D_{n-1}, \ldots, Y_{m-1}D_{2n}$, $Y_{m-1}D_1, \ldots, Y_1D_{n+1}$, Y_1D_n, Y_1D_{n+2}, $Y_1D_{n-1}, \ldots, Y_1D_{2n}$, Y_1D_1.

For convenience, these segments are called B_1, $B_2, \ldots B_{(2m-1)n}$, and a $(2m - 1)n$-element historical training data sequence **BB** is constituted:

$$[B_1, B_1 + B_2, B_1 + B_2 + B_3, \ldots, B_1 + B_2 + \ldots + B_{(2m-1)n}],$$

or simply $[BB_1, BB_2, \ldots, BB_{(2m-1)n}]$. Those data corresponding to the segments will be trained in turn until the satisfactory prediction is obtained.

To determine what is a satisfactory prediction, two parameters are set beforehand. TEI_{max} is the upper limitation of total error indicator; and RN_{min}, which is the lower limitation of the result number. In the final result, it is believed that when

$$TEI < TEI_{max} \text{ and } k + m + 1 \ge RN_{min},$$

the result is reliable.

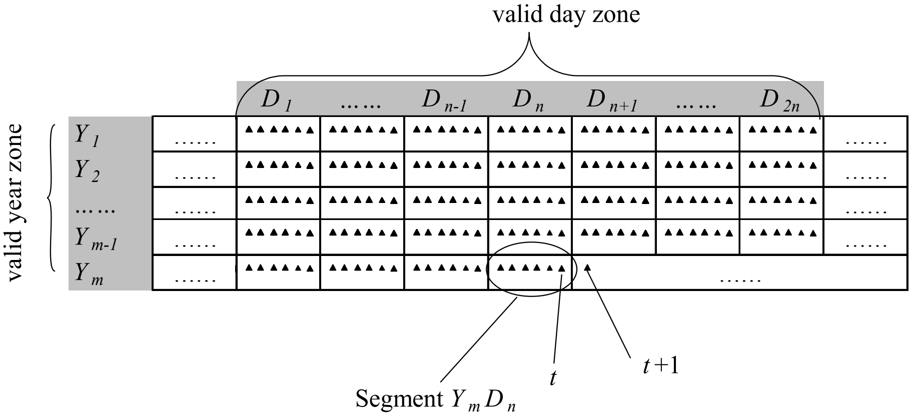


Fig. 4.6 Demonstration of selection of the historical days

The following is the procedure of main forecasting. First use BB_1 as training set, and find out if the forecasting result has satisfactory total error indicator and result number. If not, use BB_2 to enhance the historical date scale. Do it repetitiously until the good estimated result is obtained. Fig. 4.7 shows the process. If the elements of ***BB*** have been gone through and no satisfactory result has been found, the standard can be lowered or other forecasting ways can be sought for a better result.

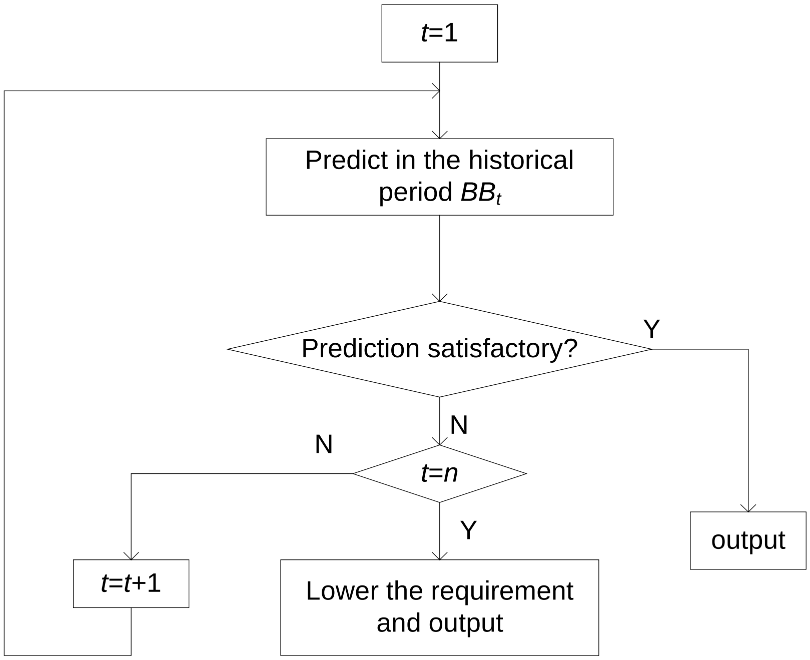

t=1
Predict in the historical period BBt
Prediction satisfactory?
Y
N
N
t=n
Y
t=t+1
Lower the requirement and output
output

Fig. 4.7 Process of using the data in BB

The above-mentioned way of choosing historical sample days is limited to increment RT in this research. Non-increment regression tree can only take the samples within one year because of the possible load change of each year.

Any of the following treatments can be done as a compromise when the elements of ***BB*** have been used and no satisfactory result has been found:

- Increase the TEI_{max} and calculate again.
- Decrease RN_{min} and calculate again.
- Seek for other calculating methods not involving iteration.

Fig. 4.8 shows the RT load forecasting system schematic diagram proposed in this chapter.

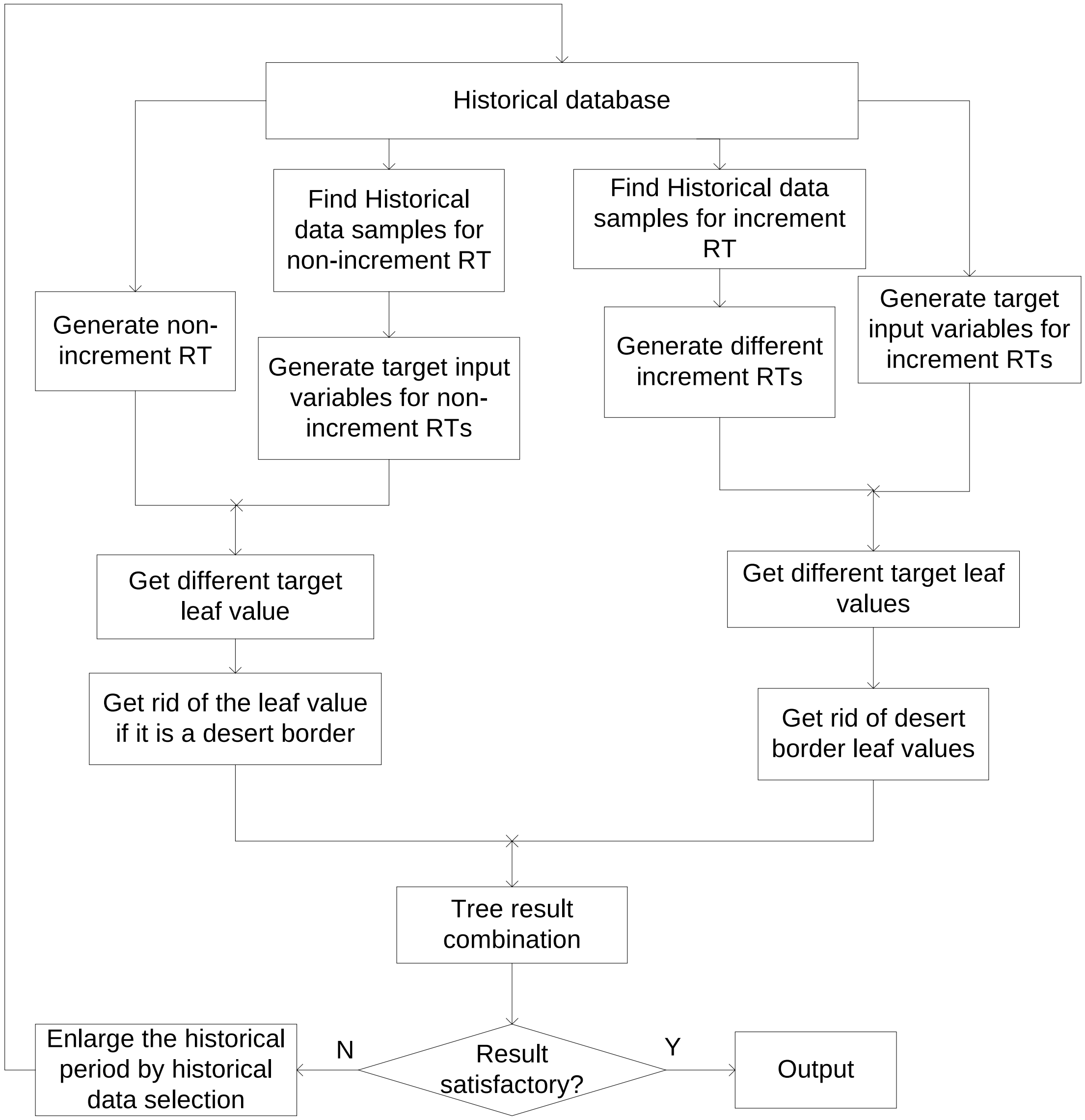

Historical database
Find Historical data samples for non-increment RT
Find Historical data samples for increment RT
Generate non-increment RT
Generate target input variables for non-increment RTs
Generate different increment RTs
Generate target input variables for increment RTs
Get different target leaf value
Get different target leaf values
Get rid of the leaf value if it is a desert border
Get rid of desert border leaf values
Tree result combination
Enlarge the historical period by historical data selection
N
Result satisfactory?
Y
Output

Fig. 4.8 Regression tree load forecasting system schematic diagram

4.4 Case Study

The proposed method is applied to Shanghai Power Grid historical load data of 2000-2002 to test its effectiveness.

Fig. 4.9 shows the non-increment tree of 2002.04.26(Friday), 12:30PM. Tab. 4.5 shows the statistics of the leaf nodes of Fig. 4.9. Nodes *e*, *h* and *k* are not qualified due to a small node sample number. The others are accepted nodes. From this example it can be found that the regression is intuitively easy to understand in analyzing the relationship between input and output variables. The input variable of 2002.04.26 is

$$[TH, TL, THP, HU, WR] = [16.8℃, 11.9℃, 12.7℃, 70.4\%, 5],$$

so it falls on node *i*, and the predicted load is 7115MW.

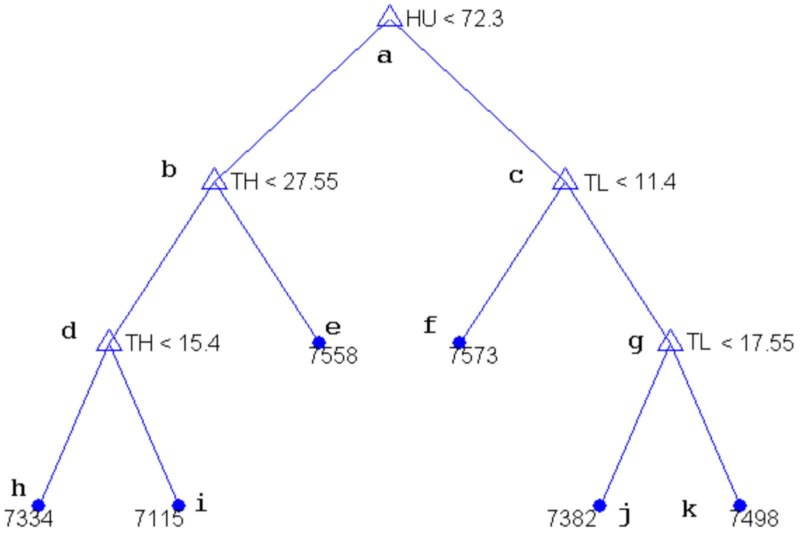


Fig. 4.9 Non-increment tree of 2002.04.26, 12:30PM

Tab. 4.5 Statistics of the leaf nodes of Fig. 4.9

Leaf Node Name	**Node Sample Number**	**Node Dispersion (MW)**	**Node Output (MW)**	**Input Variable Range**
e	2	106	7558	$TH \geq 27.55℃\ HU < 72.3\%$
f	5	146	7573	$TL < 11.4℃\ HU \geq 72.3\%$
h	4	111	7334	$TH < 15.4℃\ HU < 72.3\%$
i	8	69	7115	$15.4℃ \leq TH < 27.55℃\ HU < 72.3\%$
j	9	79	7328	$11.4℃ \leq TL < 17.55℃\ HU \geq 72.3\%$
k	2	45	7498	$TL \geq 17.55℃\ HU \geq 72.3\%$

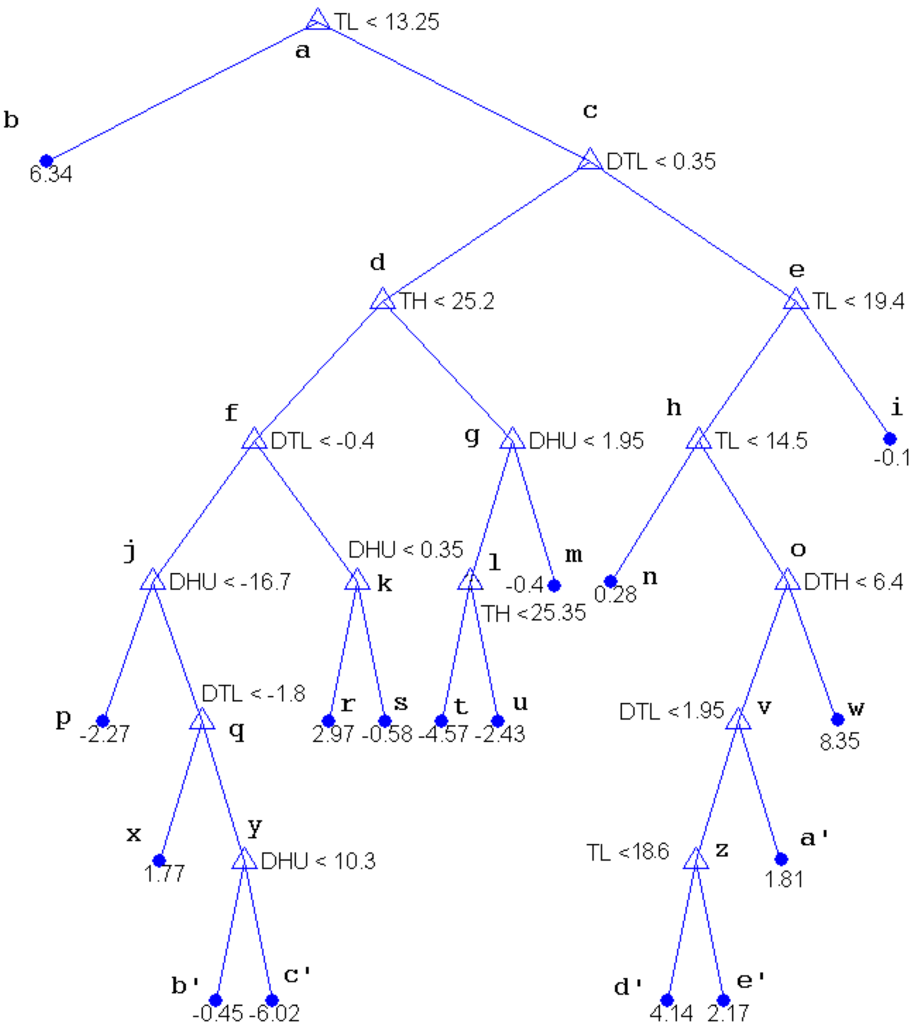


Fig. 4.10 Increment tree of 2002.04.26, 12:30PM

Fig. 4.10 shows the relative value incremental tree of the same point. Since the complete tree is too large to be shown in the thesis, only a small fraction of the tree is shown. Note that node *a* in the tree is not the root node, but one of the intermediate nodes of the tree. It satisfies the condition

$$DTL \geqslant -2.65^{\circ}\mathrm{C},\ TH \geq 12.25^{\circ}\mathrm{C},\ TL \geqslant 13.15^{\circ}\mathrm{C},\ HU \geqslant 65.7\%.$$

Tab. 4.6 shows the statistics of the leaf nodes of Fig. 4.10. Due to large dispersion or small sample number, nodes *b*, *i*, *r*, *s*, *t*, *w*, and *c'* are not qualified. Every leaf node has a “ if…then…” rule, and a corresponding load percentage increment. For example, leaf node *m* corresponds to rule:

“if $-2.65^{\circ}\mathrm{C} \leq DTL < 0.35^{\circ}\mathrm{C}$ and $TL \geq 13.25^{\circ}\mathrm{C}$ and $HU \geq 65.7\%$ and $TH \geq 25.2^{\circ}\mathrm{C}$ and $DHU \geq 1.95\%$, then the load increment is –0.4%”.

Tab. 4.6 Statistics of the leaf nodes of Fig. 4.10

Leaf Node Name	**Node Sample Number**	**Node Dispersion (%)**	**Node Output Value (%)**
b	10	3.5	6.34
i	11	3.2	-0.1
m	12	2.75	-0.4
n	7	2.2	0.28
p	10	1.86	-2.27
r	11	3.1	2.97
s	5	1.68	-0.58
t	6	1.76	-4.57
u	12	1.54	-2.43
w	3	2.76	8.35
x	10	1.44	1.77
a'	13	2.26	1.81
b'	14	2.25	-0.45
c'	2	1.19	-6.02
d'	11	1.76	4.14
e'	10	1.72	2.17

To show an example of relative value increment RT method, the load of 2002.04.24 is used. The input value

$$[TH, TL, THP, HU, DTH, DTL, DTHP, SR, DHU]$$

$$= [20.8℃, 14.0℃, 19.7℃, 95.3\%, -4℃, -2.1℃, -1.9℃, \text{false}, -24.9\%].$$

According to the regression tree, the input value leads to node *p*. The reference load in 2002.04.21 is 7426MW, so the forecasted result is

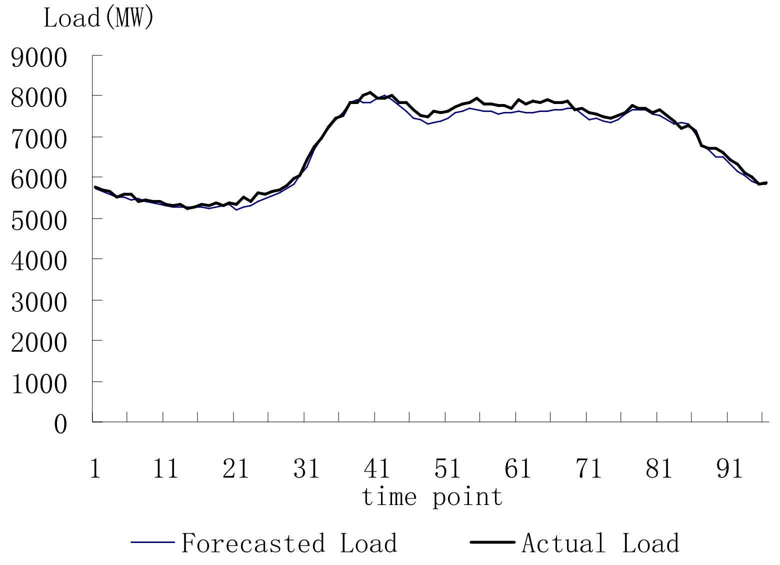
$$7426 \times (1 - 2.27\%) = 7257.2 \text{ MW}.$$

Tab. 4.7 shows the weighted average forecasting result of several trees to predict the load of 2002.04.26, 12:30.

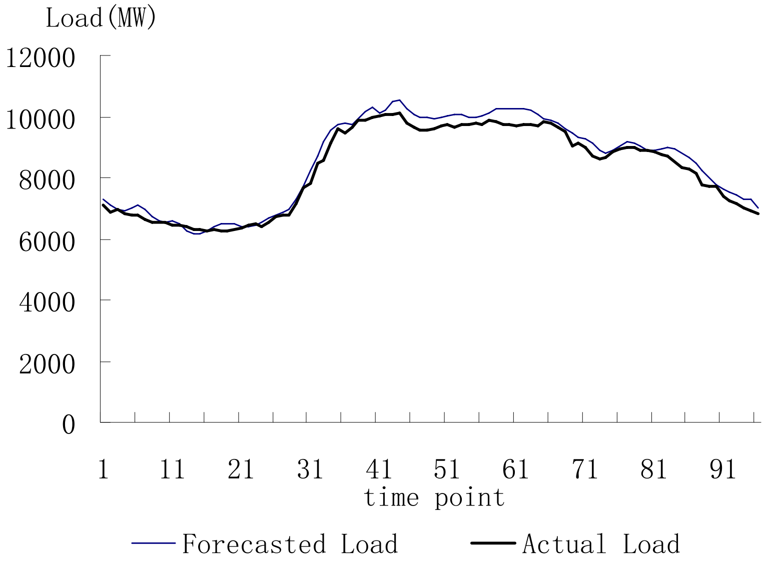
Fig. 4.11 shows four typical seasonal weekday forecasted load curves in comparison with the actual load curves. Weekend load curves are relatively difficult to be forecasted than the normal days due to limited sample numbers. Fig. 4.12 shows four typical seasonal weekend forecasted load curves in comparison with the actual load curves. The figures indicate the error of these days is acceptable.

Tab. 4.7 Integrated weighted average forecasting result of 2002.04.26, 12:30PM

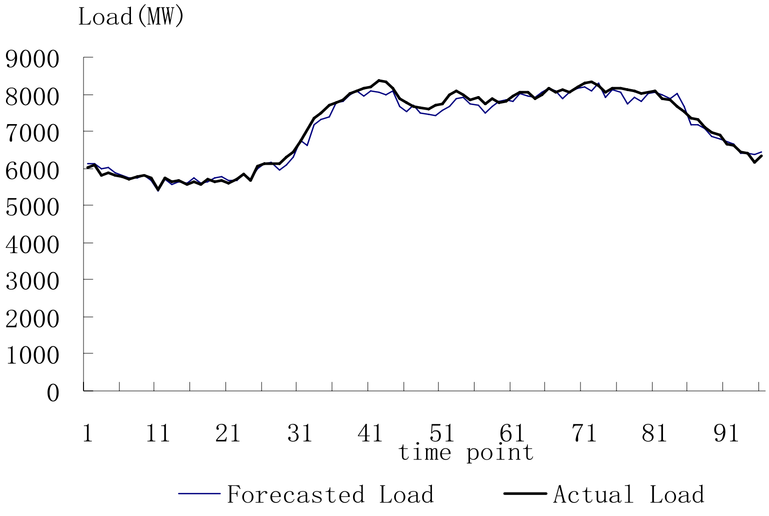
Parameter	Non-increment Tree	Increment Tree 1	Increment Tree 2	Increment Tree 3	Increment Tree 4
Reference load(MW)	No	7466	7426	7485	7218
Reference date	No	02.04.22	02.04.23	02.04.24	02.04.25
Leaf output	7115(MW)	-3.3	-2.27	-1.8	3.4
Forecast result (MW)	7115	7218.2	7257.2	7346.7	7463.4
Dispersion	69(MW)	1.77(%)	1.85(%)	1.54(%)	1.29(%)
Weight	0.29	0.16	0.15	0.17	0.22
Integrated result (MW)	7271.64	**Actual load (MW)**	7395	**Error (%)**	–1.67



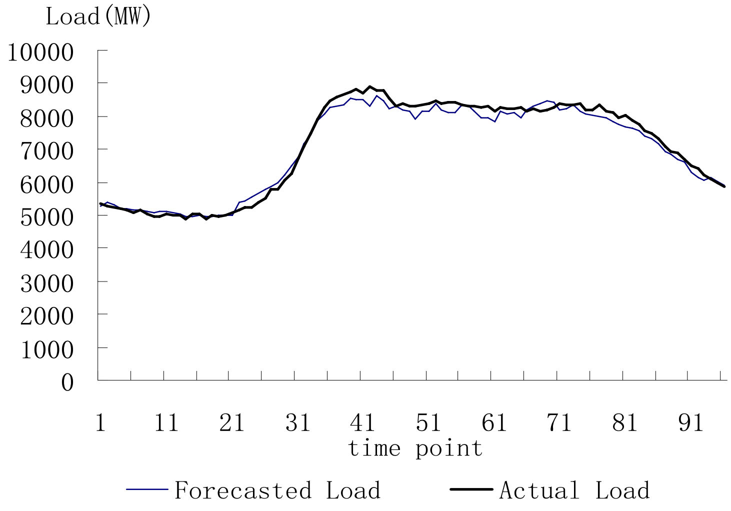
(a) 2002.05.24(typical spring weekday)



(b) 2002.08.08 (typical summer weekday)

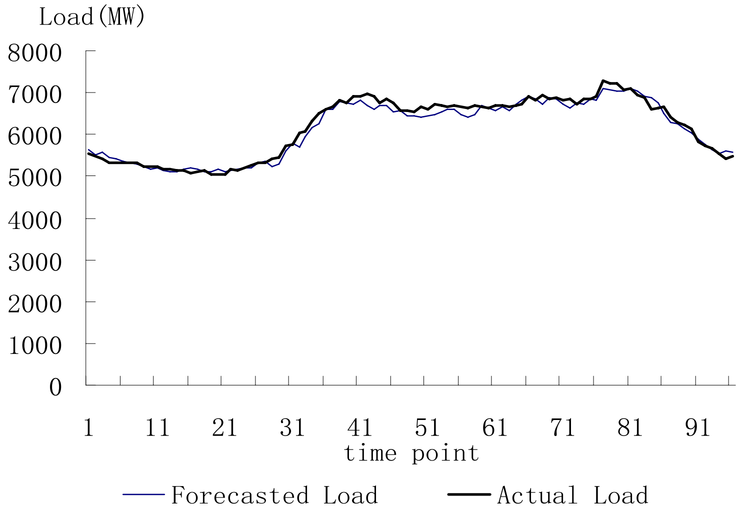


(c) 2002.10.25(typical autumn weekday)

