2018

Innovative Machine Learning Methods for Demand Management in Smart Grid Market

Xishun Wang
University of Wollongong

Innovative Machine Learning Methods for Demand Management in Smart Grid Market

Xishun Wang

Supervisor:
Prof. Minjie Zhang
Co-supervisor:
Dr. Fenghui Ren

This thesis is presented as required for the conferral of the degree:

Doctor of Philosophy

The University of Wollongong
School of Computing and Information Technology

June 14, 2018

Declaration

I, Xishun Wang, declare that this thesis submitted in fulfilment of the requirements for the conferral of the degree Doctor of Philosophy, from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

Xishun Wang

June 14, 2018

Abstract

Smart Grid has been widely acknowledged as an efficient solution to the current energy system. Smart Grid market is a complex and dynamic market with different types of consumers and suppliers under an uncertain environment. An efficient management of Smart Grid market can benefit Smart Grid in multiple aspects, including reducing energy cost, improving energy efficiency and enhancing network reliability. This thesis focuses on improving demand management in Smart Grid market through developing innovative machine learning methods.

Specifically, this thesis

1. studies Smart Grid market and proposes an intelligent broker model for Smart Grid market management. In the proposed broker designs, the challenges that a smart broker faces in Smart Grid market are comprehensively considered, and an adaptive and systematic model is constructed to surmount the challenges. Experimental results demonstrate that the proposed broker model can not only make much profit but also keep a good supply-demand balance. Through the study of broker models, two empirical laws in Smart Grid market have been discovered i.e. Law 1: profit margin shrinks in a competitive market environment and Law 2: the imbalance rate of supply demand increases when a market environment is more competitive.

2. studies how to accurately predict power demand of Smart Grid considering customer behaviors. A sparse Continuous Conditional Random Fields(sCCRF) model is proposed to explore customer behaviors. A load forecasting method through learning customer behaviors (LF-LCB) is proposed to effectively predict the demand of Smart Grid. Experimental results show the superiority of LF-LCB to other methods. Learning customer behaviors to aggregate customers in fact can be a general method to assist decision makings towards various customers in a complex market environment. Evaluation results also indicate that the proposed sCCRF is effective in feature selection and prediction. Thus, sCCRF can also be applied in other related research fields.

3. studies effective renewable energy prediction methods through deep learning. A Deep Regression model for Sequential Data (DeepRSD) is proposed for renewable energy prediction. An alternative dropout is also proposed to effectively improve the generalization of DeepRSD. According to the experimental results, DeepRSD shows two major advantages over other known methods. 1) DeepRSD can simultaneously represent step features and temporal information. 2) DeepRSD has a strong nonlinear presentation capacity to achieve a good performance without feature engineering. As renewable energy prediction is a regression problem on sequential data, DeepRSD can be an effective solution to problems of this kind.

4. investigates state-of-the-art time-series prediction models and proposes two new effective models for time-series prediction, applying to demand prediction in Smart Grid market. The first model is Sparse Gaussian Conditional Random Fields (SGCRF) on top of Recurrent Neural Networks (RNN), short as CoR. CoR integrates the advantages of RNN and SGCRF and shows excellent performance in demand prediction. The second model is CoR with attention (CoRa). CoRa further improves CoR by introducing an attention mechanisms for RNN. CoR can effectively make use of temporal correlations, nonlinearities and structured information in time-series prediction. CoRa can further improve CoR in terms of prediction accuracy via the attention mechanism. With sufficient experiments and analysis, this thesis concludes that CoR and CoRa can be new effective models for time-series prediction in Smart Grid and broad domains.

In summary, this thesis proposes several effective machine learning methods to ameliorate demand management in Smart Grid market. The proposed machine learning methods not only contribute to effective demand management of Smart Grid market in practice, but also contribute to machine learning research, as they can be applied to broad domains.

Acknowledgments

First and foremost, I am deeply grateful to my supervisor Professor Minjie Zhang, who was very supportive and helpful in my Ph.D study. I feel so fortunate to have such a responsible supervisor to encourage me to excel in three and a half years. Minjie is kind to everyone in life while strict in work. She revised every word of my each manuscript for several times and explained patiently, which are valuable moments curved in my mind.

I also express my gratitude to my co-supervisor Dr. Fenghhui Ren for his research advice and discussions. Fenghui also helped me a lot to quickly accommodate to Australian life. I can recall the happy moments in his house on festivals.

Moreover, I would like to thank Associate Professor Lei Wang and Dr. Luping Zhou. Lei and Luping have got great passion in research and significantly influenced me. I have learned a lot from their paper reading seminars.

Additionally, I wish to thank all my friends and colleagues. Thanks for Jihang Zhang, Lei Niu, Yuchen Wang, Yan Kong, Ahmed Moustafa and Dien Tuan Le for their company in the lab.

I am also very thankful to my friends from IFIS team, just to name a few: Jan Kemper, Peter Beattie, Kel Magrath, Bob Colvin and Andrea Kincoff. There are also many international friends from IFIS, enriching my life in Australia. Last but not least, friends from badminton club also brings a lot of fun in my spare time.

Moreover, I would like to express my gratitude to IPTA and UPA scholarships from University of Wollongong to support my Ph.D study. I am also grateful to the ARC Discovery Project for support of my research activities in my Ph.D project. I am also thankful to the friendly and relaxing environment in Australia, in which I studied and experienced.

Last but not least, I deeply thank my parents and my girl friend Qiushi Ning for their unconditioned love on me. With their support and encouragement, I could overcome the difficulties in my Ph.D study.

Publications

This section lists publications/manuscripts completed in this Ph.D study.

Referred journal articles:

1. Xishun Wang, Minjie Zhang and Fenghui Ren, A Hybrid-learning Based Broker Model for Strategic Power Trading in Smart Grid Markets. *Knowledge-Based Systems*, Vol. 119, pp. 142-151, 2017.

Referred conference papers:

2. Xishun Wang, Minjie Zhang and Fenghui Ren, Sparse Gaussian Conditional Random Fields on Top of Recurrent Neural Networks. Accepted by *Association for the Advancement of Artificial Intelligence (AAAI 2018)* as oral presentation.

3. Xishun Wang, Minjie Zhang and Fenghui Ren, Load Forecasting through Customer Behaviour Learning Using L1-Regularized Continuous CRF. In *Proceedings of 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS2016)*, pp. 817-825, Singapore, 2016. (The Best Student Paper Nomination)

4. Xishun Wang, Fenghui Ren, Chen Liu and Minjie Zhang, L1-Regularized Continuous Conditional Random Fields. *The 14th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2016)*, Lecture Notes in Artificial Intelligence, Springer, pp. 793-804, 2016.

5. Xishun Wang, Minjie Zhang, Fenghui Ren, and Takuyaki Ito. Gongbroker: A Broker Model for Power Trading in Smart Grid Markets. In *Proceedings of International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2015)*, vol. 2, pp. 21-24, 2015.

Papers under review:

6. Xishun Wang, Minjie Zhang and Fenghui Ren, Effective Load Forecasting through Learning Customer Behavior. Submitted to *IEEE Transactions on Knowledge and Data Engineering* in September, 2017. Revised version submitted *IEEE Transactions on Knowledge and Data Engineering* in February, 2018.

7. Xishun Wang, Minjie Zhang and Fenghui Ren, An Integration of Recurrent Neural Networks and Gaussian Graphical Models for Time-series Prediction. Submitted to *IEEE Transactions on Neural Networks and Learning Systems* in February, 2018.

8. Xishun Wang, Minjie Zhang and Fenghui Ren, DeepRSD: A Deep Regression Method on Sequential Data. Submitted to *the 15th Pacific Rim International Conference on Artificial Intelligence* (PRICAI 2018) in February, 2018.

List of Abbreviations

The following abbreviations will be used in this thesis. Some special abbreviations will be defined when they are first used.

ARIMA	Auto-Regressive Integrated Moving Average
RNN	Recurrent Neural Network
LF-LCB	Load Forecasting through Learning Customer Behavior
CRF	Conditional Random Fields
CCRF	Continuous Conditional Random Fields
sCCRF	sparse Continuous Conditional Random Fields
RSD	Regression on Sequential Data
DeepRSD	Deep Regression model on Sequential Data
SGCRF	Sparse Gaussian Conditional Random Fields
CoR	Sparse Gaussian Conditional Random Fields on Top of Recurrent Neural Networks
CoRa	CoR with attention

[This page is intentionally left blank]

Contents

List of Figures

Chapter 1

Introduction

Smart Grid is a mixture of entities of various types of energy suppliers and consumers. It has been widely acknowledged as an efficient solution to modern energy systems. Smart Grid market is a complex and dynamic market that is very challenging to manage. This thesis focuses on solving challenging problems in demand management of Smart Grid market. Different from the traditional grid market, Smart Grid market shows three distinctive features. 1) There are many types of customers in Smart Grid market. 2) Customers in Smart Grid can be adaptive to the dynamic market. 3) Smart Grid market is sensitive to a range of external factors, such as time and weather conditions. Due to these features, it is very challenging to manage Smart Grid market.

An efficient demand management of Smart Grid market is of great significance in Smart Grid research. It can maintain a good supply-demand balance, in order to alleviate energy wasting and reduce emissions to the environment. This thesis focuses on the studies of demand management and develops innovative machine learning methods to improve demand management in Smart Grid market. The rest of this chapter is organized as follows. Section 1.1 introduces the background of Smart Grid. Section 1.2 briefly discusses demand management in Smart Grid market. Section 1.3 depicts the research issues and objectives of this thesis. Section 1.4 summaries the contributions of this thesis. Finally, Section 1.5 shows the structure of this thesis.

1.1 Smart Grid

A Smart Grid is a modern grid system that introduces analog or digital technology to collect, communicate and process information, such as information related to the patterns of of consumers and suppliers, in an automatic mode, in order to enhance sustainability, reliability and efficiency of the production and distribution of energy [oE12].

Figure 1.1 illustrates the components of Smart Grid, including traditional users such as householders and offices, a range of energy producers such as wind and solar energy generators. Smart Grid brings revolutions to the traditional grid. It has the following distinctive features, which are the advantages over the traditional power grid.

(1) Smart Grid can increase the reliability of the network.

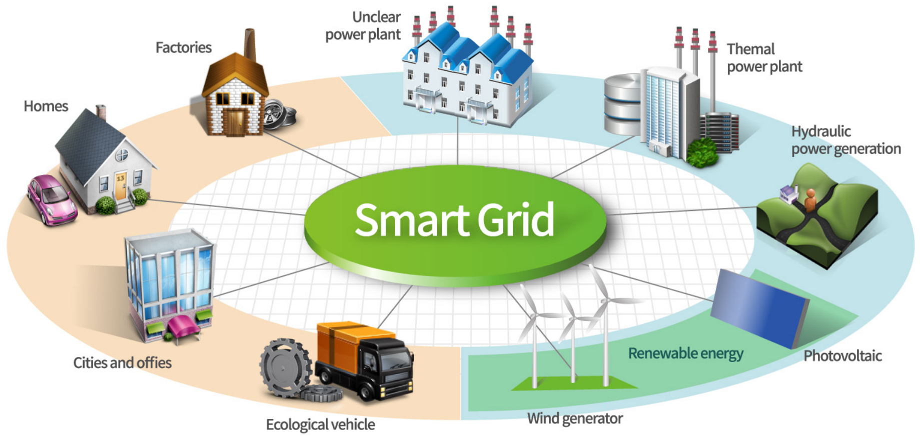

Figure 1.1: An illustration of Smart Grid. This figure is adapted from http://www.editiontruth.com/smart-grid-security-market-technological-progress-energy-power-industry-trends-2025/.

Smart Grid utilizes current technologies to result in fault detection. Besides, the network can also heal itself without technicians' interventions. The fault detection and self-healing of Smart Grid can ensure a reliable power supply, and increase robustness to natural disasters or attacks [Sia14].

(2) Smart Grid can be more flexible in network topology.
The transmission and distribution infrastructure in Smart Grid can handle bi-directional energy flows. Therefore, energy can easily flow between suppliers and consumers in two directions. Besides, energy suppliers and consumers can easily migrate between grids [SS12]. As a result, the network topology of Smart Grid can be dynamic and flexible.

(3) Smart Grid can improve the efficiency of energy.
The demand side management can enable efficient use of power energy. Possible solutions include demand balancing, demand adjustment, peak-usage reduction, and time-of-use pricing. These solutions can reduce energy transmission and distribution, and lead to efficient utilization of generators.

(4) Smart Grid can supply a sustainable solution for energy generation.
The flexible Smart Grid allows the penetration of renewable energy sources, which have large but variable amount. Typical sustainable energy includes hydroelectric power, wind power and solar power [JMI13].

(5) Smart Grid can offer a friendly power market environment.
Smart Grid supplies convenient communications between consumers (in terms of their intention-to-use) and supplier (in terms of energy price), and enables both consumers and suppliers to be adaptive in decision-makings in Smart Grid market.

The innovative features of Smart Grid bring about enormous research opportunities. There has been much work on the hardware infrastructure [BI12, FZXC11], the power flow control [Age14], and the market control and decision making. This thesis focuses on demand management in Smart Grid market.

1.2 Demand Management

In a broad view, energy demand management [SS12] involves making good use of various energy resources, efficient energy resource management, reliable energy supply, efficient renewable energy systems, independent energy transmissions, etc. Demand management in Smart Grid market has to consider a range of factors, which can be technical, social and organizational solutions to result in efficient energy consumption.

1.2.1 Significance of demand management

Demand management in Smart Grid market can significantly contribute to the e-conomy, environment and network reliability.

- ***Economy***
  From the perspective of customers, they can get the freedom to rebate the tariff price and can achieve savings by reducing peak hour consumptions. The overall change in customers will result in reduction on peak hour demand, as to reduce generation units. In the end, the cost of energy can be reduced in Smart Grid market.

- ***Environment***
  Efficient demand management can help to balance local supply and demand, so as to reduce the transmission cost. The efficient distribution of renewable energy is environment-friendly , implying a promising way for sustainable energy.

- ***Network reliability***
  By reducing the peak consumptions, the Smart Grid can reduce risk of huge power load. Besides, the reduced peak usage will lead to reduced power generation, and the local balance can result in efficient transmission. Generally, demand management can decrease the burden of the grid system and enhance network reliability.

1.2.2 Challenging issues in demand management

In traditional grid systems, the power flows from generators to consumers, and consumers are passive. Smart Grid significantly differs from traditional grid systems in that power flows in two directions and customers can be active and adaptive. Besides, there are more types of customers in Smart Grid than those in traditional grids. Due to the differences from traditional grids, several challenging issues arise in demand management in Smart Grid market.

Issue 1: It is challenging to balance supply and demand in Smart Grid market.
For power trading in Smart Grid market, there is a wholesale and a retail market. It is important to keep a balance of supply (from the wholesale market) and demand (from the retail market). As customers can adapt their behaviors dynamically to the changes in the market, such as changing tariffs and reducing peak hour usages, it is challenging to effectively maintain a good supply-demand balance in Smart Grid market.

Issue 2: It is difficult to accurately predict power demand.
Demand prediction is a fundamental problem for the management of Smart Grid market. Due to the various types of customers with irregular power consumptions or productions, it is very difficult to accurately predict the power demand of a grid sytem.

Issue 3: It is hard to accurately predict the production of renewable energy.
The fluctuation of the production of renewable energy is a critical factor that causes the imbalance of supply and demand. As renewable energy can vary significantly under the influence of weather conditions, it is hard to accurately predict its production.

1.3 Research Questions and Objectives

The purpose of this thesis is to solve the challenges in demand management in Smart Grid market by developing innovative machine learning methods. Based on the challenging issues in Subsection 1.2.2, four concrete research questions in this Ph.D. study are identified.

(1) How to keep a good supply-demand balance in Smart Grid market?
In Smart Grid market, the demand of customers fluctuates because customers are adaptive to the changes in Smart Grid market. In the supply side, the production of renewable energy changes with weather conditions. Therefore, it is challenging to keep the balance of supply and demand in Smart Grid market. Brokers are widely introduced to manage the complex and dynamic market [KCRW14]. A good broker model should maintain a good supply-demand balance and make profit to survive. An intelligent broker model can introduce supervised learning to predict the demand

of customers and use reinforcement learning for market managements. Therefore, it is possible to design a broker model to balance supply and demand in Smart Grid market.

(2) How to utilize customer behaviors for effective demand prediction?
In Smart Grid, there are different types of customers exhibiting various behaviors. Two instances are given here. Example 1: more and more householders have equipped photovoltaic power generation systems, which may lead to variable power usages under the influence of weather factors [TB03], such as cloudiness and humidity. Example 2: some customers with storage capacity may recharge or use their power according to the varying prices at different times of the day (Time-of-Use [PD11], a pricing mechanism used in Smart Grid markets). Intuitively, customers can be aggregated by their different behaviors. Accordingly, decision-makings based on the aggregated customer groups can be more effective than the decisions based on the whole grid system or individual customers. Therefore, it is meaningful to develop an effective method to discover customer behaviors for an accurate demand prediction.

(3) How to improve renewable energy prediction via deep learning?
In Smart Grid market, renewable energy prediction plays a critical role for decision makings. There has been much work on how to accurately predict energy production. However, the performances of existing models rely heavily on feature engineering. Recently, deep learning is under fast development and shows significant success in computer vision, natural language processing etc. Deep neural networks gain strong capacity to represent non-linear features, usually outperforming human designed feature engineering. It is possible to utilize deep neural networks to improve the prediction accuracy of energy production.

(4) How to build a new effective time-series model to improve demand prediction?
Demand prediction is intrinsically a multi-step time-series prediction problem. In this kind of problems, there are three key factors that affect the prediction accuracy, which are temporal correlations in the observed data sequence, nonlinearities in features and structured information in the output. In current time-series prediction models, none of them could effectively make use of the three types of information. This thesis tries to design a new model that can utilize the three types of information to further improve demand prediction in Smart Grid market.

To provide solutions to the above four research questions, this Ph.D. study establishes the following four research objectives:

Objective 1: to design an effective broker model in Smart Grid market, which can both make large profit and maintain a good supply-demand balance.

Objective 2: to utilize an efficient machine learning method to explore customer behaviors in Smart Grid, in order to accurately predict power demand.

Objective 3: to propose an effective deep learning method to improve renewable energy prediction in Smart Grid market.

Objective 4: to propose a new time-series prediction model in theory to further improve demand prediction.

1.4 Contributions of This Thesis

Focusing on the four research questions and objectives, this thesis has the following four contributions.

(1) An effective broker model in Smart Grid market
To resolve the first research issue, an intelligent broker model, utilizing hybrid learning methods, is proposed for strategic power trading in Smart Grid markets. In the proposed design, the challenges that brokers face in Smart Grid markets are comprehensively considered, and an adaptive and systematic model is constructed to surmount the challenges in Smart Grid market. The proposed broker shows excellent performance on both profit making and supply-demand balancing. In this study, two empirical laws in Smart Grid market have been discovered, i.e. Law 1: profit margin shrinks in a competitive market environment; and Law 2: the imbalance rate of supply demand increases when a market environment becomes more competitive.

(2) An efficient machine learning method to explore customer behaviors
To resolve the second research issue, an sCCRF is proposed to explore customer behaviors. This study also proposes a load forecasting method through learning customer behaviors (LF-LCB), which utilized the proposed sCCRF to analyze customer behaviors by using the learned weights which can reflect different energy consumption patterns of various customers. Through this study, two conclusions are drawn: 1) Learning customer behaviors to aggregate customers can improve the prediction precision and lead to a reasonable computation cost. 2) The proposed sCCRF is an efficient learning tool with feature selection capacity.

This work can also potentially facilitate research in related domains. Learning customer behaviors to aggregate customers in fact can supply a general methodology to assist better decision making towards various customers in a complex market environment. This is worth further explorations in other market domains. Evaluation results also indicate that the proposed sCCRF is effective in feature selection and prediction. Thus, sCCRF can also be applied in other related research fields.

(3) A deep learning method for renewable energy prediction

To resolve the third research issue, a Deep Regression method for Sequential Data (DeepRSD) is proposed for renewable energy prediction. DeepRSD uses stacked bi-directional RNNs to represent the sequential data. Four conditions are explored to ensure a plausible training. An alternative dropout is also proposed to effectively improve the generalization of DeepRSD. DeepRSD is evaluated by two real-world solar energy prediction problems. DeepRSD shows two major advantages over other state-of-the-art methods. 1) DeepRSD can simultaneously represent step features and temporal information. 2) DeepRSD has strong nonlinear presentation capacity to achieve a good performance without feature engineering. Therefore, DeepRSD provides an effective solution for regressions on sequential data.

(4) A new time-series prediction model for demand prediction
For the second and third research issues, CoR is proposed. CoR, which integrates the advantages of RNN and SGCRF, is applied for demand prediction. Two training methods are designed for CoR, i.e. an end-to-end training and an alternative training. CoRa is also proposed, which improved CoR by introducing an attention mechanisms for RNN. Evaluation results show that CoR and CoRa can outperform other state-of-the-art methods in demand prediction.

CoR can effectively make use of temporal correlations, nonlinearities and structured information in multi-step time-series prediction. CoRa can further improve upon CoR in terms of prediction accuracy. With sufficient experiments and analysis, this thesis concludes that both CoR and CoRa can be new effective models in time-series prediction in other related fields.

1.5 Structure of This Thesis

The remaining chapters are organized as follows.

Chapter 2 reviews demand management in Smart Grid market and related machine learning methods. In particular, the reviews focuses on: the broker models in Smart Grid market, traditional demand prediction methods and emerging machine learning models for energy prediction.

Chapter 3 presents an hybrid-learning based intelligent broker model that can effectively surmount the challenges in Smart Grid market.

Chapter 4 proposes a machine learning method that can explore customer behaviors in Smart Grid market. This method can accurately predict the demand of a whole grid system.

Chapter 5 designs a deep learning method that can effectively predict energy production and power demand in Smart Grid market.

Chapter 6 proposes a new time-series model in theory for demand prediction.

Chapter 7 concludes the thesis with a summary of contributions and directions of future work.

Chapter 2

Literature Review

This chapter provides a thorough literature review for the identified research problems. Section 2.1 reviews the management of Smart Grid market with a focus on broker models. Section 2.2 discusses different demand prediction methods in Smart Grid market. Section 2.3 reviews renewable energy prediction methods. Section 2.4 discusses the advantages and shortcomings of existing time-series prediction models. Section 2.5 wraps up this chapter.

2.1 Management of Smart Grid Market

Smart Grid market includes a wholesale market and a retail market [KCRW14]. The wholesale market sells power to broker agents through auction. The hourly power of the next 24 hours is bid on price and amount by brokers. The retail market contains a variety of customers, such as general consumers, interruptible consumers, and even renewable energy producers, such as solar power systems and wind turbines. The broker agent interacts with a variety of customers in the retail market.

In the management of Smart Grid market, demand response and broker technology are widely used in both research and applications. In the following two subsections, major approaches of demand response and broker models are reviewed in detail.

2.1.1 Demand response

Demand Response refers to adapt usage of energy by end-use customers due to the changes of energy price. In a wider view, demand response can be regarded as strategies for efficient energy using in end-use customers and efficient energy distribution in grids [PD11, Sia14, JMI13].

Demand response benefits Smart Grid in multiple aspects. Demand response can be divided into three levels:

- **Customer's energy optimization:** The customer uses an intelligent strategy to use the electrical equipments with lower power consumptions.
- **Price-driven response:** Smart Grid provides a smart pricing mechanism to drive the using strategies of customers, leading to reduction of the peak usage in the grid system.

- **Network level response:** The grid system integrates the information of energy producers and consumers for efficient demand response.

In the following, literatures regarding to the three aspects are reviewed.

- **Customer's energy Optimization**
  Several research projects have been carried on for smart energy management for homes and buildings. With the objective of optimizing energy cost for users, Rad et al. proposed an autonomous energy management system on demand side [MRWJ$^+$10]. In their work, game theory was introduced to formulate a scheduling game of energy consumption, where game players were the users whose strategies were the daily schedules of house appliances. Using binary particle swarm optimization to schedule the resources of customers, Pedrasa et al. proposed a decision support tool for customers to acquire optimized energy services [PSM09]. Rastegar et al. developed a load commitment framework allowing minimizing the cost of householder by automatically shifting responsive electrical loads, including battery storage and plug-in hybrid electric vehicles [RFFA12].

- **Price-driven response**
  Many methods have been developed for efficient demand responses by changing the energy prices in the market. There are various ways of pricing, such as time of use, critical peak pricing, and real-time pricing. Khodaei et al. investigated the real-time pricing, which is a widely used demand response method [KSB11]. Their work implied that demand response could lead to peak load reduction and therefore reduce the price spikes to benefit the whole grid system. Conejo et al studied demand response under the influence of price uncertainty [CMB10]. They introduced robust optimization with dual property to model price uncertainty. Beside, linear programming was introduced in their approach to solve the proposed model for practical implementations.

- **Network level response**
  The network level responses take both energy producers and consumers into consideration, aiming at efficient demand response over the gird network. Ilic et al. first investigated demand response methods integrating wind power generations [IXJ11a], and then developed a new decision-making and dynamic monitoring system to achieve a near-optimal energy dispatch. Later, Ilic et al. in [IXJ11b] showed that up to 50% of generated wind power could be accommodated using the method proposed in [IXJ11a]. They also showed an effective way to implement their method by multi-directional information exchange between decision makers and the control center, so that the global supplied demand function could be optimized under constraints.

The three categories of demand response methods gain their own advantages in management of Smart Grid market. Customer's energy optimization tries to improve efficiency of energy in the end-user side. Pricing mechanisms are suitable to manage retail market with adaptive customers in Smart Grid. Network level response tries to optimize energy usage from the view of the whole grid system.

To effectively manage the retail market, the pricing mechanism is introduced in the proposed broker model in Chapter 3 of this thesis. Pricing mechanisms are further discussed in the next subsection.

2.1.2 Broker models

To ameliorate the management of Smart Grid market, brokers are widely employed [KCRW14].

Brokers, who buy energy from the wholesale market and sell it to the retail market, form a decentralized mode to enhance the efficacy of Smart Grid market. However, a broker faces complex situations in Smart Grid market as illustrated in Figure 2.1. A broker has to simultaneously interact with the wholesale and retail (tariff) market with a variety of customers. The balancing market can supply unbalanced demand to the broker with a high price.

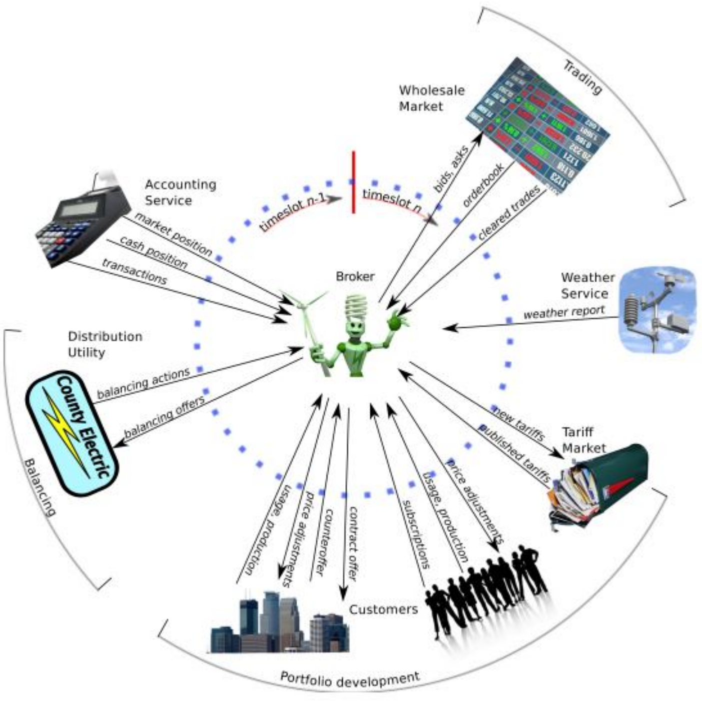

Figure 2.1: An illustration of a broker Smart Grid market. This figure is adapted from [KCRW14].

A successful broker should not only maximize his own profit, but also keep a good supply-demand balance in the two markets to improve the energy efficiency.

However, the excellent broker has to cope with the omnifarious challenges. For the wholesale market, there are dynamics in energy price, quantity and stability because of the various energy suppliers. To purchase a proper amount of energy for the each of coming 24 hours, a customer demand prediction is needed, but is a very challenging issue in Smart Grid due to the various behaviors of different customers and customers' migrations between brokers. Moreover, a bidding strategy is required to optimize the bidding prices in auctions, with the considerations of competitions among other brokers. For the retail market, there are variety of consumers with various behaviors. An excellent broker should consider the different types of consumers and their different power usages. Besides, the broker should also deliberate on the uncertainty of customers' behaviors in power usage. Moreover, the broker also needs an efficient strategy to compete with other brokers to attract customers in the retail market.

Broker modeling for the management of Smart Grid market is an emerging research field. In 2012, Power Trading Agent Competition (Power TAC) started and supplied a simulated real-world Smart Grid market environment. Some competitive broker models are reviewed in the following.

AstonTAC team [KHCW13] developed a broker model that introduced MDP approach for auctions in the wholesale market. AstonTAC also employed different Hidden Markov Models (HMM) to predict the price of energy and the customer demand. AstonTAC could keep a good supply-demand balance, but it did not provide an effective strategy to attract customers in the retail market. Their broker model got the second place in Power TAC in 2012, fell behind of the first place which used some designed tricks to attract customers. Therefore, AstonTAC had a weakness for the competitions in retail market, while its strength was the accurate prediction of power demand. AstonTAC employed HMM for customer demand prediction. The proposed broker model in Chapter 3 also makes effort to efficiently predict the customer demand, but it uses light-weight linear regressions for classified customer groups to enhance the competition capabilities in dynamic Smart Grid market.

Urieli and Stone developed a broker model called TacTex and won the Power TAC in 2013 [US14]. They decomposed the global optimization into sub-optimizations in the wholesale and retail markets. Locally weighted linear regression was introduced to predict if the customers would subscribe his tariffs. The TacTex won in profit making, but it did not make much effort on supply-demand balance. If the balance of supply and demand were further taken care of, TacTex could achieve even better performance than the current version.

The CwiBroker team [LHL14] used game theory in both wholesale and retail markets to maximize the profit. Compared to reinforcement learning used in Tac-

Tex, game theory is less competitive to the dynamic market changes. Peters et al. introduced on-policy reinforcement learning (SARSA) to optimize price in the retail market and showed a competitive performance [PKSTC13]. Based on previous studies, in the designed broker model in Chapter 3, reinforcement learning is introduced to generate policies in the competitive retail market.

In Chapter 3 of this thesis, a new broker model is developed for management of Smart Grid market. Different from previous broker models that might emphasize on one or two issues in Smart Grid market, the proposed broker model efficiently tackles aforementioned challenges in Smart Grid market. It defines some customer usage patterns to cluster customers, and then predict the demand for each customer group to obtain an accurate estimation of power usage. The proposed broker also introduces reinforcement learning to manage the dynamic retail market. In this end, the proposed broker can keep a good supply-demand balance between the wholesale and retail market. Therefore, the developed broker model can be an effective solution to the first research issue identified in Section 1.2.2.

2.2 Demand Prediction

Demand prediction plays a key role in supply-demand balance. It is the basis to make effective decisions on energy production and distribution. For its critical significance in Smart Grid, demand prediction has been deeply studied for decades. Demand prediction methods in the literature are reviewed, including regression methods, neural networks and other machine learning methods. Specially, methods that consider customer behaviors to improve demand prediction are also reviewed.

2.2.1 Regression methods

Regression is a deeply studied method in both statistics and machine learning societies. As a basic method, regression can be applied into different problems. Researchers in Smart Grid have developed many variants of regression models for energy demand prediction.

Alex et al. introduced binary coding for accurate holiday modeling, and degree functions for temperature modeling [PH90]. They employed robust parameter learning method is to reduce the effects on precision of load forecasting caused by explanatory variables. Haida et al. developed a regression with transformation for the prediction of daily peak load [HM94]. To precisely predict the demand through a year, they considered daily load changes, seasonal load changes and annual load growth. They further used a transformation function with translation and reflection to transform the former data. Charytoniuk et al. presented a nonparametric regression approach for short term demand forecast [CCVO98]. Their approach was derived from a load model in the form of a probability density function between energy load and its affecting factors. In their approach, a load forecast was a conditional expectation of load, given a series of explanatory variables, such as time and weather conditions.

Lam et al. examined the commercial and residential sector power consumption patternin [LTL08], in which principle component analysis of five major climate variables was conducted. In their method, the sector-wide energy consumption with the corresponding two principle components were determined using multiple regression. Jonsson et al. used a non-parametric regression model [JPM10] for efficient wind power prediction, in which the conditional distribution of price was found to be non-Gaussian.

Some research predicted the energy demand in a large scale and long term level. Egelioglu et al. examined how annual electricity consumptions are influenced by the economic variables [EMG01]. Using multiple regression analysis, their approach could determine the relationship between energy consumptions, the price of electricity, the number of customers and the number of tourists.

Standard regression has been widely used for solving prediction problems in many domains. Though much effort has been tried to adapt regression to energy demand prediction, the prediction accuracy is not very satisfactory, because energy demand prediction is a time-series prediction problem. A standard regression model could not effectively make use of temporal correlations in energy demand prediction. The auto-regressive method, which can improve the naive regression by considering temporal correlations in time-series and thus becomes more suitable for energy demand prediction, is discussed in the following.

2.2.2 ARIMA model

An Auto-Regressive Integrated Moving Average (ARIMA) model is a class of models for time-series prediction. The acronym ARIMA contains three major aspects [BP70], which are:

- **Auto-Regression (AR).** A model that makes use of the dependency between the current observation and lagged observations.
- **Integrated (I).** Using the differences in observations to make time-series stationary.
- **Moving Average (MA).** A model that utilizes the dependency between the residual error and the observation from a moving average of lagged observations.

As ARIMA is specially designed for time-series prediction, it has been widely used to demand prediction problems. Sumer et al. used three models, ARIMA, seasonal ARIMA and regression model for energy demand forecasting [SGH09]. They made extensive comparisons on the three models and got the conclusion that regression model with seasonal latent variable showed better results. Saab et al. used AR(I) model with a finite impulse response filter for energy demand prediction, and studies the case of electricity consumptions in Lebanon [SBN01]. Their hybrid model was compared with AR and ARIMA, and showed higher prediction accuracy. Cho et al. combined ARIMA and transfer function model for short-term load forecasting [CHC95]. Their model studied the effect of temperature on power consumption and concluded that the introduction of transfer function model could improve ARIMA by considering the causality between power consumption and temperature.

ARIMA has a long history in energy demand prediction. Though ARIMA is a tailored regression model for time-series data, it is weak in dealing with feature nonlinearities. To achieve a good performance, ARIMA requires human-designed feature engineering. With the combination of machine learning approaches, more complicated models can achieve more accurate predictions than ARIMA.

2.2.3 Neural networks

Neural networks have been recognized as universal function approximators and have been widely used for solving prediction problems in various domains. There are also a number of literatures on energy demand prediction using neural networks.

Lu et al. evaluated artificial neural network to testify if it was a system independent model or a case dependent model [LWV93]. Their work evaluated the performance of neural network in practical short-term energy demand forecasting. Chow et al. proposed a method which was a nonlinear generalization of Box and Jenkins approach for non-stationary time-series prediction [CL96]. In their approach, weather compensation neural network was developed for one-day ahead energy load prediction. Vermaak and Botha introduced Recurrent Neural Network (RNN) for energy demand prediction [VB98]. In their work, RNN showed the nature of modeling dynamic nonlinear systems and obtained good performance in short-term load forecasting.

Gao et al. proposed a neural-wavelet approach for energy demand forecasting [GT01]. The wavelet transform was proven to be able to capture key features of various types of loads and could supply promise for an on-line wavelet-based discriminator. Mandal et al. developed a neural network based algorithm for a deregulated electricity market [MSF06]. Kandil et al. proposed an efficient neural network for energy demand forecasting [KWSG06], where only weather data was used. Their results indicated that temperature played the key role in energy demand prediction. Amin et al. combined unsupervised learning and supervised learning to predict daily peak demand [ANS08], in which the patterns of daily peak load were classified and a hybrid neural network was introduced to predict the peak consumptions. Xiao et al. presented an approach to accurate prediction of short-term load. Their approach combined neural network and rough set to deal with dynamic and non-linear factors in load forecasting [XYZS09].

Neural networks are also introduced for mid-term and long-term energy demand predictions. Azadeh et al. integrated neural network, time-series model and ANOVA model to predict monthly and seasonal energy prediction [AGS07]. Carpinteiro et al. proposed a new hierarchical hybrid neural for the forecasting of long-term load [CLdS+07]. In their approach, the neural model was built by two self-organizing map networks and then connected to a perceptron. Xia used radial basis function neural network to predict short, medium and long term electricity demand in [XWM10].

Though neural networks have strong nonlinear representational capacities, they do not model temporal correlations in energy demand prediction. RNN can model temporal correlations, yet it is very hard to train. Recently, deep neural networks have been developed and could be efficiently trained, showing competitive performances in various domains. It is also possible to introduce deep neural networks for energy demand prediction. The deep learning methods for prediction are further discussed in Section 2.3.

2.2.4 Continuous conditional random fields

In the Conditional Random Fields (CRF) research community, Qin et al. proposed Continuous CRF (CCRF) in 2009, which extended CRF to be capable of solving real-value regression problems. Since then, CCRF has been widely applied to various domains, such as learning to rank [QLZ+09], expression recognition [BBR13], and social recommendation [XKDL09]. CCRF has two potential functions, i.e. node potential and edge potential. A node potential maps input features to output, playing a similar role to regression. A edge potential models the dependencies in the output variables. As CCRF integrates the two potentials, it gains an advantage of modeling dependencies in the output over regression models.

Guo [Guo15], [Guo16] used CCRF to forecast the short-term power and gas usage in a building. His work also introduced predictive clustering trees to tackle the weak feature constraint problem in CCRF. Guo's work demonstrated the advantages of CCRF and achieved superior performances in load forecasting in a small area. Based on Guo's work, a new variant of CCRF is proposed in Chapter 4 of this thesis for energy demand prediction in a complex Smart Grid market. In the proposed method, an L_1 norm penalty term is introduced to CCRF to result in a sparse learning model called sparse CCRF (sCCRF). sCCRF is applied to explore customer behaviors to aggregate different customers into groups, in order to improve prediction accuracy.

Sparse Gaussian Conditional Random Fields (SGCRF) [WK13] is also an important variant of CCRF. Wytock and Kolter used it in energy demand forecasting and wind energy prediction [WK13]. Different from CCRF, SGCRF introduced parameters in each time step, and resulted in many more parameters than CCRF. Such a large number of parameters may lead the customer clustering step to the “curse of dimensionality”, and therefore, SGCRF is not suitable to analyze customer behaviors. On the other hand, SGCRF does gain advantage in energy demand prediction because it can effectively model the dependencies of output variables. To make use of this advantage, Chapter 6 of this thesis proposes a new model, an integration of SGCRF and RNN, which is a new effective model for energy demand prediction.

2.2.5 Other machine learning methods

With the developing of machine learning technologies, new machine learning models are also introduced to energy demand prediction. Representative methods include support vector regression, Bayesian vector auto-regression and particle swarm optimization methods.

Wang et al. modeled energy consumption as a function of gross national production, imports and exports, and population [WZZS09], and then introduced support vector regression for the input variables to predict the energy demand in Turkey. Crompton et al. used bayesian vector auto-regression model to predict energy requirement in China [CW05]. In their work, the past trends were analyzed and future directions were suggested based on the predictions. Unler [Ünl08] developed energy prediction based on swarm intelligence, in particular, particle swarm optimization. Their work first introduced swarm intelligence to energy demand prediction, and showed the merits of the advanced optimization method in prediction.

The above methods tried to introduce new machine learning models to energy demand prediction. Those models might achieve some improvement compared to traditional models, but they were not tailored for time-series prediction, so they might not make full use of the information in sequential data. To utilize the information in time-series data to improve energy demand prediction, Section 2.4 further reviews latest time-series prediction models.

2.2.6 Methods considering customer behaviors

A customer's energy consumption pattern under the influence of a range of factors (such as time and weather conditions) in Smart Grid is defined as *customer behaviors*. The complexity of customer behaviors comes from two aspects: vast types of customers and irregular behaviors of each customer type. In Smart Grid, the concept of "customer" has extended to include not only general energy consumers, but also interruptible consumers, consumers with storage capacity and even small renewable energy producers. Due to the complex customer behaviors, traditional load forecasting methods, which target the grid system or a particular customer, face challenges to precisely forecast the load of Smart Grid. Intuitively, if customers with similar behaviors could be aggregated into groups, the predictions towards customer groups would improve the final load forecasting accuracy.

There has been some work on studying customer behaviors based on human classification or historical usage data. Srinivasan [Sri08] manually divided different customers in a grid system into six groups, and introduced a Group Method of Data Handling (GMDH) neural networks for load forecasting. Their methods obtained better load forecasting accuracy than forecasting towards the whole grid system, but customer groups are divided manually.

Alzate et al. [AEDS09] used spectral clustering to cluster customers with respect to the historical load data and reported improved accuracy in load forecasting. Alzate and Sinn [AS13] further explored kernel spectral clustering to aggregate customers, and their work showed improved results compared to Alzate's previous work [AEDS09]. Both of the two methods tried to aggregate customers based on the historical usage data. The customers were classified by an unsupervised clustering method. For customer aggregation, clustering according to historical usage data could be more accurate than manual divisions, but clustering could still not reflect the relations between customer usages and external influences, such as weather conditions.

Fiot and Dinuzzo [FD16] used multi-task kernel learning to predict long-term load. Their method tried to discover the similarity of nodes in Smart Grid and thus to improve the prediction of each node (customer). In predicting the long-term load, only time and calendar features were considered. It is known that individual customer may have some random behaviors that are hard to predict. Their method predicted the load of each customer and thus could not make use of the averaging effect to reduce the randomness of customer behaviors.

In summary, there have been some approaches concerning customer behaviors, in which the behaviors were defined by human or unsupervised clustering. Different from previous work, Chapter 4 of this thesis proposes Load Forecasting through

Learning Customer Behavior (LF-LCB). LF-LCB aggregates different types of customers by their identified behaviors, and then predicts the load of each customer cluster, so as to improve load forecasting accuracy of Smart Grid. LF-LCB introduces supervised learning to analyze the relations between external features and customer energy usages, which is an accurate classification of customer behavior. Based on the identified customer behaviors, customers can be accurately aggregated, so as to improve load forecasting precision. LF-LCB can be an effective solution to the second research issue identified in Subsection 1.2.2.

2.3 Renewable Energy Prediction

Renewable energy is an important part in Smart Grid market. The fluctuations of production of renewable energy significantly affect energy supply in Smart Grid market. In order to keep a good supply-demand balance, it is significant to accurately predict the production of renewable energy.

Most of energy demand prediction methods can be applied to renewable energy prediction. Compared to energy demand, the production of renewable energy is more sensitive to weather factors and shows larger fluctuations. Thus, it is necessary to further investigate methods for renewable energy prediction.

Some machine learning methods have been applied to renewable energy prediction. Kalogirou summarized the applications of neural networks in renewable energy modeling and prediction [Kal01], and concluded that the strong nonlinear neural networks could fit different tasks such as modeling and forecasting in renewable energy studies. Mohandes et al. introduced Support Vector Regression (SVR) to wind energy prediction [MHRH04]. Their work reported that SVR could achieve better accuracy than multi-layer neural networks in wind energy forecasting.

Recently, more competitive prediction models have been introduced to renewable energy prediction. Gradient boosting [Fri01] was an effective function approximator through the ensemble of boosting trees. In the solar energy prediction contest on Kaggle, which aimed at predicting the daily solar energy production given several temporal weather forecastings in a day, all of the top three winning solutions in this contest used Gradient Boosting Decision Trees (GBDT) [MGB$^+$15]. The best solution used GBDT as the learning tool, and they trained 13 models at each solar station. After that, two step optimizations were introduced to fine-tune the trained models. The first step was to use Nelder and Mead nonlinear optimization to tune the obtained GBDT models for each solar station, and the second step was to optimally weight the predictions by the results of nearby solar stations.

In the probabilistic energy forecasting competition 2014, the winning solution was also based on GBDT [HP16]. The winning solution deeply studied the features for solar power generation and introduced an extra blue sky model. GBDT and k-Nearest Neighbors were used to build the model for each hourly production with the weather feature in current hour and the previous three hours. Similar to neural networks, gradient boosting [CG16] has been an effective method to handle nonlinear features, but it cannot naturally model temporal information. Feature engineering has been introduced to compensate the weakness of gradient boosting in handling temporal information.

SGCRF, proposed by Wytock and Kolter [WK13], was applied to the prediction of wind farm energy. SGCRF can effectively make use of the dependencies in the output variables, and shows competitive performance in wind energy prediction. SGCRF can also be applied to demand prediction with satisfactory performance. However, SGCRF constructs a linear mapping from input features to output variables, thus feature engineering is necessary to cope with feature nonlinearities.

Vermaak et al. introduced RNN for renewable energy prediction [VB98]. In general, RNN can be a suitable model for renewable energy prediction because it can model temporal correlations and handle nonlinear features. However, RNN could not be reliably trained before the development of deep learning technologies. Particularly, Vermaak et al. only used one layer RNN, which had less representational capacities than deep RNNs.

Different from previous work on renewable energy prediction, Chapter 5 of this thesis studies renewable energy prediction from the perspective of deep learning. Deep learning has shown advantages in several fields, such as computer vision, speech recognition and natural language processing [Ben09]. Chapter 5 makes use of the latest deep learning method to improve energy prediction. Deep RNNs are introduced in Chapter 5 to predict the production of renewable energy and obtain excellent results. The developed method in Chapter 5 can be an effective solution to the third research issue identified in Subsection 1.2.2.

2.4 Time-series Prediction Models

Energy demand prediction is a time-series prediction problem. To further improve the accuracy of demand prediction, this section summarizes the advantages and disadvantages of state-of-the-art methods for time-series prediction in broad domains. Then a new effective time-series prediction model is sketched, which contributes to time-series prediction in machine learning research and achieves a competitive performance in energy demand prediction.

A multi-step time-series prediction problem can be formally defined as follows:

$$\mathbf{y} = f_W(\mathbf{X}), \tag{2.1}$$

where f_W is a transformation (to be learned) parameterized by W, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T]$ is a $T \times D$ matrix, representing T steps and D observed features in each step, and $\mathbf{y}$ is a T-dimension structured output.

In multi-step time-series prediction, there are three kinds of critical information, which are temporal correlations in the observed data $\mathbf{X}$, nonlinearities in features $\mathbf{x}_t$ and structured information in the output $\mathbf{y}$. *Temporal correlations* refer to that step features $\mathbf{x}_t$ are correlated in close time steps, which can be measured by auto-correlation [TEL+92]. *Nonlinearities* refer to nonlinear correlations in some features in $\mathbf{x}_t$, which are usually accomplished by feature engineering. *Structured information* refers to the dependencies in the output variables $\mathbf{y}$, which can be interpreted in a geometric view. Figure 2.2 illustrates the outputs of multi-step time-series prediction. The blue dots are ground-truths and the red triangles are predictions. For the multi-step prediction problem, the output can be imagined as a curve. In this scenario, the structured information can be regarded as the shape of the curve.

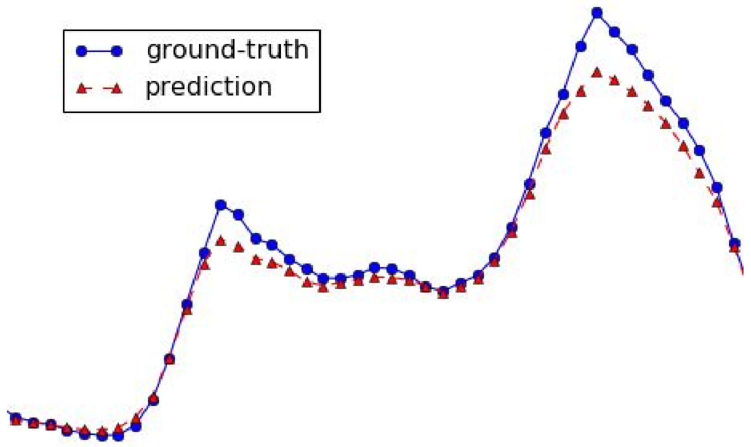

Figure 2.2: An illustration of time-series prediction. The groud-truth and prediction are imagined as a curve.

2.4.1 Typical time-series prediction models

There have been several methods for multi-step time-series prediction problems. ARIMA is a traditional model for time-series prediction [BP70], which utilizes temporal correlations of the observed data and partially structured information for prediction. However, ARIMA is weak in dealing with nonlinearities and thus it often resorts to feature engineering. Gradient boosting [Fri01] is a universal function approximator and achieves success in regression problems [CG16]. Although gradient boosting has strong capacities to represent nonlinearities, in multi-step time-series prediction, it cannot fully handle temporal correlations and miss the structured information.

Recently, deep RNNs [PGCB13] and SGCRF [WK13] have been introduced to time-series prediction, showing promising results. SGCRF is an effective structured learning model, while it is weak in representing nonlinearities in features. As a consequence, SGCRF relies heavily on feature engineering. RNN can effectively model temporal correlations and represent nonlinearities in features, while it misses the structured information in the output. We can also diagnose RNN and SGCRF in a geometric view by Figure 2.2. RNN tries to make each individual predicted point close to the ground-truth point, without the consideration of curve shape. In contrast, SGCRF tries to match the shape of predicted curve and the ground-truth curve. Intuitively, if the structured information was incorporated into RNN, the performance of time-series prediction could be improved.

There are some very recent studies of time-series prediction based on deep neural networks. Osogami and Otsuka proposed Dynamic Blotzmann Machine (DyBM) for time-series prediction in general domains [OO15]. The most notable character of DyBM is online updating, while its performance is not as good as RNN in off-line learning. Lin et al. used attention-based RNN and introduced dual-stage attention to improve the accuracy of time-series prediction [LGA17]. Their work tried to model the observed data in a more effective way, but lacked consideration of structured information of output.

2.4.2 A new time-series prediction model

Based on the thorough analysis of previous time-series prediction models, Chapter 6 of this thesis proposes a new prediction model, Sparse Gaussian **C**onditional Random Fields **o**n top of **R**ecurrent Neural Networks (CoR). CoR can make use of temporal correlations, nonlinearities and structured information in multi-step time-series prediction problems. CoR has gained striking advantages over RNN and SGCRF. Compared to RNN, CoR can learn structured information to result in a significant boost in performance. Compared to SGCRF, CoR can effectively represent nonlinear temporal features and greatly outperforms SGCRF. In the end, CoR achieves better performance than state-of-the-art methods in time-series predictions. Specially, CoR and other models are further compared in energy demand prediction in real-world data. CoR shows the best performance in energy demand prediction among various models.

In 2014, attention mechanism was proposed by Bahdanau et al. to improve RNN in machine translation [BCB14]. Attention mechanism can measure the importance of temporal features to the current output, which can also be introduced to CoR to improve the performance of time-series prediction.

In Chapter 6, attention mechanism is also introduce to CoR, resulting an improved model CoRa. Attention mechanism improves RNN by associating the importance of input features to the current output, while SGCRF layer compensates the weakness of RNN in structured learning. Attention and structured information improve RNN from two separate ways. Compared to CoR, CoRa has extra attention to measure of the complex features in Smart Grid market, and thus achieves even better performance in energy demand prediction.

2.5 Summary

In this chapter, current literature regarding the three research issues in this thesis was thoroughly reviewed. Specially, the review concentrated on related studies in Smart Grid management via broker models, demand prediction methods, renewable energy prediction methods and discussions of time-series prediction models.

Chapter 3

A Strategic Broker Model for Management of Smart Grid Market

This chapter proposes a strategic broker model for management of Smart Grid market. Three goals are established in the designation of a good broker model: **Goal 1** is to efficiently predict the energy demand of customers, so as to keep a good balance of supply and demand; **Goal 2** is to obtain energy as the required demand in the wholesale market with a lower price; and **Goal 3** is to sell energy to customers with a proper price, which can ensure a good profit and attract customers.

Effective methods are proposed to achieve the above three goals. For Goal 1, a data-driven method is proposed to first cluster various customers according to their energy consumption patterns, and then predict the one-day-ahead hourly demand of subscribed customers. For Goal 2, the Markov Decision Process (MDP) is employed for energy auctions in the wholesale market. For Goal 3, independent reinforcement learning processes are introduced to optimize prices for different types of customers.

The rest of this chapter is organized as follows. Section 3.1 presents the definitions in this chapter and framework design of the proposed broker model. Section 3.2 describes the detail designs of three modules in the broker model. Section 3.3 reports experiments that evaluate the proposed broker model. Section 3.4 provides a discussion of experimental results. Finally, Section 3.5 summaries this chapter.

3.1 Definitions and Framework Design

In this section, the terms that are used in the rest of the chapter are defined, and the framework of the proposed broker model is described.

3.1.1 Definitions

Definition 1. (**Bootstrap Data**) Bootstrap data $\mathbf{B}_D$ are the historical data of customer usages in the retail market. It is represented as the following matrix,

$$\mathbf{B}_D = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1T_b} \\ u_{21} & u_{22} & \cdots & u_{2T_b} \\ \vdots & \vdots & \vdots & \vdots \\ u_{N1} & u_{N1} & \cdots & u_{NT_b} \end{pmatrix}, \tag{3.1}$$

where u_{ij} is the hourly usage of customer i in hour j, N is the total number of customers, and T_b is the length of a usage sequence of customer i.

Definition 2. (**Usage Gradient**) Usage gradient $\mathbf{G}_i = [g_1, g_2, \cdots, g_{24}]$, is a 24-dimension vector that reflects how the hourly usage amount of customer i changes in 24 hours. Each element g_k is calculated in a usage sequence of customer i.

Assuming there is a N day length usage sequence $\{u_k^n\}$, where u_k^n is the usage of hour k in day n. g_k is the average of g_k^n, $g_k = \frac{1}{N}\sum_{n=1}^{N} g_k^n$, where g_k^n is calculated by $g_k^n = (u_k^n - u_{k-1}^n)/\overline{u^n}$ (for u_0^n the usage of hour 0 is the usage of hour 24 of last day), where $\overline{u^n}$ is the average usage of the 24 hours in day n.

Definition 3. (**Usage Variance**) Usage variance $\mathbf{V}_i = [v_1, v_2, \cdots, v_{24}]$, is a 24-dimension vector regarding customer i. Each element v_k reflects how much the usage in hour k varies in different days. v_k is calculated in a usage sequence of customer i.

In a N day length usage sequence $\{u_k^n\}$, v_k is a standard variance, calculated by $v_k = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(u_k^n - \overline{u_k})^2}$, where $\overline{u_k}$ is the average of usage in hour k of N days.

Definition 4. (**Energy Consumption Pattern**) Energy consumption pattern $\mathbf{P}_i = \{\mathbf{G}_i, \mathbf{V}_i, T\}$ for customer i, is defined as a collection of usage gradient $\mathbf{G}_i$, usage variant $\mathbf{V}_i$ and time period T. It indicates how a customer consumes energy in a certain time period.

3.1.2 Framework design

The broker model is designed to efficiently adapt to the dynamics in both the retail and wholesale markets. To achieve the three goals, three modules are designed in the broker model, which are: 1) customer demand prediction module, 2) wholesale market module, and 3) retail market module. The three modules incorporate to cope with market dynamics. The customer demand prediction module predicts one-day-ahead customer demand, and outputs the predicted demand to the wholesale market module. The wholesale market module aims to obtain such amount of energy with a low price in auctions. Then energy cost is calculated and passed to the retail market module. Based on the known energy cost, the retail market module makes decisions to attract customers and gain profit. The framework of the proposed broker model is shown in Figure 3.1.

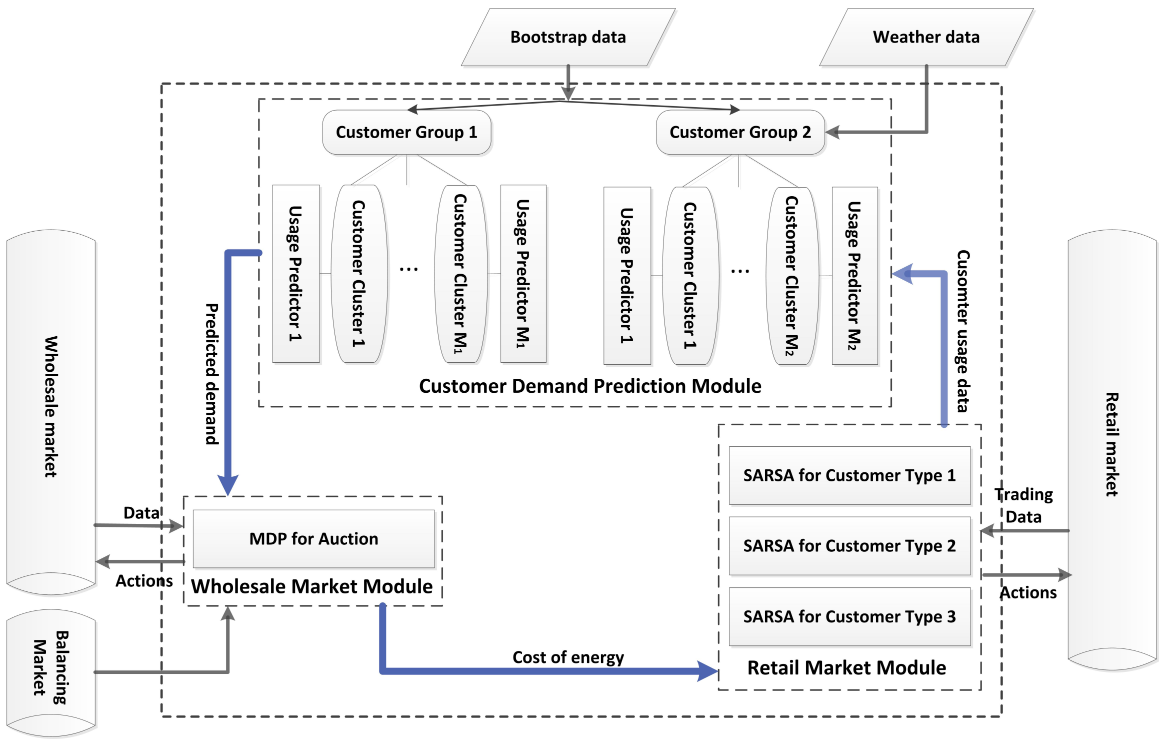

Figure 3.1: The framework of the proposed broker model

The three modules work as follows.

- **Customer Demand Prediction Module**

  The customer demand prediction module makes use of the data from two stages: the bootstrap stage and the dynamic market stage. It first explores the retail market based on bootstrap data, from which the energy consumption pattern for each customer is calculated. According to the different usage patterns, all the customers are clustered into clusters through a two-layered clustering process. For each customer cluster, a usage predictor, which is learned from the bootstrap data, is used to predict the future usage. In dynamic market environments, the retail customers who subscribe tariffs are maintained, and then these customers are assigned to the corresponding customer clusters with respect to their energy consumption patterns. Using the learned usage predictor for each cluster, the broker predicts the energy usage for each customer cluster, and accumulates all the predicted usages to obtain total demand.

- **Wholesale Market Module**

  Given the predicted demand, the wholesale market module employs a Markov Decision Process (MDP) to bid for the predicted amount of energy in one-day-ahead auction in the wholesale market. If the obtained energy amount is less than the actual usage, the imbalance amount will be compensated by the balancing market with a balancing price [a], which might be much higher than the market price. After the auction, the cost of the obtained energy is calculated and passed to the retail market module.

- **Retail Market Module**

  The retail market module calculates the profit based on the given energy cost from the wholesale market module. To compete with other brokers, this module uses three independent SARSA control processes to optimize the tariff prices for three types of customers, which are prosumers, general consumers (customers who require consistent power supply, such as householders and office users) and interruptible consumers (customer with energy storage capacity, such as cold storage companies). The calculated profit is regarded as the immediate reward in the SARSA process.

From the perspective of an integrated system, the three modules are connected and can exchange information. The information flow is highlighted in Figure 1. The customer demand prediction module gathers customer usage data from the retail market module and outputs the predicted energy demand to the wholesale market module. The wholesale market module tries to obtain the predicted amount of energy in the auction process, and passes the energy cost to the retail market. The retail market module uses the cost of energy in reinforcement learning to calculate rewards and makes decisions to different customers. The retail market module also maintains the data of customers for the customer demand prediction module. Through the communications and cooperations of the three modules, the broker model can efficiently response to the dynamics in market, so as to make more profit and keep supply-demand balance. In the following section, technical designs of the three modules are introduced in details.

3.2 The Detail Design of Three Modules

This section introduces the three modules of the proposed broker model in details.

3.2.1 The detail design of the customer demand prediction module

In this module, a data-driven method is proposed to predict customer demand, which is critical for supply-demand balancing. The data-driven method hierarchically clusters the customers based on their energy consumption patterns, and then

[a]The balancing price is for the power from high-level grid, for example, the state grid. As the delivery distance is long, balancing price may be much higher than the market price.

different prediction methods are introduced to predict the energy demand for different customer clusters 24 hours ahead. The procedure of customer demand prediction is described in Algorithm 1.

Algorithm 1 A data-driven method for customer demand prediction

```
Input: Bootstrap data $\mathbf{B}_D$;
Output: Predicted energy demand $U_t, t = 1, \cdots, 24$;
 1: Initialize: $U_t = 0, t = 1, \cdots, 24$;
 2: Calculate customer energy consumption pattern $\mathbf{P}_i$ for each customer based on $\mathbf{B}_D$;
 3: Cluster the customers into customer group $\mathrm{C}_1$ and $\mathrm{C}_2$ according to customer usage variance $\mathbf{V}_i$;
 4: Cluster the customers in $\mathrm{C}_1$ and $\mathrm{C}_2$ according to customer usage gradient $\mathbf{G}_i$, and $M_1$ and $M_2$ customer clusters are obtained, respectively;
 5: Assign the subscribed customers into the $M$ customer clusters;
 6: for each hour $t \in [1, 24]$ do
 7:     for each customer cluster $c_i \in \mathrm{C}_1$ do
 8:         Predict future usage $U_t^i$ based on $\mathbf{B}_D$;
 9:         $U_t = U_t + U_t^i$ ;
10:     end for
11:     for each customer cluster $c_i \in \mathrm{C}_2$ do
12:         Predict $U_t^i$ with the regressor learned from $\mathbf{B}_D$;
13:         $U_t = U_t + U_t^i$ ;
14:     end for
15: end for
```

In Algorithm 1, The broker model predicts next day energy demand $U_t(t = 1, \cdots, 24)$ according to the input bootstrap data $\mathbf{B}_D$. In Line 1, the energy demands to be predicted are initialized to 0. Line 2 calculates the energy consumption pattern $\mathbf{P}_i$ of each customer. Then customers are clustered according to $\mathbf{P}_i$ in Lines 3-4. Line 3 clusters the customers into customer group C_1 and C_2 according to customers' usage variances, and Line 4 further clusters the customers in C_1 and C_2 according to customers' usage gradients, respectively. The customers who subscribe tariffs are assigned into the customer clusters in Line 5. In Lines 7-10, the energy demand for each customer cluster in customer group C_1 is predicted and accumulated to the total energy demand. In Lines 11-14, similar routine is executed for each customer cluster in customer group C_2. The details of clustering customers (Lines 3-4), prediction method in C_1 (Line 8), and prediction method in C_2 (Line 12) are further introduced as follows.

Clustering customers: According to pattern $\mathbf{P}_i$ obtained in Line 2, a two-layered clustering method is used to cluster all the customers. There are two layers in the cluster method. For the first layer, customers are clustered according to usage variance $\mathbf{V}_i$, and two customer groups are obtained, i.e. C_1 and C_2 (see Line 3). For the second layer, customers in C_1 and C_2 are further clustered according to usage gradient $\mathbf{G}_i$, and M_1 and M_2 clusters are formed, respectively (see Line 4).

In the first layer, it is aimed to cluster all the customers into two groups, where one group of customers (C_1) shows stable hourly usages in different days, and another group (C_2) has variable hourly usages. A hard threshold is used to cluster all the customers in the first layer. For customer i, if any element v_k in his usage variance $\mathbf{V}_i$, satisfies $v_i \leq \Delta_1$, the customer is assigned into C_1. Otherwise, the customer will be assigned into C_2. A strict threshold Δ_1 is used to ensure that the hourly usage of customers in C_1 is stable. The cluster procedure in the first layer makes sense for the reason that two prediction methods can be tailored for the customers in C_1 and C_2. As customers in C_1 show stable hourly usages everyday, the future hourly usages can be predicted according to the historical usage data.

In contrast, the hourly usages of customers in C_2 change with respect to other factors, thus supervised learning can be introduced to learn predictors for those customers. In the second layer, it is aimed to find customers sharing similar usage gradients. K-means with a Euclidean distance criterion Δ_2 is used to cluster the customers in C_1 and C_2, respectively. It is obvious that Δ_2 affects the numbers of final clusters, i.e. M_1 and M_2. The parameter Δ_2 is further analyzed in the experiment section. The reason of further clustering customers and the meaning of clusters in the second layer are explained as follows. Similar usage gradients indicate that customers in the same cluster show similar responses to the outside factors, thus one predictor is adequate for one customer cluster.

Prediction method in $\mathbf{C}_1$: Customers in C_1 show stable hourly usages in different days. For this kind of customers, accumulated historical data are used to predict their one-day-ahead usage. For the M_1 customer clusters, $M_1 \times 24$ predictors are constructed. Equation 3.2 is designed to predict one-day-ahead usage U_{mt} of hour t for cluster m.

$$U^f_{mt} = \lambda U^c_{mt} + (1 - \lambda)\overline{U^h_{mt}} \tag{3.2}$$

In Equation 3.2, U^f_{mt} is the future one-day-ahead usage, U^c_{mt} is the current usage, $\overline{U^h_{mt}}$ is the average of historical usage within a time window of one week, and λ is a weight parameter. A large λ indicates a strong emphasis on the current usage. Equation 3.2 recurs while time goes forward.