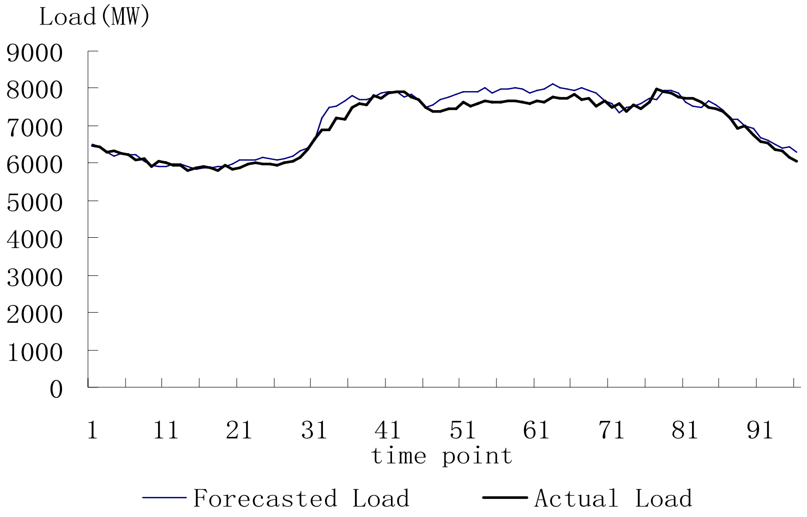


(d) 2002.01.21(typical winter weekday)

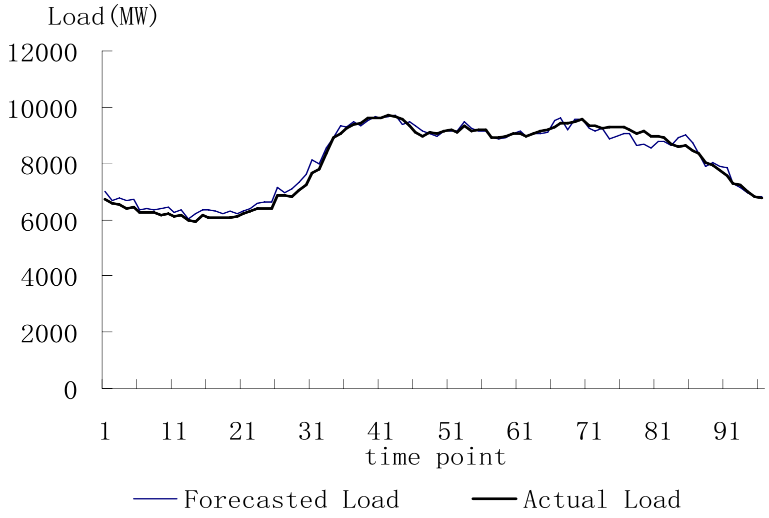
Fig. 4.11 Four typical seasonal weekday forecasted load curves



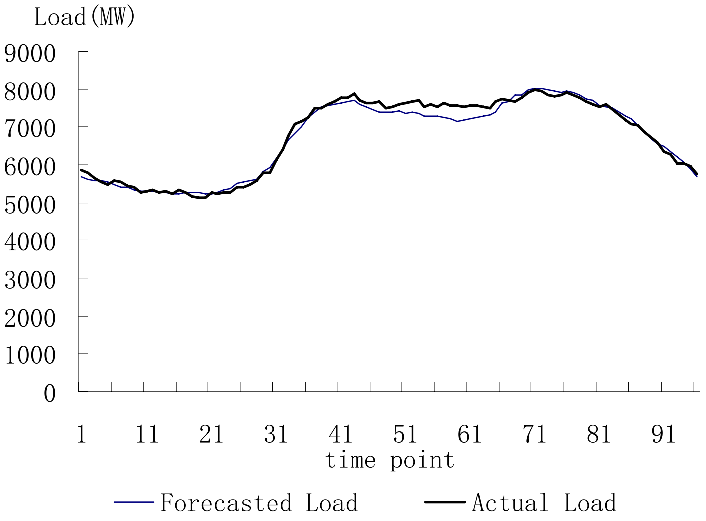
(a) 2002. 05.26(typical spring weekend)



(b)2002.08.10(typical summer weekend)



(c) 2002.11.23(typical autumn weekend)



(d) 2002.01.26 typical winter weekend

Fig. 4.12 Four typical seasonal weekend forecasted load curves

As mentioned before, non-increment RT is good at predicting normal days with large sample number, but behaves poorly for small number sample training. This can be illustrated by Tab. 4.8, which shows the comparison of pure non-increment method forecasting result and the increment + non-increment method forecasting result. The MAPE of every day maximum absolute error is shown, because the non-increment RT behaves especially poorly for this quantity.

Tab. 4.8 MAPE (%) prediction result for the whole year in 2002 in comparison with pure non-increment RT

Method		**Jan**	**Feb**	**Mar**	**Apr**	**May**	**Jun**
1^*	Tree combination	3.27	2.71	1.92	1.54	1.96	1.56
	Non-increment	4.18	3.56	2.07	1.58	2.28	1.62
2^*	Tree combination	10.75	9.39	7.09	7.47	7.28	8.32
	Non-increment	16.25	14.93	12.71	7.74	7.53	9.08
Method		**Jul**	**Aug**	**Sep**	**Oct**	**Nov**	**Dec**
1^*	Tree combination	3.02	3.83	3.63	2.51	2.16	3.43
	Non-increment	3.84	4.20	3.82	2.63	2.49	4.68
2^*	Tree combination	11.32	11.53	8.72	6.77	6.96	10.50
	Non-increment	14.61	17.17	10.96	6.65	7.66	14.42
Method		**Average**					
1^*	Tree combination	2.63					
	Non-increment	3.08					
2^*	Tree combination	8.84					
	Non-increment	11.64					

*1: MAPE for 96 points; 2: MAPE for maximal absolute error of the daily 96 points

Tab. 4.9 shows the prediction result of the whole year in 2002. To show the effectiveness of the proposed method, ANN method was also employed as a comparison. Like RT method, ANN trains the weekday, Saturday and Sunday separately. The input variables employed by non-incrment RT are also employed by ANN.

Since weekend always lead to larger prediction error, Tab. 4.10 also shows the MAPE of weekend prediction. The average monthly MAPE of RT method is 2.63, lower than the error of 2.80 in ANN method. The average monthly weekend MAPE of RT method is 2.99, much lower than the error of 4.07 in ANN method. The comparison indicates the excellence of the RT method.

Tab. 4.9 MAPE (%) prediction results for the whole year in 2002.

Method	**Jan**	**Feb**	**Mar**	**Apr**	**May**	**Jun**
Proposed method	3.27	2.71	1.92	1.54	1.96	1.56
ANN	3.58	2.49	2.11	1.51	1.89	1.90
Method	**Jul**	**Aug**	**Sep**	**Oct**	**Nov**	**Dec**
Proposed method	3.02	3.83	3.63	2.51	2.16	3.43
ANN	3.23	4.13	3.48	2.35	3.07	3.87
Method	**Average**					
Proposed method	2.63					
ANN	2.80					

Tab. 4.10 Prediction results for the whole year weekend in 2002.

Method	**Jan**	**Feb**	**Mar**	**Apr**	**May**	**Jun**
Proposed method	3.25	2.98	2.05	2.01	2.21	2.15
ANN	5.66	4.34	2.41	2.74	2.88	2.56
Method	**Jul**	**Aug**	**Sep**	**Oct**	**Nov**	**Dec**
Proposed method	3.85	4.16	3.79	3.31	2.50	3.64
ANN	5.02	5.71	4.37	4.29	3.86	4.94
Method	**Average**					
Proposed method	2.99					
ANN	4.07					

In this research$\lambda = 2$. Here the forecasting 17:30 load of 2002-9-25 is taken as an example to illustrate the effectiveness of desert point detection. The highest temperature of this day is 20.1 ℃, which is a sudden drop compared with the previous days. In the formation of an increment absolute value regression tree, the leaf node is met, the property of which is shown in Tab. 4.11. The target input value is

[*DTH, DTL, DHU, TH, TL, HU*] = [-4℃, 1.1℃, 13.8%, 21.9℃, 20.1℃, 68.4%]

the target day to be compared is 2002-9-23, the target load to be compared is 7800.3MW. For variable *TH*, $[x_1 - 2DEV_1, x_1 + 2DEV_1]$ = [23.7℃, 31.0℃]. It can be seen that *TL* is smaller than the lower limitation of *TL*. Therefore it is regarded as "desert border point". If this point is not neglected, its forecasted value should be: 7800.3 - 560.245 = 7240.06MW, which is far away from the real load 8722MW with error of -17.0%.

Tab. 4.12 and Tab. 4.13 show respectively the two comprehensive results of load forecasting: abandoning and not abandoning the desert border point.

Tab. 4.11 Property of the example leaf node

Node	***DTH***	***DTL***	***DHU***	***TH***	***TL***	***HU***	***DL***
1	-3.9	-0.8	10.6	25.5	23.9	94.5	-242.24
2	-4.3	-1.8	14.5	25.5	23.9	94.5	-517.1
3	-2.2	-0.5	17	27.5	26.2	89.8	-786.52
4	-1.5	-0.7	16.5	28.2	26	89.3	-777.65
5	-2.8	-1.4	17.4	30.5	27.1	81	-166.03
6	-1.5	-0.1	14.2	30.5	27.1	81	-216.32
7	-3	-1.4	9.3	29	25.8	76.1	-730.69
8	-1.5	-1.3	-4.9	29	25.8	76.1	-514.38
9	-2.8	-1.2	-7.7	26.2	24.6	68.4	-943.43
10	-1.8	-1.3	-10.6	27.1	25.6	73.5	-697.65
11	-2.5	-1.1	-5.8	27.1	25.6	73.5	-338.17
12	-2.1	-1.2	9.3	25.3	21.2	80.7	-1144.8
13	-1.6	-0.3	-3.9	25.8	22.1	67.5	-282.06
14	-1.6	0	-6.1	25.8	22.4	65.3	-486.39
Average	-2.36	-0.93	4.99	27.36	24.81	79.37	-560.245
Min	-4.18	-2.02	-16.46	23.71	21.09	59.98	
Max	-0.54	0.15	26.43	31.01	28.53	98.76	

Tab. 4.12 Forecasting result of 2002-9-25, without desert border detection

Leaf value	-560.24	-12.11	-12.11	-12.11	-1007.5	-1007.5	-1007.5	-30.32
DEV	296.89	1.95	1.95	1.95	149.9	149.9	149.9	161.55
Reference load	7800.3	8679.6	8568.9	8475.2	8679.6	8568.9	8475.2	7800.3
Method	A*	B*	B*	B*	A*	A*	A*	A*
Base load date	09.23	09.17	09.19	09.20	09.17	09.19	09.20	09.23
Forecasted value	7240	7628.4	7531.2	7448.8	7672.1	7561.5	7467.8	7770
Error Estimated		0.75%		**Forecast Load**		7549.2	**Error**	-13.45%

* A: non-increment absolute value; B: non-increment relative value

Tab. 4.13 Forecasting result of 2002-9-25, with desert border detection

Leaf value	-1.38	-0.84	-0.84	-121.8	-42.80	-192.54	-42.80
DEV	0.73197	1.0816	1.0816	184.35	80.792	261.35	80.792
Reference load	7816.8	8679.6	8475.2	7816.8	8679.6	8568.9	8475.2
Method	B*	B*	B*	A*	A*	A*	A*
Date	09.16	09.17	09.20	09.16	09.17	09.19	09.20
Forecasted value	7708.8	8606.4	8403.8	7695	8636.7	8376.4	8432.4
Error Estimated	0.46%	**Forecast Load**			8261.5	**Error**	-5.28%

* A: non-increment absolute value; B: non-increment relative value

5 Short-term Load Forecasting Based on Support Vector Machine

5.1 Support Vector Machine Theory

Support vector machine (SVM) is a statistical learning method based on statistical analysis and robust regression theory, which has many advantages such as structural risk, high training speed, simple mathematical models with global optimal solution and better learning results than other pattern recognition and regression forecasting methods. It can be employed to deal with three sorts of problems: classification, regression and clustering. In this section the theories of support vector regression (SVR) and support vector clustering are briefly introduced.

5.1.1 Support vector regression

For l training data, in which the i^{th} datum includes independent variable $\mathbf{x}_i \in R^n$ and corresponding dependent variable $y_i \in R$, to seek for the most suitable parameters $\mathbf{x}'$ and b in the regression form of $y(\mathbf{x}_i) = \langle \phi(\mathbf{x}'), \phi(\mathbf{x}_i) \rangle + b$, the model can take the following form

$$\begin{aligned} &\min \frac{1}{2}\|\phi(\mathbf{x}')\|^2 + \frac{C}{l}\sum_{\mathrm{i}=1}^{l}(\varepsilon_{\mathrm{i}} + \varepsilon_i^*) \\ &\text{subject to} \quad \langle \phi(\mathbf{x}'), \phi(\mathbf{x}_i) \rangle + b - y_i \le \varepsilon + \varepsilon_{\mathrm{i}} \\ &\qquad\qquad\quad y_i - \langle \phi(\mathbf{x}'), \phi(\mathbf{x}_i) \rangle - b \le \varepsilon + \varepsilon_i^* \\ &\qquad\qquad\quad \varepsilon_{\mathrm{i}}, \varepsilon_i^* > 0 \end{aligned} \tag{5.1}$$

ϕ is a nonlinear transformation to some high dimensional feature-space. For simplicity, define $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}'), \phi(\mathbf{x}_i) \rangle$. It is the kernel function (or simply kernel) that satisfies the Mercer's condition [40]. ε is the permissive error. ε_i and ε_i^* are the slack errors for the i^{th} training point. C is the punishment constant of the slack errors. According to dual optimization theory, the model can be transformed to the following problem

$$\begin{aligned} &\max -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(\mathbf{x}_i, \mathbf{x}_j) - \varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) - \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \\ &\text{subject to} \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0, \quad \alpha_i, \alpha_i^* \in [0, \frac{C}{l}] \end{aligned} \tag{5.2}$$

where α_i and α_i^* are Lagrange multipliers of (5.1).

Eq. (5.2) is a maximization optimization problem. Multiply the objective function with –1 and preserve the conditions, then a minimization optimization problem can be obtained:

$$\begin{aligned}&\min \frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)k(\mathbf{x}_i,\mathbf{x}_j)+\varepsilon\sum_{i=1}^{l}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{l}y_i(\alpha_i-\alpha_i^*)\\&\text{subject to } \sum_{i=1}^{l}(\alpha_i-\alpha_i^*)=0,\quad \alpha_i,\alpha_i^*\in[0,\frac{C}{l}]\end{aligned} \tag{5.3}$$

This is a standard quadratic programming problem, for which many tools, such as sequential minimal optimization (SMO), interior points and Quasi-Newton, can be used to solve easily.

For certain simple types of algorithms, statistical learning theory can identify rather precisely the factors that need to be taken into account to learn successfully. Real-world applications, however, often mandate the use of more complex models and algorithms, such as neural networks, that are much harder to analyze theoretically. The support vector algorithm achieves both. It constructs models that are complex enough: it contains a large class of neural nets, radial basis function nets, and polynomial classifiers as special cases. Yet it is simple enough to be analyzed mathematically, because it can be shown to correspond to a linear method in a high-dimensional feature space nonlinearly related to input space. Moreover, even though it can be regarded as a linear algorithm in a high-dimensional space, in practice, it does not involve any computations in that high dimensional space. By the use of kernels, all necessary computations are performed directly in input space. This is the characteristic twist of support vector method. People are dealing with complex algorithms for nonlinear pattern recognition, regression, or feature extraction, but for the sake of analysis, people can “pretend” that they are working with a simple linear algorithm.

The objective function of Eq. (5.1) is composed of two parts. The first part measures the smoothness of the feature space (and also that of the input space). It is referred to as “structural risk”; the minimization of this part is to make the final regression form as simple as possible. The second part measures the degree of point error deviations from the tolerated error. This part is regarded as “empirical risk”, the minimization of which is to make the regression as accurate as possible. Other traditional regression tools, such as ANN, linear and nonlinear regression, only deal with the empirical risk. That’s why they often lead to the unexpected overfitting. One outstanding property of SVM is to avoid overfitting by the utilization of structural risk together with empirical risk.

5.1.2 Support vector clustering

As a kernel-based algorithm, support vector clustering [55] is a novel clustering method of kernel-based learning. The algorithm maps the data points from input space to a high dimensional feature space, where support vector machine theory is used to define a sphere enclosing them. The boundary of the sphere in the feature space is mapped back to a set of closed contours containing the data in input space. The dataset enclosed by each contour is defined as a cluster.

Suppose $\{\mathbf{x}_j\}$ is a dataset with N points, with $\mathbf{x}_j \in \mathbb{R}^d$. Using a nonlinear transformation ϕ to map the points from the input space $\mathbb{R}^d$ to a high dimensional feature space, the algorithm looks for the smallest sphere enclosing all the points, described by the constraints: $\left\|\phi(\mathbf{x}_j)-\mathbf{a}\right\|^2 \le R^2$ $\forall j$, where R is the radius of the sphere, $\|\cdot\|$ is the Euclidean norm and $\mathbf{a}$ is the center of the sphere. For the j^{th} point slack variables $\xi_j \ge 0$ is employed to get the soft constraints:

$$\left\|\phi(\mathbf{x}_j)-\mathbf{a}\right\|^2 \le R^2 + \xi_j \tag{5.4}$$

To solve this problem Lagrange multipliers $\alpha_j, \mu_j \ge 0$ and penalty coefficient $C \ge 0$ are introduced to get the following Lagrangian:

$$\min F = R^2 - \sum_j (R^2 + \xi_j - \left\|\phi(\mathbf{x}_j)-\mathbf{a}\right\|^2)\alpha_j - \sum_j \xi_j \mu_j + C\sum_j \xi_j \tag{5.5}$$

In support vector machine, kernel function is defined: $k(\mathbf{x}_1,\mathbf{x}_2) = <\phi(\mathbf{x}_1),\phi(\mathbf{x}_2)>$. There are basically four kinds of kernels: linear function, polynomial function, Gaussian function and sigmoid function. As can be seen from the calculation results in section 5, all of these four functions are employed for support vector clustering, and the Gaussian kernel gives much better results than the other three. So in this section, the Gaussian kernel $k(\mathbf{x}_1,\mathbf{x}_2) = e^{-\gamma\left\|\mathbf{x}_1-\mathbf{x}_2\right\|^2}$ is mainly dealt with and regarded as the default kernel function.

By setting the derivative of (5.5) to zero and applying the Karush-Kuhn-Tucker method [40], the above minimum can be transformed into its dual form, which is a quadratic programming problem:

$$\begin{aligned} &\min \sum_j \left\langle \phi(\mathbf{x}_j),\phi(\mathbf{x}_j)\right\rangle \alpha_j - \sum_{i,j} \alpha_i\alpha_j \left\langle \phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\right\rangle \\ &\text{subject to } \sum_i \alpha_i = 1 \text{ and } 0 \le \alpha_i \le C \end{aligned} \tag{5.6}$$

Eq. (5.5) is a minimization with respect to variables $R, \xi_j, \mathbf{a}, \mu_j, \alpha_i$. But when it is changed to its dual form of Eq. (5.6), only α_i becomes the variable and the other variables are given in terms of α_i.

In the high dimensional feature space, the distance between a point $\mathbf{x}$ and the center of the sphere is defined: $R(\mathbf{x}) = \|\phi(\mathbf{x}) - \mathbf{a}\|$. For the j^{th} point $\mathbf{x}_j$ if $R(\mathbf{x}_j) > R$, point $\mathbf{x}_j$ lies outside the sphere, it is then considered as an "outlier" with corresponding $\xi_j > 0$. In the latter part of the thesis it will be discussed how to deal with the outliers.

Although the mapped data (except the outliers) in the high dimensional feature space are inside a sphere, they might be within different contours in their original input space [56]. In order to classify data points into clusters, the connected components method is introduced as follows.

The criteria of deciding whether two points belong to one cluster is to judge if all the points on the line segment between them can be matched back inside the sphere in the high-dimensional space. If two points $\mathbf{x}_i$ and $\mathbf{x}_j$ are arbitrarily selected from the training dataset, to determine whether they are in the same cluster, firstly set some sampling points on the line segment connecting the two points. An adjacency matrix $\mathbf{A}$ is setup. If any sampling point is an outlier, $\mathbf{x}_i$ and $\mathbf{x}_j$ are regarded to be in different clusters, and element A_{ij} and A_{ji} are set to be false. Otherwise, it is considered that all the points on the line segment connecting point $\mathbf{x}_i$ and point $\mathbf{x}_j$ are not outliers, then $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same cluster, and A_{ij} and A_{ji} are set to be true.

When every element of $\mathbf{A}$ is calculated, the breadth-first-search technique [63] is used to determine different clusters. First put point 1 into cluster 1, and then put all the points that are connected with 1 (all the points $\mathbf{x}_j$ that satisfies A_{1j} = true) into cluster 1. Similarly, for every $\mathbf{x}_j$ that have been detected to be connected with point 1, search all the points that are connected to it and put them in cluster 1. Do this repeatedly until no more points can be put in cluster 1. Now have a look at all the points. If there is still one (or more) that doesn't belong to cluster 1, put it in cluster 2 and put all the points connected with it in cluster 2. The loop is broken until every point has been assigned a cluster number.

Compared with the conventional clustering method, support vector clustering has the following advantages: it doesn't rely on the initial values; the quadratic programming problem of the algorithm is convex and has a globally optimal solution; and it can deal with outliers, making it robust with respect to noise in the data.

5.2 LIBSVM Solution

5.2.1 Basic solution

In section 5.1 the theory of support vector regression and support vector clustering was briefly reviewed. For support vector regression, model (5.3) is obtained; and for support vector clustering, model (5.6) is obtained. Both (5.3) and (5.6) are quadratic optimization problems. For the sake of simplicity and generalization, they can be generalized into the following form:

$$\begin{aligned} &\text{minimize } \frac{1}{2}\alpha^T Q\alpha + p^T\alpha \\ &\text{subject to } \begin{cases} s^T\alpha = \Delta \\ 0 \le \alpha_t \le C, t = 1,2...l \end{cases} \end{aligned} \tag{5.7}$$

The difficulty of equation (5.7) is, when there are many samples, matrix Q is a large-dimensioned, non-sparse matrix. Lin [52] proposed a decomposition algorithm in LIBSVM (a library for support vector machines). The procedure of the algorithm is introduced as follows.

1) Set $q \le l$ as the dimension of the working set, and α^k as the original value of the final solution, and set $k = 1$.

2) If α^k is the optimal solution of equation (5.7), the calculation is stopped. Otherwise, define a q-dimensional working set $B \subset \{1,...l\}$; define $N = \{1,...l\} \setminus B$; define α_B^k and α_N^k as the sub-vectors corresponding respectively to B and N.

3) Solve the following optimization problem with the variable α_B.

$$\begin{aligned} &\mathit{minimize}\ \frac{1}{2}\alpha_B^T Q_{BB}\alpha_B + (p_B + Q_{BN}\alpha_N^k)^T\alpha_B \\ &\mathit{subject\ to} \begin{cases} 0 \le (\alpha_B)_T \le C \quad t = 1,....,q \\ s_B^T\alpha_B = \Delta - s_N^T\alpha_N^k \end{cases} \end{aligned} \tag{5.8}$$

where $\begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix}$ is the reorganization of Q.

4) Define α_B^{k+1} as the optimal solution of (5); let $\alpha_N^{k+1} = \alpha_N^k$, and go to step 2.

5.2.2 Selection of working set

To get the optimal solution of (5.7) quickly and conveniently, the working subset should be carefully selected. Apply the Karush-Kuhn-Tucker (KKT) condition to model (5.7), there should be a scalar *b*, and non-negative vectors λ and μ , which satisfy：

$$\begin{aligned} & Q\alpha + p + bs = \lambda - \mu \\ & \lambda_i \alpha_i = 0 \\ & \mu_i (C - \alpha_i) = 0 \\ & \lambda_i \ge 0, \mu_i \ge 0,\ \ i = 1,...,l \end{aligned} \tag{5.9}$$

Eq. (5.9) can be rewritten as

$$\begin{aligned} & \text{If } \alpha_i\text{=0, } (Q\alpha + p + bs)_i \ge 0 \\ & \text{If } \alpha_i \in (0, C)\text{, } (Q\alpha + p + bs)_i = 0 \\ & \text{If } \alpha_i = C\text{, } (Q\alpha + p + bs)_i \le 0 \end{aligned} \tag{5.10}$$

According to Eq. (5.10)， $s_i = \pm 1, i = 1,...,l$ ， and this further implies：

$$\begin{aligned} & s_t = +1, \alpha_t < C \Rightarrow b \ge -(Q\alpha + p)_t = -\nabla f(\alpha)_t \\ & s_t = -1, \alpha_t > 0 \Rightarrow b \ge (Q\alpha + p)_t = \nabla f(\alpha)_t \\ & s_t = -1, \alpha_t < C \Rightarrow b \le (Q\alpha + p)_t = \nabla f(\alpha)_t \\ & s_t = +1, \alpha_t > 0 \Rightarrow b \le -(Q\alpha + p)_t = -\nabla f(\alpha)_t \end{aligned} \tag{5.11}$$

where $f(\alpha)$ is the objective function of (5.7), and $\nabla f(\alpha)$ is the gradient of $f(\alpha)$ at α .

Define

$$\begin{aligned} & i \equiv \arg\max(\{-\nabla f(\alpha)_t | s_t = +1, \alpha_t < C\}, \{\nabla f(\alpha)_t | s_t = -1, \alpha_t > 0\}) \\ & j \equiv \arg\min(\{\nabla f(\alpha)_t | s_t = -1, \alpha_t < C\}, \{-\nabla f(\alpha)_t | s_t = -1, \alpha_t > 0\}) \end{aligned} \tag{5.12}$$

and select B＝{i,j}. Here the dimension of B is chosen to be 2 rather than other values, so that Eq. (5.8) become a typical quadratic optimization with the analytical solution. Compared with the common optimal problem solutions, this algorithm needs less memory space and is more precise and faster.

5.3 Application of Support Vector Machine to STLF

In this thesis, a support vector clustering and regression method for short-term load forecasting is proposed. The method consists of three cascaded major modules: support vector clustering, decision tree classification [39] and support vector regression, shown in Fig. 5.1. The left part of

Fig. 5.1 illustrates how to generate the clusters and classification rules, and the right part shows how to apply these clusters and rules to forecast a future load. The general idea of the proposed method is introduced as follows.

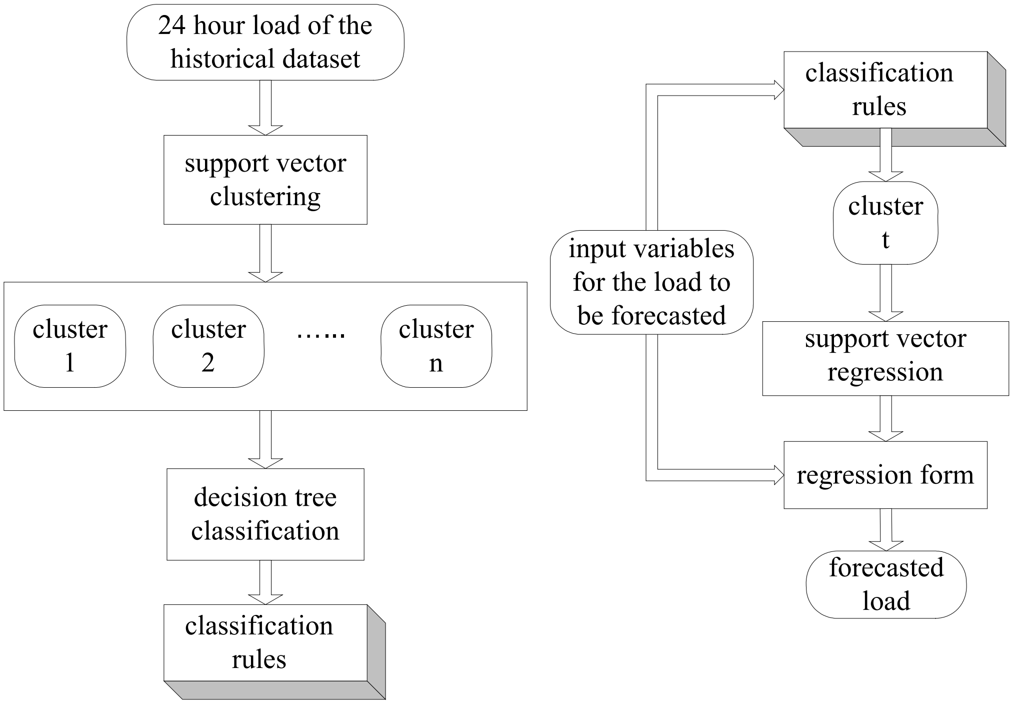

24 hour load of the historical dataset
support vector clustering
cluster 1
cluster 2
……
cluster n
decision tree classification
classification rules
input variables for the load to be forecasted
classification rules
cluster t
support vector regression
regression form
forecasted load

Fig. 5.1 Block diagram of the proposed method

Firstly the daily load curves in the historical database are divided into several clusters using the algorithm of support vector clustering. The data in every cluster have more similarity than in the original dataset. Secondly the decision tree classification algorithm generates the rules to map the input variables (e.g. weather condition, day rank of the week etc.) to their corresponding cluster. To predict a future day load, its input variables are examined according to these rules to match the associated cluster. Then support vector regression is executed applying the historical data in this cluster. The regression form that shows the relationship between the input variables and the output load is achieved. Now the SVR input variables are utilized, namely they are applied to the regression form to get the expected load value.

In the following subsections, the three modules of the proposed hybrid methods will be explained in more detail.

5.3.1 Clustering of historical load

In the clustering procedure, a vector of several continuous load sample points of every historical day is regarded as a training object. Suppose there are altogether d days in the historical database, and L_{ij} is the i^{th} real load of the j^{th} day, $i = 1, 2, ... n$, $j = 1, 2, ... d$; n is the number of sample load points in a day. If the target to be forecasted is the p^{th} load of day d + 1, for a certain j, elements

$$\{L_{ij} \mid i = p-2, p-1, p, p+1, p+2\}$$

are regarded as the input variables for the j^{th} day. Here five points are employed instead of only the p^{th} load, because the similar loads not only have similar values on the p^{th} point, but also have similar neighbours. In other words, the similar loads have the similar changing trend in the load curve. Support vector clustering described in section 2 is applied to these data to get the clusters. But too long an interval is also not employed (for example, 24 hours as described in [41]) because similar loads do not need to be associated with a very long similar load curve. How to select the points of the interval depends on experience. After some experiments, it is found that around one hour (the span of five points) is quite satisfactory.

The concepts of repetitious clustering and overlapping clusters are presented to get better prediction precision. The effectiveness of the proposed method is demonstrated through calculation of forecasted data error.

5.3.2 Repetitious clustering

With power load data, the tiny cluster problem is encountered, which is not mentioned in other references. That means, in the result of support vector clustering, there are too many clusters that include few points. Moreover, there might be many outliers in the clustering result. In this thesis both kinds of points are referred to as “isolated points”. This is a drawback for the later training of these samples, because with insufficient number of samples, no training algorithm can get satisfactory results. To solve this, this thesis presents the repetitious clustering method. Instead of clustering only once, it is done in an iterative way. Because too few samples in one cluster would affect the training and prediction precision, the lower limitation of the point number for clusters is set up. However if the point number for a cluster is too large, the point number for the other clusters might be too small. So the upper limitation of the point number for clusters is also set up. Each time the samples are clustered, the resultant clusters are dealt with.

Clusters with proper point number are reserved as qualified ones, and the remaining data are clustered again. If, as a result of the clustering the point numbers of all the clusters are too large or too small, parameter γ of kernel function

$$K(\mathbf{x}_1, \mathbf{x}_2) = e^{-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|^2}$$

can be increased or decreased to get a cluster with reasonable number of members [55]. This repeats until every datum is in a proper cluster.

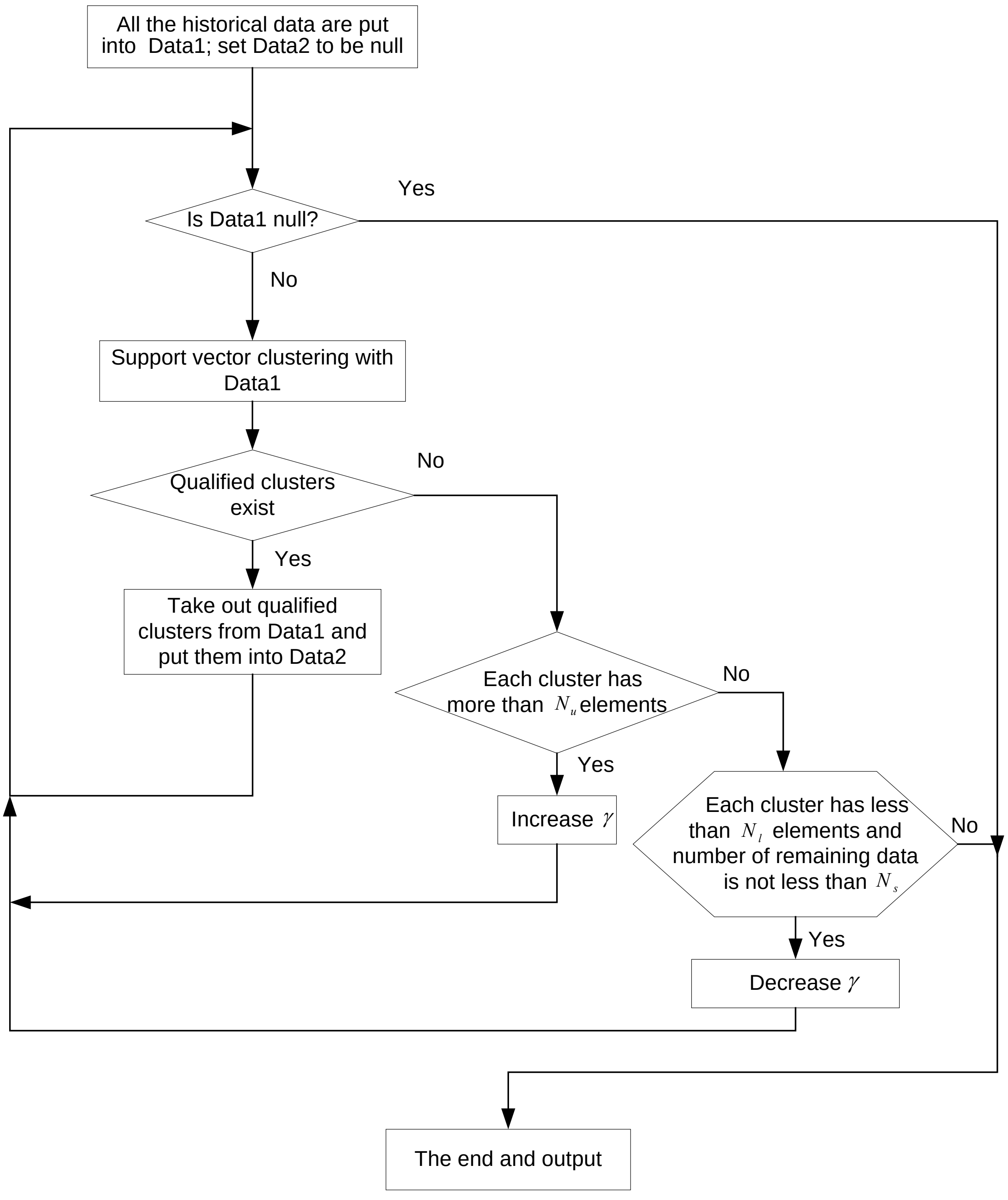

All the historical data are put into Data1; set Data2 to be null
Is Data1 null?
Yes
No
Support vector clustering with Data1
Qualified clusters exist
No
Yes
Take out qualified clusters from Data1 and put them into Data2
Each cluster has more than N_u elements
No
Yes
Increase γ
Each cluster has less than N_l elements and number of remaining data is not less than N_s
No
Yes
Decrease γ
The end and output

Fig. 5.2 Flowchart of repetitious clustering

Although the purpose of repetitious clustering is to avoid isolated points, it is very natural that for holidays, special events or abnormal weather, the corresponding load curve is abnormal and a cluster with many members that contains it cannot be found. Here the upper limitation of the number of isolated points is set up, and repetitious clustering comes to an end when the number of the rest data outside any cluster is below this value. These data are regarded as acceptable isolated points. Actually this is another advantage of the support vector clustering method: being able to find the real isolated points. Instead of using training methods, the prediction of these data should employ some expert knowledge, which will be another subject of our work in chapter 6. The prediction method presented in this chapter is not suitable for the isolated points.

Fig. 5.2 is the flowchart of repetitious clustering. Its related term definition and variable explanation are given in Tab. 5.1.

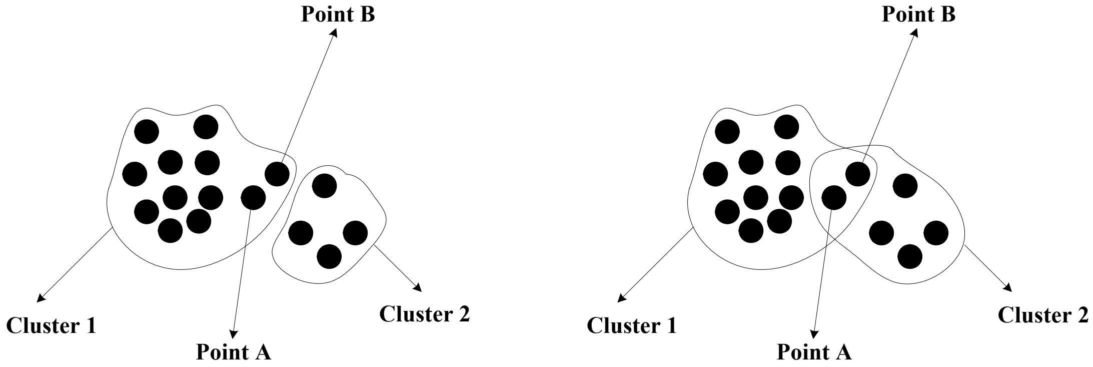
Tab. 5.1 Term definition and variable explanation of repetitious clustering

$Data1$	The dataset to store all the data to be clustered
$Data2$	The dataset to store the feasible clusters
N_u	Upper limitation of the member number in a cluster
N_l	Lower limitation of the member number in a cluster
Tiny cluster	Cluster whose member number is under N_l
Remaining data	The data in $Data1$
N_s	Upper limitation of the isolated points
Qualified cluster	Cluster whose member number is in the rang of $[N_l, N_u]$

5.3.3 Slight revision of the algorithm to allow overlapping clusters

With the repetitious clustering method, the load data are divided into several qualified clusters. After a lot of simulation, it is found that the application of overlapping clusters is better than completely separate clusters. This can be illustrated in Fig. 5.3. Fig. 5.3(a) and (b) show several points in a 2-D area. In Fig. 5.3(a), firstly the support vector clustering is done with all the points in the input space and cluster 1 is formed. Then it is done again with the remaining points and cluster 2 is formed. Points A and point B are classified in cluster 1. But it can be seen from the figure that points A and B also have some property of cluster 2. If the points are clustered in another way shown in (b), points A and B are in the overlapping part of clusters 1 and 2. Although Fig. 5.3 only shows the overlapping of two clusters for simplicity, the method can be generalized to more clusters. In other words, two or more clusters can share some same points.

The employment of overlapping clusters is quite useful in short-term load forecasting, as can be seen later in the table of forecasting result.



(a) without overlapping (b) with overlapping

Fig. 5.3 Clustering without and with overlapping clusters

To revise the algorithm presented in last subsection, some concepts are introduced as follows:

The distance $R(\mathbf{x})$ from the map of point $\mathbf{x}$ to a cluster center in the high dimensional feature space is defined by:

$$R^2(\mathbf{x}) = \|\phi(\mathbf{x}) - \boldsymbol{\alpha}\|^2 = k(\mathbf{x},\mathbf{x}) - 2\sum_j \alpha_j k(\mathbf{x}_j,\mathbf{x}) + \sum_{i,j} \alpha_i \alpha_j k(\mathbf{x}_i,\mathbf{x}_j) \tag{5.13}$$

Cluster sphere: when a cluster has been formed, support vector clustering can be executed again with the data inside the cluster in the input space, and the corresponding sphere in the high dimensional feature space is the cluster sphere [55].

Inner sphere: suppose for a cluster m the radius of its cluster sphere O_1 in the high dimensional feature space is R, inner sphere O_2 is a smaller sphere which has the same center as O_1, but its radius is less than R. Points inside O_2 are thought to only belong to cluster m. Points inside O_1 but out of O_2 are thought to not only belong to m, but also possibly belong to other clusters in the dataset.

In order to get the overlapping clusters, part of the repetitious clustering algorithm should be slightly revised. If a cluster whose member number lies in the area $[N_l, N_u]$ exists, it is regarded as a qualified cluster. The data corresponding to this cluster are put in $Data2$, and the distance between each point in the cluster and the cluster center, namely $R(\mathbf{x})$ in Eq. (5.13), is calculated. If it is less than a predefined inner sphere radius, it is taken out from $Data1$; otherwise, it is left in *Data*1 to take part in the next clustering. Some points might be left in *Data*1 for more than one time, which proves that the number of overlapping clusters can be greater than 2.

5.4 Regression of the Clustered Data

5.4.1 Decision tree application

After clustering, there are several calculated clusters. In the decision tree classification period, each sample takes the cluster ID it belongs to as its classification property according to the support vector clustering result. Because the decision tree method requires that every sample should have only one unique classification property, for the sample in overlapping clusters, it must be decided which of these clusters should be regarded as its cluster property. This thesis presents the method of relative distance calculation to decide it. For a point $\mathbf{x}$ which belongs to cluster m, if the radius of the cluster sphere for cluster m is r_m, and the distance between $\mathbf{x}$ and the cluster centre is l, then the relative distance of the point $\mathbf{x}$ to the cluster sphere centre is defined as:

$$l' = l / r_m \tag{5.14}$$

Suppose $\mathbf{x}$ is located in the intersection of several clusters, the cluster corresponding to the smallest relative distance will be selected as the winning one. It is regarded as the classification property of point $\mathbf{x}$.

For the load to be forecasted the grown decision tree is used to find the cluster it belongs to. The output variable is the cluster ID number. The following variables are chosen as the input variables of the decision tree.

TH: the highest temperature of the sample day

TL: the lowest temperature of the sample day

HU: the average humidity of the sample day

WE: Weekend property: if the sample day is Saturday or Sunday, the corresponding value is true; otherwise false

Monday, Tuesday, Thursday, Wednesday, Friday, Saturday, and Sunday: 7 weekday properties indicating day rank in a week. Note that every day can only belong to one weekday rank, so if one of these seven properties for a certain day is “true”, then the others must have the “false” value.

TH and *TL* are continuous numerical variables. The other variables have discrete property, and there are only two possible values for them: true and false.

5.4.2 Support vector regression for the clustered data

When the decision tree is built up, the input variables of the day whose load is to be forecasted are employed to determine which cluster it belongs to. With the advantages of structural risk, simple mathematical models and short training time, SVR is used to train all the data in this cluster and give the predicted result.

Suppose the p^{th} load of the day $d + 1$ is to be forecasted, the following variables are selected as the input variables of SVR: forecasted highest temperature *TH*, forecasted lowest temperature *TL*, the p^{th} load in the past 2 days, and of the last week, the corresponding highest temperature difference and the corresponding lowest temperature difference:

$$L_{p,d}, L_{p,d-1}, L_{p,d-6}, DTH_d, DTH_{d-1}, DTH_{d-6}, DTL_d, DTL_{d-1}, DTL_{d-6}$$

A linear function is selected as a kernel. This is also an experience---it is found out that in the clustered data, the input variable and the output variable has an approximate linear relationship.

5.5 Calculation Results

5.5.1 Conditions

1) The proposed method is applied to the Shanghai Power Grid real load data of 24 hour load forecasting. The goal is to predict 96 points load in every day in 2002.

2) Suppose day ID of the target load is t, the time point of the target load is p, then days with ID from $t-730$ to $t-1$ are regarded as 730 historical data objects. Every object has 5 components, namely the 5 real load sampling values:

$$\{L_{ij} \mid i = p-2, p-1, p, p+1, p+2\}, j = t-730, t-729, ...t-1$$

3) In the program some predefined parameters are set as follows: $N_l = 6$, $N_u = 75$, $N_s = 25$.

4) This thesis employs the following different methods for the short-term load forecasting to make a comparison and verify the feasibility of the presented method:

- Method A: The presented method (repetitious clustering + decision tree + SVR), with overlapping clusters, Gaussian kernel function applied
- Method B: support vector clustering + decision tree + SVR, clustering is completed in one time instead of repetitious clustering, Gaussian kernel function applied
- Method C: repetitious clustering + decision tree + SVR, without overlapping clusters, Gaussian kernel function applied
- Method D: K-means clustering + decision tree + SVR
- Method E: using SVR directly without finding clusters, using the data of two months before the date of the load to be forecasted.

5.5.2 Prediction results

Tab. 5.2 displays the clustering results of daily curves of point 56 for 730 days with different kernel methods. The Gaussian kernel clustering divides the curves into 13 clusters. Linear kernel, polynomial kernel and sigmoid kernel can only find one cluster, which doesn't help to classify the data at all. This shows that among the four kernel functions, only Gaussian is able to fulfill the task of clustering.

Tab. 5.2 Clustering results of different kernel functions

Function name	**Gaussian**	**Linear**	**Polynomial**	**Sigmoid**
Cluster numbers	13	1	1	1
Relative outlier number (%)	4.5	0	0	0

In Tab. 5.3, the clustering results of point 56 for 730 days with method A and method B are displayed and compared to show the effectiveness of overlapping. Note that for method A the sum of member numbers in all the clusters is above 100%, due to the permission of overlapping clusters. There are 4.5% isolated points in the result of method A. To go into details it is found out that these sample days include the Spring Festival holidays, National Day holidays, Labour Day holidays and the winter and summer days in which load control was employed due to insufficient electricity supply. These would be treated separately with expert knowledge. The result of method B shows that there are 21.3% outliers and 3.7% points in tiny clusters. Not only the isolated points of method A are included, many relatively normal days are also regarded as isolated points in the result. For example, July 3rd is regarded as an outlier in method B and cannot be trained. But in the result of method A it is in cluster 10, and the MAPE prediction result is 3.8%, which is quite acceptable.

Tab. 5.3 Clustering results of methods A and B

Method A	**Cluster ID**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	Relative member number (%)	17.4	16.0	11.9	3.1	12.5	11.4	13.7
	Cluster ID	**8**	**9**	**10**	**11**	**12**	**13**	**Isolated points**
	Relative member number (%)	9.0	5.8	2.4	4.5	5.5	3.1	4.5
Method B	**Cluster ID**	**1**	**2**	**3**	**4 to 33**			**Outliers**
	Relative member number (%)	60.4	11.6	3.0	each cluster < 0.5 Altogether 3.7			21.3

For every cluster, a decision tree algorithm is employed in the presented method, and every cluster corresponds to a classification rule. As examples, the classification rule of cluster 1 with method A in Tab. 5.3 is written as follows:

$(WE = \text{false AND } Monday = \text{false AND } TL \in (11,15])$ OR

$(WE = \text{false AND } Monday = \text{false AND } TH \in (18,25]$ OR

$(WE = \text{false AND } Monday = \text{false AND } TH \in (25,27] \text{ AND } TL \in (15,18])$ OR

$(WE = \text{false AND } Monday = \text{false AND } TH \in (14,18] \text{ AND } TL \in (7,11])$

Fig. 5.4 shows the prediction MAPE of every month in 2002 with different methods. Tab. 5.4 shows the prediction results of the load in a randomly selected number of 10 days in 2002. The MAPE of the prediction results for everyday are shown in the table. This thesis only compares the result of methods A, C, D and E, since there is an unacceptable large number of isolated points in method B.

Tab. 5.4 Prediction MAPE (%) of different methods

Day (Date)	Method A	Method C	Method D	Method E
1(18th January)	3.2	3.9	3.7	4.8
2(17th, Februay)	2.8.	3.1	3.2	3.5
3(26th, March)	2.6	2.6	2.1	3.4
4(9th, April)	2.2	2.8	2.5	4.6
5(23rd May)	1.6	2.1	1.9	4.2
6(11th,June)	2.7	3.5	2.5	4.6
7(3rd, July)	3.8	4.1	4.2	5.4
8(14th,August)	3.7	4.3	4.5	5.6
9(23rd, September)	2.5	3.1	2.8	3.5
10(11th, October)	3.3	4.4	4.1	5.6
Average	2.8	3.4	3.2	4.5

From Tab. 5.4 and Fig. 5.4 it can be seen that methods A, C, D are much better than method E. This proves the advantage of clustering. In method E, which is without clustering, all the data two months ahead are trained together. The diversity of these data acts on the training parameters and finally affects the precision. The average error of the ten days predicted by methods A, C, D are respectively 2.8%, 3.4% and 3.2%. It is obvious that the presented method has the best precision.

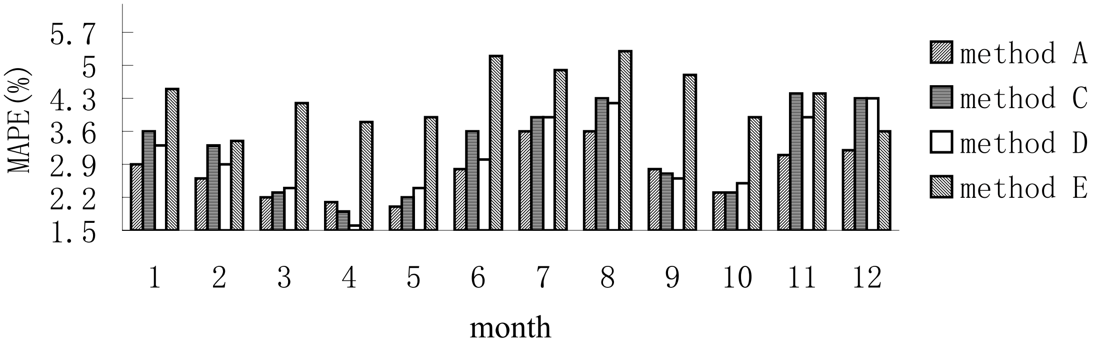


Fig. 5.4 Prediction MAPE of every month in 2002 with different method

To show the comparison of the SVM method proposed in this chapter and the RT method proposed in the last chapter, Fig. 5.5 shows the monthly MAPE result of the two approaches. RT is better than SVM for half of 12 MAPE results, and SVM better than RT for the other half. In chapter 6 the combination of different methods will be covered.