Prediction method in $\mathbf{C}_2$: Customers in C_2 show variable hourly usages, which might be affected by weather conditions and dynamics of the retail market. The influence of weather conditions is considered first. A supervised learning method—LASSO regression [Tib96] is used to estimate the future usage. The regressors are learned from the historical usage data and weather data. $M_2 \times 24$ predictors are learned for each customer clusters in C_2, respectively. The predictor, which predicts the hourly usage in hour t for cluster m, is illustrated as follows. The customer usage in hour t of all the customers in cluster m from bootstrap data $\mathbf{B}_D$ is extracted. A new data matrix $\mathbf{B}_d$ is obtained,

$$\mathbf{B}_d = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1L} \\ u_{21} & u_{22} & \cdots & u_{2L} \\ \vdots & \vdots & \vdots & \vdots \\ u_{K1} & u_{K2} & \cdots & u_{KL} \end{pmatrix}, \tag{3.3}$$

where u_{kl} is the hourly usage, and K is the total number of customers in this cluster, L is the total number of usage records of hour t contained in a line vector of bootstrap data $\mathbf{B}_D$. L can also represent the length of bootstrap data measured by days. Adding up each line of $\mathbf{B}_d$, a vector $\mathbf{U}$ is obtained, $\mathbf{U} = [u_1, \cdots, u_L]$. In $\mathbf{U}$, each element u_l is the lth day hourly usage of all customers in this cluster. u_l fluctuates with respect to weather conditions. Following the traditional procedure in data mining, u_l is normalized by deducting the mean value and then dividing the standard variance. u'_l denotes the obtained elements. φ^l_t is from weather forecast of hour t of the lth day. φ_t is a 4-dimension real value vector: $\varphi_t = [T, W, D, C]$, where T refers to temperature, W refers to wind speed, D refers to wind direction, and C refers to cloudiness. Each weather feature is also normalized as u_l. A LASSO regressor uses the following equation to estimate u'_l given φ^l_t.

$$\tilde{u'_l} = w_t \cdot \varphi^l_t + b \tag{3.4}$$

In Equation 3.4, w_t is the learned weight vector, and b is the bias item. A LASSO regressor for time t is learned as the following process. The weather feature and the corresponding normalized usage at time t form a pair $(\varphi^l_t \rightarrow u'_l)$, which is regarded as a training sample. L training samples are be obtained in the end. LASSO algorithm [Tib96] is employed to minimize the following regularized cost function,

$$J(w_t) = \frac{1}{L}\sum_{l=1}^{L}(u'_l - \tilde{u'_l})^2 + \frac{\theta}{2L}\|w_t\|_1 \tag{3.5}$$

In Equation 3.5, θ is a parameter to compromise the cost function and the regularization term, and $\|\cdot\|_1$ is the L_1 norm which can perform feature selection. With the trained regressor, power usage u_l can be predicted. Finally, all the predicted usages of M clusters are collected, and the estimated energy demand U_t for hour t is obtained.

The performance of the predictors are monitored to adapt to the dynamics in the retail market. If the predicted result differs from the actual usage to more than a certain criterion for a certain time period, the prediction module is updated with the historical data of subscribed customers. For customers in C_1, the predictors will instantly update their values accordingly with Equation 3.2. For customers in C_2, the average usage $\overline{u}$ is updated by Equation 3.2, and all the predictors for C_2 are re-trained using Equation 3.5. In this manner, the predictors can catch up with the dynamics in the retail market.

3.2.2 The detail design of the wholesale market module

In the wholesale market, the design aims to minimize the cost of energy in one-day-ahead auctions. When energy demand U_t for hour t is estimated, U_t is immediately sent to the auction process. Bidding occurs at the beginning of each hour, thus the broker has 24 opportunities to buy energy of amount U_t. This is a sequential bidding problem. Vytelingum [VCJ08] studied continuous auctions in 2008. Tesauro et al. [TB02] proposed a dynamic programming strategy to optimize the bidding price. In this situation, the auction repeats every 24 hours. Thus, it would be beneficial to employ an MDP [Put14] to model the bidding process. The bidding form is $bid(Q, p)$, where Q is the energy amount and p is the bidding price. The proposed broker model uses Algorithm 2 for an energy auction process in the wholesale market.

Algorithm 2 Algorithm for one energy auction in the wholesale market

```
Input: Historical auction data $\mathbf{A}_t$; Predicted energy demand $U_t$;
Output: Bidding $bid(Q, p_t)$;
1: Initialize: $Q = U_t$;
2: if the length of $\mathbf{A}_t < l$ then
3:     Set $p_t$ as the clear price $p_c$ of the former successful auction
4: else
5:     Optimize $p_t$ using MDP [US14]
6: end if
7: return $bid(Q, p_t)$;
```

Algorithm 2 outputs the optimal bidding price according to the historical auction data and predicted energy demand. Line 1 sets the bidding amount $Q = U_t$. Lines 2-3 deal with the situation lack of historical data, and the bidding price p_t is set as the clear price of the former successful auction p_c. If enough historical data are available, the MDP designed in TaxTac13 [US14] is employed for auction in Lines 4-5.

The MDP is defined as a five tuples $S \times A \times T \times R \times T_S$ as follows.

- **States:** $s \in \{0, 1, \cdots, 24, success\}$, $s_0 := 24$
- **Actions:** bidding price $p_t \in R^+$
- **Transition:** a state $s \in \{1, \cdots, 24\}$ is transited to the terminal state *success* if a bid is fully or partially cleared. Otherwise, the state s is transited to state

$s-1$. The transition probability is defined by $P_T(s, p_t) = m'/m$, where m' is the total cleared energy amount when the clear price is less than the bidding price, and m is the total cleared energy amount. m' and m are accumulated from the historical data of state s.

- **Reward:** The reward for states $s \in \{1, \cdots, 24\}$ is 0, and the reward for state $s = 0$ is the balancing price p_b (initially unknown, accumulated from historical balancing data). For the state *success*, the reward is the p_t.
- **Terminal States:** $\{0, success\}$

The MDP can be solved by a back-sweep, from state 0 to state 24, with the following value function operator:

$$V(s) = \begin{cases} p_b & if \quad s = 0 \\ min_{p_t}\{P_T \times p_t + & \\ (1 - P_T) \times V(s-1)\} & if \quad 1 \le s \le 24 \end{cases} \tag{3.6}$$

At time $t+24$, the bought energy is consumed by customers. Assuming that the actual customer usage is Q, S_A is the cost spent in the auction and S_B is the cost spent for balancing amount (when the obtained energy in auction is less than the actual customer usage). The per kW energy cost W_t is computed, $W_t = (S_A + S_B)/Q$. The cost W_t is sent to the retail market module for profit computing.

3.2.3 The detail design of the retail market module

In the dynamic retail market, there are two objectives to be achieved in the competitions, which are: 1) to keep and attract more customers against other brokers; and 2) to make more profit. There are mainly three types of customers, which are *prosumers*, *general consumers* and *interruptible consumers*. Prosumers can produce energy, such as solar systems. Interruptible consumers can tolerate the energy interruptions for a certain period. General consumers require continuous energy supply. As the three types of customers exhibit different attributes in consuming energy, it is reasonable to publish different tariffs for different types of customers. To optimize different tariffs, independent SARSA [SB98] processes are introduced for different customers. Each SARSA is used to explore the dynamics of its corresponding customers and take proper actions (optimizing energy price). One SARSA process is taken as an instance to detail the strategy in the retail market. This process is illustrated in Algorithm 3.

Algorithm 3 SARSA control for a type of customer in the retail market

```
1: **Initialize:** $Q(s,a) = 0, a = a_6$ (See Table 3.2);
2: **Repeat:**
3:     Take action $a$, and observe the reward $r$ and the next state $s'$;
4:     Calculate $Q(s,a)$ according to Equation 3.7;
5:     Choose the next action $a'$ with $\epsilon$-*greedy* policy based on $Q(s,a)$;
6:     Update: $a = a', s = s'$;
```

In Algorithm 3, Line 1 initializes $Q(s,a)$ and a. Lines 3-6 repeat for the life time of the broker model. Line 3 calculates the immediate reward r and next state feature s' after taking action a. Line 4 calculates $Q(s,a)$ according to Equation 3.7, which will be introduced later. Line 5 chooses next action using ϵ-*greedy* policy based on the value of $Q(s,a)$ in last step. In Line 6, state and action are updated.

The action-value function $Q(s,a)$ is calculated by the following equation,

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)], \tag{3.7}$$

where r is the immediate reward, which is the profit when taken action a to reach state s', α is the learning rate, and γ is a discount parameter. A smaller γ makes stronger emphasis on the current reward.

For state s, it is referre to the work of Peters et al. [PKSTC13], which has explored effective features in the retail market. With the selected features (and the corresponding presentations), Peter et al. has achieved competitive performance in the retail market, so these features are inherited in this work. The selected features for s and the corresponding feature representations are listed in Table 3.1.

Table 3.1: Features and feature representations for state s

Feature	**Definition**	**Representation**
Bias	Constant 1.0	Plain
ActionIndex	Index of selected action	Plain
MarketBreadth	Range from the lowest to highest rate	RBF
MarketShare	Customer percentage of the market	Plain

Actions are designed to effectively response to the dynamic market states [PKSTC13]. Typical actions include increasing price, decreasing price, maintaining current price etc. The full list of action a is illustrated in Table 3.2.

Table 3.2: Actions and action definitions

No.	**Action**	**Definition**
a_1	*Maintain*	keep the same price at time $t+1$ as at time t
a_2	*Max*	publish a tariff with price $p_h + \varepsilon$, where p_h is the highest price, ε is a small value
a_3	*Min*	publish a tariff with price $p_l - \varepsilon$, where p_l is the lowest price
a_4	*Increase*	publish a tariff with price $p_c + \delta$, where p_c is the current price, δ is the amount to increase price
a_5	*Decrease*	publish a tariff with price $p_c - \delta$
a_6	*Average*	publish a tariff with an average market price p_a

As continuous control problems are dealt with in the retail market, function approximation [RN94] is employed. A linear function approximation is used, shown in Equation 3.8.

$$Q(s,a) = \theta F(s,a)^T, \tag{3.8}$$

where $F(s,a)$ is a feature vector, and θ is continually updated by the reinforcement learner.

In above, one SARSA control process has been illustrated. For the three types of cusomters, three SARSA control processes, as illustrated above, are used independently. The three independent SARSA control processes work in parallel, taking the best strategy for each type of customers. Experiments demonstrate that this is a more effective way to compete with other brokers.

3.3 Experiments

The proposed broker model is evaluated by the following four experiments from different perspectives.

- **Experiment 1** aims at testing the performance of the proposed broker model competing with other brokers in Smart Grid market.
- **Experiment 2** is to evaluate the on-line performance of the broker model on supply-demand balance.
- **Experiment 3** aims at evaluating the data-driven method for demand prediction and reveal the interactions between customer clustering and demand prediction.

- **Experiment 4** is to evaluate the performance of the strategy that the proposed model takes in the retail market.

For convenience, the implemented broker model is called **GongBroker**. GongBroker is evaluated on the platform of Power TAC, which simulates real-world Smart Grid market. Power TAC started in 2012 and held every year, and has drawn wide attentions throughout the world. Dozens of brokers have participated in the competition, making Power TAC an authentic platform to evaluate the performances of brokers. This is the reason for us to evaluate GongBroker on the platform of Power TAC.

The game process of Power TAC is briefly described as follows. Power TAC simulates the real-world one hour with a 5-second timeslot, and it simulates 60 days in one game. At the beginning of the competition, each broker receives the bootstrap data with two weeks' length from the Power TAC server. The server also informs the properties of all consumers in the retail market. Besides, the server sends the hourly weather data with a length of two weeks. In each timeslot, brokers get public information including wholesale market clearing price and trading quantities, total energy production and consumption and weather forecasting data. In every 6 timeslots, brokers get the public information on tariffs in the retail market, including new tariffs, revoked tariffs and superseding tariffs. In each timeslot, each broker also gets the private information, including tariff transaction states of all his customers, the production and consumption of his current customers, the wholesale market transactions and the balancing transaction, and the current bank balance. Basing on the market information, brokers can make decisions on actions, such as bidding or asking in the wholesale market, and publishing or modifying tariffs in the retail market. When a game ends, brokers are evaluated by the amount of profit and the balance of supply and demand.

The Power TAC game is utilized to design experiments in this chapter. In Power TAC games, brokers compete in the market to make more profits. This is a straight way to test the performances of broker models. Experiment 1 is taken in the game mode to compare the performance of GongBroker with others. Experiment 4 also makes use of the game mode to evaluate GongBroker's retail strategy. Besides, there are sufficient logs of historical market data supplied by Power TAC game server to analyze a broker model. The logs are used to analyze the performance of demand prediction module in GongBroker in Experiments 2 and 3.

3.3.1 Experiment settings

Four experiments were setup to evaluate and analyze the proposed broker model.

1. **Evaluation of Competition with other brokers**: GongBroker was put

into the Power TAC environment to compete with other excellent brokers. This experiment was a direct evaluation of the overall performance of GongBroker in Smart Grid market.

2. Evaluation of on-line supply-demand balance: A new broker was implemented by replacing the customer demand prediction module by a baseline prediction method. Then the new broker is compared with GongBroker in Power TAC games.

3. Evaluation of off-line demand prediction: GongBroker itself dominates the Power TAC market. Different parameters in customer demand prediction module were set to evaluate the proposed data-driven method according to the logs of Power TAC server.

4. Evaluation of Retail Market Control: A new broker was constructed by replacing the retail market module by only one SARSA control process. Then GongBroker is compared with the new broker in Power TAC games.

The common parameters for the four experiments are set in Table 3.3

Table 3.3: Common parameters for GongBroker in experiments

Module	Parameter	Value
Customer Demand Prediction Module	clustering criterion Δ_1	0.15
	clustering criterion Δ_2	0.05
	weight parameter λ	0.70
Wholesale Market Module	length of time l	5
Retail Market Module	learning rate α	0.66
	discount parameter γ	0.71
	ϵ in ϵ-*greedy*	0.04

In the customer demand prediction module, the clustering criterion is set as $\Delta_1 = 0.15$ for the first layer clustering. For the second layer clustering, the clustering criterion is set as $\Delta_2 = 0.05$. In this setting, $M_1 = 11$ customer clusters were obtained in C_1, and $M_2 = 12$ customer clusters were formed in C_2. To predict the usage amount for customers in C_1, $M_1 \times 24$ predictors were used, and the weight parameter λ in Equation 3.1 was set as 0.70. In the prediction method for customers in C_2, $M_2 \times 24$ LASSO regressors were used. The above settings in customer demand prediction module maintained for experiment 1, 2 and 4. In experiment 3, the parameter Δ_2 was set to different value to evaluate the data-driven demand prediction method. In the wholesale market module, the length l was set to 5, which indicated if there was enough historical auction data. In the retail market module, the parameters for SARSA control was set as follows: $\alpha = 0.66$, and $\gamma = 0.71$. These settings were referred to [PKSTC13]. For the ϵ-*greedy* policy, ϵ was set as 0.04. There are some random factors in the game, but the results are generally consistent. For a fair comparison, 10 games are tested and the average result is reported. In the following subsections, the four experiments are reported and analyzed.

3.3.2 Experiment 1: evaluation of competition with other brokers

In this experiment, other four brokers [b] are introduced to compete with GongBroker in Power TAC games. The four brokers are Sample, LargeBroker, TacTex, and cwiBroker [pt]. Sample broker was provided by the Power TAC, and the other three brokers were excellent brokers in former Power TAC competitions. Two different game modes are designed to test the performance of GongBroker. In Mode 1, GongBroker competed with each of the other four brokers. Each competition was repeated 3 times to avoid the random factors. The averaged profits and imbalance rates of each broker are shown in Figure 3.2 and Figure 3.3, respectively. The imbalance rate here is defined as $r = |U_a - U_p|/U_a$, where U_a is the actual hourly usage and U_p is the predicted usage. In Mode 2, all the brokers joined the game. Three repeated games were held to avoid the randomness.

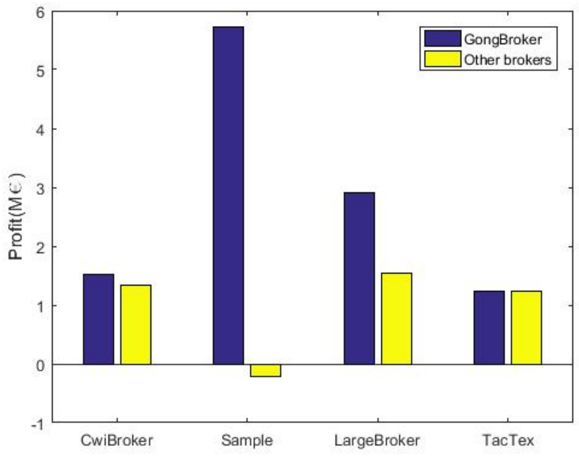

Figure 3.2: Results of averaged profits of 5 brokers in Mode 1

From Figure 3.2, it is seen that in Mode 1, when competing with other brokers, GongBroker made a profit of 1.53, 5.72, 2.90 and 1.23M€. The other four brokers,

[b]There are some excellent broker models in [pt]. Excellent brokers whose relevant literatures can be found were picked, so that they can be compared and analyzed. TacTex [US14] was the champion broker in Power TAC 2013. CwiBroker [LHL14] got the second places in Power TAC 2013 and 2014. LargeBroker was an excellent broker in Power TAC 2012, and its literature is [PKSTC13]. Sample broker was a base line version supplied by Power TAC.

CwiBroker had a profit of 1.33M€, Sample broker made -0.22M€, LargeBroker got 1.55M€, and TacTex gained 1.24M€. GongBroker made a competitive profit with TacTex, and gained larger profits than the other three brokers.

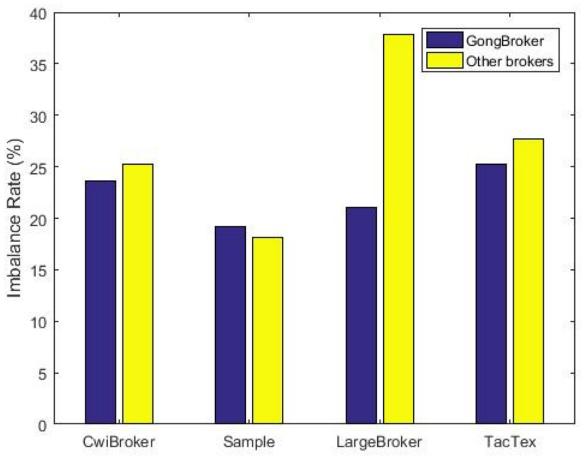

Figure 3.3: Results of averaged imbalance rates of 5 brokers in Mode 1

In Figure3.3, the imbalance rates of GongBroker were 23.6%, 19.2%, 21.1%, 25.2%, respectively, while the imbalance rates of CwiBroker, Sample, LargeBroker and TacTex were 25.3%, 18.1%, 37.8% and 27.7%, respectively. GongBroker showed less imbalance rate against other three brokers but Sample. Sample broker kept a good imbalance rate but it owned only a few customers after a short period of the game. In summary, GongBroker shows the excellent performance in both profit making and supply-demand balancing in games in Mode 1.

The average profits and imbalance rates of each broker in games in Mode 2 are illustrated in Figure 3.4.

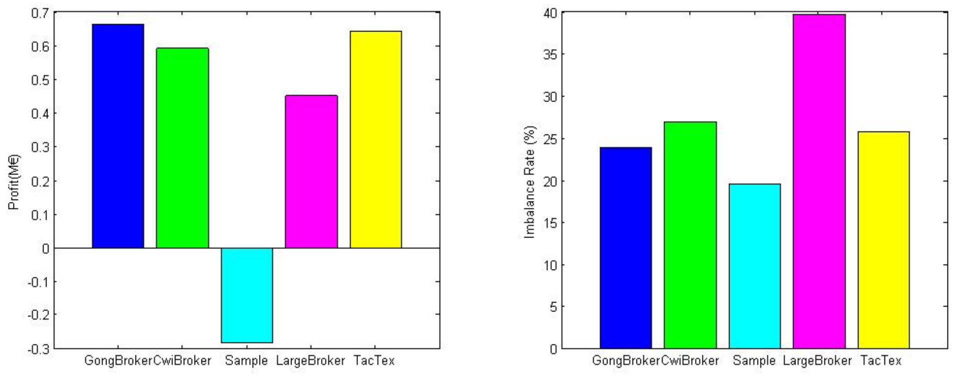

Figure 3.4: Results of averaged profits and imbalance rates of 5 brokers in Mode 2

In games in Mode 2, all the brokers participated in one game. From Figure 3.4, it can be seen that GongBroker made a profit of 0.67M€ with an imbalance rate 23.9%. CwiBroker gained 0.59M€, and its imbalance rate was 26.8%. Sample broker made a profit of -0.27M€, with an imbalance rate 19.3%. LargeBroker had a 0.45M€ profit, and a 39.6% imbalance rate. TacTex gained 0.63M€, and had an imbalance rate of 25.6%. In Mode 2, GongBroker made the most profit and kept a good balance of supply and demand.

Combining the games in Mode 1 and Mode 2, the conclusion is that GongBroker gains advantages in both profit making and supply-demand balance can be drawn. The performance of the five brokers can be ranked in the following order: GongBroker, TacTex (the first two brokers show competitive performances), CwiBroker, LargeBroker and Sample broker. Moreover, some interesting empirical laws in S-mart Grid market can be drawn combining Mode 1 and Mode 2. One is that the profit margin shrinks when there are fierce competitions in market. Games in Mode 2 exhibited stronger competitions than games in Mode 1. The total market profit in Mode 2 is 2.07M€, less than the total profit in any games in Mode 1. Taking a further analysis of the market profits in games in Mode 1, it can be seen from Figure 3.2 that profit margins become smaller when the participated brokers are more competitive. For instance, in the game between GongBroker and TaxTex, the total market profit is 2.47M€, while the market profit is 5.50M€ in the game between GongBroker and Sample. In a competitive environment, brokers publish their tariffs with low prices to attract customers from time to time, thus the profit margin shrinks. Another law is that the imbalance rate of supply demand increases when the market environment is more competitive. In Mode 2, the imbalance rates of all brokers slightly increased comparing to those of in games in Mode 1. In games in Mode 1, GongBroker showed a higher imbalance rate when competing with TacTex than the one when competing with Sample broker. In order to attract customers in the competitive retail market, brokers have to publish cheaper tariffs timely. In this situation, customers may migrate frequently among different brokers, which brings an extra difficulty in demand prediction. That is the reason why imbalance rate increases in a more competitive environment.

3.3.3 Experiment 2: evaluation of supply-demand balance

In this experiment, the performance of GongBroker is evaluated on supply-demand balance by a comparison broker. From this evaluation, the contribution of the customer demand prediction module can be demonstrated. The comparison broker retained the other modules of GongBroker, but replaced the customer demand prediction module with a baseline prediction method. The baseline prediction method was designed as follows. For each customer i, the next-day usage $u_{it}(t = 1, ..., 24)$ was predicted the same as the hourly usage in current. Then by adding up the results for all the customers, the next-day energy demand $U_t, t = 1, ..., 24$ was obtained. The modified broker model was called Broker with Baseline Predictions (BrokerBP). In the comparison experiment, the two brokers participated the game. Three games were repeated. The average results are shown in Table 3.4.

Table 3.4: Performance comparison of GongBroker and BrokerBP

Broker	**Profit (M€)**	**Imbalance (%)**
GongBroker	2.41	32.8
BrokerBP	0.76	81.4

As shown in Table 3.4, GongBroker made a profit of 2.41M€, while BrokerBP had a profit of 0.76M€. The imbalance rate of GongBroker was 32.8%, while imbalance rate of BrokerBP was 81.4%. GongBroker beat BrokerBP in both profit and imbalance rate. As GongBroker showed significant advantages over BrokerBP on balancing supply and demand, the contribution of the customer demand prediction module can be demonstrated. This result also reveals that an effective demand prediction method is critical for supply-demand balance. Moreover, though GongBroker and BrokerBP applied the same strategies to the retail market, GongBroker made much more profit than that of BrokerBP. The poor performance of BrokerBP in supply-demand balance results in a smaller profit, which demonstrates that supply-demand balance is also a critical factor for profit making.

To further analyze the customer demand prediction module, GongBroker and BrokerBP are tested in two independent games. In each game, there was only one broker, GongBroker or BrokerBP, in Smart Grid market. In this game setting, there was relatively no competition. Figure 3.5 illustrates the hourly imbalance rate of GongBroker and BrokerBP in a non-competitive environment.

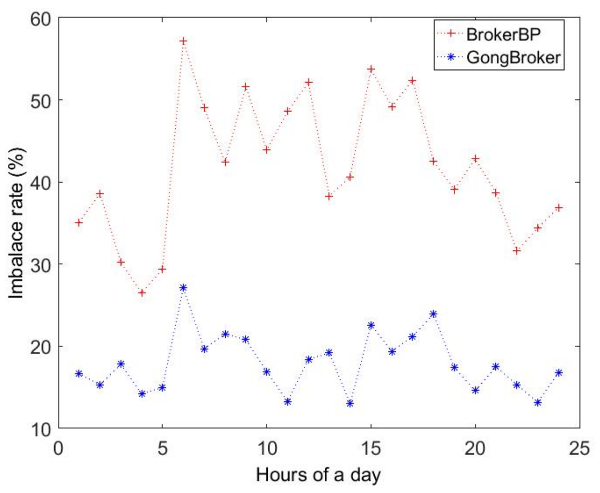

Figure 3.5: Results of imbalance rate of GongBroker and BrokerBP in a non-competitive environment

Figure 3.5 shows that imbalance rates vary in 24 hours. According to commonsense, peak loads occur in 6-8 o'clock in the morning and 6-8 o'clock in the evening. It can be seen that the imbalance rates are higher in the peak hours than those of in other time periods. This phenomenon may attribute to that customers' behaviors are more chaotic in the peak hours. The total averaged imbalance rate of GonBroker is 19.1% and that of BrokerBP is 48.4%. In the environment with no competitions, imbalance rates are less than those in the competitive environment. This result reveals that strong competition brings an extra difficulty in supply-demand balance.

3.3.4 Experiment 3: evaluation of off-line demand prediction

This experiment evaluated the performance of the proposed data-driven method according to the off-line log data. The log data from the server recorded the actual hourly usages of all customers, which could be regarded as the ground-truth of customer demand prediction. GongBroker was configured to dominate the whole Smart Grid market, thus there was no competition in the market. Excluding all the other varying factors, the robustness of the data-driven method can be analyzed. There are two parameters in the hierarchical customer clustering process, i.e. Δ_1 and Δ_2. For Δ_1, it can be determined by constraining the error between the prediction value and the ground-truth value. But for Δ_2, it cannot be determined with ease. Δ_2 is a critical parameter which influences the granular of final customer clusters and affects the final prediction precision. Therefore, Δ_2 is analyzed in details in the following.

Δ_2 is set to three different values: 0.05, 0.10, and 0.15. Apparently, the smaller value of Δ_2 leads to the finer granular of customer clusters. Table 3.5 summarizes how Δ_2 influences the number of clusters and the MAPE of 24 hours. The prediction precision is measured by Mean Absolute Percentage Error (MAPE).

Table 3.5: The influence of Δ_2 on the MAPE and the number of clusters

Δ_2	**Cluster No. in C_1**	**Cluster No. in C_2**	**MAPE(%)**
0.05	11	12	9.6
0.10	8	10	10.7
0.15	6	6	14.4

Table 3.5 shows how Δ_2 influences the MAPE of a day and the number of clusters. A larger Δ_2 results in a less number of customer clusters, thus less predictors are required. However, large value of Δ_2 causes the declination of prediction precision. On the contrary, a too small Δ_2 may result in too many customer clusters

and the required predictors. The above analysis reveals that $0.05 \leq \Delta_2 \leq 0.10$ is reasonable to ensure the performance of the data-driven demand prediction method. In practice, the value of Δ_2 can be compromised between the prediction precision and the number of predictors.

Figure 3.6 shows how MAPE of each hour changes with the value of Δ_2.

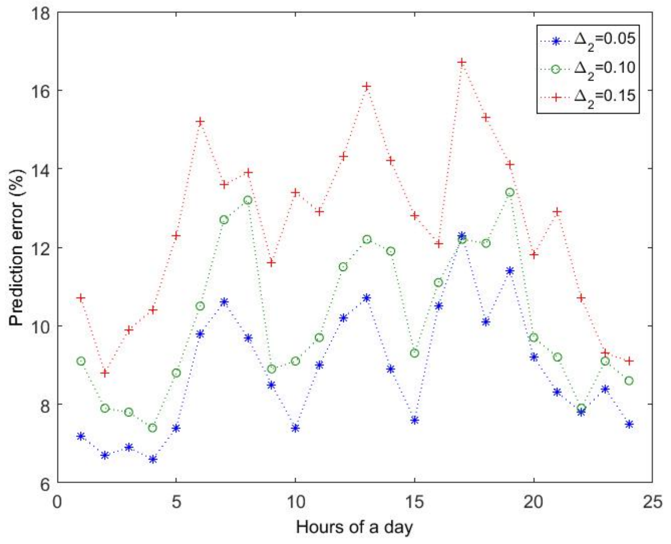

Figure 3.6: Results of prediction errors under different values of Δ_2

In Figure 3.6, when $\Delta_2 = 0.05$, the data-driven method shows the best performance. The prediction error increases when Δ_2 becomes larger. When Δ_2 increases from 0.10 to 0.15, the MAPE of each hour increases more. When Δ_2 ranges from 0.05 to 0.10, the prediction precision is acceptable.

Some profiles of customer clusters are shown when $\Delta_2 = 0.05$. Customer clusters in C_1 show stable power usages, and such clusters include cold storage company, householders with power storage and some office users. Cluster in C_2 have unstable power usages. These clusters are householders, office users and renewable energy producers.

3.3.5 Experiment 4: evaluation of retail market control

This experiment evaluated the retail market control strategy of GongBroker by comparing GongBroker with a newly constructed broker. The comparison broker retained the other modules of GongBroker, but replaced the retail market module with only one SARSA control for all customers. The newly introduced SARSA control shares the same parameters with the three SARSA controls in GongBroker. The modified GongBroker is called Broker with One SARSA Control (BrokerOSC). The performance of the two brokers are compared in three games. The average results are shown in Table 3.6.

Table 3.6: Performance comparison of GongBroker and BrokerOSC

Broker	**Profit (M€)**	**Imbalance (%)**
GongBroker	2.43	27.3
BrokerOSC	1.70	24.7

In Table 3.6, GongBroker made a profit of 2.43M€, while BrokerOSC had a profit of 1.70M€. The imbalance rate of GongBroker was 27.3%, and the imbalance rate of BrokerOSC was 24.7%. It can be seen that imbalance rates of the two brokers were very close, but GongBroker made 0.73M€ profit more than BrokerOSC. Therefore, the profit gap evidences that GongBroker uses better strategies than BrokerBP in the retail market. These comparison games demonstrate that it is an effective method to use independent SARSA controls for different types of customers in the retail market.

3.4 Discussions

Competitive market environments were populated to test the performance of the proposed broker model. Through the competitions with other successful brokers, the advantages of the proposed broker model were demonstrated . The results of the games in two modes show that GongBroker model can make a leading profit and keep a good supply demand balance in different competitive market environments.

More importantly, through the experiments, two empirical laws can be drawn from the competitive market environments.

Law 1: profit margin shrinks when there are fierce competitions in market. In the competitive retail market, brokers offer cheap tariffs to attract customers, resulting in small profit margin.

Law 2: the imbalance rate of supply demand increases when the market environment is more competitive. In the competitive environment, as brokers publish cheap tariffs from time to time, the customers migrate frequently between brokers. The frequent migrations of customers bring extra difficulties in demand prediction, resulting in a high imbalance rate.

A good supply-demand balance can enhance the energy using efficiency. A comparison experiment demonstrated the advantage of the proposed broker model in supply-demand balance. This experiment also verified the effectiveness of the proposed data-driven method for demand prediction. Besides, the comparison experiment showed that supply-demand balance was also a critical factor to gain a high profit for a broker. To further analyze the data-driven method for demand prediction, the key parameter Δ_2, which affected the number of predictors and the prediction precision, was deeply analyzed. A proper value range for Δ_2 is suggested in practice based on evaluation results.

It is vital for a broker to keep and attract customers in the retail market. Reinforcement learning can be applied to generate adaptive tariff prices based on the changing market prices. Through a comparison experiment, it was demonstrated that using independent SARSA controls for different customers was superior than using only one SARSA control for all customers. Independent SARSA controls introduce additional computational cost, but this disadvantage can be easily compensated by parallel computing.

3.5 Summary

This chapter proposed an intelligent broker model for management of Smart Grid market. In proposed broker design, the challenges that brokers face in Smart Grid market were comprehensively considered, and an adaptive and systematic model was constructed to surmount the challenges. The proposed broker model was tested and evaluated on the platform of Power TAC. From the evaluation results, two empirical laws are discovered in the retail market. There is still room to improve the proposed broker model, for instance, the demand prediction part can be further studied for more precise predictions.

Chapter 4

Demand Prediction through Learning Customer Behavior in Smart Grid Market

This chapter develops Load Forecasting through Learning Customer Behavior (LF-LCB) for energy demand prediction, which can further improve demand management in Smart Grid market. A sparse Continuous Conditional Random Fields (sCCRF) is proposed to identify customer behavior through learning. sCCRF analyzes a range of features that are possibly related to the customer's power usage through feature selection and feature weighting. Feature selection determines whether the customer behavior is influenced by a certain feature, while feature weighting further shows how much the power consumption is related to the selected feature. Consequently, all customers can be hierarchically clustered according to feature binarization (whether this feature is selected) and then feature weights. For each customer cluster, a representative sCCRF is fine-tuned to predict its load. Finally, the load for the grid system is obtained by summing the loads of all customer clusters.

The rest of this chapter is organized as follows. Section 4.1 gives a brief introduction to Continuous Conditional Random Fields (CCRF). Section 4.2 proposes sCCRF. In Section 4.3, customer behavior learning using sCCRF is described in detail. The load forecasting process is introduced in Section 4.4. Experimental results are reported in Section 4.5. Section 4.6 further discusses the experimental results and the potential to extend customer behavior learning to other market domains. Finally, conclusions are drawn in Section 4.7.

4.1 An Introduction to CCRF

In this section, the concept of CCRF is introduced. As CCRF is originated from Conditional Random Fields (CRF), this section first introduces CRF, and then extends CRF to CCRF.

4.1.1 Conditional random fields

CRF [Laf01] was initially proposed for labeling sequence data. The chain-structured CRF, as illustrated in Figure 4.1, is widely used.

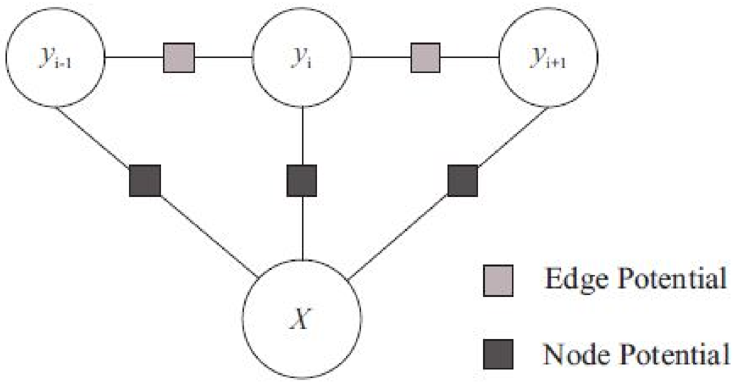

Figure 4.1: An illustration of a CRF with a chain structure.

Assume that $X = \{x_1, x_2, \cdots, x_m\}$ is the given sequence of observations, and $Y = \{y_1, y_2, \cdots, y_n\}$ is the label sequence to be predicted. CRF defines the conditional probability $P(Y|X)$ in Equation 4.1.

$$P(Y|X) = \frac{1}{Z(X)} exp(\Psi), \tag{4.1}$$

where Ψ is the energy function, and $Z(X)$ is the partition function that normalizes $P(Y|X)$.

The energy function Ψ is further defined as

$$\Psi = \sum_i \sum_{k=1}^{K_1} \alpha_k f_k(y_i, X) + \sum_{i,j} \sum_{k=1}^{K_2} \beta_k g_k(y_i, y_j, X), \tag{4.2}$$

where $f_k(y_i, X)$ is called node potential, $g_k(y_i, y_j, X)$ is called edge potential, and α_k and β_k are corresponding weight parameters. In the energy function, the node potential captures the associations between inputs and outputs, while the edge potential captures the interactions between conditioned outputs.

The partition function $Z(X)$ is defined in Equation 4.3.

$$Z(X) = \sum_Y exp(\Psi) \tag{4.3}$$

CRF explicitly defines $P(Y|X)$, which indicates that Y is determined by the whole observation X. Therefore, CRF can model the whole observed sequence for the output.

4.1.2 Continuous conditional random fields

The CRF model outputs discrete values, while CCRF extends CRF to output real values. The definition of CCRF [QLZ+09] differs from CRF in three aspects. 1) The output $Y = \{y_1, y_2, \cdots, y_n\}$ can be a real value sequence. 2) The partition function $Z(X)$ is alternatively defined as:

$$Z(X) = \int_Y exp(\Psi) \tag{4.4}$$

3) The weights $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ must be positive to ensure the partition function is integrable.

To learn a CCRF model, maximum log-likelihood is used to find the optimal weights $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Given training data $D = \{(X, Y)\}_1^Q$, where Q is the total number of training samples, the log-likelihood $L(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is maximized:

$$(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) = \mathrm{argmax}_{(\boldsymbol{\alpha},\boldsymbol{\beta})}(L(\boldsymbol{\alpha}, \boldsymbol{\beta})), \tag{4.5}$$

where

$$L(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{l=1}^{Q} logP(Y_q|X_q) \tag{4.6}$$

The inference of CCRF is to find the most likely value for Y_k, provided an observed sequence X_k:

$$\hat{Y_k} = \mathrm{argmax}_{Y_k}(P(Y_k|X_k)) \tag{4.7}$$

Radosavljevic et al. [RVO10] have shown that $P(Y|X)$ with quadratic potentials can be transformed into a multivariate Gaussian, which facilitates the learning and inference processes. Therefore, this chapter designs quadratic node and edge potentials to take advantage of the multivariate Gaussian form.

4.2 Sparse Continuous Conditional Random Fields

sCCRF is proposed in this section. L_1 norm is introduced to regularize CCRF because L_1 penalty can result in a sparse model. However, L_1 norm is not differentiable at zero, thus the previous learning method for CCRF can no longer be applied to sCCRF. Some special methods have been proposed to tackle the learning with L_1 norm penalty [AG07, YVGS10]. Orthant-Wise Limited-memory Quasi-Newton (OWL-QN), proposed by Andrew and Gao [AG07], has been verified an advantageous algorithm for L_1-regularized log-linear model in [LCY10]. Therefore, OWL-QN algorithm is employed to train sCCRF.

4.2.1 Introducing L_1 norm to regularize CCRF

In previous CCRF [QLZ+09, RVO10], the partition function takes the following form:

$$Z(\mathbf{X}) = \int_{\mathbf{y}} exp(\sum_i \sum_{k=1}^{K_1} -\alpha_k(y_i - \mathbf{X}_{i,k})^2 + \sum_{i,j} \sum_{k=1}^{K_2} -\delta_k \beta_k (y_i - y_j)^2) \tag{4.8}$$

In Equation 4.8, $\mathbf{X}$ denotes the feature matrix and $\mathbf{y}$ denotes the sequence to be predicted. When the variables in $\mathbf{X}$ and $\mathbf{y}$ are defined in infinite domains, both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are required to be positive to ensure that the partition function is integrable. However, this sufficient condition seems too strict. Glass et al. [GGVO16] extended the definitions of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ so as to extend the modeling capacity of CCRF. The constraints of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ will be discussed in Subsection 4.3.2 after CCRF is transformed to a multivariate Gaussian.

In machine learning, regularization has been commonly used in the learning process to achieve a model that generalizes to unseen data. L_2 norm regularization has been used in [BBR13, RVO10, Guo15] in learning CCRF. L_1 norm regularizer has been theoretically studied in [Ng04] by Ng, and in practice, the L_1 norm regularizer has gained roughly the same accuracy as the L_2 norm regularizer [LCY10]. L_1 norm also has a favorable property of selecting effective features, which can be utilized to analyze customer behavior in this research. Therefore, L_1 norm is introduced to regularize the CCRF in the learning procedure. $\boldsymbol{\lambda} =< \boldsymbol{\alpha}, \boldsymbol{\beta} >$, a concatenation of vector $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, is introduced to compactly represent the weights. The objective function to be minimized for sCCRF is designed in Equation 4.9,

$$F(\boldsymbol{\lambda}) = -L(\boldsymbol{\lambda}) + \rho \|\boldsymbol{\lambda}\|_1, \tag{4.9}$$

where $\|\|_1$ stands for L_1 norm. In the objective function, the first term is the loss function, which is a negative log-likelihood of the training set (see Equation 4.6), and the second term is the L_1 norm of $\boldsymbol{\lambda}$, used as a regularization term. The parameter ρ compromises the loss and the regularization term.

4.2.2 Learning sCCRF

For sCCRF learning, this chapter introduces the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) algorithm, which in fact extends L-BFGS [LN89] algorithm for convex functions with L_1 penalty. The quasi-Newton algorithms gain the advantage of the second-order convergence rate with a small computation cost. These algorithms construct an approximation of the second-order Taylor expansion of the objective function, and then try to minimize the approximation. In the approximated Taylor expansion, the Hessian matrix is constructed with the first-order information gathered from previous steps. OWL-QN, which modifies L-BFGS, is motivated by the following basic idea. When the orthant is given, L_1 norm can be determined and be differentiable. Furthermore, L_1 norm is not related to the Hessian, which can be approximated by the loss term alone. Thus, OWL-QN in fact imitates L-BFGS steps in a chosen orthant.

The learning procedure for sCCRF using OWL-QN is summarized in Algorithm 4.

Algorithm 4 sCCRF learning using OWL-QN

```
Input: Training samples $D = \{(X, Y)\}_1^Q$;
Output: Weight parameter vector $\boldsymbol{\lambda}$;
1: Initialize: Initial point $\boldsymbol{\lambda}^0$; $S \Leftarrow \{\}$, $R \Leftarrow \{\}$.
2: for $k = 0$ to $T$ do
3:     Compute the pseudo-gradient $\diamond F(\boldsymbol{\lambda})$
4:     Choose an orthant $\boldsymbol{\xi}^k$
5:     Construct $\mathbf{H}_k$ using $S$ and $R$
6:     Compute search direction $\mathbf{p}^k$
7:     Find $\boldsymbol{\lambda}^{k+1}$ with constrained line search
8:     if termination condition satisfied then
9:         Stop and return $\boldsymbol{\lambda}^{k+1}$
10:    end if
11:    Update $S$ with $\mathbf{s}^k = \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k$
12:    Update $R$ with $\mathbf{r}^k = -\nabla L(\boldsymbol{\lambda}^{k+1}) + \nabla L(\boldsymbol{\lambda}^k)$
13: end for
```

Before the explanation of Algorithm 4, two special functions [AG07] are introduced for convenience. The first one is *sign function* σ: $\sigma(x)$ results in a value in $\{-1, 0, 1\}$ according to whether x is negative, zero or positive. The second one is *project function* π: $\pi(\mathbf{x}; \mathbf{y})$, $\mathbb{R}^n \mapsto \mathbb{R}^n$, is parameterized by $y \in \mathbb{R}^n$, where

$$\pi_i(x; y) = \begin{cases} x_i & if \quad \sigma(x_i) = \sigma(y_i) \\ 0 & otherwise \end{cases} \tag{4.10}$$

It can be interpreted as projecting $\mathbf{x}$ onto the orthant defined by $\mathbf{y}$.

In Algorithm 4, Line 1 chooses initial $\boldsymbol{\lambda}$, and initialize sets S and R. S is for displacements $\mathbf{s}^k = \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k$, and R is for changes in gradient $\mathbf{r}^k = -\nabla L(\boldsymbol{\lambda}^{k+1}) + \nabla L(\boldsymbol{\lambda}^k)$. Lines 2-13 are the main iteration loop. Line 3 calculates the pseudo-gradient of $F(\boldsymbol{\lambda})$ at $\boldsymbol{\lambda}$, according to the following equation:

$$\diamond_i F(\lambda) = \begin{cases} \partial_i^- F(\lambda) & if \quad \partial_i^- F(\lambda) > 0 \\ \partial_i^+ F(\lambda) & if \quad \partial_i^+ F(\lambda) < 0 \\ 0 & otherwise \end{cases}, \tag{4.11}$$

where

$$\partial_i^{\pm} F(\lambda) = -\frac{\partial}{\partial \lambda_i} L(\lambda) + \begin{cases} \rho\sigma(\lambda_i) & if \quad \lambda_i \neq 0 \\ \pm\rho & if \quad \lambda_i = 0 \end{cases} . \tag{4.12}$$

In Equation 4.12, the term $\partial L(\lambda)/\partial \lambda_i$ is derived with respect to the specified model. In Line 4, an orthant $\boldsymbol{\xi}^k$ is chosen based on $\diamond F(\boldsymbol{\lambda})$,

$$\xi_i^k = \begin{cases} \sigma(\lambda_i^k) & if \quad \lambda_i^k \neq 0 \\ \sigma(- \diamond_i F(\boldsymbol{\lambda}^k)) & if \quad \lambda_i^k = 0 \end{cases} . \tag{4.13}$$

Line 5 constructs the inverse of Hessian $\mathbf{H}_k$, which is constructed the same as the traditional L-BFGS [LN89]. Line 6 then determines the search direction $\mathbf{p}^k$, formulated by

$$\mathbf{p}^k = \pi(\mathbf{H}_k v^k; v^k), \tag{4.14}$$

where $v^k = - \diamond F(\boldsymbol{\lambda}^k)$. Lines 7-10 aim at finding the next point $\boldsymbol{\lambda}^{k+1}$ using constrained line search, in which each point explored is projected back onto the chosen orthant: $\boldsymbol{\lambda}^{k+1} = \pi(\boldsymbol{\lambda}^k + \alpha\mathbf{p}^k; \boldsymbol{\xi}^k)$, where α controls the search step. Lines 11 and 12 update sets S and R, respectively.

4.3 Learning Customer Behavior to Aggregate Customers

Customer aggregation tries to "smooth" the random behaviors of customers by clustering similar customers into the same group. Based on the "averaged" data of a customer cluster, a better predictor can be learned. Figure 4.2 shows the pipeline of the customer behavior learning process, which is composed of four steps. **1)** The load forecasting problem for each customer is modeled using sCCRF; **2)** the sCCRF model is initially learned for behavior analysis; **3)** all the customers are hierarchically clustered based on different customer behavior patterns; and **4)** the representative sCCRF is fine-tuned for each customer cluster, In the following subsections, the four steps are described in details.

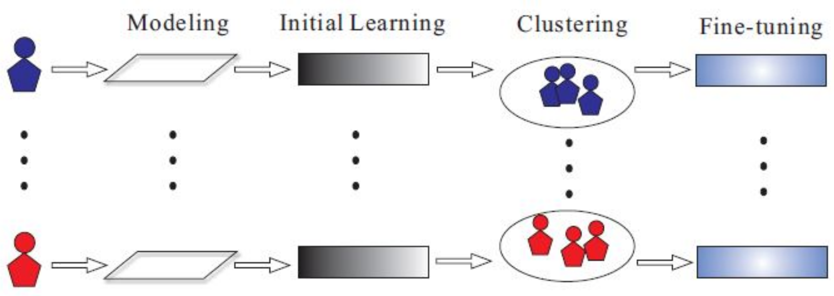

Figure 4.2: The pipeline of learning customer behavior to aggregate customers.

4.3.1 Model design

CCRF discriminatively models the conditional probability $P(Y|X)$. In this research, a vector $\mathbf{y} = [y_1, y_2, \cdots, y_n]^T$ is used to denote the hourly power usages to be predicted. The observations X are specified with a matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]^T$, where each row $\mathbf{x}_i$ represents the observed D-dimension feature vector for hour i.

In the proposed model, both node potentials and edge potentials bear a quadratic form. The node potential is defined as follows.

$$f_k(y_i, \mathbf{X}) = -(y_i - \mathbf{X}_{i,k})^2 \tag{4.15}$$

For an output y_i, node potential associates it with the current observed feature vector $\mathbf{x}_i$. As the number of features in $\mathbf{x}_i$ is D, D node potentials are generated for the current observation. The raw features after normalization are directly used in the node potential, so that how much a feature relates to certain customer's load can be analyzed.

The edge potential, which captures the interactions between outputs, is defined as follows.

$$g_k(y_i, y_j, \mathbf{X}) = -\frac{1}{2} s^k_{i,j} (y_i - y_j)^2, \tag{4.16}$$

where $s^k_{i,j}$ is an indicator function defined as follows,

$$s^k_{i,j} = \begin{cases} 1 & if \quad |i-j| \leq m \\ 0 & otherwise \end{cases}, \tag{4.17}$$

where m is the number of neighboring variables taken into account.

This study takes m closest neighboring variables into account. With the consideration of multiple neighboring variables, load forecasting for each customer can be modelled more accurately, and hence improve the accuracy of load forecasting in a grid system. Here, m is a critical parameter that influences the performance of LF-LCB. In the experiment part, Subsection 4.5.2 further analyzes the choice of a proper m.

For regression problems, edge potentials suffer a weak feature constraint problem [Guo15], which is briefly explained as follows. CRF is a maximum entropy model with feature constraints to perform structural learning. CRF learning aims to force the expectation of each feature with respect to the model to equal to that with respect to the learning data. For the binary features in conventional CRF, knowing the mean is equivalent to knowing its full distribution. In contrast, the mean does not contain much information of the distribution of a continuous variable in CCRF. This is the cause of the weak feature constraint problem.

Following the work by Guo [Guo15], Predictive Clustering Trees (PCTs)

[VVL07] are introduced to tackle this problem. PCTs use a tree structure to supply rich feature constraints, which indeed divide the distribution of a continuous variable into several sub-distributions. Here, Δx is introduced to denote the change of neighboring features, and Δy is to denote the change between y_i and y_j. The PCTs provide more sophisticated relationships between Δy and Δx through dividing $s_{i,j}^k$ into more indicator functions $\delta_k^{(p)}$. The value of $\delta_k^{(p)}$ is determined by its corresponding assertion. When the assertion holds, its value is "1"; otherwise "0". Figure 4.3 uses the temperature feature as an example to illustrate how PCTs work. When the assertion that Δx is small (similar temperatures in the two hours) holds, the value of $\delta_k^{(1)}$ is "1" and PCTs stop expanding. When Δx is not small, $\delta_k^{(1)}$ is "0" and PCTs continues expanding. Similar processes repeat for $\delta_k^{(2)}$ and $\delta_k^{(3)}$.

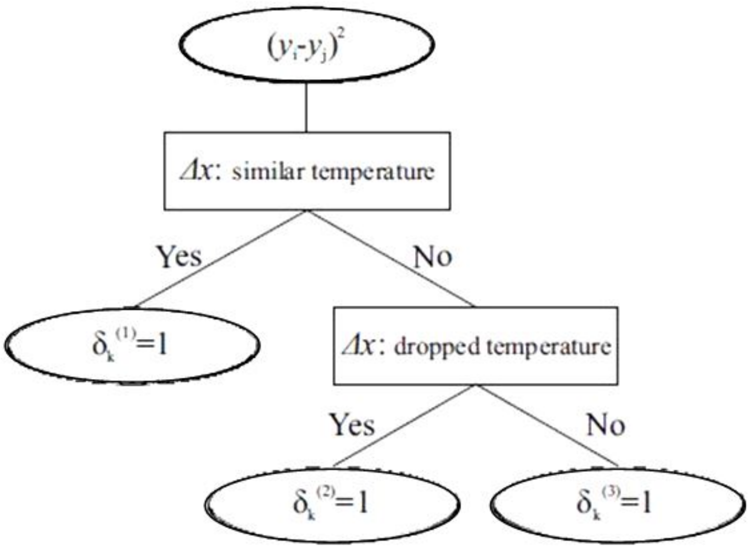

Figure 4.3: An illustration of how PCTs work with respect to the temperature feature. When the temperature does not change much, the branch $\delta_k^{(1)}$ holds. If temperature goes down, it goes to the branch of $\delta_k^{(2)}$. If temperature goes up, the branch $\delta_k^{(3)}$ holds. Therefore, three levels of PCTs are sufficient to supply information of temperature change.

In this situation, three different indicator functions ($P = 3$) can represent the changes (going up, going down or staying similar) of a feature, and therefore are adequate to supply sufficient relationship information between Δy and Δx. As the assertions in PCTs are mutually exclusive, only one assertion holds in the end. Thus, it is obtained $s_{i,j}^k = \sum_{p=1}^{P} \delta_k^{(p)}$. In another view, through the divisions by PCTs, more indicator functions $s_{i,j}^k$ are obtained. In the following text, this chapter can only use $s_{i,j}^k$ and no longer mention $\delta_k^{(p)}$.

PCTs are applied to some features that may fluctuate so as to influence Δy, such as the temperature, the market price and so on. Meanwhile, it is not necessary to introduce PCTs to temporal features. The features, which PCTs are applied to, are further described in experiment settings in Subsection 4.5.1.

With the node potential in Equation 4.15 and the edge potential in Equation 4.16, the resultant CCRF model is formulated as follows.

$$\begin{aligned} P(\mathbf{y}|\mathbf{X}) = \frac{1}{Z(\mathbf{X})} \cdot exp(-\sum_{i=1}^{n}\sum_{k=1}^{D} \alpha_k (y_i - \mathbf{X}_{i,k})^2 - \\ \frac{1}{2}\sum_{i,j}\sum_{k=1}^{S} \beta_k s_{i,j}^k (y_i - y_j)^2) \end{aligned} \tag{4.18}$$

Note that β_k is shared across different pairs of neighboring variables. It is tried to introduce a parameter for each neighboring variables, but slightly performance decay is observed. Therefore, β_k are shared in edge potentials.

Following Radosavljevic's work [RVO10], the CCRF in Equation 4.18 can be derived into the following multivariate Gaussian form to facilitate learning and inference

$$P(\mathbf{y}|\mathbf{X}) = \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}|^{1/2}} \cdot \quad exp(-\frac{1}{2}(\mathbf{y} - \mu(\mathbf{X}))^T \mathbf{\Sigma}^{-1} (\mathbf{y} - \mu(\mathbf{X}))) \tag{4.19}$$

In this Gaussian form, the precision matrix $\mathbf{\Sigma}^{-1}$, is the sum of two $n \times n$ matrices, further expressed as follows.

$$\begin{aligned} \mathbf{\Sigma}^{-1} &= 2(\mathbf{M}^1 + \mathbf{M}^2), where \\ \mathbf{M}_{i,j}^1 &= \begin{cases} \sum_{k=1}^{D} \alpha_k & if \quad i = j \\ 0 & if \quad i \neq j \end{cases} \\ \mathbf{M}_{i,j}^2 &= \begin{cases} \sum_{k=1}^{S} \beta_k \sum_{r=1}^{n} s_{i,r}^k - \sum_{k=1}^{S} \beta_k s_{i,j}^k & if \quad i = j \\ -\sum_{k=1}^{S} \beta_k s_{i,j}^k & if \quad i \neq j \end{cases} \end{aligned} \tag{4.20}$$

The diagonal matrix $\mathbf{M}^1$ represents the contribution of $\boldsymbol{\alpha}$ terms (node potentials), and the symmetric matrix $\mathbf{M}^2$ represents the contribution of $\boldsymbol{\beta}$ terms (edge potentials). The mean $\mu(\mathbf{X})$ is computed by

$$\mu(\mathbf{X}) = \mathbf{\Sigma} \cdot \boldsymbol{\theta}, \tag{4.21}$$

where $\boldsymbol{\theta}$ is an n-dimension vector, where each element is calculated by

$$\theta_i = 2\sum_{k=1}^{D} \alpha_k \mathbf{X}_{i,k} \tag{4.22}$$

4.3.2 Learning an sCCRF model

Algorithm 4 is used to optimize the weights of built CCRF model. The constraints on parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are discussed first. Here, a simple way is proposed to tackle these constraints. CCRF has been derived into a multi-variable Gaussian form, the constraint inherently come from the positive-definiteness of the precision matrix $\boldsymbol{\Sigma}^{-1}$ [RVO10]. This constraint can be incorporated into the line search step in OWL-QN optimization process. In the line search step, the step size is chosen to ensure both descent of objective function (Equation 4.9) and positive-definiteness of the precision matrix. This is similar to the trick used in [WK13]. Therefore, this chapter does not explicitly derive the constraints on parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

The gradient of α_k and β_k of $-L(\boldsymbol{\lambda})$ are derived with respect to the model in Equation 4.18.

$$\nabla\alpha_k = -\frac{\partial L(\boldsymbol{\alpha},\boldsymbol{\beta})}{\partial\alpha_k} = -\frac{\sum_{q=1}^{Q}\partial logP(\mathbf{y}^{(q)}|\mathbf{X}^{(q)})}{\partial\alpha_k} \tag{4.23}$$

$$\nabla\beta_k = -\frac{\partial L(\boldsymbol{\alpha},\boldsymbol{\beta})}{\partial\beta_k} = -\frac{\sum_{q=1}^{Q}\partial logP(\mathbf{y}^{(q)}|\mathbf{X}^{(q)})}{\partial\beta_k} \tag{4.24}$$

Based on $\nabla\alpha_k$ and $\nabla\beta_k$, Algorithm 4 (refer to Subsection 4.2.2) is used to minimize the objective $F(\boldsymbol{\lambda})$ to obtain the optimal weights for sCCRF.

4.3.3 Aggregating customers

Customer behavior can be represented by the weight vector $\boldsymbol{\lambda}$ of sCCRF. In the learned weights $\boldsymbol{\lambda} =< \boldsymbol{\alpha}, \boldsymbol{\beta} >$ for a certain customer, each α_k reveals how much a feature influences the power usage of the customer, and each β_k reveals how much the feature variance influences the change of hourly usage. As sparse CCRF is used, the weights of unrelated features have been pushed to zero, and the rest features with non-zero weights reflect how much the load is influenced by the related features.

With the obtained sparse feature weights, a hierarchial clustering process can be designed to aggregate customers, which is illustrated in Figure 4.4.

First, each weight in $\boldsymbol{\lambda}$ is binarized. To be specific, all the non-zero weights are converted into "1", and the zero values remain. In the first layer, all customers are clustered according to the binarized weights. Customers who have the same binarized weights fall into the same cluster. Afterwards, each cluster in the first layer is further clustered according to the customers' non-zero weights to form clusters in the second layer. K-means, with a Euclidean distance criterion Δ, is utilized in the second layer clustering. Δ is a critical parameter, which determines the number of final clusters and influences the final accuracy of load forecasting. This parameter

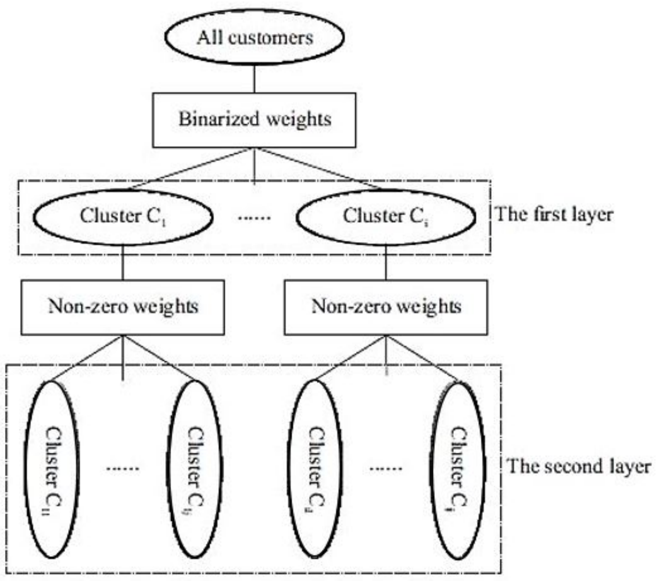

Figure 4.4: An illustration of clustering customers. All customers are first clustered according to the binarized weights (feature selection results). Then each cluster in the first layer is further clustered according to the non-zero weights using K-means method.

is further analyzed in Subsection 4.5.2 in experiments.

A two-layer clustering tree is obtained (see Figure 4.4), and clusters in each layer indicate clear physical meanings. In the first layer clustered by the binarized weights, the features with weights “1” are related to the customers’ power usage. Thus, customers in the same cluster are influenced by the same range of features. In practice, customers in this layer can be certain customer genres such as wind power producers, householders and office users. In the second layer, each cluster C_k is further divided into smaller clusters according to the non-zero weights, which indicate how much each feature influences the customer’s power usage. After a second clustering, customers in each smaller cluster C_{kl} share similar sensitivity to the range of features. One example is the office buildings. Office buildings in one cluster may adaptively adjust their power usage according to temperature, while buildings in another cluster are less sensitive to the influence of temperature.

The two-layered clustering process results in reliable customer clusters since it circumvents the “curse of dimensionality” in cluster problems. Benefiting from sCCRF, the resultant weight parameters are sparse and consequently two-layered clustering can be designed to aggregate customers. The introduction of the sparse learning model and the two-layered clustering construct a coherent process to effectively aggregate customers.

4.3.4 Fine-tuning sCCRFs

After obtained the clustering tree, an sCCRF is fine-tuned for each cluster in the second layer. In Subsection 4.3.2, an sCCRF has been learned for each customer. For a customer cluster, a learned sCCRF is selected and fine-tuned. Fine-tuning an sCCRF for each cluster gains two advantages. 1) Increasing prediction precision: for an individual customer, its behavior is chaotic, and its load is hard to predict. On the contrary, the customers' usage data seem to be "smoothed" in a customer cluster. 2) Reducing computational cost: for a cluster with N customers, only one fine-tuned sCCRF is needed in the end.

An effective way is presented to select representative sCCRF in order to improve the efficiency of fine-tuning. The new selection rule is as follows. The cluster center is defined as the center of the minimal sphere that contains all the customers' non-zero weight vector. For a cluster with N customers, the cluster center is found first. The customer whose non-zero weight vector is closest to the center is targeted, and its corresponding sCCRF is selected.

Fine-tuning an sCCRF for a cluster is quite straight-forward. For the selected sCCRF, the input feature $\mathbf{X}_q$ remains, while the truth of load becomes the average load of all customers in the cluster (note that load data are normalized). Then Algorithm 4 is employed to fine-tune this sCCRF with the input features and new ground-truth. The fine-tuning process results in a quick convergence, because the weights in the selected sCCRF have been close to the optimal weights of the final sCCRF.

4.4 Load Forecasting

With the learned sCCRF for each customer cluster, the hourly load can be predicted. Summing the predicted load for each cluster, the final load for the grid system can be obtained.

To predict the load for each customer cluster, it is tried to find the most likely $\mathbf{y}$ given the observed feature $\mathbf{X}$, as formulated in Equation 4.7. Benefiting from the multivariate Gaussian form, the inference becomes quite tractable. To maximize $P(\mathbf{y}|\mathbf{X})$ in the multivariate Gaussian (see Equation 4.19), it simply makes $\mathbf{y}$ equal to $\mu(\mathbf{X})$,

$$\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}}(P(\mathbf{y}|\mathbf{X})) = \mu(\mathbf{X}) = \mathbf{\Sigma} \cdot \boldsymbol{\theta} \tag{4.25}$$

Assuming there are N customer clusters formed in the grid system, adding up the predicted load of each cluster $\hat{\mathbf{y}^i}$ in element wise, the final load $\mathbf{y}^W$ of the grid system is obtained by the following equation.

$$\mathbf{y}^W = \sum_1^N \hat{\mathbf{y}^i} \tag{4.26}$$

Besides the exact inference, due to the Gaussian distribution, the 95%-confidence intervals of the approximate outputs can be obtained by the following Equation:

$$\tilde{\mathbf{y}} = \hat{\mathbf{y}} \pm 1.96 \times diag(\mathbf{\Sigma}) \tag{4.27}$$

Equation 4.27 may assist decision makings in uncertain environments.

4.5 Experiments and Analysis

Four experiments were conducted from different perspectives to evaluate LF-LCB.

Experiment 1: Analysis of internal parameters. As both the number of neighboring variables m and clustering criterion Δ affect the final load forecasting result, this experiment tries to find optimal values of m and Δ for practical use of LF-LCB. This experiment also justified the value of m and Δ used in the following experiments.

Experiment 2: Comparing sCCRF to state-of-the-art methods in load forecasting. This experiment compares sCCRF with CCRF [Guo15], Support Vector Regression (SVR) [DBK$^+$97] and Convolutional Neural Networks (CNN) [DQH17] on load forecasting for two typical customers.

Experiment 3: Evaluation of customer behavior learning. Two other prediction methods based on sCCRF without the consideration of customer behavior were constructed. This experiment compares LF-LCB with the two methods to demonstrate the advantage of introducing customer behavior learning to load forecasting. Moreover, customer clusters and load forecasting results of LF-LCB are visualized and analyzed.

Experiment 4: Comparisons with related customer aggregation methods. LF-LCB is compared with hand-crafted customer clusters for load forecasting, a simplified LF-LCB that considers only closest neighboring variables. and Alzate and Sinn's work that utilized spectral clustering to aggregate customers [AS13].

Experiments were conducted on the platform of Power Trading Agent Competition (Power TAC) [KCRW14]. Power TAC has drawn wide attentions and has become a benchmark in the Smart Grid research community. Power TAC simulates a variety of customers with various behaviors in a grid system. There are also rich features, including real-world weather conditions and real-time market status. Besides, the Power TAC server supplies rich logs of customers' hourly power usage, which are regarded as the ground-truth to evaluate the proposed LF-LCB.

4.5.1 Experiment settings

Features used in this study include temporal features, calendar features, weather features and market features. Table 4.1 lists the contents of the four types of features, where indexes are used for further discussion.

Table 4.1: Features used in LF-LCB

Feature	**Content**	**Index**
Temporal feature	hour of a day	t_1
	day of a week	t_2
Calendar feature	is or not holiday	c_1
Weather feature	temperature	w_1
	wind strength	w_2
	wind direction	w_3
	cloudiness	w_4
Market feature	lowest price	m_1
	average price	m_2

This study configured the Power TAC server and weather data server, and utilized Power TAC games to generate training and test data. Training data were generated by six games based on the data in 2009. Test data were from six games according to the data in 2010. The logged customers' usages were regarded as the ground-truths of loads to be predicted. To induce rich features, three broker models, TacTex [US14], cwiBroker [LHL14] and GongBroker [WZRI15] were introduced to compete in the games. 40 customers, each with a certain population, were simulated in Power TAC. Two customers with populations up to tens of thousands, were split into customers with population of 100. The split customers were then rendered with some random behaviors in the log data, such as power usage decreasing for going out at night, or load increasing for having a party at home. In the end, 538 customers were obtained. These customers were manageable for experiments and sufficient to analyze LF-LCB.

The configurations in LF-LCB are described in detail. sCCRF modeled 24-hour power usage sequence under the influences of hourly features. For the features in each hour, 9 node potentials (corresponding to 9 features) were generated. Edge potentials were generated in m closest neighboring variables, where m is determined in

Experiment 1. and PCTs were applied to weather and market features, which fluctuate to influence the power usage. Meanwhile, it was not necessary to apply PCTs to temporal and calendar features. To ensure that the sCCRF for each customer was converged, 50 iterations were set for OWL-QN. For the fine-tuned sCCRF for each customer cluster, 10 iterations were sufficient to ensure convergence. sCCRF was implemented in Matlab, and OWL-QN was implemented based on minFunc [Sch05].

In the proposed LF-LCB, there are three hyper-parameters. One is ρ in the cost function of sCCRF (Equation 4.9), which can be determined by cross-validation. The other two are m that defines the number of neighboring variables taken into account and Δ that determines the granularity of the customer clusters. As the two parameters greatly affect the performance of LF-LCB, This experiment shows how to choose proper m and Δ in Experiment 1. For the other experiments, this chapter sets $m = 2$ and $\Delta = 0.05$, which are optimal values found in Experiment 1.