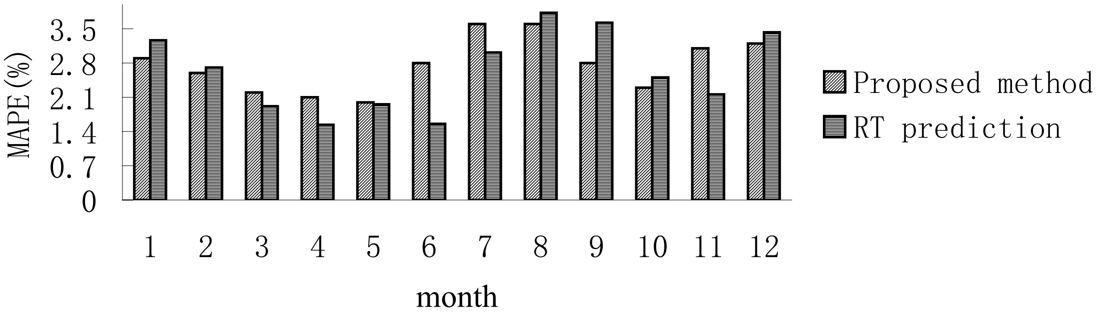


Fig. 5.5 SVM and RT prediction comparison MAPE of every month in 2002

6 Integrative Algorithm

6.1 Load Forecasting for Holiday and Anomalous Days

The basis of load forecasting is to find historical points that are similar to the target load and do the training employing these points. Therefore having enough historical training samples is a precondition for a good forecasting result. For normal days this is not a difficulty, but for anomalous days it is much more difficult to find enough similar sample points in the historical database.

Anomalous days include public holidays, consecutive holidays, days preceding and following holidays, days with rare weather or special events. Every year these kinds of anomalous days appear only one or a few times, therefore a large enough training set can not be obtained within one or two years. Although the sample number can be increased by including more historical days (e.g. five years' historical data can be applied rather than only one or two years), it would lead to decentralization of training samples: because of the change of economic situation and, in turn, the corresponding change of consuming electricity, two different days with similar weather and holiday conditions may have totally different load curves if the time interval between them is too long.

[10] classifies the anomalous days into different types. Based on this theory, this research improves it by replacing the "anomalous day" with "anomalous period", since the anomalous load curves do not always appear in the unit of one day. For example, January 1st is a public holiday, but from the previous evening people begin to celebrate it, so in this research the period from the 19:00 of December 31st to 24:00 of the next day is regarded as a holiday period.

In the proposed method several kinds of anomalous periods are defined and every period is associated with an anomalous period index number, shown in Tab. 6.1. There are altogether 11 kinds of anomalous periods for the Shanghai Power Grid load. Periods with the same anomalous period index number have a similar load behavior. Deciding how many indices there should be and how to define the period of each index is more or less an expert system domain problem, since it is dependent on peoples' experience and the acquaintance of load curves.

Tab. 6.1 Anomalous periods for Shanghai Power Grid load

Anomalous period index	**Description**	**Corresponding holiday name**
1	From 19:00 Dec 31st to 24:00 Jan 1st	New Year's day
2	From 12:00 lunar new year eve to 2nd day of lunar year	Lunar new year holidays
3	3rd or 4th day of lunar year *	
4	0:00 5th of the lunar year to 24:00 of 8th lunar year	
5	15th day of lunar year	Festival of lanterns
6	From 12:00 30th Apr to 24:00 2nd May	May golden week
7	3rd May or 4th May *	
8	0:00 5th May to 12:00 of 8th May	
9	From 12:00 30th Sep to 24:00 2nd Oct	October golden week
10	3rd May or 4th Oct *	
11	0:00 5th Oct to 12:00 of 8th Oct	

* "or" is used because either day corresponds to the same index

Similar to the method of regression tree proposed in Chap. 4, in this research "weekday-holiday", "Saturday-holiday", "Sunday-holiday" and "holiday-holiday" increment trees are generated in addition to pure holiday non-increment trees. The increment trees usually have more training samples than non-increment trees. Besides, the increment trees usually lead to the target leaf node with higher sample number and lower estimated error (dispersion) than non-increment trees for holiday load prediction. In other words the rule generated by increment RT is more convincing than the non-increment RT. This is especially true when the historical database span is largely enhanced.

The load curve of holidays is affected not only by the common factors of load such as climate and recent load, but also the holiday property. To decouple the problem, another way of anomalous day load forecasting is presented.

Suppose the p^{th} load of target anomalous day with the anomalous period index l is to be forecasted. First find in the historical database all the p^{th} load historical period with the anomalous period index l. These real loads are named as $RL(1)$, $RL(2)$,…$RL(n)$.

Then suppose these periods are not anomalous days but common weekdays. For example, all of them are supposed to be common Tuesday. For every p^{th} load of the "supposed Tuesday" the forecasting methods proposed in the preceding chapters are employed to get the imaginary loads: $IL(1), IL(2),\ldots,IL(n)$.

The imaginary loads are mapped to the corresponding real loads.

$$IL(1)\rightarrow RL(1),\ IL(2)\rightarrow RL(2),\ldots,\ IL(n)\rightarrow RL(n) \tag{6.1}$$

They are regarded as the input variables and the corresponding output variables of a dataset. To find the input variables of the target load, the target day is also supposed to be an ordinary Tuesday. Use the normal prediction method presented in the former chapters to get the imaginary load $IL(n + 1)$. Train the input and output variables in Eq. (6.1) with the support vector machine, and use $IL(n + 1)$ as the target input variable. A predicted $RL(n + 1)$ is generated and regarded as the forecasting result of the target load.

6.2 Integration of SVM, RT and Other Traditional Algorithms

6.2.1 Integration of SVM and RT

As mentioned in Chap. 5, SVM is employed as a tool for STLF. Taking advantage of structural risk, simple mathematical models which can be solved easily, the application of SVM to STLF has shown good results with small errors and high training speed. But unlike RT, it is not able to find the suitable input variables and partition the input variable space. RT can do this well. The disadvantage of RT is that, for every predictor value that falls into the leaf node, it can only use the samples’ average dependent value as its output value. If the dispersion in the node is large, this can cause a large error. If the sample number in leaf node is small, the result lacks statistical significance. If the target point is a desert border point of the leaf node, the leaf output might be quite different from the real target load. Although DEV_{max} and N_{min} as well as desert border detection can be employed in RT to prevent large error as mentioned before, sometimes, especially when dealing with the holidays, days with rare weather or other anomalous days, maybe too few qualified (or even no) leaf nodes can be reached. To solve this problem, this thesis presents the combined RT and SVM method (RTSVM) to take use of their advantages for better results.

When the regression tree algorithm is used to forecast a future load, a leaf node can be matched. According to the principle of determining whether a node is a leaf node, the leaf node should satisfy one of the following two conditions:

- $DEV < DEV_{max}$, $N \geq N_{min}$

- $N < N_{min}$

Condition 1 is very ideal because it shows a large number of similar samples with very low dispersion. The target load that falls in this kind of leaf node can take the leaf node output value as its forecasted value.

If the target load falls into the second kind of leaf node, it is not so reliable to regard the output value of the leaf node as the forecasted load, no matter whether the dispersion is greater or less than DEV_{max}, since the statistical significance of the samples is not obvious. In this thesis, RTSVM method is presented to deal with this kind of node, which is described as follows.

For a load to be forecasted, generate the regression tree as described earlier. Suppose the forecasted load falls in the leaf node ND_0 that satisfies condition 2. Backdate toward the root node. Suppose node ND_2 is the parent of ND_0, ND_1 is ND_2's parent, and ND_3 is ND_2's sibling, shown in Fig. 6.1. Calculate separately the dispersion of ND_1, ND_2 and ND_3: DEV_1, DEV_2 and DEV_3. Define split dispersion ratio (SDR) as the DEV of a parent node divided by the average DEV of its two siblings

$$SDR = (DEV_2 + DEV_3) / DEV_1 / 2 \tag{6.2}$$

Set ρ as the maximum limitation of split dispersion ratio. In this thesis it is set:

$$\rho = 0.25 \tag{6.3}$$

If the $SDR < \rho$, this implies that the split of ND_1 is efficient in partitioning the subset ND_1 into two distant subsets: ND_2 and ND_3. Therefore it might not be appropriate to train the data of ND_2 and ND_3 together. Otherwise, it implies that the data in ND_2 and ND_3 have similarity, so it is proper to train them together.

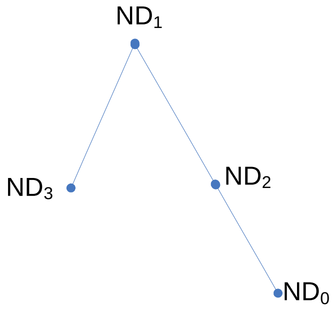


Fig. 6.1 Subtree with node ND_0, ND_1, ND_2 and ND_3

Here the notation *T* is used to represent the node that is regarded as a complete cluster, the samples of which will be trained in SVM. Therefore, when $SDR < \rho$, let $T = ND_2$. If not, backdate toward the root node. Do this repeatedly until a good node *T* (or the root node) is reached. Fig. 6.2 shows the tree backdating flowchart to find the appropriate node samples for further SVM.

Since *T* is not a leaf node, it must have some splits. Collect the split of *T* and all the splits of its offspring and regard all the related split variables as the important variables. For example, in Fig. 4.1, if *a*, *b, c* or *d* is regarded as the complete cluster node, the important variables are x_1 and x_2; while if *f* or *h* is regarded as the complete cluster node, the important variable is only x_2 because no further split is related to x_1. The split input variable of a node is thought to be the influential variable to the dataset corresponding this node. Therefore the important variables are regarded as the input variable components of SVM. This is an effective way to reduce the input variable number.

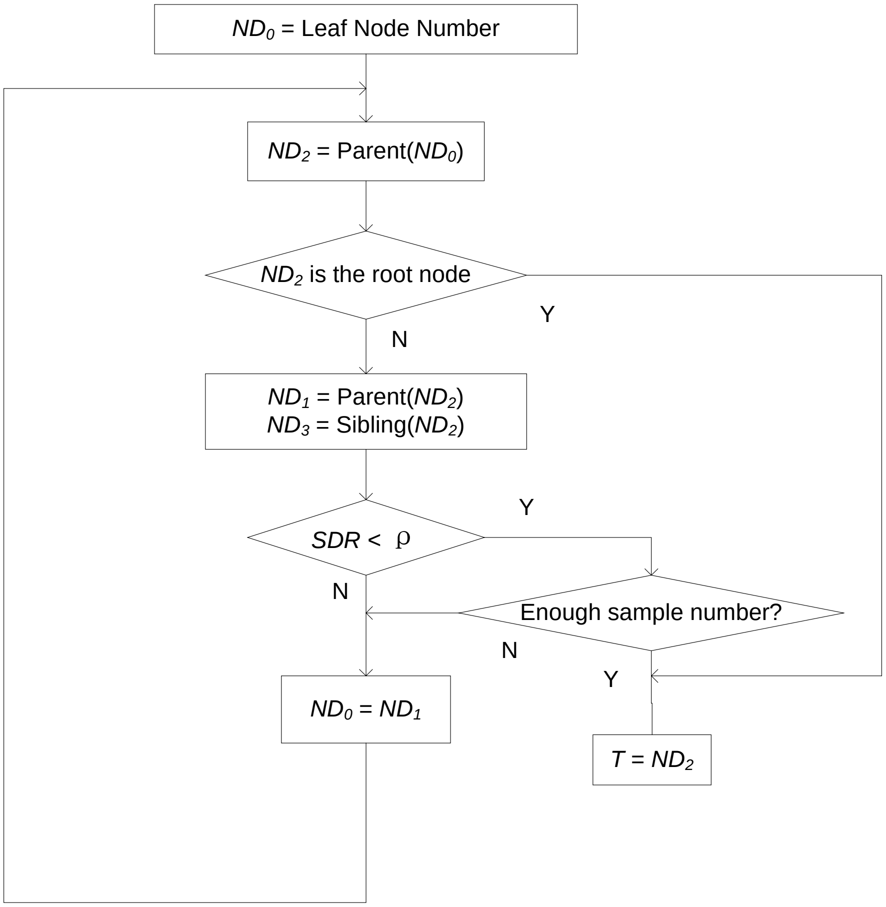


Fig. 6.2 Tree backdating flowchart for finding the cluster node for further SVM

Train the samples in node *T* with SVM. The process of RTSVM is shown in Fig. 6.3. But here enough sample numbers in the SVM training must be assured. Here “enough” means at least the sample number should be no less than the input variable [51]. If this condition of node *T* is not satisfied, backdate to find a new appropriate node. Apply the independent variables (only the important variables derived from the tree) to the final regression form and the forecasted load can be obtained.

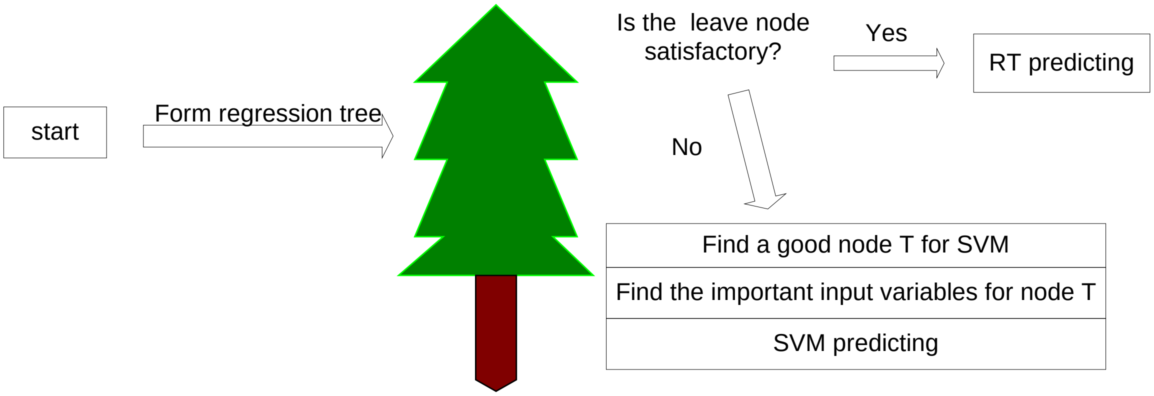

start
Form regression tree
Is the leave node satisfactory?
Yes
RT predicting
No
Find a good node T for SVM
Find the important input variables for node T
SVM predicting

Fig. 6.3 Process of RTSVM prediction

6.2.2 Extended dispersion calculation in RTSVM forecasted result

The weighted average method in RT, which integrates the different forecasted results, has been introduced in Chap. 4. According to Eq. (4.4) - (4.7), the weight of every forecasted result of RT can be calculated. This method can also be extended to the integration of the RTSVM method. The problem is, the RTSVM forecasted result doesn’t correspond to a dispersion value. In this thesis a way of measuring the accuracy of RTSVM is presented.

Suppose *T* is the RT node in which SVM has been carried out, and there are *m* samples in *T*. Suppose the input variables of these *m* samples are

$$[x_1, x_2, \ldots x_m]$$

and the corresponding output invariables are

$$[y_1, y_2, \ldots y_m]$$

By training these samples in SVM the regression form indicating the input-output relationship $y = f(x)$ is obtained. Now apply $[x_1, x_2, \ldots x_m]$ to the regression form $y = f(x)$ and the forecasted values $[y_1', y_2', \ldots y_m']$ are calculated. To compare the real output variable values and the forecasted ones, define the extended dispersion of node *T* as

$$DEV(T) = (\sum_{i=1}^{m} \left| y_i^{'} - y_i \right|) / m \tag{6.4}$$

Apply the weighted average method to both the RT and RTSVM results (Eq. (4.4) - (4.8)). In the equations *DEV* is replaced by an extended dispersion for the RTSVM result. Similarly the weighted average method is applied and the RTSVM and TEI (total error indicator) can also be obtained (Eq. (4.9)). They serve as a final result of the RTSVM forecasting.

6.2.3 Integration of different algorithms

Different methods outperform in different conditions. For example, in April, the difference of different day load curves is not obvious, even if the highest or lowest temperature changes by 5 ℃. Linear regression or non-increment RT might perform well in such a situation. But in summer, only a 1℃ change of the highest temperature might cause a great change in the load curve, and the load value is a nonlinear response of the temperature. In this case the increment RT method might outperform. In this subsection the integration method presented in 6.2.2 is generalized to employ more single load forecasting algorithms and take more advantage of the more appropriate ones.

Suppose there are n forecasting methods in a forecasting system: $M_1, M_2, \ldots, M_n$. Every method has its own advantages and disadvantages. Now these methods are used to predict the historical loads of the past s days, "as if" the real data are unknown. To distinguish this kind of prediction from the normal future load forecasting, it is named "past forecasting".

Suppose the historical days are day 1,day 2, …day t, and the target is the p^{th} load of day $t + 1$. Past forecasting is done to "forecast" the p^{th} load for s days before the target load. In other words, from day t back to day ($t - s$), all the p^{th} loads of the load curve are predicted by past forecasting. Due to the limitation caused by different algorithms, sometimes less than s results might be obtained for some algorithms. For example, if an algorithm is specially designed for weekend load curve forecasting, normally there are much less than s forecasting results since weekends occupy only around 2/7 of the total days.

Suppose N_i past forecasting results are obtained for the method M_i ($N_i \le s$, which means some days might not satisfy the forecasting condition). In addition, N_i past forecasting absolute percentage errors are achieved by comparing the forecasting result with the real data: E_1, E_2, …,$E_{Ni.}$ The average error

$$AVE_i = (E_1 + E_2 + \ldots + E_{Ni}) / N_i \tag{6.5}$$

is regarded as an approximate measurement of the possible error of the target forecasting result for method M_i. Like dealing with the RT in Chap. 4, the upper limitation of AVE_i (AVE_{max}) and the lower limitation of N_i (N_{min}) are set to filter out the results with large error or small sample number. Excluding the methods with AVE_i or N_i breaking bounding, k results, M_{q1},M_{q2},…M_{qk}, are obtained. The following calculation is done to combine the k prediction results and calculate the total error indicator.

$$CONF_i = 1/(AVE_{qi}), \quad i = 1,...k \tag{6.6}$$

$$TOTAL\_CONF = \sum_{i=1}^{k} CONF_i \tag{6.7}$$

$$W_i = CONF_i / TOTAL\_CONF \qquad i = 1,...k \tag{6.8}$$

$$L = \sum_{i=1}^{k} W_i L_{qi} \tag{6.9}$$

$$TEI = \sum_{\mathrm{i}=1}^{k} W_i AVE_{qi} \tag{6.10}$$

$CONF_i$ is the confidence of the qi^{th} method. $TOTAL\_CONF$ is the sum of all the confidence values. W_i is regarded as the weight of the qi^{th} forecasting result L_{qi}, and TEI is the total error indicator of the final integrative result. Fig. 6.4 shows the process of integration of different algorithms.

The integrative method integrates several different methods. Different methods have different forecasting error for the same target load. As an average method, the integrative result is a value between the maximal error prediction and the zero error prediction. Although it might be worse than the prediction result of the single prediction that leads to the minimal error for any single point, for the future forecasting people can never know in advance which one is with the minimal error. Moreover, an individual algorithm, which performs better than the other for one point, might show poor performance for another point. With the average property of the integrative method, it is more effective to decrease the maximal error of the daily load curve than the single methods. For load forecasting the maximal error of the forecasted daily load curve is a very important measurement of the forecasting result, since larger maximal error leads to improper unit commitment and, in turn, causes a higher cost of real time dispatch. Therefore the integrative method is very effective. On the other hand, since the algorithm applies weights in averaging all the results instead of a simple average, it pays more attention to the potentially better results and, in turn, usually leads to better prediction precision.

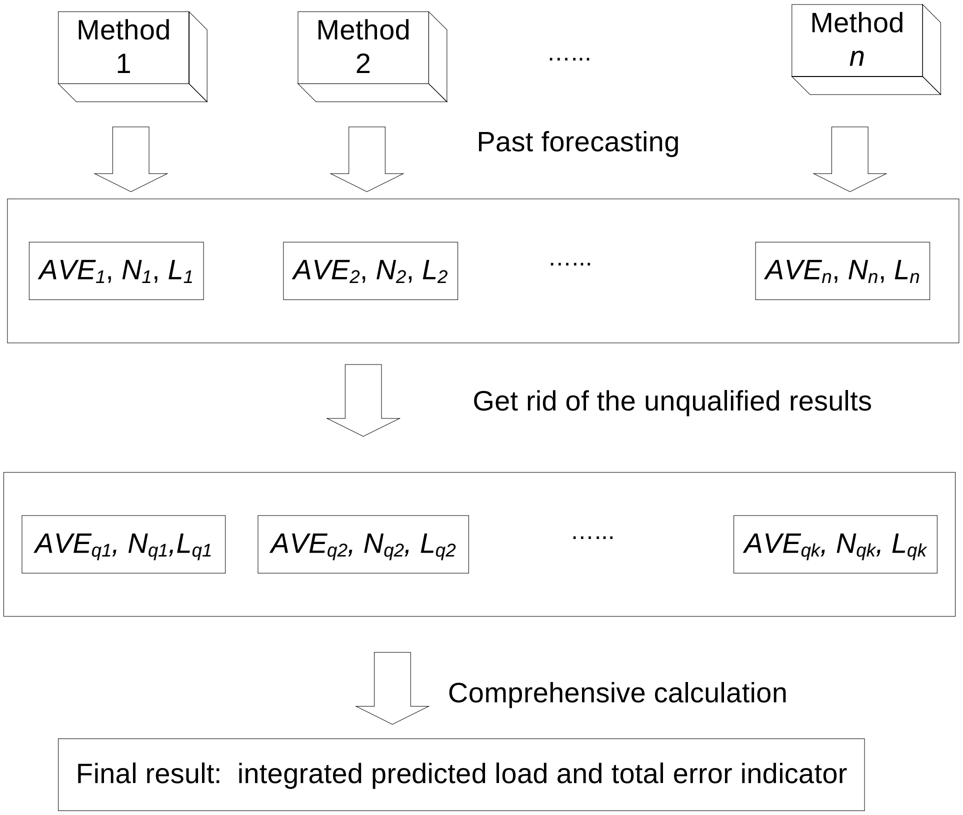


Fig. 6.4 The integration process of different algorithms

6.2.4 Smoothing of the forecasted load curve

For STLF input variable selection there is a contradiction. If the p^{th} load of the target day is to be forecasted, although the samples near the p^{th} points are influential on the output load, too many input variables might interfere with the forecasting result precision. In the previously mentioned methods of short-term load forecasting, decoupling was utilized in treating the input and output variables. For example, if the target load is the p^{th} point of a day, in selecting the input and output variables only the p^{th} point of the historical days are taken into consideration. The purpose is to decrease the scale of the problem. The deficiency of not using the historical load of p^{th} points' neighbours can be compensated by the smoothing method described as follows.

In Chap. 3 the second order difference was employed to detect the outliers, and weighted least square quadratic fitting was employed to smooth the real load curve. Here, to improve the STLF result, these two tools are also applied to the forecasted load curve.

Suppose the forecasted time sequence load for a future day is L_1, L_2,…,L_{96}. Employ the second order difference to detect if there are sudden change points in the curve. They are considered to be bad forecasting results and are replaced by a least square quadratic regression. Then the forecasted load curve is smoothed by weighted least square quadratic fitting. The application of "smoothing" can take the effect of historical points near the p^{th} point. In other words, without increasing the complexity of the problem, the application of smoothing can naturally "add" some "input variable effect" of the nearby loads of the p^{th} point in the historical database.

6.3 Generalized Programming System Design

Different power systems have different load behaviours. In RT forecasting, a way of generating "weekday-Saturday", "weekday-Sunday" tree has been proposed in Chap. 4. This implies that the weekdays, Saturdays and Sundays are in different clusters. The clusters are obtained from experience. But the clusters are not always in the unit of a day, and they do not always obey the division of weekday, Saturday and Sunday. For example, it is found that in terms of wee hours, Tuesday, Wednesday, Thursday, Friday and Saturday have more similarity than from Monday to Friday in many regions. Therefore, the clustering of the time period shouldn't be fixed in the programs for better generalization. In addition, many other parameters, such as the selection of calculation methods and the maximal acceptable estimated error, also differ from one system to another. This inspires the authors to devise a tabular data format, for the users to decide the calculation mode.

In this research the three-table frame and the related programming modules are designed for different users to input the system load properties and the calculation requirement. With these three tables, users can apply the proposed integrative algorithm easily without any modification of the programs themselves. The three tables, which are a cluster description table, a time schedule table and a method description table, are explained respectively as follows.

The cluster description table defines the load curve clusters of the system, shown in Tab. 6.2. Every column represents the property of the investigated load, and every row corresponds to one cluster. If the element of the x^{th} row and the y^{th} column has a "1" value, it means the y^{th} property for x^{th} cluster has a true value. "0" means false value. "-1" means the y^{th} property doesn't affect the decision of the x^{th} cluster. Moreover, there are also some numerical value columns. For example, MinTime is the lower limitation of the time point of the investigated load and MaxTime is its upper limitation. All the integration of these columns are in "and" relationship.

For example, cluster 3 has the rule “week rank is not Sunday and week rank is not Monday and holiday is false and time point ≥ 1 and time point ≤ 20”. From this table any load at any time belongs to its cluster(s). The design of this table doesn’t require any point to be in a unique cluster. In other words, one point can be in more than one cluster. But it is necessary that every point belong to at least one cluster.