4.5.2 Experiment 1: analysis of internal parameters

Δ and m are two important parameters in LF-LCB. Grid search is used to find the optimal values for Δ and m. m was searched in the outer loop and Δ was search in the inner loop. Three different values were set for m: 1, 2, 3; while four different values were set for Δ: 0.025, 0.05, 0.75, and 0.10. LF-LCB process repeated with different Δ and m. For the predicted power usage in each hour, Mean Absolute Percentage Error (MAPE) is used to measure the precision of load forecasting. The overall MAPE (an average of MAPE in 24 hours) indicates the performance of load forecasting. With the grid search, optimal parameters $m = 2$ and $\Delta = 0.05$ are found.

This experiment further analyzes the influence of m and Δ independently. Table 4.2 shows the number of clusters and the overall MAPE under different Δ values when $m = 2$.

Table 4.2: The influence of Δ on the MAPE and the number of clusters

Δ	**Number of clusters**	**overall MAPE(%)**
0.025	60	5.33
0.05	24	4.08
0.075	23	4.33
0.10	17	5.41

In Table 4.2, when Δ was set as 0.025, 60 customer clusters were formed, and the total MAPE was 5.33%. When Δ was 0.05, 24 customer clusters were obtained, and the total MAPE became much better. Comparing the above two settings, it can be seen that a small Δ results in fine granularity of customer clusters. When the customer cluster is too small, the "smoothness" of load data is compromised. That is why a small Δ leads to a less competitive prediction result. When Δ was set to 0.075, 23 customer clusters were formed, and the total MAPE was 4.33%. This indicates that when Δ changes from 0.05 to 0.075, the performance of LF-LCB does not change much. Besides, as the number of clusters also determines the required final sCCRFs, the range from 0.05 to 0.075 can result in a small computational cost. When Δ was 0.10, prediction accuracy greatly declined. From the above analysis, it is suggested that the reasonable range of Δ is $[0.05, 0.075]$.

Table 4.3 shows the number of clusters and the overall MAPE under different m values when $\Delta = 0.05$.

Table 4.3: The influence of m on the MAPE and the number of clusters

m	**Number of clusters**	**overall MAPE(%)**
1	26	4.26
2	24	4.08
3	27	4.21

It can be seen from Table 4.3 that m does not influence much on the number of customer clusters, but does affect the final load forecasting accuracy. As the MAPE has been quite small, different values of m result in considerable relative changes in prediction accuracy.

In summary, Δ greatly affects the performance of LF-LCB. It significantly influences the number of customer clusters and consequently affects load forecasting results. With a proper Δ, an optimal m can further improve the accuracy of load forecasting.

4.5.3 Experiment 2: comparing sCCRF to state-of-the-art methods

This experiment compares the proposed sCCRF with other state-of-the-art prediction methods, including Support Vector Regression (SVR) [DBK+97], CCRF [Guo15] and Convolutional Neural Networks (CNN) [DQH17]. For a fair comparison, this experiment chose two typical customers in Smart Grid, and used the four models to predict the load for the customers respectively.

Householders and office users are two types of customers exhibiting irregular behaviors. This experiment, therefore, chose one householder and one office user as samples to evaluate performances of above four models in load forecasting. For different models, cross-validation was used to tune hyper-parameters. Table 4.4 lists the best results achieved by each model.

Table 4.4: The performances of four models on load forecasting of two customers. The results are measured by overall MAPE (%).

Customer	**SVR**	**CCRF**	**CNN**	**sCCRF**
Householder	4.81	4.88	5.45	4.40
Office user	3.89	3.76	4.48	3.69

In Table 4.4, for householders, sCCRF achieves better accuracy than SVR and CCRF. CNN does not perform well, which may due to the limited number of training samples. For office users, sCCRF, SVR and CCRF get close results, while CNN is less competitive. When training samples are limited, sparse model, sCCRF, has advantages in generalization. It is widely known that deep neural networks require a large quantity of training samples to achieve good performances. In this evaluation, the power of CNN is not fully shown, and it is hasty to come to the conclusion that CNN is not competitive. Besides the competitive accuracy in load forecasting, sCCRF has another advantage that it simultaneously selects effective features during training, which can be utilized in customer behavior analysis.

4.5.4 Experiment 3: evaluation of Learning Customer Behavior

4.5.4.1 Comparing LF-LCB to baselines

The contribution of learning customer behavior in load forecasting was evaluated by comparing LF-LCB with two other configurations, both of which used sCCRF without considering customer behavior. In method 1, one sCCRF was trained for each customer. The load of the grid system was the sum of all individual customers' loads predicted by sCCRFs. This method was named LF-S. In method 2, one sCCRF was trained towards the grid system, regardless of any individual customer behavior. LF-W is used to denote this method. MAPE for each hour of the three methods are illustrated in Figure 4.5. It can be seen that LF-S performs slightly better than LF-W, and LF-LCB outperforms the other two methods.

LF-S used sCCRF to predict the load for each customer, but some behaviors of an individual customer were random and might be impossible to predict. For instance, some household customers may occasionally go out for parties on any weekday. Thus, the weakness of LF-S came from many accumulated errors resulting from the random customer behaviors. For LF-W, it utilized one sCCRF to predict the load for all the customers, but a single sCCRF failed to handle the various customers with different behaviors. In the end, the final load prediction result of LF-W was not satisfactory.

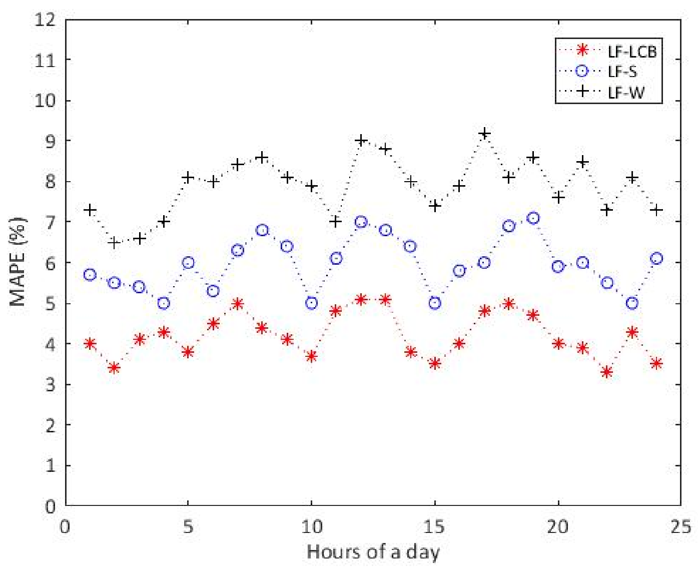

Figure 4.5: Performance comparison of LF-S, LF-W and LF-LCB

In contrast, LF-LCB performed well because it overcame the disadvantages in the above two methods. LF-LCB used sCCRF to first analyze customer behavior, then grouped customers with similar behaviors into one cluster. In a cluster of customers, the chaotic random behaviors were averaged, resulting in "smooth" power usage data. With a fine-tuned sCCRF for each customer cluster, the final prediction result was more accurate than that of the other two methods.

4.5.4.2 Customer behavior analysis

Customer aggregation is based on customer behavior, which is a vector that reflects how the power usage of a customer is influenced by a series of external factors. Customer behavior vector summaries the power consumption patterns of different types of customers. For instance, the behavior of solar energy producer is strongly affected by weather conditions, and the corresponding vector has large weights on weather features. Customer behavior is analyzed in the two-layered clustering process. Figure 4.6 shows the clustering tree of LF-LCB.

In the clustering tree, 7 clusters $(C_1, \cdots, C_7)$ are obtained in the first layer. In the second layer, the digit on each ellipse indicates the number of clusters, and 24 clusters are formed in the end.

In the first layer of the clustering tree, based on the binarized feature weights, it can be determined whether customer behavior in one cluster is influenced by certain features. Table 4.5 uses a binary matrix to show the relationships between the clusters and features.

In Table 4.5, "1" indicates that the cluster is influenced by this feature, while "0" means that the feature is not related to this cluster. The clusters in the first

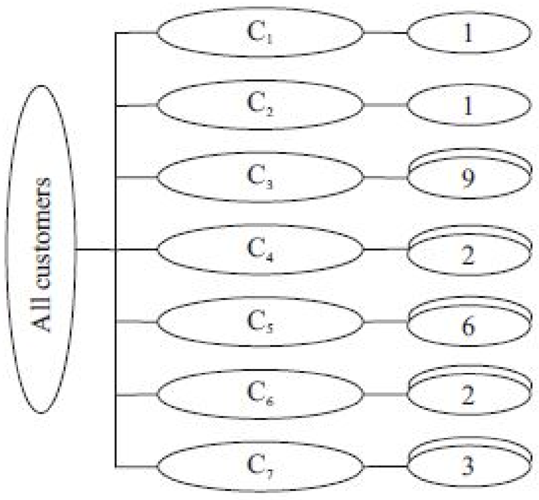

Figure 4.6: Clustering tree of LF-LCB. C_i represents the cluster in the first layer. The digits indicate numbers of clusters in the second layer.

Table 4.5: Cluster and feature relation matrix

	t_1	t_2	c_1	w_1	w_2	w_3	w_4	m_1	m_2
C_1	1	0	0	0	1	1	0	0	0
C_2	1	0	0	1	0	0	1	0	0
C_3	1	1	1	0	0	0	0	0	1
C_4	1	0	0	0	0	0	0	1	1
C_5	1	1	1	1	0	0	1	0	0
C_6	1	1	1	1	0	0	1	1	0
C_7	1	0	0	1	0	0	0	1	0

layer show clear physical meanings. For instance, in Figure 4.6, customers in C_1 are wind power producers, and their behavior is influenced by the related wind features. Customers in C_2 are solar energy producers whose power usage is affected by time, temperature and cloudiness. Observing the types of customer in each cluster, it can be seen that they generally belong to the same category of customer. For example, customers in C_4 are thermal storage customers, while customers in C_5 are householders and office users.

In the second layer of the clustering tree in Figure 4.6, customers are further clustered, resulting in 24 final clusters. Take C_5 as an example. C_5 is further clustered into 6 clusters based on the non-zero weights. In each customer cluster C_{5j}, customers show similar responses to the influences of outside features. For the 24 customer clusters, accordingly, 24 corresponding sCCRFs are fine-tuned. In the end, only 24 sCCRFs are maintained for load forecasting for the grid system. Therefore, LF-LCB results in a reasonable computational cost.

The stability of customer clustering is also analyzed. In the two-layered clustering, the first layer is actually a classification by binarized weights, which is totally stable. Customers in each cluster are further clustered according to the non-zero weights, whose dimension has been greatly reduced. Moreover, the number of customers in each cluster is not large. Thus, the clustering processes in the second layer have quite stable results.

4.5.4.3 Visualizations

Load forecasting is a non-linear prediction. Figure 4.7 visually shows the predicted loads and actual loads in two sample days, which are 1^{st} and 2^{nd} of December in 2010. It can be seen that the curve of predicted loads follows the trends of the

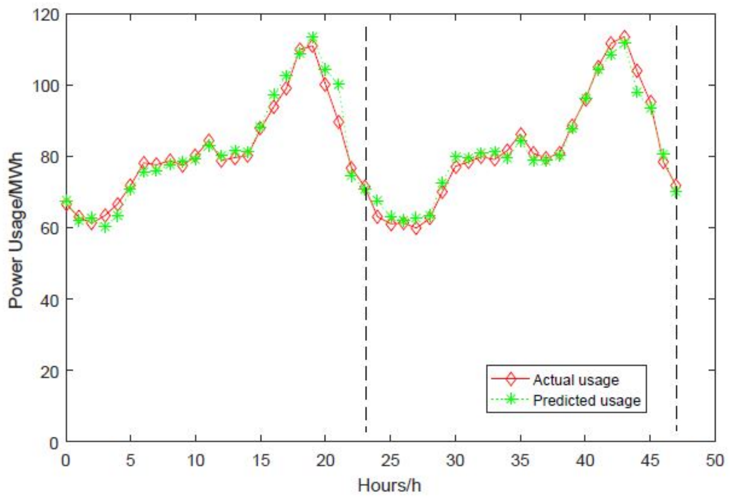

Figure 4.7: The predicted loads and actual loads in two sample days on the 1^{st} and 2^{nd} of December in 2010.

actual load sequence, which demonstrates that the proposed method captures the fluctuations of actual loads.

Figure 4.8 shows the predictions of 95%-confidence intervals on the 10^{th} of November in 2010. The predictions of 95%-confidence intervals may assist decision making in uncertain environments.

4.5.5 Experiment 4: comparing to other customer aggregation methods

In this experiment, LF-LCB was compared to two other customer aggregation methods. The first one was customer aggregation by manual rules. This method classifies

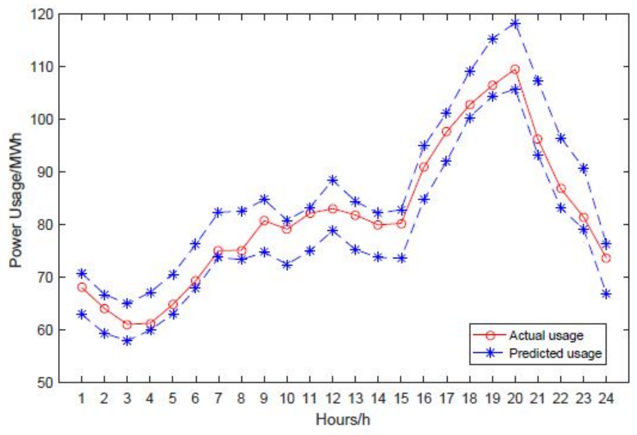

Figure 4.8: The predicted 95%-confidence intervals on the 10^{th} of November in 2010.

customers by their known attributes, such as householder, office user and solar power producer, and then sCCRF is introduced to predict the load of each cluster. This method is called LF-M. The other method was Alzate and Sinn's work [AS13], which used kernel spectral clustering to aggregate customers with respect to the historical usage data, and then used periodic auto-regression to predict the load of each cluster. This method is referred as LF-KSC for convenience.

For LF-M, only the customer attributes were used to aggregate customers, which might contain partially information on customer behavior, but certainly not complete. For LF-KSC, the RBF kernel with Spearman's distance, which was reported best performance [AS13], was used. Validation method was used to select the parameters for kernel.

The three methods were compared on the cluster number, and prediction accuracy. The result of LF-LCB that considers only the nearest neighboring variable is also added, denoted as LF-LCBo. Table 4.6 shows the compared items.

Table 4.6: The comparisons of LF-M, LF-KSC and LF-LCB

Method	**Number of clusters**	**overall MAPE(%)**
LF-M	8	6.82
LF-KSC	14	5.68
LF-LCBo	26	4.51
LF-LCB	*24*	*4.08*

In Table 4.6, LF-M can be regarded as a baseline customer aggregation method, where customers are aggregated manually. In LF-M, only 8 customer clusters are identified. In contrast, LF-LCB introduces learning method to identify customer behavior, and 24 clusters are obtained adaptively. It can be seen that LF-LCB significantly outperforms LF-M, which verifies the effectiveness of learning customer behavior.

In comparisons between LF-KSC and LF-LCB, LF-KSC has less customer clusters, while LF-LCB gets better accuracy in the end. LF-KSC aggregates customers only on the patterns of customers' historical usage data, while LF-LCB considers the customer behavior pattern under influences of different external factors. It is found that the learned customer behavior pattern is more effective than only using the historical usage pattern to aggregate customers. In experiments, LF-KSC has some advantages in training time. In fact, kernel spectral clustering is a quite expensive algorithm in time cost, while the time cost of LF-LCB can be further reduced by simply learning more customers in parallel.

Comparing to the simplified LF-LCB, the current method improves load forecasting accuracy by 0.43%, which is a considerable relative improvement.

4.6 Discussions

This chapter proposed a method of learning customer behavior to aggregate customers (LF-LCB) to surmount the challenges of complex customer behaviors, so as to facilitate load forecasting in Smart Grid market. Based on the experimental results, this section further discusses the key issues in customer behavior learning, the benefits gained from learning customer behavior and insights to extend learning customer behavior to broad market domains.

4.6.1 Two key issues in learning customer behavior

The essence of learning customer behavior is that it clusters similar customers into the same group based on discovered customer behavior patterns, and consequently targets each customer cluster for the optimal solution. In order to achieve a good performance, there are two key issues to be addressed.

The **first issue** is to find the robust customer behavior patterns. Only with robust behavior patterns, can customers be clustered accurately. An appropriate learning method can discover the customer behavior patterns much more robustly than some statistical criteria or manually classification.

The **second issue** is to find an appropriate granularity of the customer clusters. The clustering granularity affects the final prediction accuracy and computational cost. A proper clustering granularity should compromise between the separation of different customers and the smoothness of customer behavior in the same cluster.

An efficient learning of customer behavior should effectively resolve the above two issues. To discover robust customer behavior patterns, sCCRF is proposed in this chapter. Experiments 2 and 3 verified that sCCRF could identify key features for different customers and obtain a good accuracy in load forecasting. Thus, it can be an efficient learning method to model the load forecasting problem and to analyze customer behavior. To find a proper clustering granularity, this study used validation through experiments. Experiment 1 revealed the process to choose the proper granularity of the cluster.

It is worthwhile to notice the coherence of the sparse learning model and the two-layered clustering process. Benefiting from the sCCRF, two measures for customer aggregation were thereby obtained. As a consequence, a two-layer clustering could effectively aggregate customers, free from the “curse of dimensionality”. In Experiment 3, it was found that the two-layered clustering resulted in stable and reliable customer clusters.

4.6.2 The benefits of learning customer behavior

The efficient customer behavior learning brings benefits for load forecasting in Smart Grid.

1) LF-LCB is superior to learning towards Smart Grid market. In Experiment 3, LF-LCB outperformed LF-W. In Smart Grid market, LF-W could not perform well because of the variations in customer behavior. In contrast, LF-LCB introduced extra computations to handle the complex customer behaviors and resulted in high prediction accuracy.

2) LF-LCB performs better than learning for each single customer. LF-LCB also showed a better result than that of LF-S in Experiment 3. As some random behaviors were hard to predict, learning from each customer did not result in the highest accuracy. In comparison, LF-LCB “smoothed” the random behaviors to some extent, and thus achieved a better prediction accuracy. LF-LCB also had a much lower computational cost than that of LF-S in load forecasting. In summary, Experiment 3 demonstrated that LF-LCB had the advantages of improving load forecasting accuracy and consuming a low computational cost.

In comparison with LF-M in Experiment 4, it is verified that customer behavior learning is much more effective than manually classifications of customers. In comparison with LF-KSC in Experiment 4, LF-LCB showed better performance, which demonstrates that clustering customers with learned patterns is more effective than clustering customers using historical load data only.

4.6.3 The extensions of learning customer behavior

Learning customer behavior to aggregate customers can also be applied to other market domains. With learned customer behavior, similar customers can be aggregated.

Consequently, decision makings, such as retail strategies or demand predictions, can target the customer clusters. In this scenario, the decisions tailored for a customer cluster will be better than the decisions aimed at the whole complex market. With proper clustering granularity, there would be limited number of clusters, so the overall computational cost would be manageable.

This discussion also placed an emphasis on the two key issues in general customer behavior learning. The first key issue is to choose an appropriate learning method to model the targeted problem to discover the customer behavior patterns. The machine learning methods could be LASSO, LARS [EHJT04], L_1-SVM [HCL+08], or any method that can perform feature selection. To model a temporal sequence problem, candidate learning methods could be sCCRF or RNN [FN93]. The second key issue is to find a proper clustering granularity to cluster different customers. For different problems, the proper cluster granularity could be found through validations by experiments, as illustrated in Experiment 1. After the customers are clustered, each cluster can be treated independently. If better sale strategies are required, decisions can be tailored targeted individual customer clusters. If the total customer demand is to be predicted, fine-tuning can be applied to each customer cluster, and then the total market demand can be obtained by summing the demands of each cluster. In general, learning customer behavior can be an efficient solution to decision makings in large-scale complex markets.

4.7 Summary

This chapter proposed a load forecasting method in Smart Grid market through learning customer behavior (LF-LCB), which utilized the proposed sCCRF to analyze customer behavior by using the learned weights which can reflect different energy consumption patterns of various customers. The results of experiments conducted from several perspectives verified the following two conclusions: **1)** Learning customer behavior to aggregate customers can improve the prediction precision and lead to a reasonable computation cost. **2)** The proposed sCCRF is an efficient learning tool with feature selection capacity. Though sCCRF has been an effective demand prediction model, deep neural networks can be introduced to build stronger prediction models.

Chapter 5

Recurrent Neural Networks for Renewable Energy Prediction

This chapter models renewable energy prediction as a Regression problem on Sequential Data (RSD), which is formally defined as follows.

$$\mathbf{y} = f_\theta(\mathbf{X}), \tag{5.1}$$

where f_θ is a transformation parameterized by θ, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T]$ is a $T \times N$ matrix, representing T steps and N predictor variables (features) in each step, and $\mathbf{y}$ can be a scalar or a T-dimension vector.

A Deep Regression model on Sequential Data (DeepRSD) model is proposed to effectively solve the RSD problem. DeepRSD takes advantages of several variants of neural networks to construct an effective end-to-end learning method. An alternative dropout is proposed to improve the generalization of deep neural networks.

The rest of this chapter is organized as follows. Section 5.1 describes the designs of DeepRSD in detail. Section 5.2 summarizes the conditions to reliably train DeepRSD. Section 5.3 applies DeepRSD to two real-world problems from data science competitions. In Section 5.4, three experiments are conducted to evaluate and analyze DeepRSD. This chapter finally discusses the advantages and limits of DeepRSD in Section 5.5 and draws conclusions in Section 5.6.

5.1 Designs of DeepRSD

In this section, the standard Recurrent Neural Network (RNN) is briefly reviewed, and the architecture of the proposed DeepRSD is illustrated. How to choose a proper activation function for DeepRSD is discussed, and an alternative dropout is proposed and described in detail.

5.1.1 RNN review

As RNN is the core module for DeepRSD, it is necessary to have a brief review of standard RNN [RHW88]. For the input sequence $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$, each in $\mathbb{R}^N$, RNN computes a sequence of hidden states $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_T$, each in $\mathbb{R}^M$, and a sequence of predictions $\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \cdots, \hat{\mathbf{y}}_T$, each in $\mathbb{R}^K$, by iterating the equations

$$\mathbf{h}_i = \varphi_h(\mathbf{W}_{hx}\mathbf{x}_i + \mathbf{W}_{hh}\mathbf{h}_{i-1} + b_h) \tag{5.2}$$

$$\hat{\mathbf{y}}_i = \varphi_y(\mathbf{W}_{yh}\mathbf{h}_i + b_y) \tag{5.3}$$

where $\mathbf{W}_{hx}, \mathbf{W}_{hh}, \mathbf{W}_{yh}$ are weight matrices, b_h, b_y are bias terms, and φ_h, φ_y are activation functions. Equation 5.2 defines the input-to-hidden layer and Equation 5.3 defines the hidden-to-output layer.

5.1.2 The architecture of DeepRSD

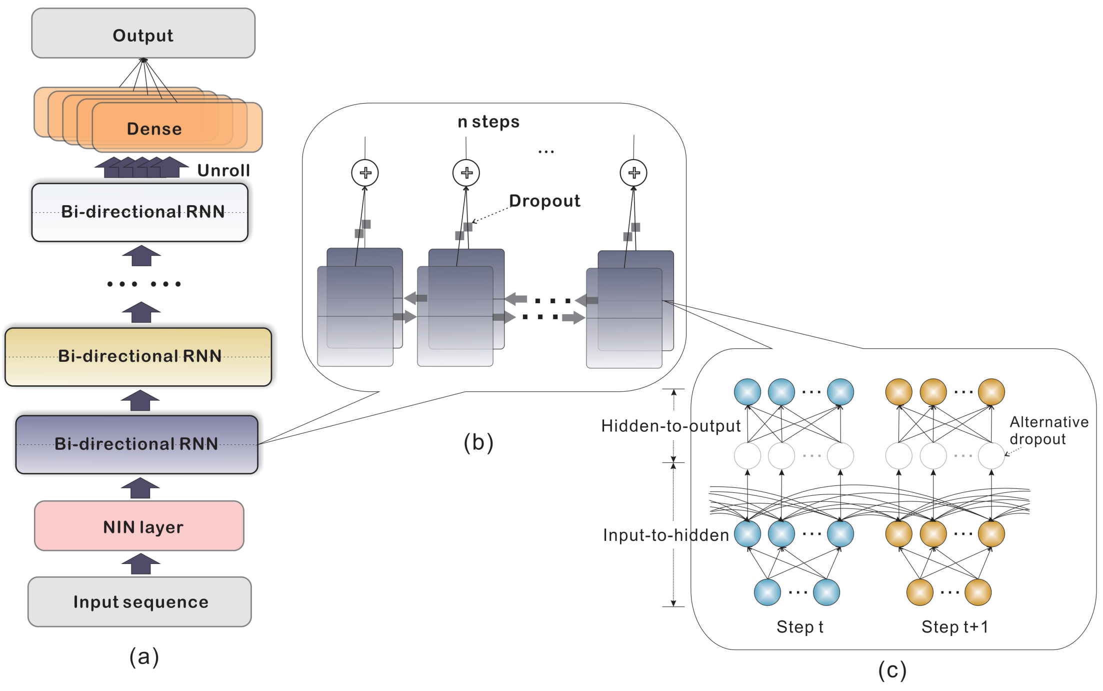

Figure 5.1: The overall architecture of DeepRSD

Figure 5.1(a) illustrates the architecture of the proposed DeepRSD. The network layers and their functions are described from bottom to top. The input data $\mathbf{X}$ are sequential data, and the output data $\mathbf{y}$ can be a vector or a scalar, as illustrated at the beginning. DeepRSD consists of three modules: input processing module, main functional module and output processing module.

Input processing module includes the input layer and the Network-In-Network (NIN) layer [LCY13]. The input layer is at the bottom, where sequential features are fed into DeepRSD. On the top of the input layer is the NIN layer that reduces the dimension of features using a linear activation function. The NIN layer contributes to accelerating the training process; meanwhile, NIN layer does not affect the precision of final predictions.

Main functional module is a stack of bi-directional RNN layers with different sizes. RNN is suitable to model sequential data because it introduces hidden layer to encode the temporal information. Bi-directional RNN [GS05] can effectively represent the correlations, and has been widely used to model sequential data. The structure of bi-directional RNN is illustrated in 5.1(b). The stacked bi-directional RNNs supply sufficient nonlinearities for the sequence and also for step feature $\mathbf{x}_i$. Even the features in $\mathbf{x}_i$ are heterogeneous and have complex functional relationships, the stacked bi-directional RNNs can learn to represent them automatically.

Output processing module is a step-wise dense layer that outputs the predicted value at each step. In Figure 5.1(a), the unrolled T dense layers are shown, corresponding to the T predictions. The T predictions can also be added up for a scalar prediction if necessary. Therefore, the final output $\mathbf{y}$ can be a T-dimension vector or a simple scalar.

Stacked bi-directional RNNs, the main functional module in DeepRSD, is designed catering the characteristics of RSD problems. In discrete problems, Pascanu et al. [PGCB13] discussed how to construct deep RNN and proposed three variants of deep RNN, which are deep input-to-hidden, deep hidden-to-output and deep transition networks. Graves [Gra13] used RNN with stacked hidden layers to generate sequences. Much effort in RNN has been made to handle long dependency in sequences in discrete problems. For RSD problems, it is necessary to consider the sequential information as well as the functional relationships among step features. For many RSD problems in practice, the sequential data is not so long that a standard hidden layer is adequate to handle the dependencies, thus no necessary to introduce deep transitions. In the stacked bi-directional RNNs, there are dense connections in input-to-hidden and hidden-to-output layers, which supply plenty of nonlinearities to encode the heterogeneous step features. Deep input-to-hidden and deep hidden-to-output networks are tried, and neither of them worked as well as stacked bi-directional RNNs in experimented datasets. The advantages of stacked bi-directional RNNs over other deep RNNs are further shown in experiments.

5.1.3 Activation function

For the nonlinear activation function, leaky rectified linear unit (leaky ReLU) [AHSB14] is used, shown in Equation 5.4.

$$\varphi(x) = \begin{cases} x & if \quad x > 0 \\ \alpha x & if \quad x \leq 0 \end{cases} \tag{5.4}$$

For RNN, traditional activations–sigmoid or hyperbolic tangent functions, which have saturated zones, may lead to the gradient vanishing problem [PMB13]. ReLU

[NH10] is advantageous since 1) it alleviates the vanishing gradient and 2) it offers a simple gradient computation. However, it often makes DeepRSD diverge. When using leaky ReLU, DeepRSD can be trained more reliably. For the above reasons, leaky ReLU is chosen as the activation function for DeepRSD.

5.1.4 Alternative dropout

Dropout [SHK$^+$14] has been a simple and effective way to improve the generalization of feed-forward neural networks. However, people struggled to develop effective dropout for RNN. Zaremba et al. [ZSV14] applied dropout to the input and hidden-to-output but not to the transition layer in RNN. Gal et al. [GG16] proposed a theoretically grounded dropout to the hidden layer in RNN and got improved performance. These methods work effectively for discrete problems. However, they do not work in DeepRSD.

Considering the structure of stacked bi-directional RNNs, a special dropout scheme is proposed in DeepRSD. For the standard RNN shown in Figure 5.1(c), dropout is applied between the input-to-hidden and hidden-to-output layers, while for the stacked second bi-directional RNN, this dropout is applied alternatively. To be specific, dropout is applied only to the *forward* RNN for the first bi-directional RNN. For the stacked second bi-directional RNN, dropout is applied to the *backward* RNN. Dropout is applied alternatively in the forward and backward directional RNNs, so this dropout scheme is named as alternative dropout.

For the stacked Bi-RNNs, the alternative dropout between input-to-hidden and hidden-to-output layer is crucial to improve generalization. A possible explanation is supplied for alternative dropout. It's assumed that the input-to-hidden layer of forward RNN outputs a vector in range $[a, b]$. The hidden-to-output layer is simplified as an identity map. If no dropout is applied, Bi-RNN sums the outputs of each direction and thus gives an output range of approximate $[2a, 2b]$ (the output range of two RNNs should be similar). For dropout with a keep rate γ ($\gamma < 1$), to keep the invariance of expectation, this method re-scales the vector and leads to an inflated approximate range $[a/\gamma, b/\gamma]$ (Re-scaling is the default manner in dropout [SHK$^+$14]). If dropout with a keep rate γ is applied to both directions, the output range of Bi-RNN will be $[2a/\gamma, 2b/\gamma]$. In contrast, for alternative dropout, the output range of Bi-RNN will be $[a/\gamma+a, b/\gamma+b]$. It can be seen that the dropout on both directions lead to a larger fluctuation range than alternative dropout. The large fluctuations may bring difficulty to optimization algorithms to find a good local minimum.

5.2 Training and Inference

This section first sets the cost function for DeepRSD. For regression problems, Mean Square Error (MSE) and Mean Absolute Error (MAE) are widely used. As it is more robust to very large or small values, MAE is used as the cost function for DeepRSD. If the output $\mathbf{y}$ is a T-dimension vector, the cost function can be written as:

$$C(\theta) = \frac{1}{ST} \sum_{i}^{S} \sum_{i}^{T} |y_i - \hat{y}_i|, \tag{5.5}$$

where S is the total number of sequence samples, and θ is parameters to be learned (see Equation 5.1). If $\mathbf{y}$ is a scalar, the summation $\sum_i^T$ can be omitted.

DeepRSD is hard to train for several reasons. Comparing to classification problems, the inputs and outputs are both unbounded in regression problems, which may lead to divergence in the training of DeepRSD. Moreover, improper initializations will result in divergence or invalid learning in deep networks. Besides, there are gradient vanishing/exploding problems in (deep) RNN, which may hamper the training process. In this study, it is found that the conditions in Table 5.1 are necessary to ensure a plausible training.

Table 5.1: Conditions for DeepRSD training

Condition	**Description**
Data preprocessing	Normalize and trim data
Initializations	Introduce proper initializations for RNN
Gradient vanishing preventing	Use leaky ReLU activation
Gradient exploding preventing	Use gradient clipping

Data preprocessing. Sutskever et al. [SMDH13] pointed out it is essential to normalize data (both inputs and outputs) to obtain a reliable training for RNN. This study follows this rule and normalizes the inputs and outputs for DeepRSD. The extreme values are trimmed to result in a bounded region $[-a, a]$. If a is set as 5, the extreme values outside the 5σ region of Gaussian will be trimmed.

Initializations. Initializations are critical to ensure the convergence of RNN [SMDH13]. Gaussian and orthogonal initializations [GB10], [SMG13] are used for the stacked bi-directional RNNs. A correct scale of Gaussian is essential for a plausible training. For normalized data, it is found that $\sigma = 0.01$ is a good scale to ensure convergence.

Gradient vanishing preventing. RNN is difficult to train because they suffer from gradient vanishing/exploding problems [PMB13]. Leaky ReLU activation is used to prevent vanishing gradients.

Gradient exploding preventing. Gradient clipping [PMB13] is introduced to prevent gradient exploding for RNN.

Applying the above conditions, DeepRSD can be trained reliably. This study resorts to mini-batch Stochastic Gradient Decent (SGD) with Nesterov's momentum [Nes83], with learning rate decay [SMDH13], to train DeepRSD. Other optimization algorithm, such as Adadelta [Zei12] and Adam [KB14] are also tried, but the results are not so good as SGD with Nesterov's momentum. Therefore, it is recommended to use SGD with Nesterov's momentum to train DeepRSD.

Inference in DeepRSD can be quite efficient. As DeepRSD introduces dropout, this study simply employs the inference process of dropout networks to predict $\mathbf{y}$ [SHK$^+$14].

5.3 Case Studies

DeepRSD is evaluated on two different real-world problems from data science competitions. **1)** DeepRSD is applied to the AMS solar energy prediction problem to predict a scalar value. **2)** DeepRSD is further evaluated on a small dataset from probabilistic energy forecasting competition 2014, where the output is structured.

For both problems, DeepRSD was used to construct a universal end-to-end learning model. The same conditions in training are set for the two problems. In data preprocessing, the data were normalized to standard Gaussian. The bound for trimming was set $a = 5$. To cope with the gradient vanishing problem, DeepRSD used leaky ReLU activation with a leakiness $\alpha = 0.15$, For gradient exploding, DeepRSD used gradient clipping to normalize the gradients if their norm exceeded 1.0 [PMB13]. In initializations, for the transition matrix W_{hh} in the RNN layer, DeepRSD used orthogonal initialization [SMG13] with a gain $(1 + \alpha^2)^{\frac{1}{2}}$. For the input-to-hidden weight matrix W_{hx}, DeepRSD used Gaussian distribution with 0 mean and 0.01 standard variance to initialize it.

5.3.1 Case study 1: evaluations on AMS solar energy prediction contest

The objective of the problem is to predict the daily incoming solar energy at 98 Oklahoma Mesonet sites [MGB$^+$15]. Input numerical weather forecast data come from the Global Ensemble Forecast System (GEFS) Reforecast Version 2. In a daily forecast, there are 5 forecasting time steps in every 3 hours. In each time step, 15 different weather factors are forecasted. The distributions of Mesonet sites and GEFS data (only available on latitude and longitude grids) are shown in Figure 5.2.

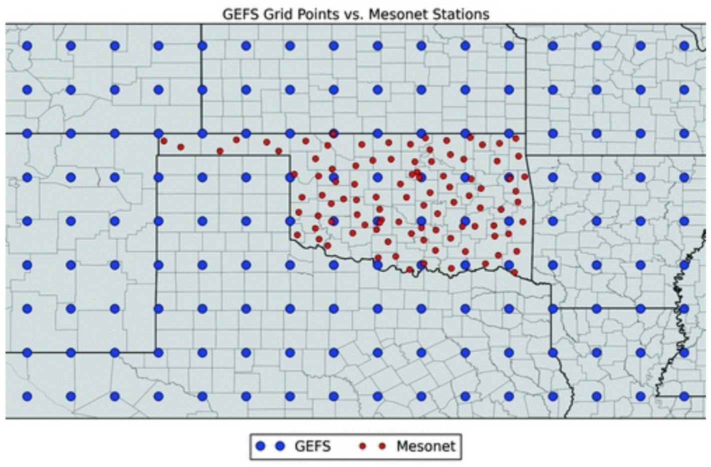

Figure 5.2: The distributions of Mesonet sites and GEFS grids

Training data (1.07GB) come from 1 January 1994 to 31 December 2007, and testing data (0.36GB) are from 1 January 2008 to 30 November 2012.

5.3.1.1 Solutions using DeepRSD

This study uses DeepRSD to construct one universal model for the 98 Mesonet sites. For each Mesonet site, the weather data is extracted from its neighbouring four GEFS grids. The distances of the Mesonet site to its neighbouring four grids are also included. Besides, the temporal and the Mesonet site specific information are added. All the above information is combined to be a step feature with 77 heterogeneous variables. The raw features are input to DeepRSD, without any feature engineering. DeepRSD uses alternative dropout to improve its generalization. DeepRSD not only learns the transitions in weather forecasting sequences, but also hiddenly learns to approximate the weather states at the Mesonet site from the heterogeneous features.

The hyperparameters were set using random search [BB12], followed by hand-craft tuning. The input dimension was 77 for the input layer. The output dimension of NIN layer was 64, reducing the input dimension by 16.9%. DeepRSD introduced 4 stacked bi-directional RNN and dense layers, with the size 128, 224, 128 and 16. The following dense layer had an output dimension of 1. The last layer was a mean-pooling layer to output the daily production. The dropout rate for both forward and backward RNN before the summation was 0.3, and the drop rate of alternative dropout was 0.5. Dropout was applied in the first 3 layers of the stacked bi-directional RNNs.

5.3.1.2 Test results

Table 5.2 shows DeepRSD's result measured by MAE and the top three results in the Kaggle competition.

Table 5.2: Ours and the top three results in the Kaggle contest

Solution	MAE(MW)	Method
1st Place	2.11	GBDT
2nd Place	2.13	GBDT
3rd Place	2.16	GBDT
Ours	*2.10*	*DeepRSD*

It can be seen from Table 5.2 that GBDT is the dominant method that holds the top three places in the past competition. The top three methods used complex feature engineering, post-processing or model combinations to improve the results [MGB+15]. In contrast, DeepRSD simply averaged 10 differently trained models and got an MAE of 2.10M.

5.3.2 Case study 2: evaluations on probabilistic solar power forecasting competition

Solar power forecasting competition aims at predicting hourly solar power production for 3 different stations given the on-site hourly weather forecasting. This competition introduced rolling forecasting, and data of one more month were released in every prediction task. To make direct comparison, this study used the data from April 2012 to May 2014 as the training data, to predict the solar power in June 2014. There were only 791 training samples for each station, which was a quite small quantity for deep neural networks.

5.3.2.1 Solutions using DeepRSD

DeepRSD is used to construct one universal model for the 3 stations. The extracted raw features include weather feature, temporal feature (hour and month), station specific feature, 19 variables in total. DeepRSD uses a data sequence with length 14, which excludes 10 hourly data whose solar power is zero or very close to zero. This competition used quantile regression measurement for the final score, while MAE is still used as the cost function to train DeepRSD. This study averages the predictions from 10 models to determine the final quantiles.

In this problem, the hyperparameters for DeepRSD were set as follows. The input dimension was 19. The output dimension of NIN layer was 16. DeepRSD introduced 2 stacked bi-directional RNNs with the size 20 and 20. The following dense layer had an output dimension of 1. The final output was a 14-dimension vector, corresponding to 14 hourly solar power predictions. DeepRSD used alternative dropout with a drop rate of 0.3.

5.3.2.2 Results

The performance of DeepRSD is compared with the winning methods in this competition. The results are shown in Table 5.3 (Methods are referred to [HPF$^+$16]).

Table 5.3: Ours and the top three results of task 16 in the contest

Solution	**Score**	**Method**
1st Place	0.01211	GBDT and k-NN
2nd Place	0.01221	GBDT and quantile regression forest
3rd Place	0.01275	Multiple quantile regression
Ours	*0.01202*	*DeepRSD*

In Table 5.3, DeepRSD achieves the best score. According to literatures [HP16], the winning team used feature engineering and extra blue-sky model in the competition. DeepRSD constructed a succinct solution and got a better result, demonstrating the effectiveness of DeepRSD in sequential regression problems.

5.4 Experiments

Three experiments are conducted to evaluate and analyze DeepRSD.

Experiment 1: Comparing DeepRSD with other deep RNNs. In this experiment, DeepRSD is compared with other deep RNN architectures to validate the choice of stacked bi-directional RNNs. The baseline performance of standard RNN is supplied to indicate the improvement by deep architectures.

Experiment 2: Comparisons of DeepRSD, GBDT and SGCRF. This experiment compares DeepRSD with two other state-of-the-art methods in sequential data regressions. Through comprehensive comparisons, the advantages and disadvantages of DeepRSD are analyzed.

Experiment 3: Analysis of alternative dropout. This experiment studies previous dropout methods and the proposed alternative dropout. The effectiveness of alternative dropout is interpreted through data visualizations.

5.4.1 Experiment 1: comparing DeepRSD with other deep RNNs

In this experiment, the performance of deep RNNs are compared in the regression problem. Besides stacked bi-directional RNNs, this study also constructed deep input-to-hidden (DeepIH) network and deep hidden-to-output (DeepHO) network.

DeepIH and DeepHO were used to replace stack bi-directional RNNs in the architecture of DeepRSD. The other parts of DeepRSD stayed the same. The three deep networks were evaluated on AMS solar energy prediction contest. Besides, standard RNN and multi-layer dense neural networks (DNN) are evaluated as the baseline.

The configurations of DeepRSD stayed the same as introduced in Subsection 5.3.1. The configurations of DeepIH and DeepHO were optimized by random search [BB12]. DeepIH used two dense layers, with size 32 and 32, for each time step; and two RNN layers, with size 192 and 128. DeepHO used two RNN layers with size 128 and 128, followed by two dense layers with size 128 and 64. The DNN had 4 dense layers, with size 128, 256, 192 and 128. The standard RNN had a size of 128. The performances of the five different RNNs are shown in Figure 5.3.

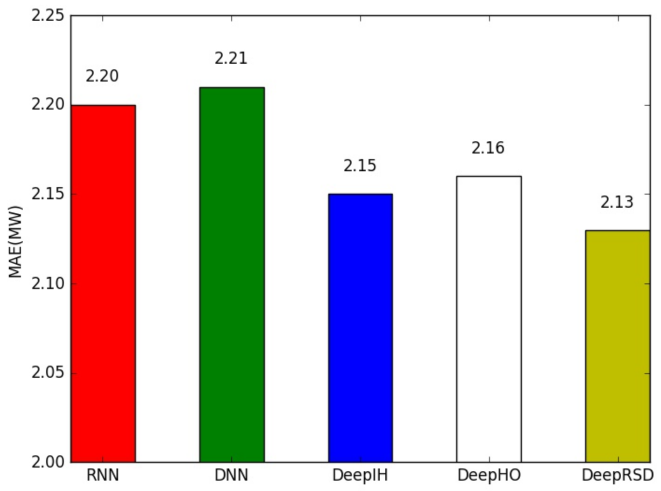

Figure 5.3: The performance comparisons of different networks

In Figure 5.3, standard RNN and DNN indicate baseline performance. RNN obtains an MAE of 2.20MW, and DNN obtains an MAE of 2.21MW. Though DNN is a deep network that has 4 layers, it does not utilize the temporal information, and therefore has a low performance. Comparing to the two baselines, DeepIH, DeepHO and DeepRSD make significant improvements on this problem.

Three deep RNNs are analyzed in detail. For DeepIH, it first uses dense networks to represent step features, and then input the representations into RNN. For DeepHO, it first represents features using two layers of RNNs, and then uses dense network to refine the output of RNN. In contrast, DeepRSD, which uses stacked bi-directional RNNs, always combines the nonlinear representations of step features and temporal information. It can be seen that DeepRSD has advantages over DeepIH and DeepHO in Figure 5.3. That is the reason we choose stacked bi-directional RNNs for sequential data regression.

5.4.2 Experiment 2: comparisons of DeepRSD, GBDT and SGCRF

In this experiment, DeepRSD, GBDT and SGCRF are compared on the data of probabilistic solar power forecasting competition. The data from April 2012 to February 2014 are training data, to predict the solar power in March to June 2014. GBDT is implemented on XGBoost [CG16], and SGCRF is implemented based on [WK13].

For a fair comparison, the same input feature data were put into the three methods. The feature was normalized by subtracting the mean and dividing the variance, without extra feature engineering. As GBDT could not directly model the sequential data, the feature of current hour was extended by adding the features of previous three hours. For DeepRSD and SGCRF, this study directly input the normalized features. The configuration of DeepRSD was the same as the one in Subsection 5.3.2. The hyperparameters of GBDT and SGCRF were optimized by grid search.

The evaluation results of the three methods are shown in Table 5.4. As the output has been normalized, MAE is calculated according to the normalized values.

Table 5.4: Evaluation results of SGCRF, GBDT and DeepRSD

Method	**MAE**
SGCRF	0.240
GBDT	0.221
DeepRSD	0.215

In Table 5.4, DeepRSD achieves better MAE than GBDT and SGCRF. For GBDT, extra effort has been made to encode the temporal information, but its result is still not so good as DeepRSD. SGCRF can handle temporal information, but the performance is not satisfactory. It was inferred that SGCRF does not perform well on vanilla features. Previous work used SGCRF with either feature engineering or other models [WK13]. DeepRSD shows best performance in Table 5.4 due to its two advantages. 1) it can naturally model temporal information. 2) it can effectively represent features by its nonlinearities. However, the learning curves in experiments reveal that DeepRSD is not so stable as GBDT and SGCRF. Model ensemble can be introduced to compensate this weakness of DeepRSD.

5.4.3 Experiment 3: analysis of alternative dropout

In this experiment, three dropout ways for DeepRSD are analyzed. The first way is naive dropout, where dropout is only applied in the input projection and output projection [ZSV14]. The second way is variational dropout, where dropout is applied to the hidden layer besides the input and output projections [GG16]. The third way is the proposed alternative dropout. The three dropout methods are applied to DeepRSD with the configurations introduced in Subsection 5.3.1.

The MAEs and epochs to converge are shown in Table 5.5 for the three dropout methods.

Table 5.5: Comparisons of different dropout ways

Methods	**MAE(MW)**	**Epoches to converge**
No dropout	2.17	4
Naive dropout	2.17	6
Variational dropout	2.17	40
Alternative dropout	2.13	6

In Table 5.5, without any dropout, DeepRSD achieves an MAE of 2.17MW. Though different dropout rates have been tried, naive dropout and variational dropout do not improve the generalization of DeepRSD. Moreover, variational dropout converges very slow, up to 40 epochs. Alternative dropout effectively enhances the generalization of DeepRSD by near 2%. In discrete problems, networks with dropout will consume more training time. This is also seen in regression problems in Table 5.5. DeepRSD without dropout converges in 4 epochs, while DeepRSD with alternative dropout converges in 6 epochs.

The alternative dropout is explained via data visualizations. When the network was near convergence in training, the output of the first step of the third bi-directional RNN was collected. The output was sampled every 100 mini-batches from batch 30000 to 35000. In the same manner, we visualized the output using naive dropout. The visualization of alternative dropout and naive dropout are shown in Figure 5.4 and Figure 5.5, respectively.

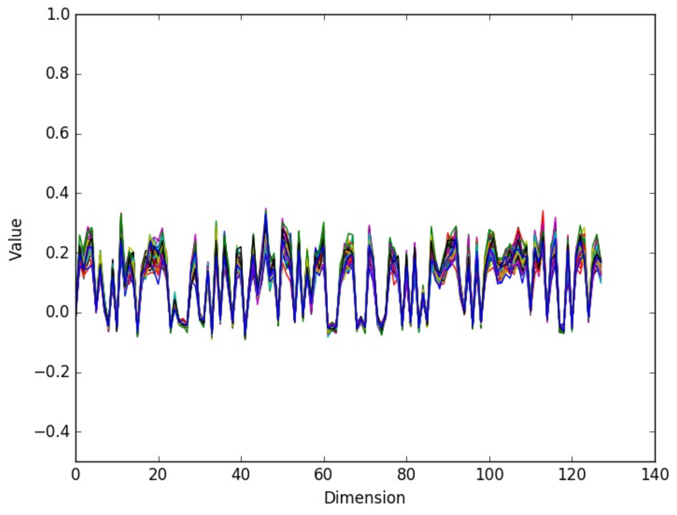

Figure 5.4: Visualizations of alternative dropout

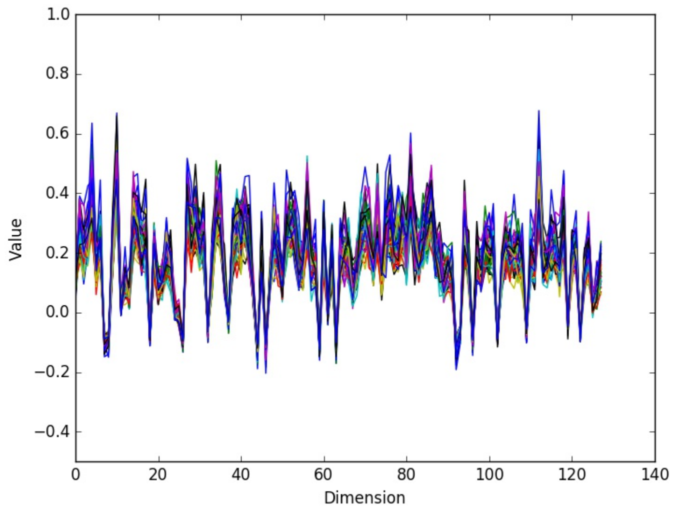

Figure 5.5: Visualizations of naive dropout

It can be seen that the outputs are consistent for the alternative dropout in Figure 5.4, while the outputs fluctuate in the naive dropout in Figure 5.5. The inconsistency and fluctuations in traditional dropout bring difficulty for the optimization algorithms, resulting in a poor local minimum. Even though the dropout rate was decreased, naive dropout does not work well. In contrast, alternative dropout provides a mild dropout way for stacked bi-directional RNNs.

5.5 Discussions

In AMS solar energy contest, DeepRSD got the best result evaluated by Kaggle server. Reviewing previous solutions, it is found that they either sought to feature engineering or introduced optimization tricks to improve performance [MGB$^+$15]. The winning solution used GBDT and similar input features as ours. They trained 13 models at each solar station. After that, two step optimizations were introduced to fine-tune the trained models. In the end, they also multiplied a coefficient 1.015 to the predictions to improve the final score. In contrast, DeepRSD constructed a universal model for all the Mesonet sites without any feature engineering.

For the probabilistic solar power forecasting problem, It is verified that DeepRSD could be competitive when the quantity of training data was small. The winning solution in this competition [HP16] deeply studied the features for solar power generation and introduced an extra blue sky model. GBDT and k-NN were used to build the model for each hourly production with the weather feature in current hour and the previous 3 hours. Comparing to their solution, ours used an end-to-end universal model and achieved better results.

Deep learning has shown advantages in several fields, such as computer vision, speech recognition and natural language processing. This chapter first introduced deep learning for RSD problems. The core functional part in DeepRSD was a deep RNN network. Though there had been studies on deep RNN in discrete problems, previous methods could not be directly applied to regressions. Instead, this chapter studied different variants of deep RNNs and chose stacked bi-directional RNNs for RSD problems. DeepRSD could simultaneously learn temporal transitions and functional relations in heterogeneous step feature, to construct an end-to-end learning solution. This study made efforts to properly train DeepRSD and applied four necessary conditions for a reliable DeepRSD training. Moreover, alternative dropout is proposed to effectively improve DeepRSD's generalization. Alternative dropout is a mild dropout way for deep bi-directional RNNs. As bi-directional RNNs are widely used in natural language processing and speech recognition, it worths a further study of alternative dropout on discrete problems.

GBDT and SGCRF are two state-of-the-art methods in RSD problems. Comparing to GBDT, DeepRSD can naturally model the temporal information. Comparing to SGCRF, DeepRSD has stronger nonlinear representation capacity for complex features. The advantages of DeepRSD have been demonstrated in the second experiment in Subsection 5.4.2. As DeepRSD is not stable, model ensemble is suggested to compensate this weakness.

In the end, the limits of this study are pointed out. DeepRSD uses vanilla RNN, which has difficulty in modeling long sequences. To compensate this, LSTM [HS97] can be employed to replace RNN in current DeepRSD architecture. Moreover, for the problems whose step feature is simple (in low dimension) and the sequence is short, it is not necessary to introduce the complex deep learning model.

5.6 Summary

In this chapter, DeepRSD was proposed for renewable energy prediction (a regression on sequential data). DeepRSD used stacked bi-directional RNNs to represent the sequential data. Four conditions were explored to ensure a plausible training. An alternative dropout was also proposed to effectively improve the generalization of DeepRSD. DeepRSD was applied to two real-world solar energy prediction problems and achieved state-of-the-art performances. The advantages and shortcomings of DeepRSD were also discussed.

Chapter 6

A New Time-series Model for Demand Prediction

This chapter develops a new time-series prediction model called Sparse Gaussian Conditional Random Fields on Top of Recurrent Neural Networks (CoR). CoR integrates the advantage of Recurrent Neural Network (RNN) and Sparse Gaussian Conditional Random Fields (SGCRF). CoR with attention (CoRa) is also developed to improve CoR by introducing an attention mechanism.

The rest of this chapter is organized as follows. Section 6.1 illustrates the overall architecture of CoR. Section 6.2 describes two training methods for CoR. Section 6.3 shows the prediction process of CoR. Section 6.4 proposes CoRa. Section 6.5 analyzes the space and time cost of SGCRF, RNN, CoR and CoRa. Section 6.6 evaluates CoR and CoRa using both synthetic and real-world data. Section 6.7 further discusses CoR and CoRa based on evaluation results. Section 6.8 summarizes this chapter.

6.1 Architecture of CoR

The overall architecture of CoR is illustrated in Fig. 6.1. CoR consists of two parts: the bottom part is stacked bi-directional RNNs (Bi-RNNs); and the top part is SGCRF. $\mathbf{X} \in \mathbb{R}^{T \times D}$ is the input observation. $\mathbf{y} \in \mathbb{R}^{T}$ is the final output, which corresponds to the observation in each time step. $\mathbf{z} \in \mathbb{R}^{T}$ is an intermediate variable, which is the output of stacked Bi-RNNs and the input of SGCRF. In the following sections, the stacked Bi-RNNs and SGCRF are described in detail.

6.1.1 Stacked Bi-RNNs

To illustrate the stacked Bi-RNNs, this chapter starts from a standard RNN [Pin87]. An input sample $\mathbf{X} \in \mathbb{R}^{T \times D}$ is a data sequence. For the sequence $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$, each in $\mathbb{R}^D$, RNN computes a sequence of hidden states $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_T$, each in $\mathbb{R}^M$, and a sequence of predictions $\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \cdots, \hat{\mathbf{z}}_T$, each in $\mathbb{R}^K$, by iterating the equations

$$\mathbf{h}_t = \varphi_h(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + b_h) \tag{6.1}$$

$$\hat{\mathbf{z}}_t = \varphi_z(\mathbf{W}_{zh}\mathbf{h}_t + b_z) \tag{6.2}$$

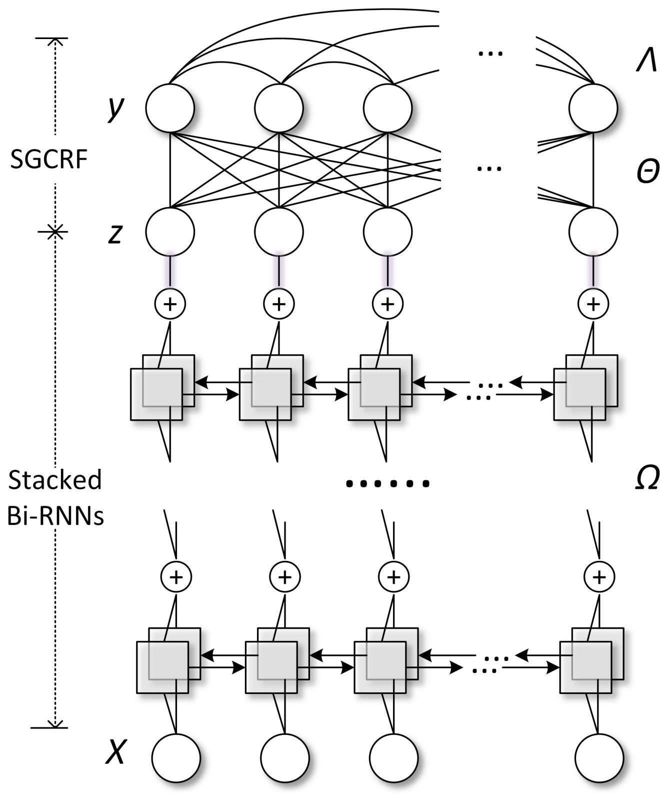

Figure 6.1: The overall architecture of CoR. The bottom part is stacked Bi-RNNs, and the top part is SGCRF. $\mathbf{X}$ is the input observation, $\mathbf{y}$ is the final output variable, and $\mathbf{z}$ is an intermediate variable, which is the output of stacked Bi-RNN and the input of SGCRF. Ω denotes all the parameters of stacked Bi-RNNs, while Θ and Λ are parameters of SGCRF.

where $\mathbf{W}_{hx}, \mathbf{W}_{hh}, \mathbf{W}_{zh}$ are weight matrices, b_h, b_z are bias terms, and φ_h, φ_z are activation functions.

A Bi-RNN is a combination of two standard RNNs in reversed directions. The performance of Bi-RNN has been better and more stable than standard RNN in speech recognition [GS05]. To adapt Bi-RNN for regression, the ouputs of two reversed RNNs are added together as the output of Bi-RNN. Multiple Bi-RNNs can be stacked to construct a deep network to further enhance representational capacity. This chapter builds stacked Bi-RNNs networks, shown in Fig. 6.1, as the bottom part of CoR. As the output of stacked Bi-RNN is in $\mathbb{R}^K$ in each time step, a dense layer is added to reduce $\mathbb{R}^K$ to $\mathbb{R}^1$ for the last Bi-RNN. The output $\mathbf{z}$ of stacked Bi-RNNs can be formulated as

$$\mathbf{z} = f_\Omega(\mathbf{X}), \tag{6.3}$$

where f_Ω is the learned mapping parameterized by Ω.

The architecture of stacked Bi-RNNs can be flexible. If the amount of training data is small, it can be reduced to a single RNN. If the number of time steps is large, Long Short-Term Memory (LSTM) [HS97] can be introduced to replace standard RNN, while the other structures remain. The flexibility of stacked Bi-RNNs enables CoR to apply to different real-world time-series prediction problems.

6.1.2 SGCRF

Gaussian CRF is a graphical model that can effectively model the structured information of output. It takes $\mathbf{z}$ as input and ouputs $\mathbf{y}$. This study resorts to the energy model to illustrate Gaussian CRF. The energy function $E(\mathbf{z}, \mathbf{y})$ has two terms,

$$E(\mathbf{z}, \mathbf{y}) = 2\mathbf{z}^T \Theta \mathbf{y} + \mathbf{y}^T \Lambda \mathbf{y}, \tag{6.4}$$

where the first term maps $\mathbf{z}$ to $\mathbf{y}$, parameterized by Θ, and the second term models the conditional dependencies of $\mathbf{y}$, parameterized by Λ. The inverse covariance matrix Λ is constrained to be positive-definite to ensure a valid multivariate Gaussian. The resultant Gaussian CRF is formulated as

$$p(\mathbf{y}|\mathbf{z}) = \frac{1}{Q(\mathbf{z})} exp\{-E(\mathbf{z}, \mathbf{y})\}, \tag{6.5}$$

where $Q(\mathbf{z})$ is the partition function,

$$\frac{1}{Q(\mathbf{z})} = c|\Lambda| exp\{-\mathbf{z}^T \Theta \Lambda^{-1} \Theta^T \mathbf{z}\}. \tag{6.6}$$

Parameter Θ encodes dense dependencies of $\mathbf{y}$ on $\mathbf{z}$, and parameter Λ also introduces dense conditional dependencies on $\mathbf{y}$. These dense dependencies are illustrated in Figure 6.1. It was observed that the connections are redundant and may cause overfitting [WK13]. Therefore, constraints for sparsity are applied to these dependencies by using L_1 norm to Θ and Λ in the training process. Gaussian CRF with sparse dependencies is thus called SGCRF.

The stacked Bi-RNNs offer powerful nonlinearities that can effectively represent the temporal and heterogeneous features, while SGCRF can model the structured output information. CoR, an integration of stacked Bi-RNNs and SGCRF, can take advantage of the two models for multi-step time-series prediction.

6.2 Training CoR

This chapter proposes two training methods for CoR, alternative training and end-to-end training. The following denotations are used in both training methods. Assuming there are N training samples, $\mathcal{X} = \{\mathbf{X}^{(1)}, \cdots, \mathbf{X}^{(N)}\}$ denotes the whole training feature set. $Y = \{\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(N)}\}$ denotes the N predictions, while $\dot{Y}$ denotes the corresponding N ground-truths. Z denotes the N intermediate predictions by stacked Bi-RNNs, which is also the input of SGCRF.

6.2.1 Alternative training

As CoR has two parts, it is natural to train the two modules alternatively. Alternative training first trains the stacked Bi-RNNs with Mean Absolute Error (MAE), and then trains SGCRF by minimizing its negative log-likelihood. Finally, it trains stacked Bi-RNNs and SGCRF alternatively until convergence, using a common loss function.

6.2.1.1 Initial training of stacked Bi-RNNs

This method first trains the stacked Bi-RNNs with the feature set $\mathcal{X}$ and ground-truth $\dot{Y}$. Here, MAE is the loss function for stacked Bi-RNNs. An L_2 norm is also introduced to regularize the parameter Ω. The resultant regularized loss is as follows,

$$\mathcal{L}(\Omega) = \frac{1}{NT} \sum_{n=1}^{N} \sum_{t=1}^{T} |\mathbf{z}_t^{(n)} - \dot{\mathbf{y}}_t^{(n)}| + \ \lambda_R \parallel \Omega \parallel_2, \tag{6.7}$$

noting that parameter Ω is hidden in $\mathbf{z}$ (see Equation 6.3).

Mini-batch Stochastic Gradient Descent (SGD) is used to train stacked Bi-RNNs. Adam [KB14] is a popular way to train deep neural networks. However, SGD with nesterov momentum [Nes83] shows a better result if momentum scheduling [SMDH13] is applied. Throughout this study, SGD with nesterov momentum is used as the default optimization method for stacked Bi-RNNs.

6.2.1.2 Initial training of SGCRF

Feeding the trained stacked Bi-RNNs with training feature $\mathcal{X}$, the intermediate prediction Z is obtained. Then SGCRF is trained according to Z and ground-truth $\dot{Y}$ by minimizing its negative log-likelihood, which is formulated as

$$L(\Theta, \Lambda) = -\log|\Lambda| + tr(S_{yy}\Lambda + \ 2S_{yz}\Theta + \Lambda^{-1}\Theta^T S_{zz}\Theta), \tag{6.8}$$

where S terms are empirical covariance,

$$S_{yy} = \frac{1}{N}\dot{Y}^T\dot{Y}, \quad S_{yz} = \frac{1}{N}\dot{Y}^T Z, \quad S_{zz} = \frac{1}{N}Z^T Z. \tag{6.9}$$

Equation 6.8 is a commonly used loss function for SGCRF [WK13], [MK16]. However, it is found that $|\Lambda|$ suffers a risk of overflow when the number of time step is large. Therefore, eigenvalue decomposition is introduced for Λ. As Λ is a positive-definite matrix, its determinant equals the multiplications of all eigenvalues. Thus,

$$\log|\Lambda| = \sum_{i=1}^{T} \log \lambda_i, \tag{6.10}$$

where λ_i is an eigenvalue. Equation 6.10 is used for $\log|\Lambda|$ to avoid the overflow problem.

Recall that it is necessary to apply penalties for sparsity to the parameters, the resultant loss function with L_1 regularization is then as follows,

$$\mathcal{L}(\Theta, \Lambda) = L(\Theta, \Lambda) + \ \lambda_T \parallel \Theta \parallel_1 + \lambda_L \parallel \Lambda \parallel_{1,*}, \tag{6.11}$$

where $\lambda_L \parallel \Lambda \parallel_{1,*}$ denotes the L_1 norm of Λ off diagonal elements.

The regularized loss $\mathcal{L}(\Theta, \Lambda)$ is optimized by Newton Coordinate Descent (Newton CD) with active set [WK13]. In each iteration, Newton CD finds a generalized Newton descent direction by forming a second-order approximation of the smooth part (i.e. $L(\Theta, \Lambda)$) and minimizes this along with L_1 penalties. Given the Newton direction, the parameters are updated with a step size found by line search. The positive-definite constraint of Λ is taken care of in the line search step.

6.2.1.3 Alternative tuning of stacked Bi-RNNs and SGCRF

Stacked Bi-RNNs and SGCRF are then tuned alternatively, using the final loss function $\mathcal{L}(\Omega, \Theta, \Lambda)$

$$\mathcal{L}(\Omega, \Theta, \Lambda) = \mathcal{L}(\Theta, \Lambda) + \lambda_R \parallel \Omega \parallel_2, \tag{6.12}$$

where the first term is seen in Equation 6.11, and the second term is the regularization term seen in Equation 6.7. Recall that parameter Ω is hidden in Z and thus also hidden in $\mathcal{L}(\Theta, \Lambda)$.

Stacked Bi-RNNs are tuned first with the loss function $\mathcal{L}(\Omega, \Theta, \Lambda)$. Θ and Λ are frozen and regarded as constants. The gradient of Ω can be derived according to the chain rule.

$$\frac{\partial \mathcal{L}(\Omega, \Theta, \Lambda)}{\partial \Omega} = \frac{2}{N}(\dot{Y}\Theta^T + Z\Theta\Lambda^{-1}\Theta^T) \cdot \frac{\partial Z}{\partial \Omega} + \ 2\lambda_R \Omega, \tag{6.13}$$

where $\partial Z/\partial \Omega$ is handled in the stacked Bi-RNNs. With the obtained gradient, SGD with nesterov momentum can be applied to fine-tune Ω. The fine-tuned stacked Bi-RNNs was fed with training feature $\mathcal{X}$ and obtain new Z. Then, SGCRF is trained according to the new Z and $\dot{Y}$. In this step, Ω is frozen, while Θ and Λ are optimized using Newton CD with active set. The alternated training of stacked Bi-RNNs and SGCRF repeats until they converged. In our study, five alternations are sufficient to reach convergence.

Algorithm 5 summarizes alternative training of CoR.

Algorithm 5 Alternative training of CoR

```
Input: Training feature set $\mathcal{X}$, ground-truth $\dot{Y}$;
Output: Parameters $\Omega$, $\Theta$, $\Lambda$;
1: With input $\mathcal{X}$ and $\dot{Y}$, initially train stacked Bi-RNNs with respect to the loss
   function $\mathcal{L}(\Omega)$ (Equation 6.7);
2: Feed $\mathcal{X}$ to stacked Bi-RNNs to obtain $Z$;
3: With input $Z$ and $\dot{Y}$, initially train SGCRF with respect to the loss function
   $\mathcal{L}(\Theta, \Lambda)$ (Equation 6.11);
4: while not converged do
5:     Freeze $\Theta$ and $\Lambda$, and train stacked Bi-RNNs with
       input $\mathcal{X}$ and $\dot{Y}$ according to the final loss $\mathcal{L}(\Omega, \Theta, \Lambda)$
       (Equation 6.12);
6:     Feed $\mathcal{X}$ to the trained stacked Bi-RNNs and obtain
       new $Z$;
7:     Freeze $\Omega$ and train SGCRF with input $Z$ and $\dot{Y}$
       according to $\mathcal{L}(\Omega, \Theta, \Lambda)$;
8: end while
```

In Algorithm 5, Line 1 is the initial training of stacked Bi-RNNs. Line 2 inputs $\mathcal{X}$ to stacked Bi-RNNs to obtain Z. Line 3 is the initial training of SGCRF. Lines 4-8 show the process of alternative training. Line 5 trains stacked Bi-RNNs with the final loss. Line 6 inputs $\mathcal{X}$ to the trained stacked Bi-RNNs to obtain new Z. Line 7 trains SGCRF with new Z. Lines 5-7 repeat until convergence.