Tab. 6.2 Example of the cluster description table

Cluster ID	**Sun**	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Holiday**	**MinTime**	**MaxTime**
Cluster 1	1	0	0	0	0	0	0	0	1	20
Cluster 2	0	1	0	0	0	0	0	0	1	20
Cluster 3	0	0	-1	-1	-1	-1	-1	0	1	20
Cluster 4	0	-1	-1	-1	-1	-1	0	0	21	96
Cluster 5	0	0	0	0	0	0	1	0	21	96
Cluster 6	1	0	0	0	0	0	0	0	1	96
……										

Corresponding to this table, a function is devised to find the cluster ID of any load. Given the date, weather, holiday, time point and other related information of the object, the function returns the cluster(s) type of the investigated point.

Suppose there are *N* kinds of calculation methods, such as absolute value increment RT, relative value increment RT, support vector machine, RTSVM, linear regression and so on, in the STLF system. As training methods, every one has its own input variables. To make things as flexible and generalized as possible, in this research the input variables are not “burnt” inside the functions. On the contrary, all the possible input variables are listed as the different columns of method description table, shown in Tab. 6.3. Every row corresponds to one method. Column “focus cluster” corresponds to the cluster ID of the target point. In the forecasting system every forecasting method corresponds to a method ID. Column “method” is the method ID. Column “compared cluster” is only valid for the increment algorithms. It defines the cluster with which the focus cluster is compared. If its value is “-1”, it means that this is not an increment algorithm. Columns *SR* to *DTHP* are related to the input values. The value “1” means the corresponding column variable is adapted by the method, and “0” means not. NumMin is the lower limitation of the sample numbers of past forecasting results. MaxAVE is the upper limitation of the average error of past forecasting. If the output result has an average error less than MaxAVE and past forecasting sample number more than NumMin for the training data, the forecasted result is regarded reasonable.

As an example, the detailed explanation of the first row of Tab. 6.3 is given here: for cluster 1, method 1(in this thesis method 1 is defined as non-increment RT) can be applied, and the following input variables are required: *TL*, *THP*, *HU*… The lower limitation of the past forecasting result number is 7. The upper limitation of the past forecasting average error is 2.5%.

For a better generalization of the system, new algorithms can be added to the system. Thus a new function for the new algorithm must be added to the project by the developers, and the total number of method IDs should be increased. Maybe the input variables in the method description table couldn't cover those in the new algorithm, in this case they have to be "fixed" in the program.

Tab. 6.3 Example of the method description table

A*	**B***	**C***	**D***	Input variable selection									**E***	**F***
				SR	*TH*	*TL*	*THP*	*HU*	*DTH*	*DTL*	*DTHP*	…		
1	1	-1	1	0	0	1	1	1	0	0	0	…	7	2.5
2	2	1	1	0	0	1	1	1	0	1	1	…	4	3
3	2	2	1	1	0	1	1	1	0	1	1	…	4	3
4	2	3	1	1	0	1	1	1	0	1	1	…	5	2

***A**: Cluster ID; **B**: Method; **C**: Compared cluster; **D**: Focus cluster; **E**: NumMin; **F**: MaxAVE(%)

Tab. 6.4 Example of the time schedule table

cluster ID	***m***	***n***	***Dd***	TEI_{max}	RN_{min}
1	5	10	5	2	2
2	4	10	5	2.5	2
3	6	10	5	2.5	3
……					

Tab. 6.4 is the time schedule table of the forecasting. Every row corresponds to a cluster. The forecasting is consistent with the principle of “the nearer historical data has higher priority in training”. This means firstly some nearby samples of the target load for forecasting are employed. If the result is not satisfactory, the allowable time period for historical training data is enlarged a bit more. A loop is executed until the satisfactory result is found. The method was introduced in Chap. 4. Tab 6.4 defines the related parameters: m, n, Dd . m is the maximal historical year count, $2n$ is column count (see Fig. 4.6) of one year, and Dd indicates how many days there are in a column. TEI_{max} and RN_{min} indicate the loop ending condition. The former is the upper limitation of TEI and the latter is the lower limitation of the calculation result number. When the calculated TEI and result number are within limit, the calculation is thought to be valid.

If a target load belongs to two or more clusters, corresponding calculations of all the clusters will be carried out and the results will be combined again using the forenamed integrative method. Fig. 6.5 shows the generalized main calculation for STLF.

The proposed method is applied to Shanghai Power Grid and the Germany E.ON. power grid. For the calculation of a new system, the programs don’t need to be modified at all; only the three tables should be refilled. This saves effort and shows good portability. Nevertheless, people have to use the expert knowledge and experience to decide the data in the three tables for a new system. From the experiment it can be found: if parameters from Shanghai are utilized instead of filling new parameters for the E.ON, the STLF result would be very poor.

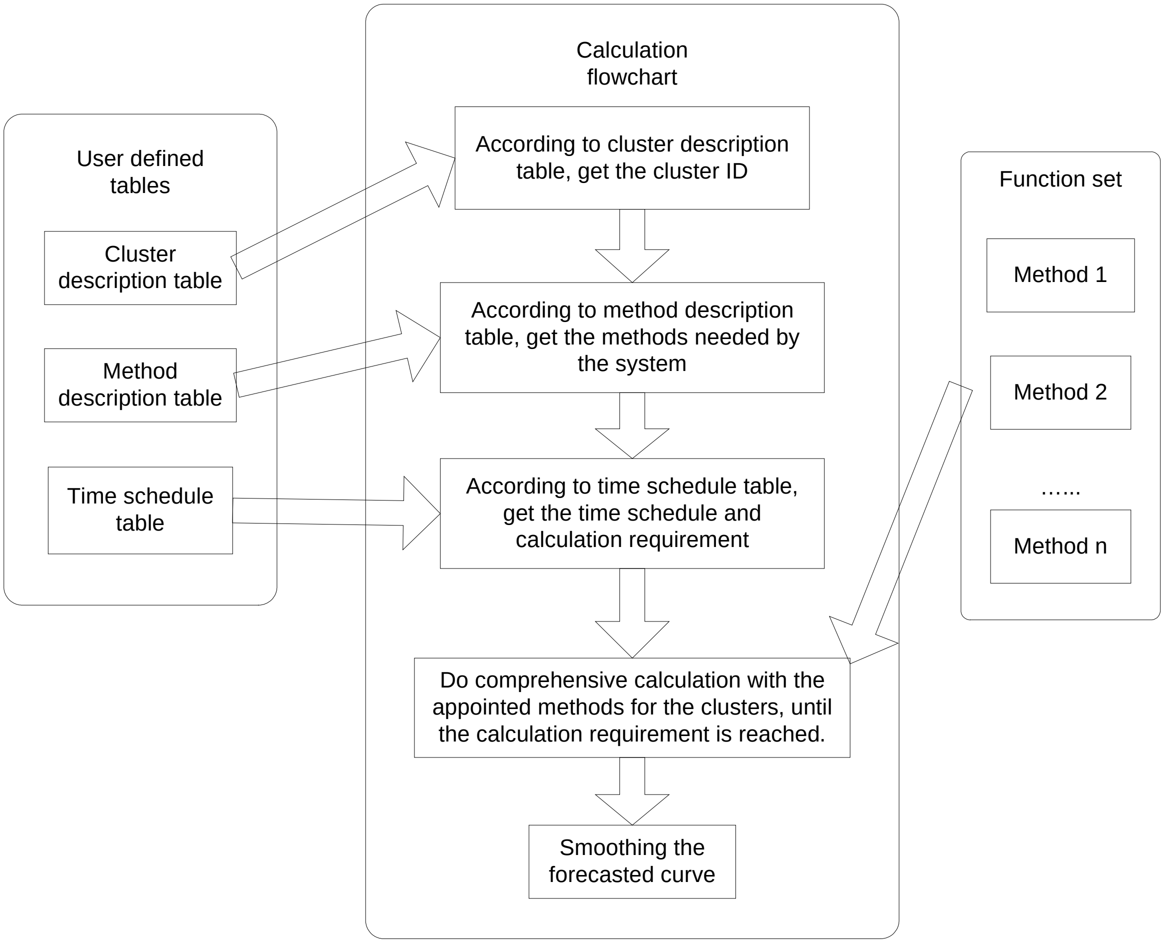

Calculation flowchart
User defined tables
Cluster description table
Method description table
Time schedule table
According to cluster description table, get the cluster ID
According to method description table, get the methods needed by the system
According to time schedule table, get the time schedule and calculation requirement
Do comprehensive calculation with the appointed methods for the clusters, until the calculation requirement is reached.
Smoothing the forecasted curve
Function set
Method 1
Method 2
……
Method n

Fig. 6.5 Schematic diagram of the generalized main calculation

6.4 Case Study

Tab. 6.5 shows the holiday forecasting result of 2002 using the integration of the two proposed holiday forecasting methods. Tab. 6.6 shows the monthly forecasting results for year 2002 with the presented integration method. It combines the results of four methods: RT method presented in Chap. 4, SVM method presented in Chap. 5, RTSVM presented in this chapter, as well as ANN. To show its effectiveness, the results of the other four individual methods are also displayed. Mean absolute percentage error (MAPE) of every month is displayed in the table to compare the results of the different methods. Mean max error is the mean value of daily maximum percentage error (absolute value). From the table it can be seen that RTSVM outperforms RT and SVM methods. The presented method has achieved the best results. From the results it can also be found that the presented method is especially effective in the mean max error calculation.

Tab. 6.5 Holiday forecasting results for 2002.

Anomalous period index	**MAPE(%)**
1	3.59
2	3.52
3	4.18
4	3.25
5	3.01
6	4.31
7	3.27
8	3.06
9	4.67
10	3.58
11	3.64

Tab. 6.6 Monthly forecasting results for the year 2002 with different methods

Month	**MAPE(%)**					**Mean max error(%)**				
	1*	2*	3*	4*	5*	1*	2*	3*	4*	5*
Jan	3.27	2.9	3.22	3.58	3.31	10.75	9.54	9.52	11.16	7.57
Feb	2.71	2.62	2.49	2.49	2.38	9.39	10.29	10.46	12.39	8.79
Mar	1.92	2.21	1.95	2.11	2.07	7.09	6.91	6.84	6.48	5.94
Apr	1.54	2.16	1.6	1.51	1.8	7.47	8.24	7.68	7.97	6.87
May	1.96	2.03	1.76	1.89	1.84	7.28	7.32	6.83	7.69	6.25
Jun	1.56	2.8	1.48	1.9	1.77	8.32	9.64	8.65	10.34	8.51
Jul	3.02	3.66	3.15	3.23	3.16	11.32	9.97	10.09	11.65	8.35
Aug	3.83	3.64	3.43	4.13	3.73	11.53	10.34	10.61	12.88	10.05
Sep	3.63	2.83	3.55	3.48	3.15	8.72	8.56	7.82	8.25	8.66
Dec	2.51	2.38	2.28	2.35	2.47	6.77	6.25	6.56	7.17	6.21
Nov	2.16	3.19	2.18	3.07	2.53	6.96	7.18	7.58	7.48	7.02
Dec	3.43	3.25	3.1	3.87	3.5	10.5	10.08	9.84	12.45	10.1
Average	2.63	2.81	2.52	2.8	2.64	8.84	8.69	8.54	9.66	7.86

* 1-RT 2-SVM 3-RTSVM 4-ANN 5-integrative method

Chap. 3 proposed the bad data detection method, together with the load curve smoothing method. In this thesis all the load forecasting examples utilize the data based on these data treatment methods. To examine the contribution of data smoothing to the prediction accuracy, Tab. 6.7 shows some comparison of applying and not applying smoothing. The research object is the Changzhou power system in China. It displays the STLF results for June 1st to 10th, 2004 with three methods. Method 1 employs bad data detection, fitting and SVM forecasting presented in Chap. 5. Method 2 neglects the fitting module. Method 3 neglects both the detection module and the fitting module, with the original data input to the SVM module directly. The MAPE is used to compare the prediction results. From the predicted data it can be seen that the presented method has the best accuracy among the three. And the effect of bad data detection is obvious through the comparison of method 2 and method 3: it can be seen that when there are no bad data in training and predicting (the date without *), the predicting results are the same; but when the bad data appear, the effect of bad data detection and revaluation is significant. Method 1 improves the accuracy of method 2 by 18.6%, and method 2 improves the accuracy of method 3 by 3.79%.

Tab. 6.7 STLF results for ten Days

Date	**MAPE(%)**		
	Method 1	**Method 2**	**Method 3**
June 1st	2.28	2.32	2.32
June 2nd	2.75	2.89	2.89
June 3rd	3.63	3.83	3.83
*June 4th	2.82	2.97	3.45
*June 5th	2.68	2.74	5.81
*June 6th	3.12	3.30	4.55
*June 7th	2.51	2.49	4.76
June 8th	3.78	3.81	3.81
*June 9th	3.43	3.69	3.83
June 10th	3.41	3.58	3.58
Average	3.04	3.16	3.88

* implies there are bad data in the training data or predicting data.

To prove the smoothing affect of forecasted load curve, proposed in subsection 6.2.4, four days in 2002, which respectively belong to spring, summer, autumn and winter, are randomly selected as target days. The cases of smoothing and no smoothing are respectively employed. Both cases take the regression tree method proposed in Chap. 4 as the forecasting algorithm.

The result in Tab. 6.8 shows that the smoothing method helps to decrease the forecasting error. This is not very obvious in the average absolute error calculation, but it shows great advantages when calculating the maximum absolute error of every day.

Tab. 6.8 RT prediction result with/without smoothing method

Date	Without Smoothing		With Smoothing	
	MAPE(%)	**Max absolute Error(%)**	**MAPE(%)**	**Max absolute Error(%)**
2002.09.03	2.74	6.44	2.73	6.24
2002.07.21	2.95	12.56	2.71	9.18
2002.02.05	2.75	8.70	2.60	7.08
2002.11.05	2.81	9.21	2.63	7.34
2008.08.02	4.08	9.01	4.05	7.65

7 Conclusion and Outlook

7.1 Conclusion

The general objective of this work is to provide power system dispatchers with an accurate and convenient short-term load forecasting (STLF) system, which helps to increase the power system reliability and reduce the system operation cost. In the modern electricity market, the energy trade and the spot price establishment are based on a precise load forecasting result. The significance of STLF inspires the author to develop this work.

On the whole, this thesis is composed of three parts: historical data treatment, individual algorithms proposed for load forecasting, and the design of an integrative and convenient system combining different algorithms.

The existence of bad data in the historical load curve affects the precision of the load forecasting result. There are two kinds of bad data in the daily load curve: false channel bad data and abnormal event bad data. The concepts of forward second order difference (FSOD) and backward second order difference (BSOD) are introduced. Bad data always correspond to the second order difference being outside a certain range $\boldsymbol{V}$. The bad data separates a load curve into several segments. The points in every segment are continuous. By calculating the second order difference, the continuous segment(s) of a load curve can be detected. Bad data between the neighbouring continuous segments are regressed in a quadratic form to revaluate the points between them. Case studies for Shanghai Power Grid and the German E.ON Grid indicate, that the second order difference bad data detection method can effectively find false channel bad data and abnormal event bad data.

After detecting the bad data and replacing them with reasonable data, the load curve might still not be very smooth because of the impulse load. This research regards a load curve as the sum of two load curves: an essential load curve that represents the basic load requirement, and a vibrating curve that contains the information of sudden change of the large consumers’ state. The former is obtained by smoothing the load curve, and is utilized in training instead of the original curve. The essential load curve is achieved through weighted least square quadratic fitting. The load forecasting methods with and without smoothing are respectively employed. Better prediction precision is acquired by the one with the smoothing treatment.

This work applies the regression tree algorithm to the load forecasting problem. The algorithm can automatically classify the data and assign a value for every tree node without a prior knowledge. The result of the algorithm has the form of "if… then…", which can be easily understood. Both continuous and categorical independent variables are acceptable in forming a regression tree. It can handle the non-homogeneous relationship between input and output variables. It can estimate the error of the prediction values. It is robust with outliers. Given a redundant set of input variables, it is able to pick up the important input variables and to ignore the redundant ones.

Although the original purpose of applying a regression tree is to avoid a prior knowledge, it is found that good understanding of the system helps to improve the regression tree design for a better forecasting result. Therefore some special treatments are added to the regression tree according to the expert experience. These treatments include: setting up a weekday tree, weekend tree, and holiday tree; setting up the relative value increment regression tree; and, setting up the absolute value increment regression tree. Many forecasting results are obtained with different trees, and they are combined to generate a combined forecasted value, together with the total error indicator. This work also presents the concept of "desert border variable", the effect of which is removed from the forecasting results. Historical data selection is done according to the expert experience of "the near date samples have more similarity to the target load than the distant date samples". A case study compares the presented regression tree method with the ANN algorithm and proves its superiority.

This work proposes an SVM-based forecasting method. Support vector training classifies the input data into clusters efficiently. The data in every cluster have good similarity for further training. A decision tree is an efficient way to decide which cluster the input data belong to. SVR is used to predict daily load due to the advantages of structural risk, simple mathematical models and short training time. Clustering classifies the data with numerical diversity into different clusters. The prediction precision of methods with clustering is higher than the methods without clustering. Support vector clustering is a useful algorithm to classify data. Compared with the conventional clustering method of k-means, support vector clustering doesn't rely on the initial values; the quadratic programming problem of the cluster description algorithm is convex and has a globally optimal solution; it can deal with outliers, making it robust with respect to the noise in the data. The repetitious support vector clustering method proposed in this thesis clusters the data in an iterative way. If the repetitious support vector method is not applied, there are too many isolated data. Less isolated points are obtained by this method and they correspond to the abnormal days very well. Many isolated data produced in a conventional support vector clustering can be predicted by the repetitious method and the result is acceptable. The points inside the intersection of overlapping clusters can be trained in different clusters. This is extremely helpful for those clusters that do not have many members. The simulation result shows the precision can be greatly improved by this method.

Holiday and anomalous day load forecasting is emphasized because this is always a difficulty for STLF. Two methods are proposed to solve this problem: a holiday regression tree and imaginary load method. In the holiday regression tree method, the concept of anomalous period is presented and every period is assigned an index. A regression tree method is employed to predict the anomalous day results. The anomalous day load is affected by not only the common factors of load, such as climate and recent load, but also the holiday property. Therefore, in imaginary load method, the relationship between the imaginary load and its corresponding real load is analyzed with SVM. Holiday load forecasting examples prove the feasibility of the two methods.

This thesis proposes to combine RT and SVM to take advantage of the merits and avoid the demerits of the two algorithms. Firstly RT is established. If the target load falls into a leaf node with a large number of similar samples and very low dispersion, the leaf node output value can be taken as its forecasted value. Otherwise, SVM is executed to analyze the behavior of the samples in the same node.

Different methods outperform others in different conditions. A combination method is proposed to employ more single load forecasting algorithms and take more advantage of the more appropriate ones.

Different power systems have different load behaviours. In this work three-table frame and related programming modules are designed for different users to input the system load properties and the calculation requirement. With these three tables, users can apply the proposed comprehensive algorithm easily without any modification of the programs themselves. This idea increases the forecasting system portability and generalization.

7.2 Experiences and Outlook

During the research procedure some unsuccessful attempts have been made. The first one is the application of the apparent temperature. The weather condition is very influential to the load. Common weather variables include temperature, humidity, sunshine duration, amount of daylight, wind velocity. In meteorology the concept of “apparent temperature” is defined to measure the people’s feeling of the environment temperature. This variable is mainly decided by the actual temperature, but it is also influenced by the environment humidity and the wind velocity.

In this research work the author tried to use “highest apparent temperature” and “lowest apparent temperature” as influential variables of load. The highest temperature and lowest temperature of the day are respectively applied in the calculation of the apparent temperature, as well as the average humidity and average wind velocity. Experiment results show that the proposed method is even less accurate than the methods employing normal weather variables. Through a further analysis it is found that using the average humidity and wind velocity to represent real time humidity and wind velocity can cause large errors. Nevertheless, in the existing weather report and forecasting systems, only the maximum, minimum and mean values of these two variables are available. It is expected that in the near future, the hourly humidity and wind velocity can be provided when recording the historical and predicting the future weather data, so that the apparent temperature might be effective in load forecasting.

Another unsuccessful attempt is to train all the historical data in one SVM frame. In the thesis, three-year historical load data are regarded as sample dependent variables regardless of the day type and time point. 25 corresponding independent variables concerning the weather condition, day type, time point and historical load, are listed. This results in an extremely large dataset. The training time was very long for one single target point (about 45 hours). The predicted loads have an average error of about 15%, which is very high. This experiment indicates that clustering, independent variable selection and human experience are crucial to load forecasting. Although the most ideal way of load forecasting is to “provide the computer with a large amount of data and let it calculate the rule while the people are drinking and chatting”, this proves impossible with the current techniques.

The following recommendations may help to further contributions in this area.

In the application of support vector machine, the parameters of the input-output function are decided by experience. Further employment of genetic or grid algorithms might help to locate the most appropriate parameters.

In the electricity market environment, the electricity price and market mechanism are also influential to the load. In this research work they are neglected due to the lack of data; only time variables are considered to contain the market information of the system. In future work the market variables can be directly considered.

The proposed SVM, RT, RTSVM and integrative forecasting are methods to find the input-output relationship. Therefore they shouldn't be limited to short-term load forecasting. Future work might employ these proposed methods to super short-term, mid-term and long-term load forecasting.

Recent research on demand side management enhancements have been applied to electrical energy consumers. The load curve of these users may have some new characteristics. Future work can focus on the load forecasting of the demand side management users. In addition, load characteristics can also be explored to find ways of lowering the system load peak.

Appendix A Methodology for building a classification tree

In constructing a classification tree, CART makes use of prior probabilities (priors). A brief review of priors and their variations as used in CART is provided.

Prior probabilities play a crucial role in the tree-building process. Three types of priors are available in CART: priors data, priors equal, and priors mixed. They are either estimated from data or supplied by the analyst.

In the following discussion, let

N = number of cases in the sample

N_j = number of class j cases in the sample, and

F_j = prior probabilities of class j cases

Priors data assumes that distribution of the classes of the dependent variable in the population is the same as the proportion of the classes in the sample. It is estimated as

$$F_j = N_j / N.$$

Priors equal assumes that each class of the dependent variable is equally likely to occur in the population. For example, if the dependent variable in the sample has two classes, then

$$\text{prob(class 1)} = \text{prob(class 2)} = 1/2.$$

Priors mixed is an average of priors equal and priors data for any class at a node.

Three components are required in the construction of a classification tree: (1) a set of questions upon which to base a split; (2) splitting rules and goodness-of-split criteria for judging how good a split is; and (3) rules for assigning a class to each terminal node. These components are discussed below.

Two question formats are defined in CART: (1) Is $X \leq d$?, if X is a continuous variable and d is a constant within the range of X values. For example, is HighTemperature ≤ 18? Or (2) is $Z = b$?, if Z is a categorical variable and b is one of the integer values assumed by Z. For example, is Holiday = false?

The number of possible split points on each variable is limited to the number of distinct values each variable assumes in the sample. For example, if a sample size equals N, and if X is a continuous variable and assumes N distinct points in the sample, then the maximum number of split points on X is equal to N. If Z is a categorical variable with m distinct points in a sample, then the number of possible split points on Z equals $2^{m-1} - 1$.CART assumes that each split will be based on only a single variable. Let

$$j = 1,2,\ldots,k$$

be the number of classes of categorical dependent variables; then define $p(j|t)$ as class probability distribution of the dependent variable at node t, such that $p(1|t) + p(2|t) + p(3|t) + \ldots + p(k|t) = 1$, $j=1,2,\ldots,k$. Let $i(t)$ be the impurity measure at node t. Then define $i(t)$ as a function of class probabilities $p(1|t)$, $p(2|t)$, $p(3|t)$,… Mathematically, $i(t)=\Phi[p(1|t), p(2|t), p(3|t), \ldots, p(k|t)]$. The definition of impurity measure is generic and allows for flexibility of functional forms.

Splitting Rules. There are three major splitting rules in CART: the Gini criterion, the towing rule, and the linear combination splits. In addition to these main splitting rules, CART users can define a number of other rules for their own analytical needs. CART uses the Gini criterion as its default splitting rule.

The Gini impurity measure at node t is defined as $i(t) = 1-S$ (the impurity function), where $S = \Sigma\, p^2(j|t)$, for $j = 1,2,\ldots k$.

The impurity function attains its maximum if each class (vulnerable or not) in the population occurs with equal probability. That is $p(1|t) = p(2|t)=\ldots= p(k|t)$. On the other hand, the impurity function attains its minimum (=0) if all cases at a node belong to only one class. That is, if node t is a pure node with a zero misclassification rate, then $i(t)=0$.

Let s be a split at node t. Then, the goodness of split s is defined as the decrease in impurity measured by

$$\Delta i(s, t)=i(t)-p_L[i(t_L)]- p_R[i(t_R)]$$

where

s = a particular split,

p_L = the proportion of the cases at node t that go into the left child node, t_L

p_R = the proportion of the cases at node t that go into the right child node, t_R

$i(t_L)$=impurity of the left child node, and

$i(t_R)$ =impurity of the right child node.

There are two rules for assigning classes to nodes. Each rule is based on one of two types of misclassification costs.

The Plurality Rule: Assign terminal node t to a class for which $p(j|t)$ is the highest. If the majority of the cases in a terminal node belong to a specific class, then that node is assigned to that class. The rule assumes equal misclassification costs for each class. It does not take into account the severity of the cost of making a mistake. This rule is a special case of rule 2.

Assign terminal node t to a class for which the expected misclassification cost is at a minimum. The application of this takes into account the severity of the costs of misclassifying cases or observations in a certain class, and incorporates cost variability into a Gini splitting rule.

The tree-building process starts by partitioning a sample or the root node into binary nodes based upon a very simple question of the form

Is $X \leq d$?,

where X is a variable in the dataset and d is a real number. Initially, all observations are placed in the root node. This node is impure because it contains observations of mixed classes. The goal is to devise a rule that will break up these observations and create groups or binary nodes that internally have more purity than the root node. CART uses a computer intensive algorithm that searches for the best split at all possible split points for each variable. The methodology that CART uses for growing trees is technically known as binary recursive partitioning. Starting from the root node, and using, for example, the Gini diversity index as a splitting rule, the tree building process is as follows:

1. CART splits the first variable at all of its possible split points (at all of the values the variable assumes in the sample). At each possible split point of a variable, the sample splits into binary or two child nodes. Cases with a “yes” response to the question posed are sent to the left node and those with “no” responses are sent to the right node.

2. CART then applies its goodness- of- split criteria to each split point and evaluates the reduction in impurity that is achieved using the formula

$$\Delta i(s, t)=i(t)-p_L[i(t_L)]-p_R[i(t_R)]$$

which was described earlier.

3. CART selects the best split of the variable as that split for which the reduction in impurity is highest.

4. Steps 1–3 are repeated for each of the remaining variables at the root node.

5. CART then ranks all of the best splits on each variable according to the reduction in impurity achieved by each split.

6. It selects the variable and its split point that most reduced the impurity of the root or parent node.

7. CART then assigns classes to these nodes according to the rule that minimizes misclassification costs. CART has a built-in algorithm that takes into account user-defined variable misclassification costs during the splitting process. The default is unit or equal misclassification costs.

8. Because the CART procedure is recursive, steps 1–7 are repeatedly applied to each non-terminal child node at each successive stage.

9. CART continues the splitting process and builds a large tree.

The largest tree is built if the splitting process continues until every observation constitutes a terminal node. Obviously, such a tree will have a large number of terminal nodes, which will be either pure or have very few cases.

Incompleteness of data may be a problem for conventional statistical analysis, but not for CART. It makes use of a surrogate variable splitting rule. A surrogate variable in CART is that variable that mimics or predicts the split of the primary variable. If a splitting variable used for tree construction has missing values for some cases, those cases are not thrown out. Instead, CART classifies such cases on the basis of the best surrogate variable (the variable with a close resemblance to the primary split variable). The surrogate may have a different cutoff point from the primary split, but the number of cases the surrogate split sends into left and right nodes should be very close to that with the primary split. By default, CART analysis produces five surrogate variables as part of its standard output. Surrogate splits are available only for splits based on a single variable.

Appendix B Forecasting Result for Frankfurt Substation

1. Data Resource

Load data

The load data is from a substation in Mainova AG. The data time span is from 1/1/1999 to 12/29/2003. There are 96 sampling load points for every day.

Weather data

The weather data is from http://www.dwd.de/de/de.htm. Four weather factors of the day are utilized: highest temperature, lowest temperature, mean humidity, and mean degree of cloud cover.

Holiday information

The holiday information is from http://www.nensel-kalender.de/. The holiday information of the Hessen state is employed

2. Data Treatment

With the proposed "second order difference" and "weighted least square quadratic fitting" methods, all the load data are inspected and the suspicious bad data are detected. If possible, the bad data are revalued with the estimated values.

3. Load Prediction

Forecast object

The everyday load curves of a year from 12/2/2000 – 12/1/2001 are forecasted. The starting time (12/2/2000) was randomly selected.

Available data for the forecast object

For the i^{th} point of the d^{th} day to be forecasted, the following information is supposed to be known:

1. The weather information from 1/1/1999 to day (d-1)
2. The forecasted weather information from 1/1/1999 to day d. If the forecasted weather

information is not available, the actual data are used instead.

3. The load data from 0:00, 1/1/1999 to the i^{th} point of the $(d-1)^{th}$ day
4. The holiday information from 1/1/1999 to i^{th} the $(d-1)^{th}$ day

Forecasting methods

The following algorithms are employed for load forecasting and derivation of an integrated forecasting result.

1. Regression tree algorithm
2. Support vector regression algorithm
3. The integrated algorithm

4. Suspected Bad Data Report

Bad data statistics

From 1/1/1999 to 12/29/2003, there are altogether 1824 load curves. 32 of them are thought to contain bad data. Among these 23 curves can be revalued and 9 cannot, shown in Tab. A. 1.

Tab. A.1 Statistics of the load bad data

	Total curve number	Curves with bad data	Curves that can be revalued	Curves that can not be revalued	Total valid curves
Actual number	1824	32	23	9	1815
Percentage number	100%	1.75%	1.26%	0.49%	99.5%

Some Load curves with bad data which can be revised

Fig. A. 1 – Fig A. 3 show some examples of the bad data load curve which can be detected and revalued. Fig. A. 4 – Fig A. 6 show some examples of the bad data load curve which can not be detected and revalued.

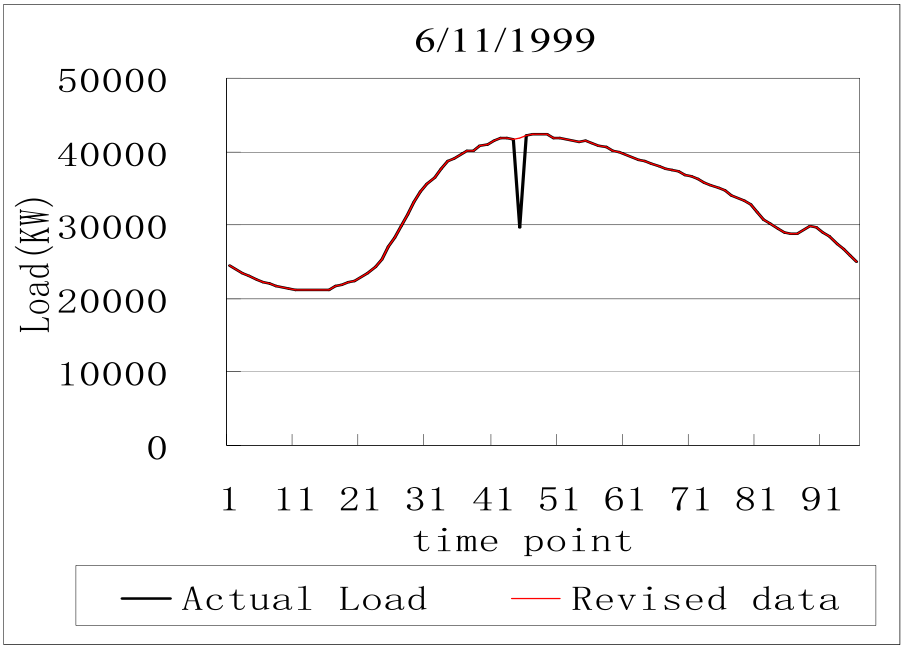


Fig. A. 1 Load correction of 6/11/1999

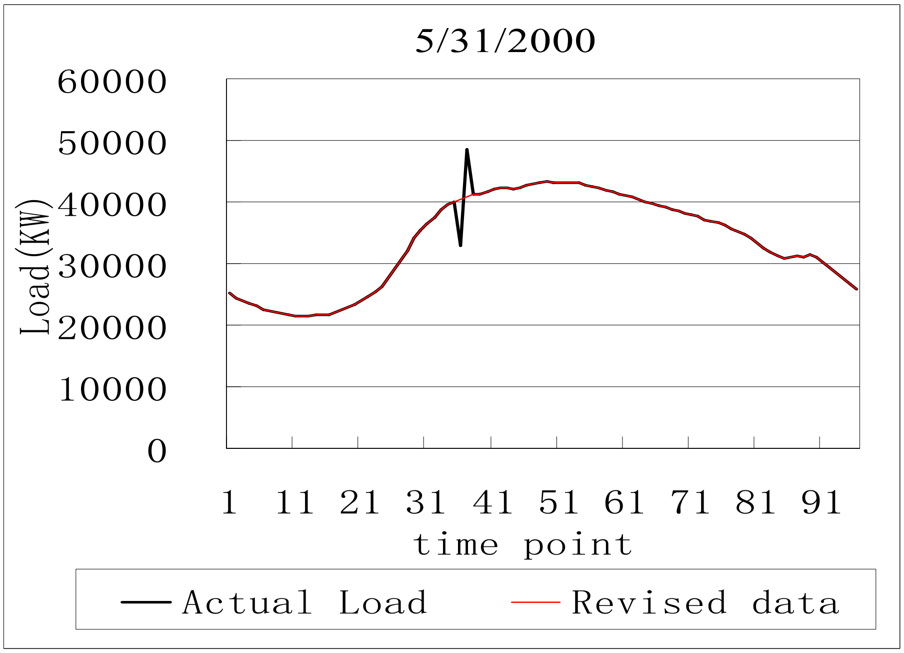


Fig. A. 2 Load correction of 5/31/2000

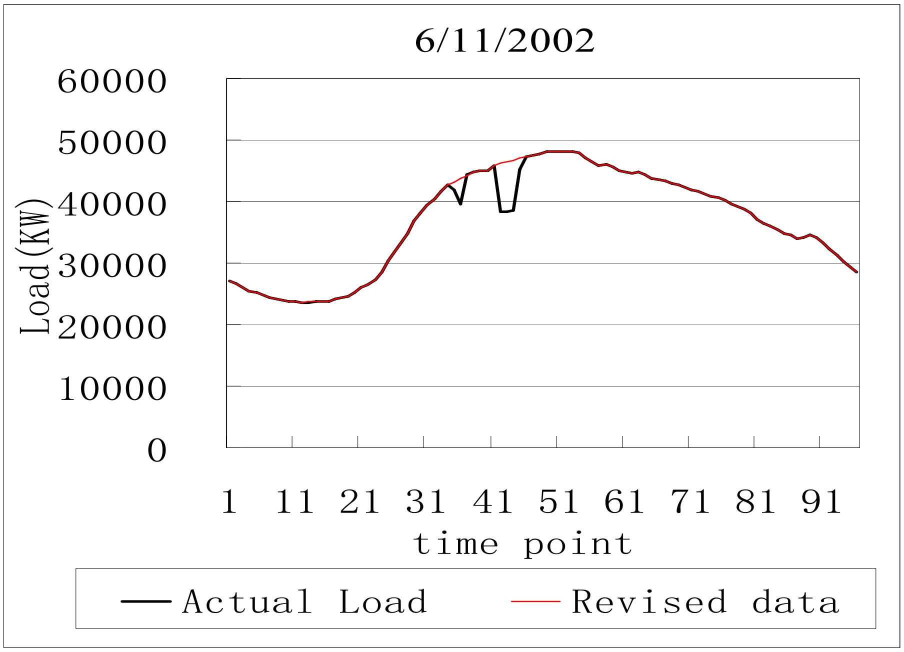


Fig. A. 3 Load correction of 6/11/2002

Some Load curves with bad data that cannot be revised

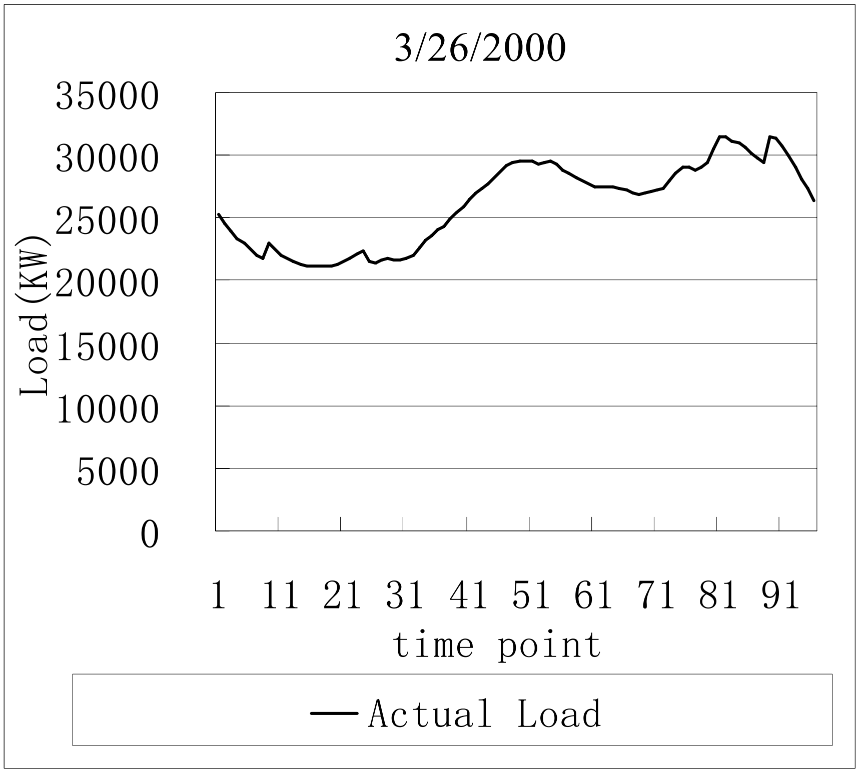


Fig. A. 4 Load curve of 3/26/2000

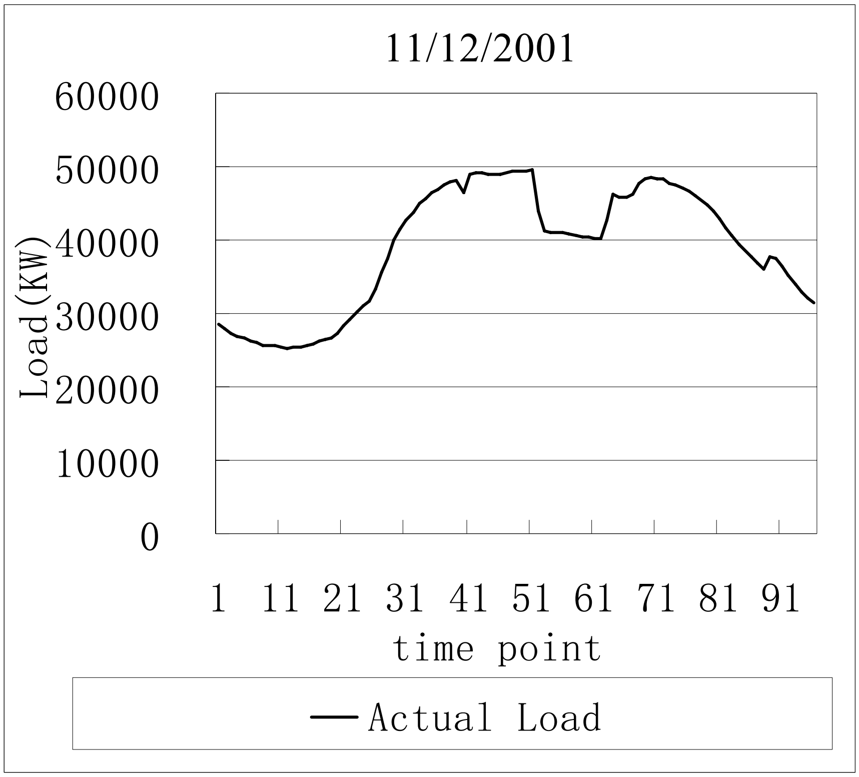


Fig. A. 5 Load curve of 11/12/2001

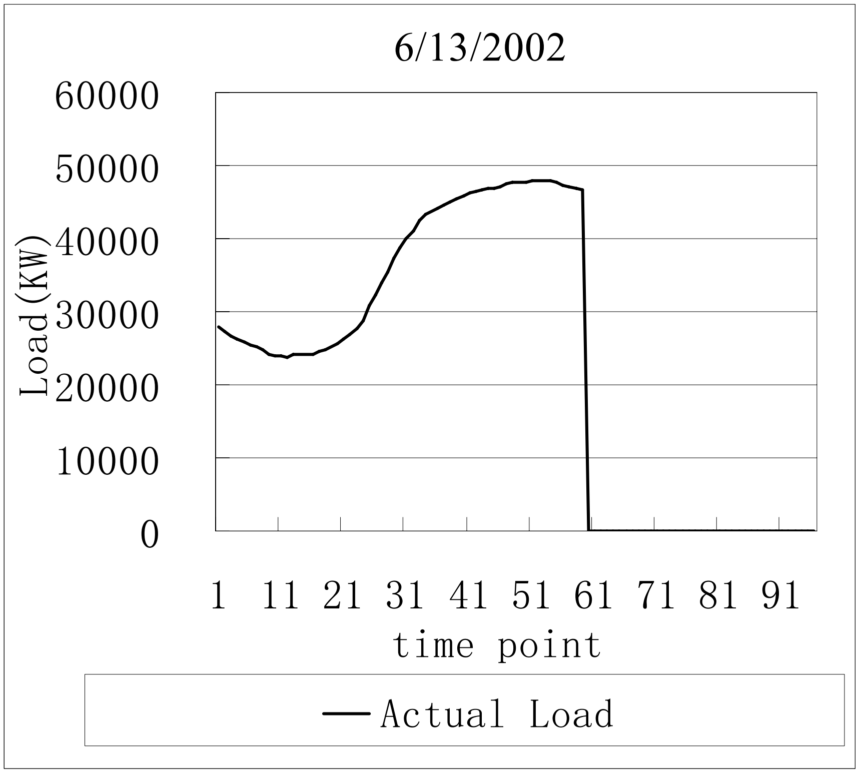


Fig. A. 6 Load curve of 6/13/2002

5. Forecasting Results

The forecasting error of every day table

The forecasting error of every day is listed in Tab. A. 2.

Tab. A. 2 Forecasting error of every day

Date	Mean Absolute Percentage Error (%)	Mean Max Absolute Percentage Error(%)	Date	Mean Absolute Percentage Error(%)	Mean Max Absolute Percentage Error(%)
12/2/2000	0.97	6.34	6/3/2001	5.00	9.59
12/3/2000	0.86	5.93	6/4/2001	1.90	3.93
12/4/2000	0.99	3.23	6/5/2001	4.68	8.83
12/5/2000	1.01	2.60	6/6/2001	1.47	5.91
12/6/2000	2.35	6.79	6/7/2001	1.76	4.83
12/7/2000	1.85	4.89	6/8/2001	1.10	4.33
12/8/2000	1.28	4.10	6/9/2001	0.95	2.68
12/9/2000	0.90	2.36	6/10/2001	1.04	3.93
12/10/2000	1.94	7.93	6/11/2001	1.43	5.51
12/11/2000	0.64	1.76	6/12/2001	1.08	2.67
12/12/2000	1.47	3.84	6/13/2001	1.04	3.46
12/13/2000	0.58	2.06	6/14/2001	9.31	14.75
12/14/2000	1.58	5.86	6/15/2001	2.46	6.35
12/15/2000	1.58	4.00	6/16/2001	3.00	7.25
12/16/2000	1.40	3.47	6/17/2001	2.79	8.63
12/17/2000	1.35	3.08	6/18/2001	0.80	2.10
12/18/2000	1.91	3.81	6/19/2001	0.95	2.15
12/19/2000	1.88	4.94	6/20/2001	1.01	2.94
12/20/2000	1.81	4.56	6/21/2001	1.22	3.08
12/21/2000	1.29	3.49	6/22/2001	0.76	2.98
12/22/2000	3.13	8.24	6/23/2001	1.50	3.94
12/23/2000	2.83	7.05	6/24/2001	0.78	3.34
12/24/2000	6.27	19.02	6/25/2001	2.21	3.18
12/25/2000	3.29	7.65	6/26/2001	2.72	3.99
12/26/2000	3.17	6.51	6/27/2001	4.57	6.68
12/27/2000	11.07	20.84	6/28/2001	4.23	7.83
12/28/2000	6.29	13.51	6/29/2001	1.25	2.78
12/29/2000	2.42	6.17	6/30/2001	1.94	3.82
12/30/2000	2.80	9.22	7/1/2001	1.78	3.87
12/31/2000	6.36	18.17	7/2/2001	0.83	2.86
1/1/2001	5.38	12.68	7/3/2001	0.53	1.43
1/2/2001	1.60	4.34	7/4/2001	0.54	1.98
1/3/2001	2.37	5.16	7/5/2001	2.39	3.63
1/4/2001	4.42	9.11	7/6/2001	2.34	4.78
1/5/2001	2.95	7.72	7/7/2001	2.79	7.05
1/6/2001	2.34	6.89	7/8/2001	1.52	5.27
1/7/2001	3.06	8.86	7/9/2001	1.27	3.65
1/8/2001	4.62	10.68	7/10/2001	0.83	3.08

1/9/2001	5.42	10.19	7/11/2001	1.32	3.93
1/10/2001	5.28	8.79	7/12/2001	3.18	5.44
1/11/2001	2.92	9.18	7/13/2001	1.68	3.84
1/12/2001	1.70	5.33	7/14/2001	1.60	4.45
1/13/2001	2.84	5.98	7/15/2001	1.63	3.53
1/14/2001	3.06	7.06	7/16/2001	2.70	4.58
1/15/2001	3.26	6.15	7/17/2001	0.86	3.13
1/16/2001	3.56	5.87	7/18/2001	1.64	3.85
1/17/2001	2.03	6.92	7/19/2001	0.52	2.28
1/18/2001	1.96	4.21	7/20/2001	0.91	2.48
1/19/2001	0.98	2.07	7/21/2001	2.11	3.50
1/20/2001	1.41	7.24	7/22/2001	0.72	2.16
1/21/2001	0.65	2.22	7/23/2001	1.82	3.68
1/22/2001	1.32	3.34	7/24/2001	3.08	5.53
1/23/2001	2.31	5.59	7/25/2001	2.19	3.92
1/24/2001	2.57	5.47	7/26/2001	2.44	4.59
1/25/2001	0.89	2.89	7/27/2001	1.13	6.06
1/26/2001	0.84	2.64	7/28/2001	1.01	3.14
1/27/2001	2.93	7.61	7/29/2001	5.54	8.59
1/28/2001	0.95	2.58	7/30/2001	2.08	5.00
1/29/2001	1.42	2.93	7/31/2001	3.91	6.47
1/30/2001	0.89	3.08	8/1/2001	3.19	8.23
1/31/2001	0.93	3.05	8/2/2001	1.63	6.03
2/1/2001	1.11	3.34	8/3/2001	5.73	10.80
2/2/2001	1.20	3.30	8/4/2001	2.13	4.19
2/3/2001	1.56	3.41	8/5/2001	2.01	6.11
2/4/2001	1.76	5.30	8/6/2001	1.89	5.76
2/5/2001	2.19	4.49	8/7/2001	1.28	3.49
2/6/2001	2.14	4.76	8/8/2001	1.35	3.35
2/7/2001	1.29	4.49	8/9/2001	2.06	6.24
2/8/2001	1.94	4.06	8/10/2001	1.54	4.31
2/9/2001	1.46	4.06	8/11/2001	3.10	6.46
2/10/2001	3.34	12.91	8/12/2001	3.14	6.76
2/11/2001	2.11	5.30	8/13/2001	1.58	3.75
2/12/2001	1.08	4.36	8/14/2001	2.90	7.11
2/13/2001	1.25	4.82	8/15/2001	3.74	7.21
2/14/2001	1.76	3.99	8/16/2001	4.20	7.51
2/15/2001	1.64	3.75	8/17/2001	1.15	3.71
2/16/2001	1.76	4.48	8/18/2001	1.18	2.94
2/17/2001	3.24	7.21	8/19/2001	3.81	7.34
2/18/2001	2.25	6.67	8/20/2001	1.54	3.57
2/19/2001	2.45	7.02	8/21/2001	1.83	5.52
2/20/2001	1.92	5.57	8/22/2001	1.85	3.82
2/21/2001	0.91	3.29	8/23/2001	2.79	4.53
2/22/2001	0.93	3.56	8/24/2001	4.61	5.78
2/23/2001	2.41	8.58	8/25/2001	4.99	6.59
2/24/2001	2.69	11.07	8/26/2001	5.80	9.76
2/25/2001	2.19	7.87	8/27/2001	4.94	9.11
2/26/2001	3.61	8.81	8/28/2001	1.04	3.34
2/27/2001	4.18	10.62	8/29/2001	3.80	6.90
2/28/2001	1.10	2.12	8/30/2001	3.15	6.00
3/1/2001	2.08	7.35	8/31/2001	1.89	3.39
3/2/2001	0.65	2.33	9/1/2001	1.83	4.23
3/3/2001	1.52	5.23	9/2/2001	2.56	5.14