6.2.2 End-to-end training

This study further proposes a more efficient end-to-end training method for CoR. Effective initial parameters are set for CoR, and then parameters are fine-tuned end-to-end.

6.2.2.1 Initializations

The first two steps of alternative training are reused to set effective initializations for CoR. Stacked Bi-RNNs are initially trained with respect to the loss function $\mathcal{L}(\Omega)$ (see Equation 6.7). The trained stacked Bi-RNNs are fed with input data $\mathcal{X}$ to obtain an intermediate prediction Z. Then SGCRF can be trained with Z and $\dot{Y}$ according to the loss $\mathcal{L}(\Theta, \Lambda)$ (see Equation 6.11). With the initial training of stacked Bi-RNNs and SGCRF, parameters Ω, Θ and Λ are effectively initialized.

The motivation of initialization is as follows. The stacked Bi-RNNs can be naturally trained with a first-order gradient descent method, while the training of SGCRF is much more challenging. Newton CD with active set has been demonstrated as an effective training method for SGCRF, which outperforms other second-order methods [WK13]. It is believed that the second-order Newton CD with active set is much more efficient than first-order methods. Consequently, Newton CD is employed to supply initializations for Θ and Λ. With well initialized parameters, first-order gradient descent is used to fine-tune the parameters end-to-end.

6.2.2.2 End-to-end fine-tuning

This study then applies gradient descent to fine-tune the parameters of CoR. However, the parameter Λ is constrained to be positive-definite, and thus the unconstrained gradient descent cannot be directly applied. This study resorts to the Projected Gradient Descent (PGD) for Λ in the end-to-end training.

PGD comprises two steps. The first step is a regular gradient descent, which updates Λ by $\Lambda^{t+1} = \Lambda^t - \alpha \nabla_\Lambda \mathcal{L}(\Omega, \Theta, \Lambda)$, where $\mathcal{L}(\Omega, \Theta, \Lambda)$ is defined in Equation 6.12, and α is the learning rate. The second step projects Λ^{t+1} back to the definition domain C of Λ. The projection is defined by $\Pi_C(\Lambda^{t+1}) = argmin_{c \in C} \parallel c - \Lambda^{t+1} \parallel^2$. The projection to a positive definite matrix has been well studied [HM12]. As Λ^{t+1} is a symmetric matrix, its eigenvalue decomposition can be written as follows

$$\Lambda^{t+1} = U \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_T \end{bmatrix} U^T \tag{6.14}$$

The projection is accordingly defined as follows,

$$\Pi_C(\Lambda^{t+1}) \Leftarrow U \begin{bmatrix} max(\mu, \lambda_1) & & \\ & \ddots & \\ & & max(\mu, \lambda_T) \end{bmatrix} U^T, \tag{6.15}$$

where μ is a small positive value.

Λ is updated by PGD, while Ω and Θ are updated by ordinary gradient descent. In implementation, both PGD and gradient descent use a mini-batch mode. In the end-to-end training, SGCRF influences what RNN learns and causes internal covariance shift [IS15] in stacked Bi-RNNs. It is beneficial to apply batch normalization [IS15] to the intermediate prediction $\mathbf{z}$. A batch normalization layer is added between stacked Bi-RNNs and SGCRF, and thereafter apply end-to-end fine-tuning.

Algorithm 6 summarizes the end-to-end training of CoR.

Algorithm 6 End-to-end training of CoR

```
Input: Training feature set $\mathcal{X}$, ground-truth $\dot{Y}$;
Output: Parameters $\Omega$, $\Theta$, $\Lambda$;
1: With input $\mathcal{X}$ and $\dot{Y}$, initially train stacked Bi-RNNs with respect to the loss
   function $\mathcal{L}(\Omega)$ (Equation 6.7);
2: Feed $\mathcal{X}$ to stacked Bi-RNNs to obtain $Z$;
3: With input $Z$ and $\dot{Y}$, initially train SGCRF with respect to the loss function
   $\mathcal{L}(\Theta, \Lambda)$ (Equation 6.11);
4: Add a batch normalization layer between stacked Bi-RNNs and SGCRF;
5: while $t < max\_iteration$ do
6:     Update $\Omega$, $\Theta$ and $\Lambda$ according to gradient descent
       rule: $\theta^{t+1} = \theta^t - \alpha\nabla_\theta\mathcal{L}(\theta)$, where $\mathcal{L}(\theta)$ is defined
       by Equation 6.12;
7:     Project $\Lambda^{t+1}$ according to Equation 6.15;
8:     $t = t + 1$;
9: end while
```

In Algorithm 6, Line 1 is the initial training of stacked Bi-RNNs. Line 2 inputs $\mathcal{X}$ to stacked Bi-RNNs to obtain Z. Line 3 is the initial training of SGCRF. Line 4 adds a batch-normalization layer for Z. Lines 5-9 show the process of end-to-end training. Line 5 trains stacked Bi-RNNs with the final loss. Line 6 updates parameters using gradient descent. Line 7 project Λ^{t+1} according to Equation 6.15 to ensure that Λ^{t+1} is positive-definite. Line 8 updates the number of steps. Lines 6-8 repeat until convergence.

Several tricks are applied to the end-to-end training. It is essential to use a smaller learning rate for the SGCRF part than the stacked Bi-RNNs part, or CoR trends to diverge. For the stacked Bi-RNNs part, it is a non-convex optimization. Therefore, learning rate decay is not used. On the contrary, the SGCRF part forms a convex optimization problem and learning rate decay should be applied.

6.3 Prediction of CoR

The prediction of CoR comprises two steps.

- **Step 1.** For a test sample, the observed feature $\mathbf{X}$ is fed to the stacked Bi-RNNs. Through a feed-forward with recurrent process, an intermediate prediction $\mathbf{z}$ is obtained.

- **Step 2.** This study predicts $\mathbf{y}$ given $\mathbf{z}$ through the prediction process of SGCRF, which seeks to maximize $p(\mathbf{y}|\mathbf{z})$. The underlying model of SGCRF is a Gaussian distribution: $\mathbf{y}|\mathbf{z} = \mathcal{N}(-\mathbf{z}\Theta\Lambda^{-1}, \Lambda^{-1})$, so the maximum of $p(\mathbf{y}|\mathbf{z})$ is the mean of the Gaussian. Thus, the final prediction $\mathbf{y}$ is formulated as

$$\mathbf{y} = -\mathbf{z}\Theta\Lambda^{-1}. \tag{6.16}$$

It can be seen that the prediction of CoR is quite efficient. Due to the Gaussian distribution, the 95% confidence interval can be derived as follows,

$$\tilde{\mathbf{y}} = \mathbf{y} \pm 1.96 diag(\Lambda^{-1}), \tag{6.17}$$

where $diag(\Lambda^{-1})$ is the diagonal elements of Λ^{-1}. This equation may assist the decision making in uncertain environments.

6.4 CoR with an Attention Mechanism

An attention mechanism is further introduced to CoR. The attention mechanism in [BCB14] was applied to an encoder-decoder framework. The principle of attention is that it associates the current prediction to the weighted input sequence, where each weight is determined by the relatedness of each step input to the current prediction. This study adapts the attention for a sole RNN network, and then extends attention to stacked Bi-RNNs, which is the bottom part of CoR.

Recall that a standard RNN is defined by Equations 6.1 and 6.2. Equation 6.1 is modified as follows.

$$\mathbf{h}_t = f(\mathbf{s}_t, \mathbf{h}_{t-1}), \tag{6.18}$$

where $\mathbf{s}_t$ represent the current attention unit. $\mathbf{s}_t$ is a combination of input step features, written as

$$\mathbf{s}_t = \sum_{j=1}^{L} \alpha_{ij}\mathbf{x}_j, \tag{6.19}$$

where L is the size of attention window, and α_{ij} is the normalized weight of step feature $\mathbf{x}_j$,

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{L}\exp(e_{ik})}, \tag{6.20}$$

where e_{ij} is learned by a feed-forward neural network a.

$$e_{ij} = a(\mathbf{h}_{i-1}, \mathbf{x}_j). \tag{6.21}$$

For the current prediction, the standard RNN only considers the current step feature, while the RNN with attention associates the current prediction with L weighted input step features. Therefore, RNN with attention can achieve better performance than vanilla RNN.

The attention mechanism can be easily extended to stacked Bi-RNNs. For the first RNN layer, attention is employed as the above procedure. For the second RNN layer, the output of the first RNN layer is regarded as the represented features, and thus attention can be introduced similarly. Similar processes repeat for more RNN layers.

When stacked Bi-RNNs with an attention mechanism are built up, SGCRF can also be put on top, and can result in CoRa. As RNN with attention can be trained by SGD similarly as RNN, stacked Bi-RNNs with attention can be trained end-to-end via SGD. As a consequence, the proposed alternative training and end-to-end training can also be applied to CoRa. For prediction, RNN with attention is similar as the standard RNN, which is accomplished via a feed-forward with recurrent process. Therefore, CoRa can also reuse the two-step prediction process of CoR.

The improvement of CoRa over CoR is that CoRa utilizes RNN with attention instead of standard RNN. RNN with attention considers wieghted input step features for the current prediction, so as to outperform the standard RNN. As a result, CoRa can achieve better prediction accuracy than CoR, with the price of more computational cost.

6.5 Model Comparisons

This section compares SGCRF, stacked Bi-RNNs, CoR and CoRa for multi-step time-series prediction. This comparison still uses the multi-step time-series prediction defined by Equation 5.1, where the total number of time steps is T, and the dimension of features in each step is D. The four models are compared in two aspects: model sizes and computational costs.

6.5.1 Model sizes

The four models, SGCRF, stacked Bi-RNNs, CoR and CoRa model the multi-step time-series prediction problem from different perspectives. As a consequence, the sizes of the four models vary significantly.

SGCRF models multi-step time-series prediction via two parameter matrices Θ and Λ. The Θ matrix linearly maps $D \times T$ input features to a output vector with length T. Thus, the size of Θ matrix is $(D \times T) \times T$. The Λ matrix models the dependencies in the T-dimension output vector, with a size of $T \times T$. In summary, the model size of SGCRF is approximated as DT^2.

Stacked Bi-RNNs is a stack of several Bi-RNNs. Firstly, the model size of RNN is analyzed. Recall that RNN is defined by Equations 6.1 and 6.2. There are three parameter matrices $\mathbf{W}_{xh}$, $\mathbf{W}_{hh}$ and $\mathbf{W}_{hz}$. $\mathbf{W}_{xh}$ is an input projection matrix that maps D-dimension features to hidden neurons. Usually, the number of hidden neurons is close to D. Thus, the size of $\mathbf{W}_{xh}$ is in proportion to $D \times D$. $\mathbf{W}_{hh}$ is a recurrent matrix that models the transitions in hidden states. As the number of hidden neurons can be approximated by D, its size is also in proportion to $D \times D$. $\mathbf{W}_{hz}$ is an output projection matrix that maps hidden states to output, whose size is also in proportion to $D \times D$. The three matrices are shared through all time steps. Therefore, the model size of RNN can be approximated as $3D^2$. Assuming there are k Bi-RNNs in the stack, The final model size of stacked Bi-RNNs is approximated as $6kD^2$.

CoR has two parts. The size of stacked Bi-RNNs has been approximated as $6k \times D \times D$. Stacked Bi-RNNs produces the intermediate T-dimension prediction, which greatly reduces the input size of SGCRF. As a consequence, the sizes of Θ and Λ matrices are both $T \times T$. In the end, the model size of CoR is approximately $6kD^2 + 2T^2$.

Compared to CoR, CoRa further introduces attention units. For one RNN, the size of attention unit is $T \times L$, where L is the size of the attention window. For stacked Bi-RNNs with stack size k, the total size of attention units is $2kTL$. In summary, the size of CoRa is an approximation of $6kD^2 + 2kTL + 2T^2$.

Table 6.1: Model size of SGCRF, stacked Bi-RNNs, CoR and CoRa

Model	SGCRF	stacked Bi-RNNs	CoR	CoRa
Size	DT^2	$6kD^2$	$6kD^2 + 2T^2$	$6kD^2 + 2kTL + 2T^2$

Table 6.1 summaries the sizes of the four models. For general multi-step time-series prediction problems, SGCRF model has the largest size. CoR and CoRa can significantly improve the performance of stacked Bi-RNNs, while the model sizes of CoR and CoRa do not increase much.

6.5.2 Computational costs

SGCRF, stacked Bi-RNNs, CoR and CoRa are trained with different methods. This study analyzes the training algorithms of the four models to show their computational cost.

SGCRF is trained using second-order Newton CD with active set. In training process, the main computation of SGCRF is the calculation of Hessian matrix with a size $(T \times D + T) \times (T \times D + T)$. This is a large matrix whose size increases quadratically with respect to T or D. Moreover, the second-order optimization consumes much more time than the first-order method in one iteration. For large time-series prediction problems, the computation cost of SGCRF can be very large.

Stacked Bi-RNNs are trained via first-order SGD. For efficiency, RNN is unrolled during training, which may take up space in proportion to $T \times D \times D$. However, this is much less than the Hessian matrix in SGCRF learning. Besides, stacked Bi-RNNs uses first-order SGD in optimization, which usually consumes less time than second-order Newton methods.

For CoR, the more efficient end-to-end training is focused on. The first step is the initial training of stacked Bi-RNNs, which has been analyzed in the above paragraph. As stacked Bi-RNNs output an intermediate prediction, the input size of SGCRF has been greatly reduced. The size of Hessian matrix in SGCRF layer is now reduced to $(2T) \times (2T)$. In the end, the training time of CoR can be much less than that of SGCRF model.

CoRa shares a similar training process as CoR. As CoRa further introduces an attention mechanism, the computational cost increases. Compared to CoR, CoRa has extra $2k$ attention matrices to learn, where k is the number of Bi-RNNs in the stack. In experiment, CoRa consumes more training time than CoR.

In the experimental section, a large synthetic time-series prediction problem is used to test the computational cost of SGCRF, stacked Bi-RNNs, CoR and CoRa. Results are shown in Table 6.2.

6.6 Experiments

CoR and CoRa are evaluated by three experiments using both synthetic data and real-world data.

Experiment 1 evaluates CoR and CoRa by synthetic data. This experiment demonstrates the advantages of CoR and CoRa over RNN and SGCRF with synthetic data.

Experiment 2 evaluates CoR and CoRa using a real-world forecasting of sea surface temperature.

Experiment 3 evaluates CoR and CoRa using a real-world electricity demand prediction competition.

In the evaluations by real-world data, CoR and CoRa show significantly better performance than state-of-the-art methods in time-series prediction.

In implementations, Theano [The16] is used for deep neural network. For the Newton CD in training SGCRF, it is referred to [WK13].

6.6.1 Experiment 1: evaluations using synthetic data

Through the evaluations using synthetic data, this experiment demonstrates the advantages of CoR and CoRa over SGCRF and stacked Bi-RNNs. As the number of time steps and training samples of synthetic time-series data can be changed, some interesting properties of evaluated models are discovered. This experiment furthermore conducts an evaluation using large-scale data to compare the computational cost of SGCRF, stacked Bi-RNNs, CoR and CoRa.

6.6.1.1 Evaluations using controlled data

The models to be evaluated include SGCRF, stacked Bi-RNNs, RNN+SGCRF (a combination of stacked Bi-RNNs and SGCRF without joint training, which can be defined by Lines 1-3 in Algorithm 5), CoR trained by alternative training (CoR_v1), CoR trained by end-to-end training (CoR_v2) and CoRa trained by end-to-end training. This evaluation changes the time steps and sample quantities of synthetic data to evaluate different models, and find some interesting properties of these models.

The synthetic data are generated as follows. The dimension of feature D is fixed as 10, while the time step T can be varied in $\{10, 20, 40\}$, and number of samples N can be varied in $\{1000, 2000, 4000, 8000\}$. The data generation includes three steps:

1) Generating temporally correlated features X. 10 random real values are sampled from a standard Gaussian $\mathcal{N}(0, 1)$, denoted as $\mathbf{x}$. The feature $\mathbf{x}_i$ for each time step is generated by $\mathbf{x}_i = (1 - 0.2\mathbf{x}) * \sin(i\pi/T)$, where the sine function models temporal correlations.

2) Introducing nonlinear transformations to features. Each $\mathbf{x}_i$ is transformed by a dense network parameterized by W, which is a $D \times D$ matrix with diagonal $W_{i,i} = 0.8$, sub-diagonal $W_{i-1,i} = -0.2$, super-diagonal $W_{i,i+1} = -0.3$, and the other elements are assigned as zeros. The following transformation is a dense network parameterized by $\mathbf{v}$ and $b = 0.3$, where $\mathbf{v}$ is a D-dimension vector $\mathbf{v}_i = 0.2$. This neural network reduces the dimension of each step output to 1. For both networks, the non-linear activation is leaky ReLU [XWCL15] with leakiness = 0.2.

3) Introducing structured information on output. The output of neural network is then put into a SGCRF parameterized by Θ and Λ. Θ is a $T \times T$ diagonal matrix with $\Theta_{i,i} = 0.2$, and Λ is a $T \times T$ matrix with diagonal $\Lambda_{i,i} = 0.8$, sub-diagonal $\Lambda_{i-1,i} = -0.5$ and super-diagonal $\Lambda_{i,i+1} = 0.2$. The output of SGCRF is the final sequence $\mathbf{y}$ to be predicted.

The simulated multi-step time-series prediction problem has temporal feature correlations, nonlinear feature transformations and structured information. It is adequate to evaluate the performances of different models on time-series prediction. Neural networks and SGCRF are used to encode feature nonlinearities and structured information for convenience, while they are actually equivalent to nonlinear functions. Figure 6.2 illustrates time-series samples generated by the three steps.

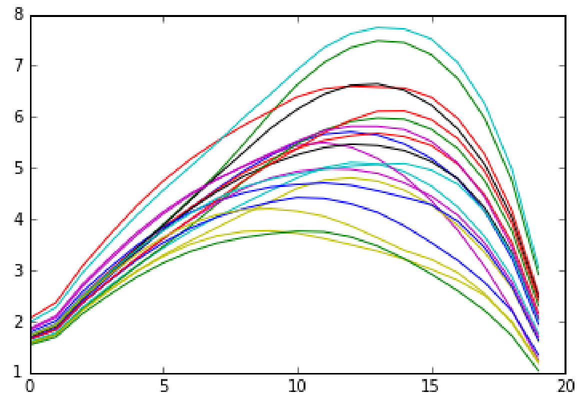

Figure 6.2: An illustration of generated time-series data. The number of time step is 20. 20 random samples are drawn in this figure.

This experiment evaluates models on different time steps with different numbers of samples. In each evaluation, random 80% samples are used for training, while the rest are for testing. Mean Absolute Percentage Error (MAPE) is used as the metric, which is defined as $|g - \hat{g}|/g \times 100\%$, where g is ground-truth and $\hat{g}$ is prediction. SGCRF does not have any hyper-parameters. The stacked Bi-RNNs has two Bi-RNNs layers, each with a hidden size of 16. CoR inherits the configurations of stacked Bi-RNNs. For CoR, the size of attention window is 16. For different models, hyper-parameters are tuned by random search [BB12].

For clarity, the evaluation results are shown in two figures, Figure 6.3 and Figure 6.4. This study analyzes the results in individual figures, and then summarize across figures.

In Figure 6.3(a), the time step of time-series data is 10. With different numbers of training samples, the MAPE of SGCRF is relatively stable, about 3.2%. Stacked Bi-RNNs achieve a lower MAPE than SGCRF. When the number of training samples increases, the MAPE of stacked Bi-RNNs decreases significantly. This is an indication that the performance of deep neural networks relies heavily on the quantity of training data. RNN+SGCRF obtains a little improvement against stacked Bi-RNNs. CoR_v1 and CoR_v2 significantly outperform the previous three models, and CoR_v2 consistently achieves the lowest MAPE. It is stressed that when training samples are few, CoR improves stacked Bi-RNNs greatly. This suggests that CoR is more robust than deep neural networks to the number of training samples.

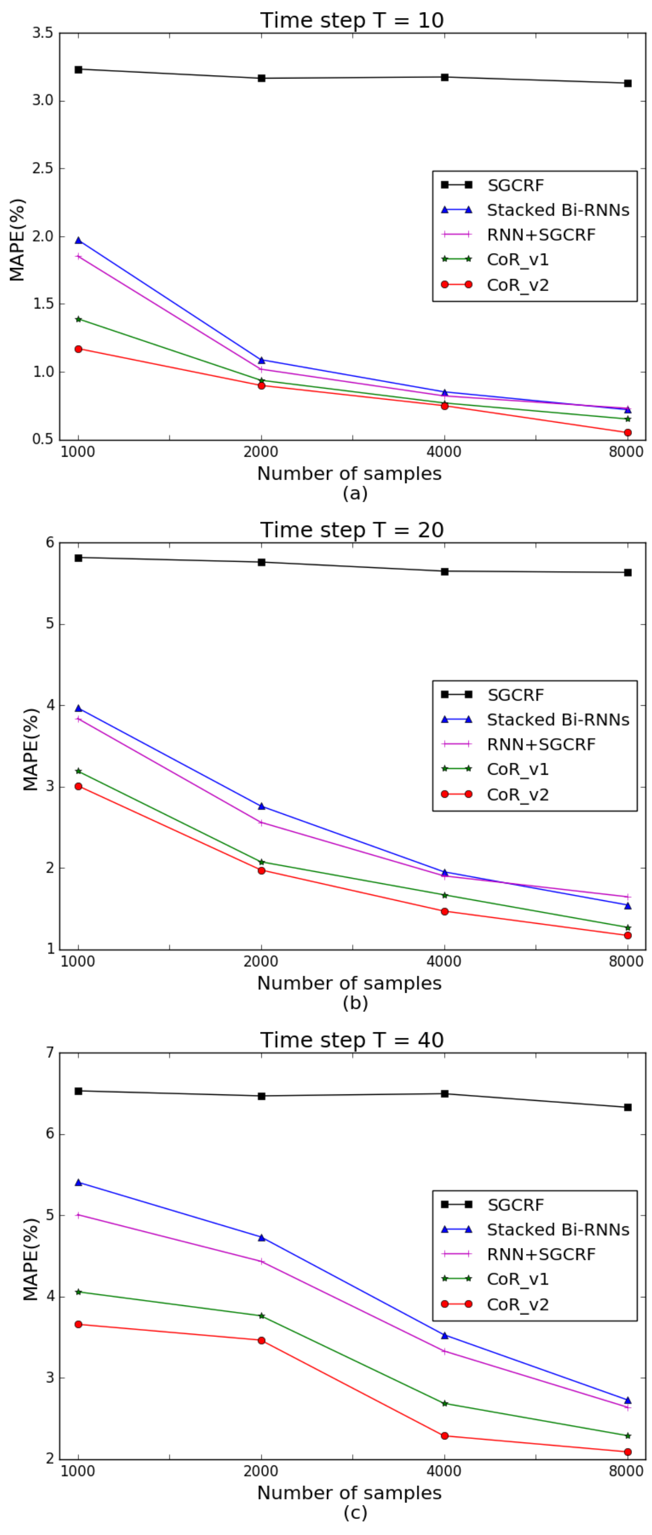

Figure 6.3: Evaluations of five models on synthetic data. Note that X axes are uneven and the scales of Y axes are different.

In Figure 6.3(b), the time step of time-series data is 20. For the five models, similar trends are observed in performance variations as those in Figure 6.3(a). When the time step increases to 40 in Figure 6.3(c), the observed rules of MAPEs of different models stay the same.

This study then summarizes evaluation results across Figure 6.3(a)-(c). From Figure 6.3(a) to Figure 6.3(c), the time step of time-series data is increasing, which indicates that the difficulty of prediction increases. It can be seen that the MAPEs of different models increase when the number of time step increases. SGCRF does not perform well because it misses the nonlinearities in features. Stacked Bi-RNNs outperform SGCRF as they have powerful nonlinearities to fit the data. It is noted that the relative gap of stacked Bi-RNNs and SGCRF decreases when the number of training sample is small (i.e. $N = 1000$). SGCRF is a sparse model that can generalize well with limited training samples, while the performance of stacked Bi-RNNs strongly depends on the number of training samples. A simple combination of stacked Bi-RNNs and SGCRF (i.e. RNN+SGCRF) can improve performance, but not always (see $N = 8000$ in Figure 6.3 (a) and (b)). CoR can reliably improve MAPE over stacked Bi-RNNs and SGCRF. The end-to-end trained CoR (CoR_v2) consistently achieves the best performance. Especially, when the training samples are limited, CoR shows significant advantage over other models. CoR's robustness to number of training samples may ascribe to SGCRF, a sparse model that generalizes well to limited training samples.

Figure 6.4 shows the results of stacked Bi-RNNs, stacked Bi-RNNs with attention, CoR and CoRa. Both CoR and CoRa are trained by end-to-end method. In Figure 6.4(a), it can be seen that the attention mechanism significantly improves the performance of stacked Bi-RNNs in terms of prediction accuracy. MAPE achieved by stacked Bi-RNNs with attention is close, but not so good as CoR. CoRa can produce even better prediction than CoR. In the case of small quantity of training data (the number of sample is 1000), CoR and CoRa show a significant advantage over stacked Bi-RNNs and stacked Bi-RNNs with attention. This may attribute to SGCRF, which is not sensitive to the number of training samples. In Figure 6.4(b) and 6.4(c), the similar rule of the performances of the four models can be observed. Through Figure 6.4(a)-(c), it can be seen that with the increase of time steps, prediction accuracies of different models decrease more or less.

6.6.1.2 Evaluations using large-scale data

SGCRF, stacked Bi-RNNs, CoR and CoRa are further evaluated on large problems to compare their precision and computation cost. 100K time-series samples are generated with time step $T = 200$ and step feature $D = 100$. The synthetic data are generated in the previous way, while the sizes of parameters are increased accordingly. In this large problem evaluation, LSTM replaces standard RNN in stacked Bi-RNNs, CoR and CoRa, because standard RNN has difficulties in modeling long sequences. CoR and CoRa are trained by the end-to-end training method, which is more efficient than alternative training. Evaluations are conducted on a server

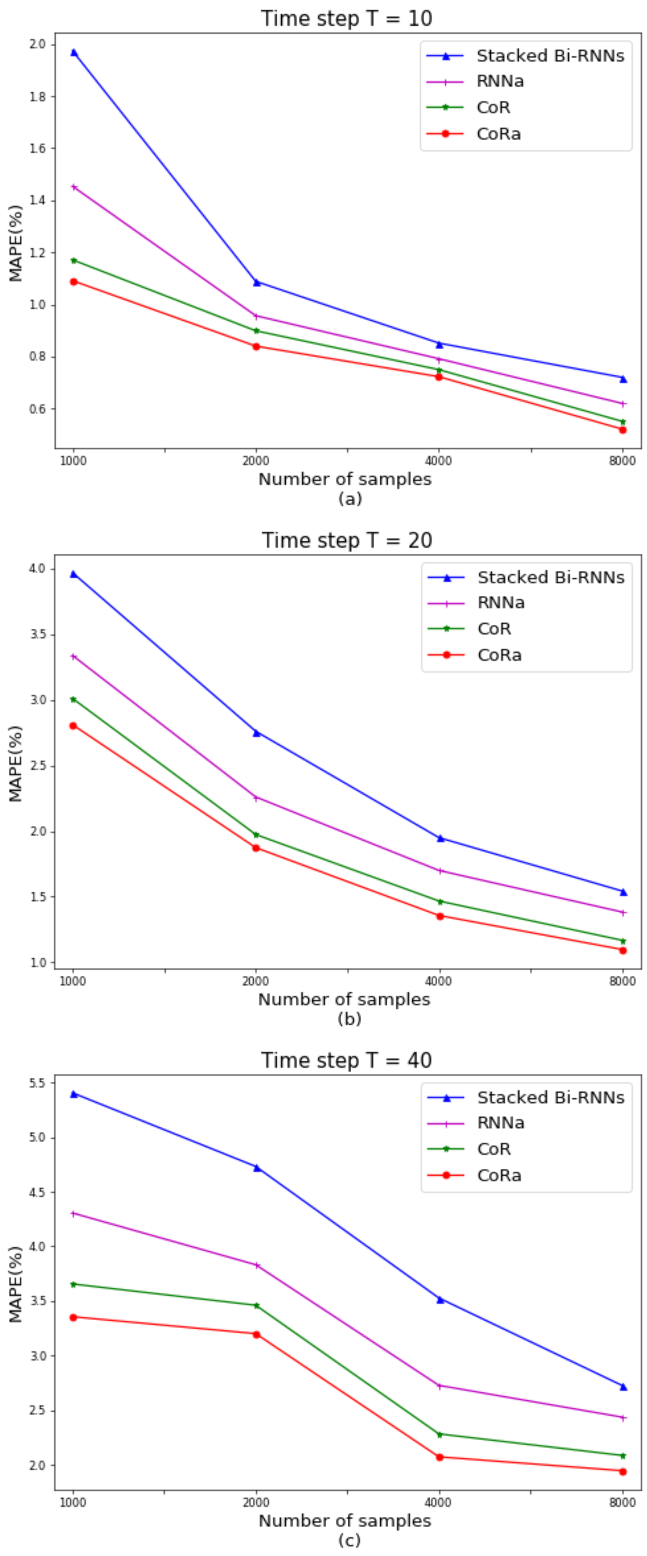

Figure 6.4: Evaluations of four models on synthetic data. Note that X axes are uneven and the scales of Y axes are different.

with 8 CPUs and 64 GB Memory. MAPE is used to measure prediction precisions of different models. Table 6.2 summarizes the precisions and computation costs of the three models.

Table 6.2: Comparison of SGCRF, stacked Bi-LSTMs, CoR(LSTM) and CoRa(LSTM) on a large time-series prediction problem.

Model	**MAPE**	**Memory usage**	**Time usage**
SGCRF	6.5	17.21G	50.33h
stacked Bi-LSTMs	3.2	3.53G	5.43h
CoR(LSTM)	2.6	3.55G	9.41h
CoRa(LSTM)	2.3	4.02G	16.21h

In Table 6.2, the prediction precisions of the four models accord with previous rules. Both CoR and CoRa achieved significant improvement over SGCRF and stacked Bi-RNNs. CoRa obtained the lowest MAPE among the four models. The four models were compared in theory in Section 6.5. SGCRF consumed the largest memory space for two reasons: 1. Its model size was the largest. 2. Its training process used second-order optimization method, in which the size of Hessian matrix was very large. The training time of SGCRF was also the largest because the second-order Newton method was slower than the first-order SGD. Stacked Bi-RNNs was the most efficient in terms of memory space and training time. In comparison, CoR did not increase much space cost, while the training time increased a lot. CoRa also did not increase much in space cost, but its training time was much more than stacked Bi-RNNs and CoR.

6.6.2 Experiment 2: evaluations on forecasting of sea surface temperature

In this experiment, CoR and CoRa are evaluated on a real-world Sea Surface Temperature (SST dataset) forecasting dataset [SLW+85] from Climate Prediction Center [a]. Sea surface temperature is closely related to the climate across the globe. Forecasting of sea surface temperature facilitates resource planning and has been widely studied [TTMH98, THMT00]. Therefore, the proposed CoR and CoRa are tested on this significant multi-step time-series prediction problem.

SST dataset contains monthly temperature (average of temperatures in one month) data from Jan, 1970 to Mar, 2003. The sampled spots range from 124°E to 70°W by two-degree intervals, 29°S to 29°N by by two-degree intervals. This experiment predicts one-year-ahead temperatures in this dataset. The data from Jan, 1970 to Dec, 1990 are used as training set, and data from Jan, 1991 to Dec, 2002 are used as testing set. The features include year, month, latitude, longitude and an El Nino indicator.

[a]https://iridl.ldeo.columbia.edu/SOURCES/.CAC/.sst/?Set-Language=en

For this multi-step time-series prediction problem, the length of time step is 12. RNN is capable of handling sequences with such length. Stacked Bi-RNNs are built with two layers, and each RNN has a hidden size of 8. CoR and CoRa are built upon the aforementioned stacked Bi-RNNs. Two training methods are applied to CoR and CoRa, respectively. In the end, four models are obtained, CoR by alternative training (CoR_v1), CoR by end-to-end training (CoR_v2), CoRa by alternative training (CoRa_v1) and CoRa by end-to-end training (CoRa_v2). Other methods that are widely used in time-series prediction are also evaluated for reference. The traditional ARIMA model is evaluated as the baseline. SGCRF, RNN and stacked Bi-RNNs are also tested for comparison.

The results of different models are shown in Table 6.3. MAPE is used to measure the prediction accuracy of tested models.

Table 6.3: Evaluation results of sea surface temperature forecasting.