3/4/2001	1.09	3.31	9/3/2001	1.76	6.25
3/5/2001	1.55	3.53	9/4/2001	1.70	3.87
3/6/2001	1.58	4.52	9/5/2001	0.72	2.24
3/7/2001	1.73	6.65	9/6/2001	1.21	4.30
3/8/2001	1.45	3.87	9/7/2001	2.78	6.07
3/9/2001	1.41	3.71	9/8/2001	2.55	6.17
3/10/2001	1.60	3.87	9/9/2001	3.02	6.46
3/11/2001	1.40	4.09	9/10/2001	3.14	6.83
3/12/2001	1.56	3.45	9/11/2001	3.39	6.38
3/13/2001	2.10	6.20	9/12/2001	3.13	6.25
3/14/2001	1.53	5.62	9/13/2001	1.65	4.01
3/15/2001	0.91	3.44	9/14/2001	0.94	2.87
3/16/2001	1.55	6.28	9/15/2001	2.70	8.27
3/17/2001	1.45	3.15	9/16/2001	3.35	5.14
3/18/2001	1.23	4.92	9/17/2001	2.14	4.44
3/19/2001	0.87	3.20	9/18/2001	1.39	3.32
3/20/2001	1.34	3.67	9/19/2001	1.67	5.52
3/21/2001	2.21	4.90	9/20/2001	2.08	4.81
3/22/2001	1.06	3.19	9/21/2001	0.99	2.34
3/23/2001	1.14	4.98	9/22/2001	1.24	4.94
3/24/2001	1.01	2.87	9/23/2001	0.85	2.42
3/25/2001	2.36	6.06	9/24/2001	1.66	4.50
3/26/2001	2.58	5.93	9/25/2001	0.87	2.78
3/27/2001	3.62	8.76	9/26/2001	1.07	2.69
3/28/2001	3.94	6.53	9/27/2001	1.19	2.68
3/29/2001	2.37	6.38	9/28/2001	0.99	2.70
3/30/2001	1.48	6.55	9/29/2001	1.36	5.01
3/31/2001	1.28	5.57	9/30/2001	0.85	3.87
4/1/2001	1.09	4.62	10/1/2001	1.65	6.45
4/2/2001	4.08	6.98	10/2/2001	1.76	3.60
4/3/2001	2.22	6.17	10/3/2001	8.69	14.55
4/4/2001	1.57	3.86	10/4/2001	2.69	4.80
4/5/2001	1.28	3.00	10/5/2001	1.59	3.87
4/6/2001	0.86	3.52	10/6/2001	1.01	4.19
4/7/2001	1.74	5.00	10/7/2001	1.23	4.19
4/8/2001	2.33	5.24	10/8/2001	0.94	3.33
4/9/2001	2.27	5.91	10/9/2001	0.93	4.57
4/10/2001	2.01	5.55	10/10/2001	1.37	4.27
4/11/2001	1.35	3.22	10/11/2001	1.45	2.98
4/12/2001	3.06	6.39	10/12/2001	1.43	3.08
4/13/2001	1.56	5.13	10/13/2001	2.90	7.43
4/14/2001	2.41	7.76	10/14/2001	1.70	7.50
4/15/2001	4.17	9.16	10/15/2001	2.12	6.02
4/16/2001	1.83	4.87	10/16/2001	1.57	5.26
4/17/2001	1.94	5.36	10/17/2001	1.03	3.13
4/18/2001	2.25	4.60	10/18/2001	0.66	1.79
4/19/2001	2.44	7.92	10/19/2001	1.22	3.58
4/20/2001	1.59	5.49	10/20/2001	2.08	5.46
4/21/2001	1.87	5.20	10/21/2001	1.40	7.92
4/22/2001	1.77	5.52	10/22/2001	0.64	3.10
4/23/2001	1.66	3.50	10/23/2001	1.20	6.34
4/24/2001	1.18	2.96	10/24/2001	0.96	3.13
4/25/2001	3.75	7.68	10/25/2001	NA*	NA*
4/26/2001	1.42	3.41	10/26/2001	0.88	2.37

4/27/2001	1.00	2.75	10/27/2001	0.86	2.33
4/28/2001	1.07	3.64	10/28/2001	1.72	10.87
4/29/2001	1.32	3.07	10/29/2001	1.95	9.34
4/30/2001	5.63	8.50	10/30/2001	1.31	9.25
5/1/2001	3.65	7.22	10/31/2001	1.65	7.90
5/2/2001	3.99	7.45	11/1/2001	1.17	2.95
5/3/2001	3.16	6.13	11/2/2001	1.26	2.96
5/4/2001	1.98	3.88	11/3/2001	1.97	8.47
5/5/2001	1.89	5.39	11/4/2001	1.77	5.16
5/6/2001	1.46	3.57	11/5/2001	2.79	5.25
5/7/2001	1.23	2.64	11/6/2001	3.56	8.00
5/8/2001	0.68	2.03	11/7/2001	3.12	6.99
5/9/2001	0.73	2.17	11/8/2001	2.78	6.65
5/10/2001	1.20	3.00	11/9/2001	1.95	4.85
5/11/2001	0.98	2.31	11/10/2001	1.35	4.84
5/12/2001	2.37	4.46	11/11/2001	0.15	0.80
5/13/2001	1.58	3.15	11/12/2001	NA*	NA*
5/14/2001	1.92	7.36	11/13/2001	NA*	NA*
5/15/2001	0.81	3.44	11/14/2001	2.68	5.66
5/16/2001	1.05	5.24	11/15/2001	0.22	1.72
5/17/2001	1.31	4.37	11/16/2001	0.21	2.91
5/18/2001	1.52	3.47	11/17/2001	1.39	5.73
5/19/2001	1.59	6.73	11/18/2001	1.98	4.14
5/20/2001	1.50	6.51	11/19/2001	1.64	4.02
5/21/2001	0.98	4.79	11/20/2001	0.73	2.09
5/22/2001	0.91	2.83	11/21/2001	0.58	1.70
5/23/2001	0.92	3.38	11/22/2001	2.53	5.92
5/24/2001	6.52	11.48	11/23/2001	1.31	3.87
5/25/2001	4.34	8.10	11/24/2001	1.19	5.96
5/26/2001	0.79	2.81	11/25/2001	1.74	6.21
5/27/2001	1.55	4.89	11/26/2001	1.37	4.35
5/28/2001	3.12	4.55	11/27/2001	1.53	2.97
5/29/2001	4.85	9.49	11/28/2001	1.22	2.89
5/30/2001	1.59	5.47	11/29/2001	1.36	5.19
5/31/2001	1.39	2.67	11/30/2001	1.18	3.49
6/1/2001	2.27	3.75	12/1/2001	1.82	6.51
6/2/2001	1.86	4.71			

NA*: Not available because the original curve is a bad load curve which is not revisable

Error of whole year is shown in Tab. A. 3 and the monthly error is shown in Fig. A. 7 and Fig. A. 8.

Tab. A.3 Mean Error of the forecasting result

Date	Mean Absolute Percentage Error (%)	Mean Max Absolute Percentage Error (%)
From 12/2/2000 – 12/1/2001	2.04	5.21

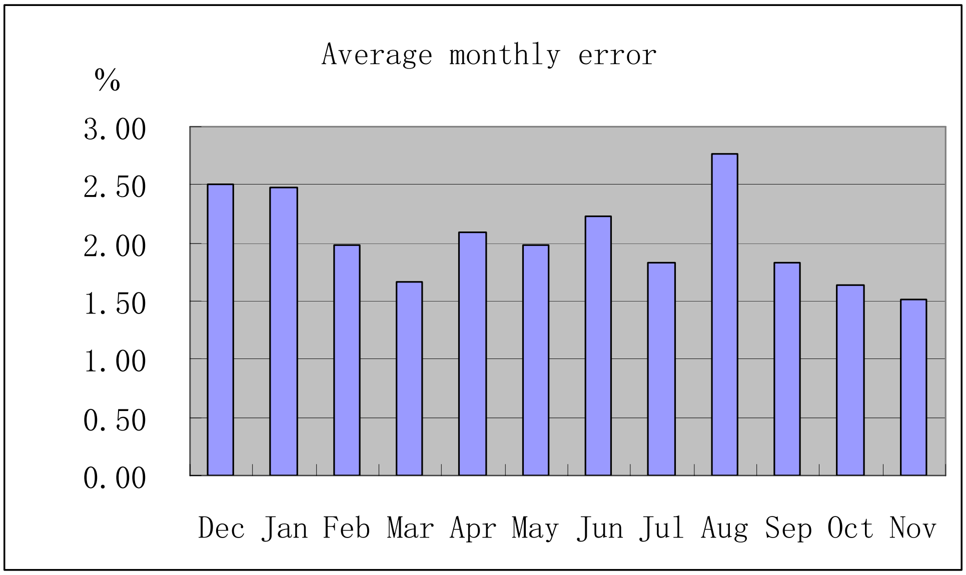


Fig. A. 7 Average monthly error

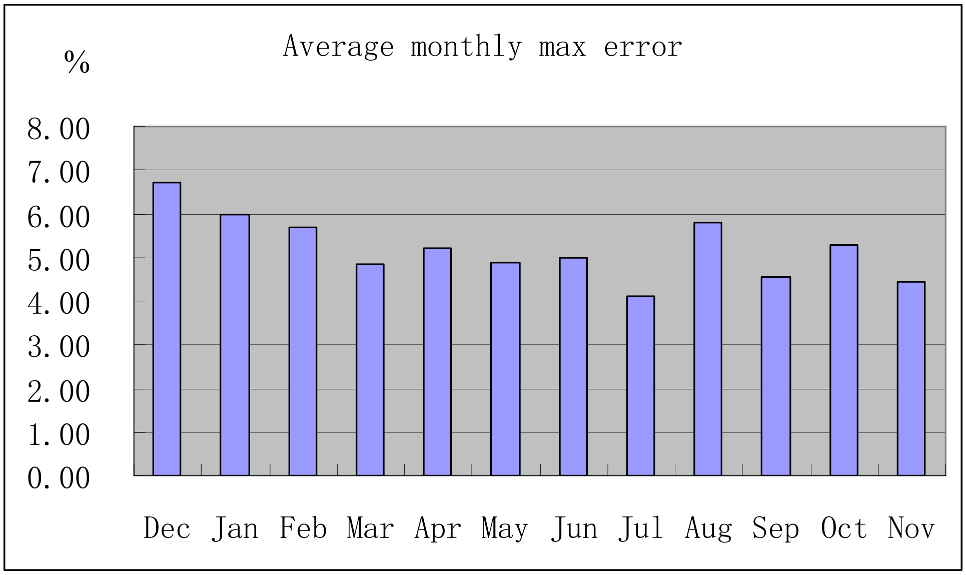
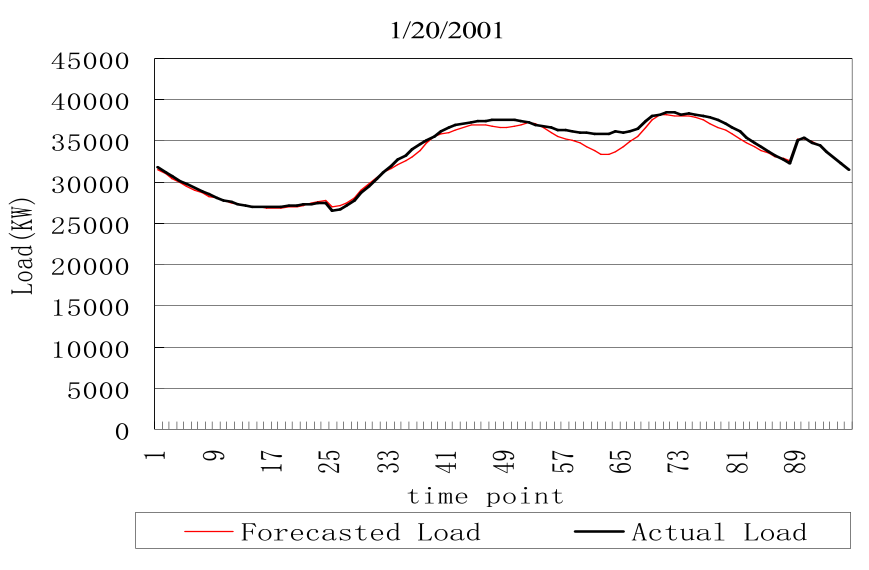
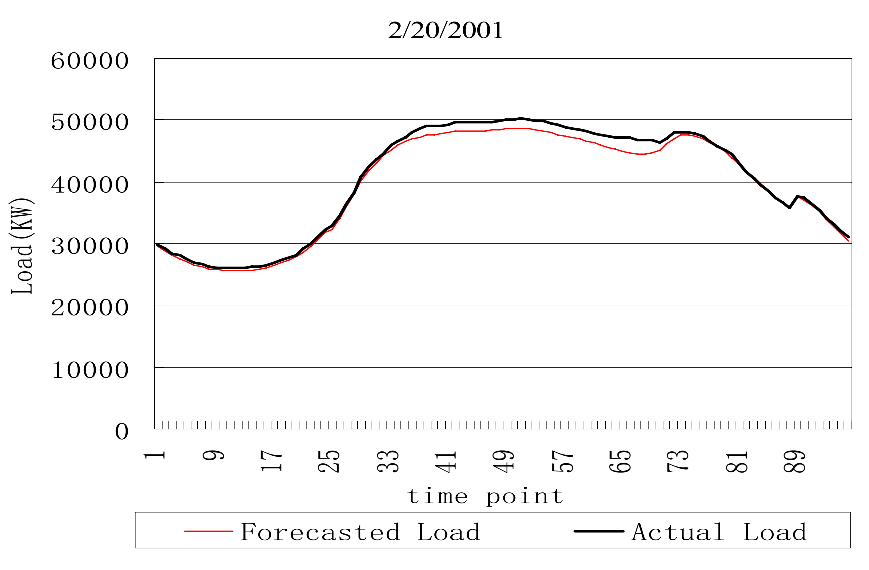


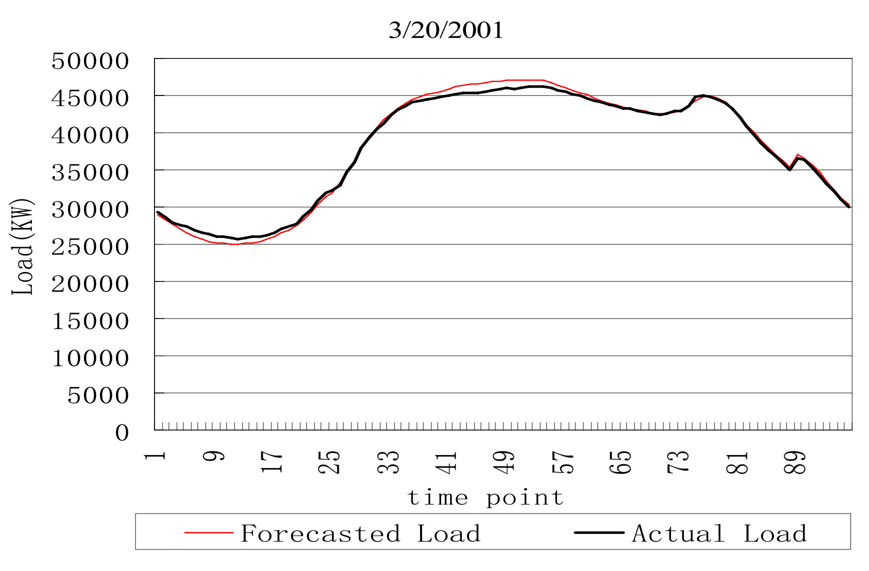
Fig. A. 8 Average monthly max error

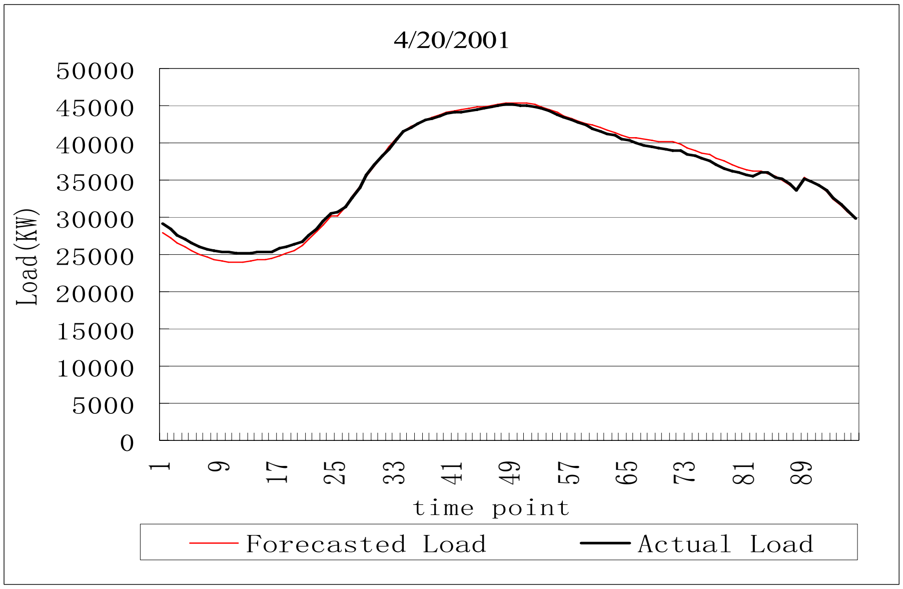
Some examples of daily load forecasting results show

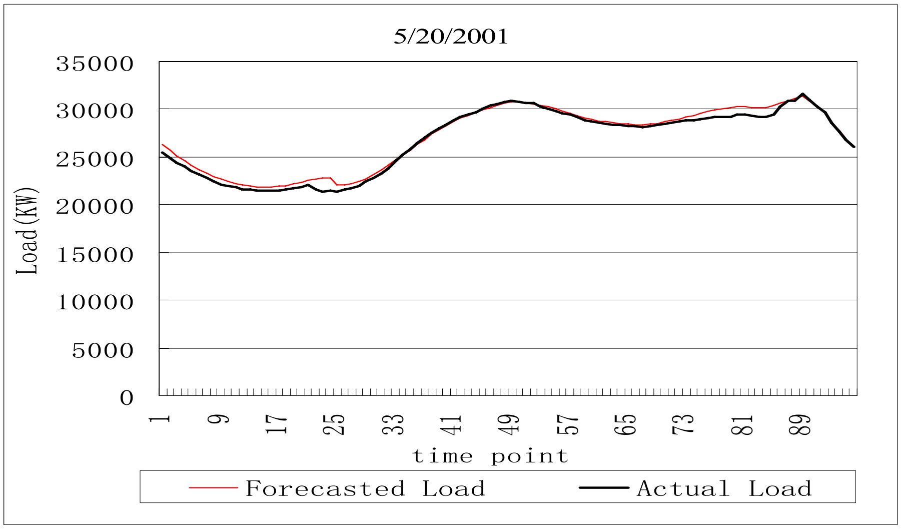
Because there are too many load curve forecasting results to be shown in the report, the 20th of every month are shown as examples. The number of “20” was just selected randomly. They are shown in Fig. A. 9.

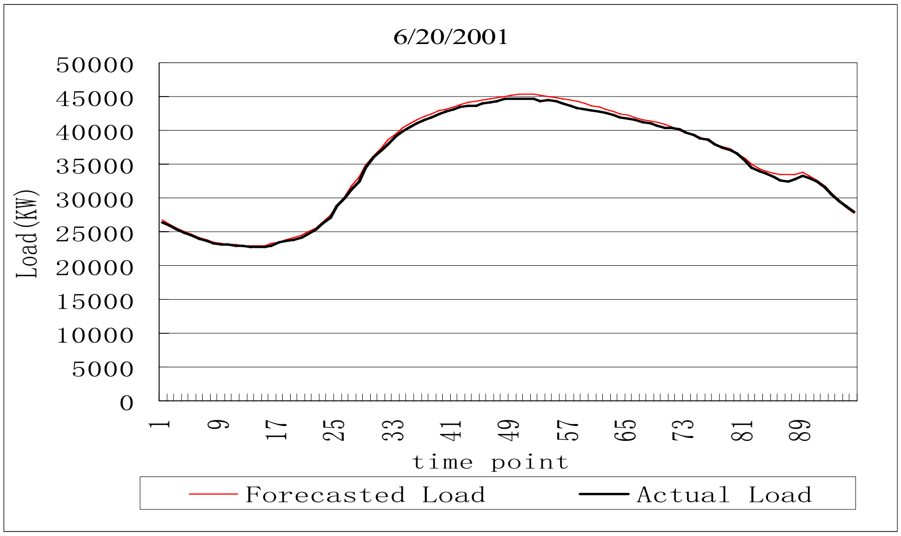

1/20/2001
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
Load (KW)
1
9
17
25
33
41
49
57
65
73
81
89
time point
Forecasted Load
Actual Load

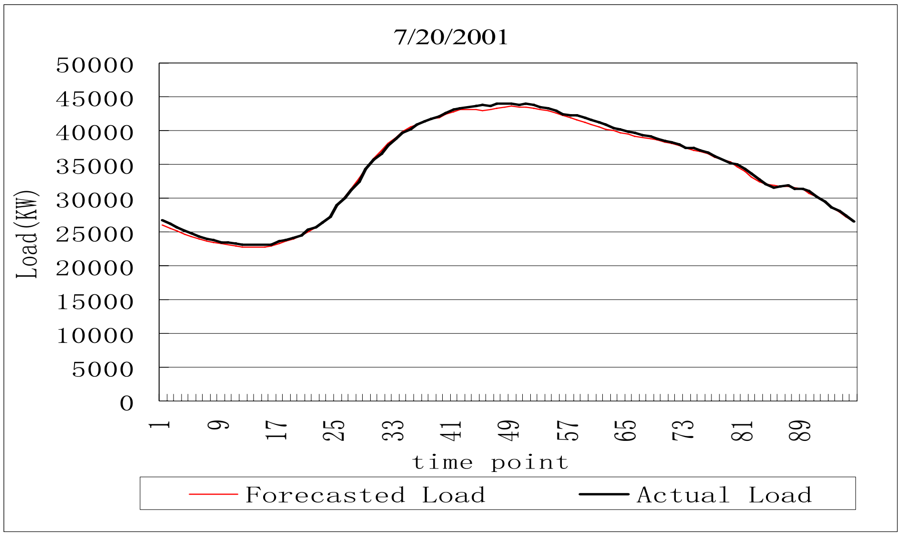

2/20/2001
60000
50000
40000
30000
20000
10000
0
Load (KW)
1
9
17
25
33
41
49
57
65
73
81
89
time point
Forecasted Load
Actual Load

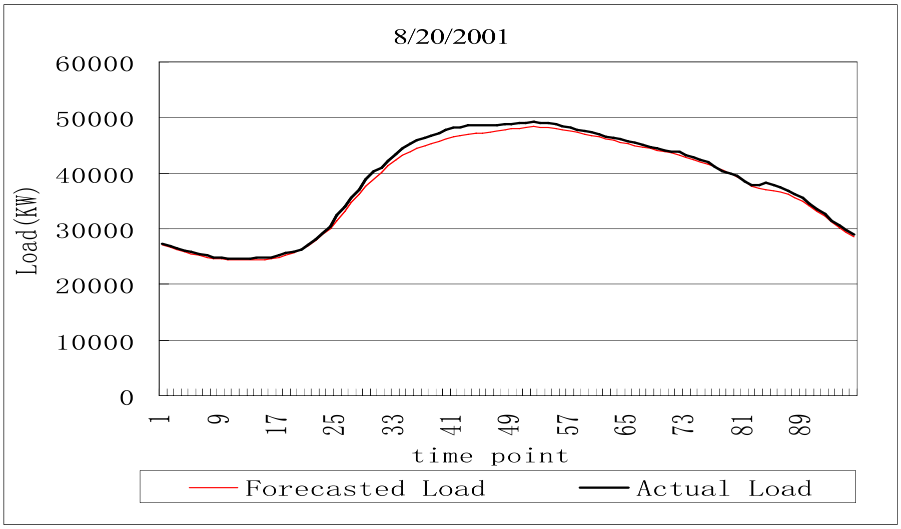

3/20/2001
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
Load (KW)
1
9
17
25
33
41
49
57
65
73
81
89
time point
Forecasted Load
Actual Load

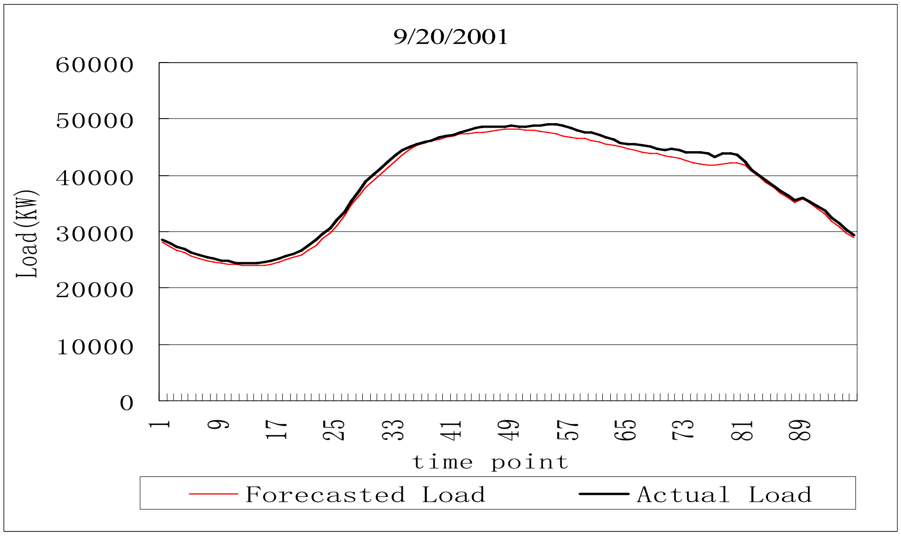

4/20/2001
Load (KW)
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
1
9
17
25
33
41
49
57
65
73
81
89
time point
Forecasted Load
Actual Load


5/20/2001
Load (KW)
35000
30000
25000
20000
15000
10000
5000
0
1
9
17
25
33
41
49
57
65
73
81
89
time point
Forecasted Load
Actual Load


6/20/2001
Load (KW)
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
1
9
17
25
33
41
49
57
65
73
81
89
time point
Forecasted Load
Actual Load


7/20/2001
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
Load(KW)
1
9
17
25
33
41
49
57
65
73
81
89
time point
Forecasted Load
Actual Load


8/20/2001
60000
50000
40000
30000
20000
10000
0
Load(KW)
1
9
17
25
33
41
49
57
65
73
81
89
time point
Forecasted Load
Actual Load


9/20/2001
60000
50000
40000
30000
20000
10000
0
Load(KW)
1
9
17
25
33
41
49
57
65
73
81
89
time point
Forecasted Load
Actual Load

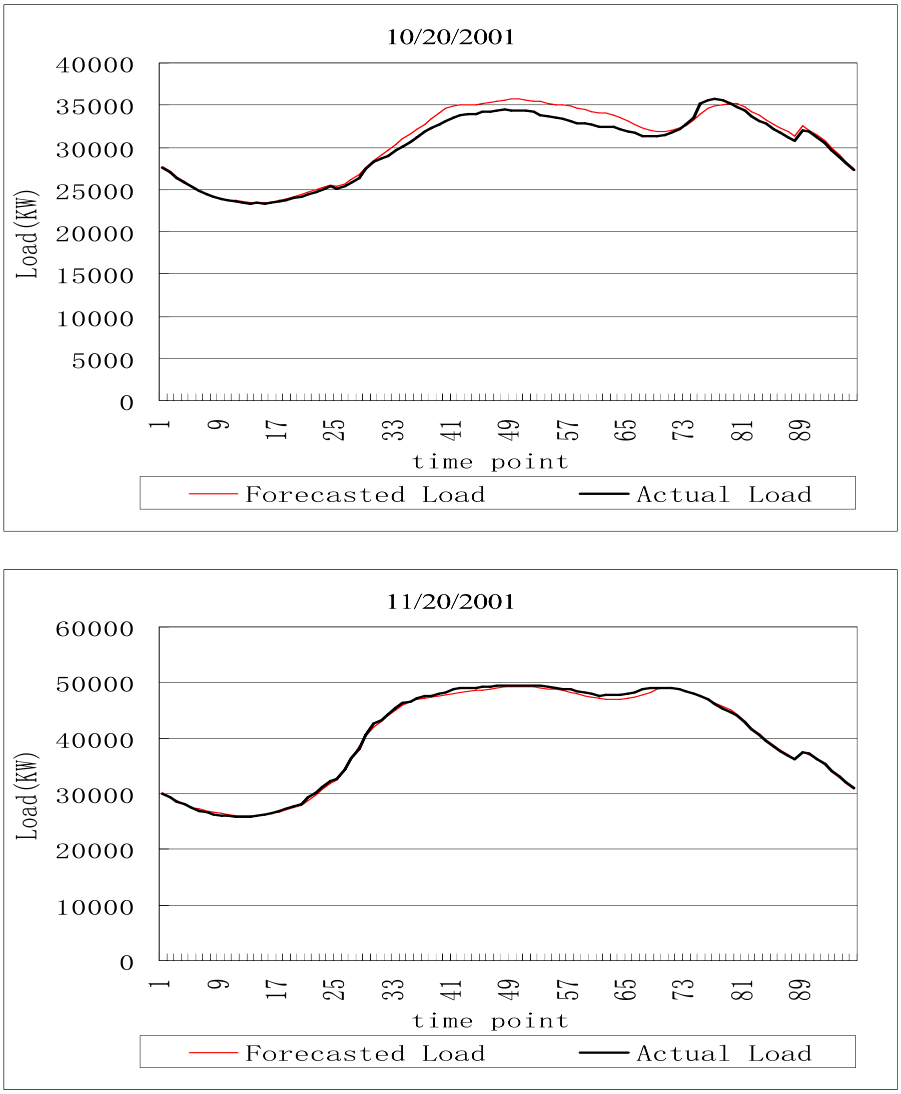


Fig. A. 9 Some forecasting result examples

Appendix C Zusammenfassung in Deutsch

Die Zielsetzung dieser Arbeit ist, Energieerzeugern und Übertragungsnetzbetreibern für die Netzleitstellen ein präzises und praktisches System zur kurzfristigen Last-Prognose zu geben. Es soll die Zuverlässigkeit des Übertragungssystems erhöhen und zur Senkung der Betriebskosten des Netzes beitragen. Im heutigen Energiemarkt hängt der Energiehandel und die Preisermittlung stark von den Ergebnissen einer exakten Lastprognose ab. Die hohe Bedeutung einer kurzfristigen Lastprognose hat die Anregung für die Entwicklung dieser Arbeit gegeben.

Die vorliegende Arbeit besteht aus drei Teilen: Analyse der historischen Daten, Vorstellung einzelner Prognosealgorithmen und Design eines integrativen Systems, das unterschiedliche Algorithmen kombiniert.

Die Existenz von falschen Werten in den Aufzeichnungen von Lastkurven aus der Vergangenheit beeinflusst die Präzision der Ergebnisse einer Lastprognose. In dieser Arbeit werden deshalb Methoden der „backward and forward second order difference“ zur Lokalisierung der falschen Werte eingeführt. Um den wahren Wert der falschen Daten zu schätzen wird eine quadratische Regression anwendet. Die Untersuchungen zeigen, dass dieses Vorgehen falsche und abnormale Werte, die z.B. durch Kurzschlüsse entstanden sind, erlaubt effizient zu detektieren.

In dieser Arbeit wird eine Lastkurve als Summe zweier Kurven betrachtet: eine Haupt-Lastkurve und eine um einen konstanten Wert schwingende Kurve. Die Haupt-Lastkurve wird durch die Methode der gewichteten kleinsten quadratischen Abweichung ermittelt. Die folgenden Lastprognose-Methoden werden jeweils mit und ohne geglättete Werte verwendet. Hierbei zeigt sich, dass mit den geglätteten Werten eine höhere Genauigkeit erreicht wird.

Für das Lastprognose-Problem wird in dieser Arbeit der „regression-tree-algorithm“ verwendet. Dieser Algorithmus kann die Daten automatisch einstufen und einen Wert für jeden Baumknotenpunkt zuweisen, ohne die Eigenschaften der einzelnen Lasten zu kennen. Das Resultat des Algorithmus wird in einfacher Form von "if... then" angegeben. Obgleich der ursprüngliche Ansatz dieser Regression darin besteht ohne detaillierte Informationen über die Lasten auszukommen, zeigt sich, dass Zusatzinformationen die Prognoseergebnisse verbessern. Deshalb werden dem Regressionsbaum entsprechend den Erfahrungen einige spezielle

Verfahren hinzugefügt. Eine Untersuchung vergleicht die dargestellte „regression-tree-algorithm" Methode mit dem ANN-Algorithmus (künstliche neuronale Netze) und belegt seine Überlegenheit.

Diese Arbeit stellt eine Prognose-Methode vor, die auf dem Ansatz der „support vector machine" basiert. Dabei werden die Beispieldaten effizient zu Clustern zusammengefasst, d.h. ähnliche Daten bilden einen Block. Mit der Methode des Entscheidungsbaums wird entschieden, in welchen Cluster die Eingangsdaten gehören. Um die tägliche Last vorauszusagen wird die „support vector regression" verwendet, da ihr ein einfaches mathematisches Modell zugrunde liegt, die Rechenzeiten sehr gering und die strukturellen Risiken klein sind.

Es werden zwei Methoden vorgestellt, um das Problem der Prognose für Feiertage und anomale Tageslasten zu lösen: „holiday regression tree" und „imaginary load method". Beispiele für Feiertags-Last-Prognosen belegen die Durchführbarkeit der beiden genannten Methoden.

Die vorliegende Thesis schlägt vor, den Regressionsbaum mit der „support vector machine"- zu kombinieren, um die Vorteile beider Algorithmen zu nutzen und deren Nachteile zu vermeiden. Als erstes wird der Regressionsbaum erstellt. Wenn die angenommene Last in einen Bereich mit einer großen Zahl ähnlicher Proben und einer sehr niedrigen Streuung fällt, kann der Ausgangswert als endgültige Aussage für die Prognose genommen werden. Andernfalls wird die support vector machine angewendet, um das Verhalten der Proben im gleichen Knotenpunkt zu analysieren.

Es ist stark von den jeweiligen Bedingungen abhängig, welche Methode die besten Ergebnisse liefert. In der Arbeit wird deshalb eine kombinierte Methode vorgestellt, um mehrere unterschiedliche Lastprognose-Algorithmen einzusetzen und den Nutzen aus den jeweils besten zu ziehen. Dazu werden drei Tabellen dem Benutzer zur Verfügung gestellt, in denen er sein zusätzliches Wissen über die betrachteten Lasten einbringen kann. Somit können Benutzer den vorgestellten Algorithmus, der verschieden Methoden miteinander verbindet, leicht anwenden ohne eine Änderung der Programme selbst vornehmen zu müssen. Dieser Ansatz erleichtert die Anpassung an unterschiedliche Prognose-Aufgaben in verschiedenen Energieversorgungsnetzen.

List of Symbols and Abbreviations

ABBREVIATIONS

ANN(NN)	Artificial Neural Network
ARIMA	Auto-Regressive Integrated Moving Average
ARMA	Auto-Regressive Moving Average
ARMAX	ARIMA with eXogenous Variables
BSOD	Backward Second Order Difference
CART	Classification And Regression Tree
EP	Evolutionary Programming
FARMAX	Fuzzy ARMAX
FSOD	Forward Second Order Difference
KKT	Karush-Kuhn-Tucker
LIBSVM	LIBrary for Support Vector Machines
MAPE	Mean Absolute Percentage Error
RT	Regression Tree
RTSVM	Regression Tree Support Vector Machine
SMO	Sequential Minimal Optimization
STLF	Short-Term Load Forecasting
SVM	Support Vector Machine
SVR	Support Vector Regression
THI	Temperature-Humidity Index
WCI	Wind Chill Index

SYMBOLS

$\nabla f(\alpha)$	Gradient of $f(\alpha)$
a,b,c	Coefficients of $L(t) = at^2 + bt + c$
$\mathbf{A}$	Adjacency matrix
AVE	Measurement of the possible error
AVE_{max}	Upper limitation of the average error of past forecasting
B	Segment of the historical training
$\boldsymbol{B}$	Working set
BB	Member of the historical training segment sequence

$\boldsymbol{BB}$	Sequence of historical training segments
C	Punishment constant of the slack errors
$CONF$	Confidence of the forecasted result
D	Historical training data column
$d(t)$	Distance from t to the most distant predictor
$Data1$	Dataset to store all the data to be clustered
$Data2$	Dataset to store the feasible clusters
Dd	Day number in a column
$DDay_{max}$	Upper limitation of the day difference
DEV	Standard deviation
DEV_{max}	Upper limitation of node sample standard deviation
DHU	Average humidity difference
DL	Increment of loads
DTH	Highest temperature difference
DTL	Lowest temperature difference
E	Past forecasting absolute percentage errors
ER	Error of forecasting
$f(\alpha)$	Objective function for LIBSVM
HU	Average humidity of the sample day
IL	Imaginary load
$k(\mathbf{x}_i, \mathbf{x}_j)$	Kernel function
k_L	Left division of a node
k_R	Right division of a node
l	Distance between a point and the cluster centre
l'	Relative distance between a point and the cluster centre
$L(i)$	Load of the point i in the load curve
$L(t)$	Quadratic regression formulation
$\ddot{L}_{\mathrm{b}}(i)$	Backward second order difference of point i
$\ddot{L}_{\mathrm{f}}(i)$	Forward second order difference of point i
M	Forecasting method code
$\boldsymbol{N}$	Non-working-set
ND	Node notation

N_{min}	Lower limitation of node sample number
N_l	Lower limitation of the member number in a cluster
N_s	Upper limitation of the isolated points
N_t	The t^{th} node of a regression tree
N_u	Upper limitation of the member number in a cluster
O	Cluster sphere
Q	Coefficient matrix of the quadratic minimal optimization
q	Dimension of the working set
R	Radius of the sphere
RL	Real load
r_m	Radius of the cluster sphere for cluster m
$RNmin$	Lower limitation of the result number
S	Segment of a load curve
SDR	Split dispersion ratio
t	Focused regression point
T	Node that is regarded as a complete cluster
TEI	Total error indicator
TEI_{max}	Upper limitation of total error indicator
TH	Highest temperature of the sample day
THP	Highest temperature of the sample day's previous day
t_i	Neighbours of t as defined by the span
TL	Lowest temperature of the sample day
$TOTAL\_CONF$	Total confidence
$\mathbf{V}$	Difference interval
v_1	Lower limitation of difference interval
v_2	Upper limitation of difference interval
WE	Weekend property
W_i	Weight of the i^{th} forecasting result in the final
$\mathbf{x}$	Independent variable vector
y	Response variable
Y	Historical training data row
$\bar{y}_k$	Average output value of node k

α_i, α_i^*, μ_j	Lagrange multiplier
α^k	Solution of the quadratic minimization problem
γ	Parameter of kernel $K(\mathbf{x}_1, \mathbf{x}_2) = e^{-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|^2}$
ε	Permitted error
ε_i, ε_i^*	Slack errors for the i^{th} training point
λ	Deviation coefficient
ρ	Maximum limitation of split dispersion ratio
w_i	The i^{th} point weight in quadratic regression
ϕ	Nonlinear transformation function
$\begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix}$	Reorganization of Q

Literature

[1] G. Gross, F. D. Galiana, ‘Short-term load forecasting’, Proceedings of the IEEE, 1987, 75(12), 1558 - 1571

[2] A.D. Papalexopoulos, T.C. Hesterberg, ‘A Regression Based Approach to Short Term Load Forecasting’, IEEE Transactions on Power Systems, 1990, 5(1), 40 - 45

[3] N. Amjady, ‘Short-term hourly load forecasting using time-series modeling with peak load estimation capability’, IEEE Transactions on Power Systems, 2001, 16(3), 498 - 505

[4] W. Christianse, ‘Short Term Load Forecasting Using General Exponential Smoothing’, IEEE Transactions on PAS, 1971, 900 - 910

[5] S.A. Villalba, C.A. Bel, ‘Hybrid demand model for load estimation and short-term load forecasting in distribution electrical systems’, IEEE Transactions on Power Delivery, 2000, 15(2), 764 - 769

[6] J. Yang, H. Cheng, ‘Application of SVM to power system short-term load forecast’, Power System Automation Equipment China, 2004, 24(4), 30 - 32

[7] Y. Li, T. Fang, E. Yu, ‘Short-term electrical load forecasting using least squares support vector machines’, International Conference on Power System Technology, 2002, 230 - 233

[8] K.J. Hwan, G.W. Kim, ‘A short-term load forecasting expert system’, Proceedings of The Fifth Russian-Korean International Symposium on Science and Technology, 2001, 112 - 116

[9] A.A. Desouky, M.M. Elkateb, ‘Hybrid adaptive techniques for electric-load forecast using ANN and ARIMA’, IEE Proceedings of Generation, Transmission and Distribution, 2000, 147(4), 213 - 217.

[10] K.H. Kim, H.A.Youn, Y.C. Kang, ‘Short-term load forecasting for special days in anomalous load conditions using neural networks and fuzzy inference method’, IEEE Transactions on Power Systems, 2000, 15(2), 559 - 565

[11] R.F. Engle, C. Mustafa, J. Rice, ‘Modeling peak electricity demand’, Journal of Forecasting, 1992, 11, 241 - 251

[12] O. Hyde, P.F. Hodnett, ‘An Adaptable automated procedure for short-term electricity load forecasting’, IEEE Transactions on Power Systems, 1997,12, 84 - 93

[13] S. Ruzic, A. Vuckovic, N. Nikolic, ‘Weather sensitive method for short-term load forecasting in electric power utility of Serbia’, IEEE Transactions on Power Systems, 2003, 18, 1581 - 1586

[14] T. Haida, S. Muto, ‘Regression based peak load forecasting using a transformation technique’, IEEE Transactions on Power Systems, 1994, 9, 1788 - 1794

[15] W. Charytoniuk, M.S. Chen, P.Van Olinda, ‘Nonparametric regression based short-term load forecasting’, IEEE Transactions on Power Systems, 1998, 13, 725 - 730

[16] J.Y. Fan, J.D. McDonald, ‘A real-time implementation of short – term load forecasting for distribution power systems’, IEEE Transactions on Power Systems, 1994, 9, 988 - 994

[17] M.Y. Cho, J.C. Hwang, C.S. Chen, ‘Customer short-term load forecasting by using ARIMA transfer function model’, Proceedings of the International Conference on Energy Management and Power Delivery, EMPD, 1995, 1, 317 - 322

[18] H.T.Yang, C.M. Huang, C.L. Huang, ‘Identification of ARMAX model for short-term load forecasting, An evolutionary programming approach’ IEEE Transactions on Power Systems, 1996, 11, 403 - 408

[19] H.T. Yang, C.M. Huang, ‘A new short-term load forecasting approach using self-organizing fuzzy ARMAX models’, IEEE Transactions on Power Systems, 1998, 13, 217 - 225

[20] M. Peng, N.F. Hubele, G.G. Karady, ‘Advancement in the application of neural networks for short-term load forecasting’, IEEE Transactions on Power Systems, 1992, 7, 250 - 257

[21] E. A. Feinberg, ‘Load Forecasting’, http://www.ams.sunysb.edu/~feinberg/public/lf.pdf

[22] A.D. Papalexopoulos, S. Hao, T.M. Peng, ‘An implementation of a neural network based load forecasting model for the EMS’, IEEE Transactions on Power Systems, 1994, 9, 1956 - 1962

[23] A. Khotanzad, et al., ‘ANNSTLF - A neural-network-based electric load forecasting system’, IEEE Transactions on Neural Networks, 8 (1997), 835 - 846

[24] A. Khotanzad, R.A.Rohani, D. Maratukulam, ‘ANNSTLF – Artificial neural network short-term load forecaster - Generation three’, IEEE Transactions on Neural Networks, 1998, 13, 1413 - 1422

[25] B.J. Chen, M.W. Chang, C.J. Lin, ‘Load forecasting using support vector machines: A study on EUNITE competition 2001’, http://www.csie.ntu.edu.tw/˜cjlin/libsvm. 2002

[26] T.W.S. Chow, C.T. Leung, ‘Nonlinear autoregressive integrated neural network model for short-term load forecasting’, IEE Proceedings, Generation, Transmission and Distribution, 1996, 143, 500 - 506

[27] S.E. Skarman, M. Georgiopoulous, ‘Short-term electrical load forecasting using a fuzzy ARTMAP neural network’, Proceedings of the SPIE, (1998), 181 - 191.

[28] Y. He, et al. ‘Similar Day Selecting Based Neural Network Model and its Application in Short-Term Load Forecasting’, Machine Learning and Cybernetics Proceedings of 2005 International Conference, 18 - 21 Aug. 2005, 4760 - 4763

[29] K.L. Ho et al, ‘Short-term load forecasting of Taiwan power system using a knowledge based expert system’, IEEE Transactions on Power Systems, 1990, 5, 1214 - 1221

[30] S. Rahman, O. Hazim, ‘Load forecasting for multiple sites: development of an expert system-based technique’, Electric Power Systems Research, 1996, 39, 161 - 169

[31] S.J. Kiartzis, A. G. Bakirtzis, ‘A Fuzzy expert system for peak load forecasting. Application to the Greek power system’,10th Mediterranean Electrotechnical Conference, 2000, 3, 1097 - 1100

[32] V. Miranda, C. Monteiro, ‘Fuzzy inference in spatial load forecasting’, Power Engineering Winter Meeting, IEEE, 2000, 2, 1063 - 1068

[33] S.E. Skarman, M. Georgiopoulous, ‘Short-term electrical load forecasting using a fuzzy ARTMAP neural network’, Proceedings of the SPIE, 1998, 181 - 191

[34] T. Hastie, R. Tibshirani, J. Friedman, ‘The elements of statistical learning: data mining, inference, and prediction’, Springer, New York, 2001

[35] J. Han, M. Kamber, ‘Data Mining: Concepts and Techniques’, Morgan Kaufmann, San Francisco, California, 2001

[36] http://www.dwd.de/de/de.htm

[37] Y. Li, T. Fang, ‘Wavelet and support vector machines for short – term electrical load forecasting’, Proceedings of the International Conference on Wavelet Analysis and its Applications, 2003, 1, 399 - 404.

[38] http://www.nensel-kalender.de/

[39] L. Breimann, J. H. Friedman, R. A Olshen, ‘Classification and regression trees’, Chapman & Hall, New York, 1984

[40] V. Vapnik, ‘Statistical Learning Theory’, Wiley, New York, 1998

[41] N. B. Karayiannis, ‘An axiomatic approach to soft learning vector quantization and clustering’, IEEE Trans on Neural Networks, 1999, 10, 1015 - 1019

[42] A. Dubinsky, T, Elperin, ‘A method for calculating a load curve using average values of load over time intervals’, Fuel and Energy Abstracts, 1993, 39, 32 - 35

[43] R. Liang, C. Cheng, ‘Short-term load forecasting by a neuro-fuzzy based approach’, International Journal of Electrical Power and Energy Systems, 2002, 24, 103 - 111

[44] B. Satish, et al., ‘Effect of temperature on short term load forecasting using an integrated ANN’. Electric Power Systems Research, 2002, 72, 95 - 101

[45] S.E. Papadakis, J.B. Theocharis, A.G. Bakirtzis, ‘A load curve based fuzzy modeling technique for short-term load forecasting’, Fuzzy Sets and Systems, 2003, 52, 279 - 303,

[46] H.A. Kamal, M.H. Eassa, ‘Solving curve fitting problems using genetic programming’, Electrotechnical Conference, 7 - 9 May 2002, MELECON, 316 - 321

[47] C. Chuang, J. Hwa-Hsia, J. Tsong, ‘A soft computing technique for noise data with outliers,’ IEEE International Conference on Networking, Sensing and Control, 2004, 1171 - 1176

[48] H. Mori, A. Yuihara, ‘Deterministic Annealing Clustering for ANN-Based Short-Term Load Forecasting’, IEEE Transactions on Power System, 2001, 16(3), 545 - 551

[49] H. Mori, and N. Kosemura, ‘Optimal regression tree based rule discovery for short-term load forecasting’, Proceedings of the IEEE Power Engineering Society Transmission and Distribution Conference, 2001, 1, 421 - 426

[50] H.M. Al-Hamadi, S.A. Soliman, ‘Long-term/mid-term electric load forecasting based on short-term correlation and annual growth’, electric power systems research, 2005,74, 353 - 361

[51] T. Cooke, ‘Two variations on Fisher's linear discriminant for pattern recognition. Pattern Analysis and Machine Intelligence’, IEEE Transactions on Power Systems, 2002, 24(2), 268～273

[52] C.-C. Chang, C.-J. Lin, ‘LIBSVM: a library for support vector machines’, http://www.csie.ntu.edu.tw/˜cjlin/libsvm. 2002

[53] D. Zhao, ‘Support vector machine approach for short term load forecasting’, Proceedings of the Chinese Society of Electrical Engineering, 2002, 22, 26 - 30,.

[54] I. Erkmen, A. Oezdogan, ‘Short Term Load Forecasting Using Genetically Optimized Neural Network Cascaded With a Modified Kohonen Clustering Process’, Proceedings of the 12th IEEE International Symposium on Intelligent Control, 1997, 1, 107 - 112

[55] A. Ben-Hur, D. Horn, H. T. Aiegelmaann, ‘A Support Vector Clustering method’, Proceedings of 15th International Conference on Pattern Recognition, 2000, 2, 724 - 727

[56] J. Chiang, P. Hao, ‘A New Kernel-Based Fuzzy Clustering Approach: Support Vector Clustering With Cell Growing’, IEEE Transactions On Fuzzy Systems, 2003,11(4), 518 - 527

[57] http://www.cs.tau.ac.il/~borens/courses/ml/cluster.html

[58] A. Sfetsos, ‘Short-term load forecasting with a hybrid clustering algorithm’, IEE Proceedings Generation, transformation distribution, 2003, 150(3), 257 - 261

[59] D. W. Bunn,’Forecasting Loads and Prices in Competitive Power Markets’, Proceedings of the IEEE, 2000, 88(2), 163 - 169

[60] R. E. Abdel_Aal, ‘Short-Term Hourly Load Forecasting Using Abductive Networks’, IEEE transactions on power systerms, 2004, 19(1), 164 - 173

[61] Y. Yohannes, P. Webb, ‘Classification and regression tress, a user manual for identifying indicators of vulnerability to famine and chronic food insecurity’, http://www.ifpri.org/pubs/microcom/micro3.pdf

[62] A.G.Bakirtzis, et al., ‘A neural network short-term load forecasting model for the Greek power system’, IEEE Transactions on Power Systems, 1996, 11, 858 - 863

[63] N. J. Nilsson, ‘Artificial intelligence: a new synthesis’, Morgan Kaufmann, San Francisco, California, 1998

[64] J. Yang, J. Stenzel, ‘Application of two-dimensional Support Vector Machine in Short-term Load Forecasting’, IEEE St.Petersburg PowerTech, 27 - 30 June, 2005, 786 - 789

[65] J. Yang, J. Stenzel, ‘Historical Load Curve Correction for Short-term Load Forecasting’, 7th International Power Engineering Conference, 29 Nov - 02 Dec 2005, Singapore, page still unknown

[66] J. Yang, J. Stenzel, ‘Short-term Load Forecasting With Increment Regression Tree’, Journal of Electric Power Systems Research, accepted

[67] J. Yang, J. Stenzel, ‘Kernel-based clustering for short-term load forecasting’, IEE Proceedings of Generation, Transmission and Distribution, submitted

Lebenslauf

Persönliche Daten

Name	Jingfei Yang
Geburtsdatum	12. Februar 1974
Geburtsort	Beijing, China
Staatsangehörigkeit	chinesisch
Familienstand	Verheiratet

Schulbildung

1979 – 1987	Grundschule und Mittelschule, Baoding
1987 – 1990	Gymnasium, Beijing

Hochschulstudium

1990 – 1994	Shanghai Hochschule für Engergie Versorgung, Shanghai Abschulss: B. Sc. in Elektrotechnik
1995 – 1998	Huabei Universität für Energieversorgung Abschulss: M. Sc. in Elektrotechnik

Berufsweg

1994 – 1995	Mitarbeiterin im Ingenieurbüro Qiji, Beijing
1998 – 2004	Mitarbeiterin an der Shanghai Jiaotong Universität, Shanghai
Seit März 2004	Wissenschaftliche Mitarbeiterin am Institut für Elektrische Energiesysteme, Technische Universität, Darmstadt, Deutschland