Model	**MAPE**
ARIMA	4.75%
SGCRF	4.13%
RNN	$(3.90 \pm 0.12)\%$
stacked Bi-RNNs	$(3.68 \pm 0.11)\%$
CoR_v1	$(3.25 \pm 0.04)\%$
CoR_v2	$(3.18 \pm 0.02)\%$
CoRa_v1	$(3.14 \pm 0.03)\%$
CoRa_v2	$(3.08 \pm 0.03)\%$

In Table 6.3, the baseline ARIMA model achieved an average MAPE (over all the sampled spots) of 4.75%. SGCRF obtained lower MAPE than the baseline, and neural network models, RNN and stacked Bi-RNNs, achieved even better prediction accuracies. The disadvantage of neural network models is the fluctuations in test results, arising from the unstable results of non-convex optimization using stochastic gradient descent.

CoR and CoRa models achieved significant better prediction accuracy than previous methods. It was notable that CoR and CoRa were built upon the stacked Bi-RNNs that produce a median prediction result among all the tests. Compared among the four models, it can be seen that end-to-end training shows slightly lower MAPE than alternative training and CoRa shows better results than CoR. Besides, the fluctuations caused by neural networks are alleviated because CoR and CoRa conduct fine-tuning upon the pre-trained neural network models.

Figure 6.5 shows a sample from the SST dataset, which is a monthly temperature time-series in 2002 at (120°W, 29°N).

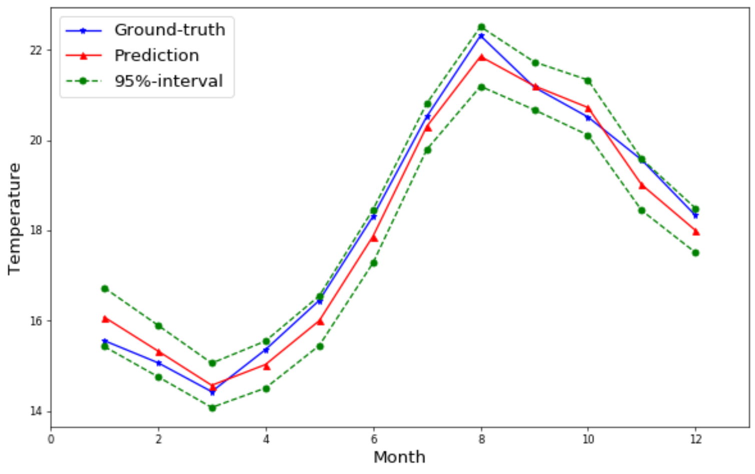

Figure 6.5: An illustration of ground-truth and prediction in sea surface temperature forecasting. The 95% confidence intervals are also shown in this figure.

In Figure 6.5, blue line is the ground-truth and the red line is the result of prediction, produced by CoRa_v2. The 95% confidence intervals can be calculated by Equation 6.17, which are shown in red dashed lines. The 95% confidence intervals can measure the uncertainty of the prediction model in theory, and assist decision-makings in uncertain environments in practice.

6.6.3 Experiment 3: evaluations on energy demand prediction

In this experiment, CoR and CoRa are applied to an electricity demand prediction problem, which is a competition called NPower Forecasting Challenge 2016 [b]. Demand prediction plays a fundamental role in power systems, and it has been studied for decades [HBA+14, MZR12, WZR16]. The improvement in demand prediction in power systems will greatly enhance the utilities of power companies and increase the energy efficiency of the whole power system. This competition adopted a rolling forecasting mode to simulate the real-world scenario. This experiment follows this mode to evaluate CoR and other comparison models in Round one and Round two. The ground-truth of Round three is not available.

The task of this competition is to predict the future power demand in every half hour according to weather data. In Round one, the training data range from 2012-01-01 to 2014-09-31, and the task is to predict power demand from 2014-10-01 to 2015-03-01. In Round two, the rolling forecasting releases the ground-truth from 2014-10-01 to 2015-03-31, and accordingly the task becomes predicting the power demand from 2015-04-01 to 2015-09-30.

[b]https://www.npowerjobs.com/graduates/forecasting-challenge. Data are publicly available. Competition results are also published on this webpage.

In this time-series prediction problem, the features extracted include the following three categories: 1) Temporal feature, including the year, the month and the number of time step; 2) Calendar feature, including the day of a week and the public holiday; 3) Weather feature, including temperature, cloudiness, altogether nine measurements. Features and ground-truths are normalized by Gaussian (subtracting the mean and dividing by variance), and then input to different models.

ARIMA is used as the baseline model. SGCRF and RNN are also evaluated. SGCRF is evaluated in two modes: SGCRF without feature engineering (SGCRF_w/o) and SGCRF with feature engineering (SGCRF_w). A standard RNN is very hard to train (note: the number of time steps is 48, which is a long sequence), and thus LSTM is evaluated instead.

A two-layer stacked Bi-RNNs can converge steadily, which is evaluated for reference. Stacked Bi-LSTMs are also evaluated as comparison.

CoR can have two variants, which are based on stacked Bi-RNNs (CoR(RNN)) and stacked Bi-LSTMs (CoR(LSTM)). The two variants are trained by alternative training and end-to-end training. Overall, there are four CoR variants (CoR(RNN)_v1, CoR(RNN)_v2, CoR(LSTM)_v1, CoR(LSTM)_v2) to evaluate. It is similar for CoRa. This experiment evaluates CoRa(RNN)_v1, CoRa(RNN)_v2, CoRa(LSTM)_v1, CoRa(LSTM)_v2.

Moreover, other two state-of-the-art methods are evaluated, namely, Gradient boosting [CG16] with feature engineering (adding previous time step features to the current), and attentional stacked Bi-LSTMs [BCB14].

The hyper-parameters of CoR and CoRa are tuned by grid search. All the hyper-parameters come from the stacked Bi-RNNs/LSTMs part. For the stacked Bi-RNNs, there are two Bi-RNN layers, where each RNN has 16 hidden neurons. For the stacked Bi-LSTMs, there are also two Bi-LSTM layers, and the size of hidden unit in LSTM is 16. Though more stacked layers in deep neural networks may increase representational capacities, it is verified that two stacked layers could produce the best results. This might be related to the number of training samples. If the training samples are limited, deep neural networks are not sufficiently trained.

Unless clearly noted with feature engineering, all models are evaluated on the raw features. This competition uses MAPE as the evaluation metric. Table 6.4 summarizes the evaluation results of different models.

Table 6.4: Evaluation results of the two rounds in NPower Forecasting Challenge 2016.

Model	MAPE in round 1	MAPE in round 2
1^{st} Place	3.14%	7.13%
2^{nd} Place	6.43%	4.89%
3^{rd} Place	7.84%	7.48%
ARIMA	8.95%	8.77%
SGCRF_w/o	5.83%	6.64%
SGCRF_w	4.91%	5.60%
LSTM	$(4.92 \pm 0.15)\%$	$(4.73 \pm 0.17)\%$
stacked Bi-RNNs	$(4.88 \pm 0.21)\%$	$(4.56 \pm 0.20)\%$
stacked Bi-LSTMs	$(4.43 \pm 0.16)\%$	$(4.31 \pm 0.14)\%$
Gradient boosting	4.90%	4.61%
Attentional LSTM	$(4.21 \pm 0.15)\%$	$(4.12 \pm 0.16)\%$
CoR(RNN)_v1	$(4.25 \pm 0.04)\%$	$(4.10 \pm 0.05)\%$
CoR(RNN)_v2	$(4.19 \pm 0.06)\%$	$(4.03 \pm 0.05)\%$
CoR(LSTM)_v1	$(4.11 \pm 0.04)\%$	$(3.94 \pm 0.03)\%$
CoR(LSTM)_v2	$(4.05 \pm 0.05)\%$	$(3.87 \pm 0.03)\%$
CoRa(RNN)_v1	$(4.11 \pm 0.05)\%$	$(4.01 \pm 0.05)\%$
CoRa(RNN)_v2	$(4.04 \pm 0.06)\%$	$(3.91 \pm 0.04)\%$
CoRa(LSTM)_v1	$(3.95 \pm 0.03)\%$	$(3.78 \pm 0.04)\%$
CoRa(LSTM)_v2	$(3.89 \pm 0.05)\%$	$(3.70 \pm 0.04)\%$

In Table 6.4, the top three results in this competition are listed for reference. These winning methods did not employ sophisticated models, but were concerned with detailed features and feature engineering [c]. The basic ARIMA model achieved MAPE of 8.95% and 8.77%, which were not as good as the top three results in the competition.

SGCRF_w/o was almost as competitive as the 2^{nd} Place, while SGCRF_w showed significantly improvement compared to SGCRF_w/o. Models based on deep neural networks outperformed SGCRF. The overall performance of LSTM was comparable to the 1^{st} Place. Stacked Bi-RNNs were slightly better than a single LSTM. In contrast, stacked Bi-LSTMs achieved lower MAPE than stacked Bi-RNNs. It suggests that LSTM is more effective than standard RNN when the number of time step gets larger. Even though feature engineering was applied to gradient boosting, it did not show any advantage over deep RNN models. The state-of-the-art

[c]http://blog.drhongtao.com/2016/12/winning-methods-from-npower-forecasting-challenge-2016.html

attentional LSTM (also in a stacked bi-directional structure) slightly outperformed stacked Bi-LSTMs.

The four variants of CoR showed better results compared to previous models. In building different CoR variants, this study used deep RNN models which achieved median performances (note the fluctuations in results). CoR(RNN)_v2 achieved an average MAPE of 4.11%, which was a relative 10.8% improvement on stacked Bi-RNNs. Similarly, CoR(LSTM)_v2 achieved a relative 9.8% improvement on stacked Bi-LSTMs. Moreover, CoR models were more stable than deep RNNs. The best model, CoR(LSTM)_v2 achieved an average MAPE of 3.96%, which was much better than 5.14%, the average MAPE of 1^{st} Place. The four variants of CoRa achieved lower MAPE than the corresponding CoR variants. The relative improvement was around 3%. The disadvantage of CoR, CoRa and deep RNN models was that the result suffers fluctuations, but this could be compensated by model ensemble.

Figure 6.6 shows three random samples predicted by CoR(RNN)_v2.

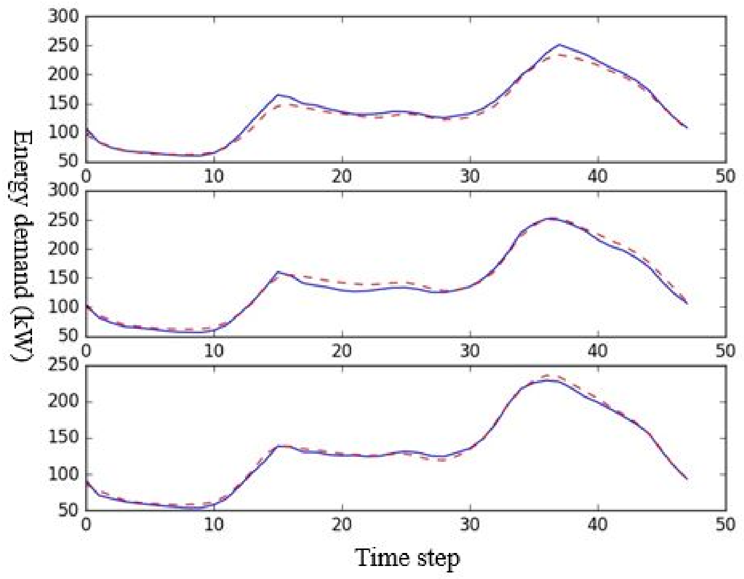

Figure 6.6: Three random samples in demand prediction. The blue solid line is the ground-truth and the red dashed line is the prediction.

This experiment did not visualize the best results by CoRa(LSTM)_v2, because the predicted values were very close to the true values, which were hard to distinguish. Through this visualization, the benefit of structured information can be further understood. In a geometric view, the cure shape can represent structured information. In CoR model, it not only tries to make each predicted value close to the truth, but also tries to match the shapes of prediction curve and ground-truth curve. That is how CoR achieves excellent performance in multi-step time-series prediction.

6.7 Discussions

This section further analyzes SGCRF, stacked Bi-RNNs, CoR and CoRa models according to evaluation results. The properties of the four models are discussed. Then how the attention mechanism and structured information can improve RNN are analyzed and compared.

6.7.1 Model properties

Summary across the evaluation results shown in Figure 6.3 and Figure 6.4 in the Experiment 1, the following rules are discovered in terms of prediction accuracy.

- SGCRF showed robust performances under the varying number of training samples.
- Stacked Bi-RNNs performed better and better with the increasing number of training samples.
- CoR showed significant improvements upon stacked Bi-RNNs. In the case of insufficient training samples, CoR outperformed stacked Bi-RNNs by a large gap.
- CoRa inherited the property of CoR, but produced more accurate predictions.

SGCRF is a sparse model that can generalize well to a small number of training samples. In accord with empirical rules of deep neural networks, prediction accuracy of stacked Bi-RNNs significantly increases with the increasing number of training samples. CoR can significantly outperform stacked Bi-RNNs because it introduces structured information. When the number of training samples is small, CoR can achieve much better performance than stacked Bi-RNNs, which may attribute to S-GCRF, a sparse model that is robust to number of training samples. CoRa performs better than CoR because it further introduces an attention mechanism.

Section 6.5 compared the four models in theory, while Subsection 6.6.1 experimentally showed the space and time cost of the four models. SGCRF took the largest space and time costs because of its large model size and second-order training method. Stacked Bi-RNNs had the least space and time cost. Though CoR was more complex than stacked Bi-RNN, it did not increase much on space and time cost. These results approve that CoRa can further improve CoR, but its time cost significantly increases.

In the real-world experiment Subsections 6.6.2 and 6.6.3, it could be seen that the four models greatly outperformed the baseline model ARIMA. CoR and CoRa consistently showed significant improvements over SGCRF and stacked Bi-RNNs. This approves that CoRa can achieve better prediction accuracy than CoR, but it indeed consumes more training time than CoR.

6.7.2 Structured information and attention

This subsection further analyzes how structured information and the attention mechanism improve RNN. In multi-step time-series prediction, the output variables predicted by RNN are independent. Actually, there are strong dependencies between output variables, termed as structured information in this study. CoR incorporates the structured information into RNN by adding an SGCRF layer, and consequently achieves more accurate predictions than RNN.

The Λ matrix in SGCRF encodes the structured information, i.e. dependencies in the output variables. The Λ matrix can be visualized to show these dependencies. Figure 6.7 shows the Λ matrix in CoR before joint training (defined by Lines 1-3 in Algorithm 5) in the demand prediction experiment. the local dependencies can be clearly seen in the visualization.

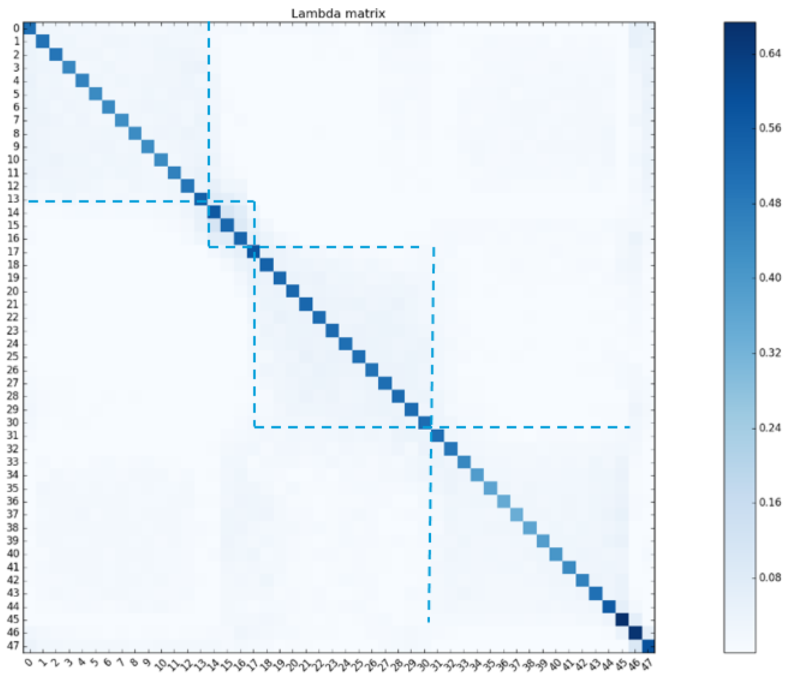

Figure 6.7: Visualization of Λ matrix in CoR before joint training.

Figure 6.8 shows the Λ matrix after end-to-end training. It can be seen that local dependencies disappear, while global dependencies come into effect. These two visualizations show how Λ matrix gradually utilizes the global structured information to improve RNN in multi-step time-series prediction.

CoR takes advantage of structured information to improve RNN, while CoRa can further improve CoR using the attention mechanism. Structured information means the dependencies in the output variables, while the attention mechanism focuses on more effective input features. Compared standard RNN defined by Equation 6.1 and RNN with an attention mechanism defined by Equation 6.18, it is found that standard RNN only considers the current step feature for the current

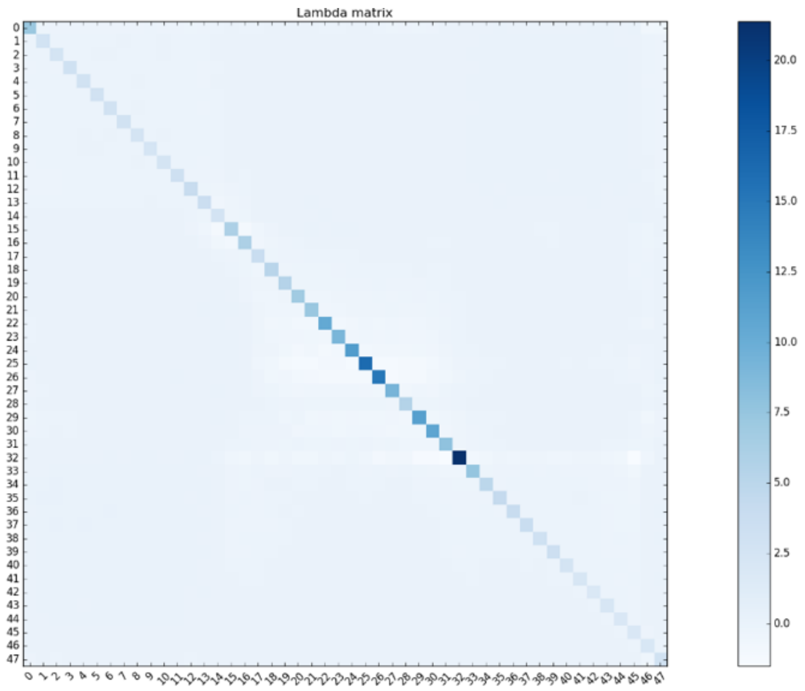

Figure 6.8: Visualization of Λ matrix in CoR after end-to-end training.

prediction, while RNN with an attention mechanism can associate the current prediction to a number of related weighted step features. This study concludes that the attention mechanism and structured information improve RNN from two different ways. The attention mechanism improves RNN from the input side, while structured information improves RNN from the output side. CoRa model utilizes both the attention mechanism and structured information, so as to achieve the best prediction accuracy among different models.

6.8 Summary

This chapter proposed CoR, which integrated the advantages of RNN and SGCRF, for multi-step time-series prediction. Two training methods were proposed for CoR. Experimental results showed that the end-to-end training was more efficient than the alternative training. This chapter also proposed CoRa, which improved CoR by introducing an attention mechanisms for RNN. The evaluations with both synthetic data and real-world data demonstrated that CoR and CoRa could significantly improve prediction precision over stacked Bi-RNNs and SGCRF. CoR and CoRa also outperformed other state-of-the-art methods in the real-world multi-step time-series prediction. Discussions were also provided to analyze the advantages of CoR and CoRa.

Chapter 7

Conclusions and Future Work

This chapter summarizes the contributions of this thesis and discusses the potential directions of future work.

7.1 Contributions of This Thesis

This research focuses on developing innovative machine learning methods to solve several challenging issues in demand management of Smart Grid market. The contributions of this thesis include:

1. An intelligent broker model was developed for effective management of S-mart Grid market. In proposed broker design, the challenges that brokers face in Smart Grid markets were comprehensively considered, and an adaptive and systematic model was constructed to surmount the challenges. The proposed broker model was tested and evaluated on the platform of Power TAC. The e-valuation results showed the advantages of the proposed broker against other excellent available brokers on both profit making and supply-demand balance. Through the experiments, two empirical laws have been discovered i.e. **Law 1:** profit margin shrinks in a competitive market environment and **Law 2:** the imbalance rate of supply-demand increases when a market environment is more competitive.

2. A new energy demand prediction method in Smart Grid market, called LF-LCB, was developed for accurate load forecasting of Smart Grid. LF-LCB utilized the proposed sCCRF to analyze customer behavior by using the learned weights which can reflect different energy consumption patterns of various customers. The results of experiments conducted from several perspectives verified the following two contributions: 1) Learning customer behavior to aggregate customers can improve the prediction precision and lead to a reasonable computation cost. 2) The proposed sCCRF is an efficient learning tool with feature selection capacity.

3. DeepRSD was proposed for renewable energy production prediction (a re-gression on sequential data). DeepRSD used stacked bi-directional RNNs to represent the sequential data. Four conditions were explored to ensure a plau-sible training. An alternative dropout was also proposed to effectively improve the generalization of DeepRSD. DeepRSD was applied to two real-world solar energy prediction problems and achieved state-of-the-art performances. According to the experimental results, DeepRSD showed two major advantages over other methods. 1) DeepRSD can simultaneously represent step features and temporal information. 2) DeepRSD has strong nonlinear presentation capacity to achieve a good performance without feature engineering. Therefore, this study concludes that DeepRSD can be an effective solution for renewable energy prediction and can be applied to regression problems on sequential data.

4. A new model for energy demand prediction was developed. Integrating the advantages of RNN and SGCRF, CoR was proposed for demand prediction. Two training methods were proposed for CoR. Experimental results showed that the end-to-end training was more efficient than the alternative training. CoRa was also proposed to improve CoR by introducing an attention mechanisms for RNN. The evaluations with both synthetic data and real-world data demonstrated that CoR and CoRa could significantly improve prediction precision over stacked Bi-RNNs and SGCRF. CoR and CoRa also outperformed other state-of-the-art methods in the real-world multi-step time-series prediction. With sufficient experiments and analysis, This study concludes that CoR and CoRa can be new effective models for demand prediction, and can also be applied to other multi-step time-series prediction problems

7.2 Future Work

Though the proposed methods in this thesis have achieved convincing results on solving several challenging issues in demand management of Smart Grid market, there is still room to improve these proposed methods.

1. **Balancing supply and demand in Smart Grid market**
   Broker agent is an effective solution to supply-demand balance in Smart Grid market. There are two possible directions to improve the proposed broker model, which are: improvement on demand side and improvement on management of retail market.

   For the improvement on demand side, prediction methods that make use of learned customer behavior can be introduced to improve demand prediction accuracy. However, the introduced method should be efficient enough for real-time decision-making.

   For the improvement on management of retail market, deep reinforcement learning can be introduced. Deep reinforcement learning [MKS+15] is under fast development in recent years. It is possible to introduce deep representations to replace human-designed function approximations. With improved representations, reinforcement learning can make efficient decisions in the retail market.

2. Demand prediction for Smart Grid
For demand prediction utilizing customer behaviors, it will be interesting to further consider customer migrations. However, customer migrations cause dynamic changes of customer groups, which will affect the demand prediction accuracy. To solve the challenge of customer migration, it is necessary to quickly assign the customer into groups. Similarity-based clustering methods may be helpful in customer assignments.

Learning customer behavior to aggregate customers can potentially facilitate research in related domains. It can offer a general methodology to assist better decision making towards various customers in a complex market environment. This is worth further explorations in other market domains. Evaluation results in this thesis also indicate that the proposed sCCRF is effective in feature selection and prediction. Thus, sCCRF can also be explored in other related research fields.

3. Prediction of renewable energy
As renewable energy fluctuates much more than power demand, it is challenging for time-series models to effectively capture correlations of greatly varied time steps. it will be interesting to study the dependencies of output in future work, so as to find the optimal number of time steps to model correlations.

Because of the strong fluctuations in the output, it is harmful to model the global dependencies and thus CoR may lose its advantage in renewable energy prediction. However, it worths to try tensor-to-tensor network [VSP$^+$17] with a proper size of attention window for renewable energy prediction.

Bibliography

[AEDS09] C. Alzate, M. Espinoza, M. De, and J. Suykens. Identifying customer profiles in power load time series using spectral clustering. In *Proceedings of International Conference on Artificial Neural Networks*, pages 315–324. Springer, 2009.

[AG07] G. Andrew and J. Gao. Scalable training of l_1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40. ACM, 2007.

[Age14] Advanced Research Project Agenc. Smart wire grid distributed power flow control. Technical report, 2014.

[AGS07] A. Ali, S. Ghaderi, and S. Sohrabkhani. Forecasting electrical consumption by integration of neural network, time series and anova. *Applied Mathematics and Computation*, 186(2):1753–1761, 2007.

[AHSB14] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.

[ANS08] M. Amin-Naseri and A. Soroush. Combined use of unsupervised and supervised learning for daily peak load forecasting. *Energy Conversion and Management*, 49(6):1302–1308, 2008.

[AS13] C. Alzate and M. Sinn. Improved electricity load forecasting via kernel spectral clustering of smart meters. In *Proceedings of the 13th International Conference on Data Mining*, pages 943–948. IEEE, 2013.

[BB12] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(2):281–305, 2012.

[BBR13] T. Baltrusaitis, N. Banda, and P. Robinson. Dimensional affect recognition using continuous conditional random fields. In *Proceedings of IEEE Conference on Automatic Face and Gesture Recognition*, pages 1–8. IEEE, 2013.

[BCB14] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[Ben09] Y. Bengio. Learning deep architectures for artificial intelligence. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[BI12] L. Berger and K. Iniewski. *Smart Grid - Applicacions, Communications and Security*. John Wiley and Sons, 2012.

[BP70] G. Box and D. Pierce. Distribution of residual auto-correlations in auto-regressive-integrated moving average time-series models. *Journal of the American statistical Association*, 65(332):1509–1526, 1970.

[CCVO98] W. Charytoniuk, M-S Chen, and P. Van Olinda. Nonparametric regression based short-term load forecasting. *IEEE Transactions on Power Systems*, 13(3):725–730, 1998.

[CG16] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[CHC95] M. Cho, J. Hwang, and C. Chen. Customer short term load forecasting by using arima transfer function model. In *International Conference on Energy Management and Power Delivery*, volume 1, pages 317–322. IEEE, 1995.

[CL96] TWS Chow and C. Leung. Neural network based short-term load forecasting using weather compensation. *IEEE Transactions on Power Systems*, 11(4):1736–1742, 1996.

[CLdS$^+$07] O. Carpinteiro, R. Leme, A. de Souza, C. Pinheiro, and E. Moreira. Long-term load forecasting via a hierarchical neural model with time integrators. *Electric Power Systems Research*, 77(3):371–378, 2007.

[CMB10] A. Conejo, J. Morales, and L. Baringo. Real-time demand response model. *IEEE Transactions on Smart Grid*, 1(3):236–242, 2010.

[CW05] P.l Crompton and Y. Wu. Energy consumption in china: past trends and future directions. *Energy economics*, 27(1):195–208, 2005.

[DBK$^+$97] H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In *Proceedings of Advances in Neural Information Processing Systems*, pages 155–161, 1997.

[DQH17] X. Dong, L. Qian, and L. Huang. Short-term load forecasting in smart grid: A combined cnn and k-means clustering approach. In *Proceedings of IEEE International Conference on Big Data and Smart Computing*, pages 119–125. IEEE, 2017.

[EHJT04] B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[EMG01] F. Egelioglu, A. Mohamad, and H. Guven. Economic variables and electricity consumption in northern cyprus. *Energy*, 26(4):355–362, 2001.

[FD16] J. Fiot and F. Dinuzzo. Electricity demand forecasting by multi-task learning. *IEEE Transactions on Smart Grid*, 2016.

[FN93] K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806, 1993.

[Fri01] J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[FZXC11] F.Yu, P. Zhang, W. Xiao, and P. Choudhury. Communication systems for grid integration of renewable energy resources. *IEEE Network*, 25(5):22–29, 2011.

[GB10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Artificial Intelligence and Statistics*, volume 9, pages 249–256, 2010.

[GG16] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1019–1027, 2016.

[GGVO16] J. Glass, M. Ghalwash, M. Vukicevic, and Z. Obradovic. Extending the modelling capacity of gaussian conditional random fields while learning faster. In *Proceedings of 30th Association for the Advancement of Artificial Intelligence*, pages 1596–1602, 2016.

[Gra13] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[GS05] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[GT01] R. Gao and L. Tsoukalas. Neural-wavelet methodology for load forecasting. *Journal of Intelligent and Robotic Systems*, 31(1-3):149–157, 2001.

[Guo15] H. Guo. Accelerated continuous conditional random fields for load forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 27(8):2023–2033, 2015.

[Guo16] H. Guo. Accelerated continuous conditional random fields for load forecasting. In *Proceedings of International Conference on Data Engineering*, pages 1492–1493. IEEE, 2016.

[HBA$^+$14] L. Hernandez, C. Baladron, J. Aguiar., R. Carro, A. Sanchez-Esguevillas, J. Lloret, and J. Massana. a survey on electric power demand forecasting: Future trends in smart grids, microgrids and smart buildings. *IEEE Communications Surveys and Tutorials*, 16(3):1460–1495, 2014.

[HCL$^+$08] C. Hsieh, K. Chang, C. Lin, S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th International Conference on Machine Learning*, pages 408–415. ACM, 2008.

[HM94] T. Haida and S. Muto. Regression based peak load forecasting using a transformation technique. *IEEE Transactions on Power Systems*, 9(4):1788–1794, 1994.

[HM12] D. Henrion and J. Malick. Projection methods in conic optimization. In *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 565–600. Springer, 2012.

[HP16] J. Huang and M. Perry. A semi-empirical approach using gradient boosting and k-nearest neighbors regression for gefcom2014 probabilistic solar power forecasting. *International Journal of Forecasting*, 32(3):1081–1086, 2016.

[HPF$^+$16] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. Hyndman. Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond. *International Journal of Forecasting*, 32(3):896–913, 2016.

[HS97] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[IS15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32th International Conference on Machine Learning*, pages 448–456, 2015.

[IXJ11a] M. Ilic, L. Xie, and J. Joo. Efficient coordination of wind power and price-responsive demand part i: Theoretical foundations. *IEEE Transactions on Power Systems*, 26(4):1875–1884, 2011.

[IXJ11b] M. Ilic, L. Xie, and J. Joo. Efficient coordination of wind power and price-responsive demand part ii: Case studies. *IEEE Transactions on Power Systems*, 26(4):1885–1893, 2011.

[JMI13] A. Jamshid and A. Mohammad-Iman. Demand response in smart electricity grids equipped with renewable energy sources: a review. *Renewable and Sustainable Energy Reviews*, 18:64–72, 2013.

[JPM10] T. Jonsson, P. Pinson, and H. Madsen. On the market impact of wind energy forecasts. *Energy Economics*, 32(2):313–320, 2010.

[Kal01] S. Kalogirou. Artificial neural networks in renewable energy systems applications: a review. *Renewable and sustainable energy reviews*, 5(4):373–401, 2001.

[KB14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KCRW14] W. Ketter, J. Collins, P. Reddy, and M. Weerdt. The 2014 power trading agent competition. *Report Series Reference No. ERS-2014-004-LIS*, 2014.

[KHCW13] R. Kuate, M. He, M. Chli, and H. Wang. An intelligent broker agent for energy trading: an mdp approach. In *Proceedings of the 23nd International Joint Conference on Artificial Intelligence*, pages 234–240. AAAI Press, 2013.

[KSB11] A. Khodaei, M. Shahidehpour, and S. Bahramirad. SCUC with hourly demand response considering intertemporal load characteristics. *IEEE Transactions on Smart Grid*, 2(3):564–571, 2011.

[KWSG06] N. Kandil, R. Wamkeue, M. Saad, and S. Georges. An efficient approach for short term load forecasting using artificial neural networks. *International Journal of Electrical Power and Energy Systems*, 28(8):525–530, 2006.

[Laf01] J. Lafferty. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, 2001.

[LCY10] T. Lavergne, O. Cappé, and F. Yvon. Practical very large scale CRF-s. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 504–513. ACL, 2010.

[LCY13] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[LGA17] T. Lin, T. Guo, and K. Aberer. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2627–2633, 2017.

[LHL14] B. Liefers, J. Hoogland, and P. La. A successful broker agent for power TAC. In *Proceedings of Agent-Mediated Electronic Commerce*, pages 99–113. Springer, 2014.

[LN89] D. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[LTL08] J. Lam, H. Tang, and D. Li. Seasonal variations in residential and commercial sector electricity consumption in Hong Kong. *Energy*, 33(3):513–523, 2008.

[LWV93] C. Lu, H. Wu, and S. Vemuri. Neural network based short term load forecasting. *IEEE Transactions on Power Systems*, 8(1):336–342, 1993.

[MGB$^+$15] A. McGovern, D. Gagne, J. Basara, T. Hamill, and D. Margolin. Solar energy prediction: an international contest to initiate interdisciplinary research on compelling meteorological problems. *Bulletin of the American Meteorological Society*, 96(8):1388–1395, 2015.

[MHRH04] M. Mohandes, T. Halawani, S. Rehman, and A. Hussain. Support vector machines for wind speed prediction. *Renewable Energy*, 29(6):939–947, 2004.

[MK16] C. McCarter and S. Kim. Large-scale optimization algorithms for s-parse conditional gaussian graphical models. In *Proceedings of Artificial Intelligence and Statistics*, pages 528–537, 2016.

[MKS$^+$15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, and G. Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[MRWJ$^+$10] A. Mohsenian-Rad, V. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *IEEE Transactions on Smart Grid*, 1(3):320–331, 2010.

[MSF06] P. Mandal, T. Senjyu, and T. Funabashi. Neural networks approach to forecast several hour ahead electricity prices and loads in deregulated market. *Energy Conversion and Management*, 47(15):2128–2142, 2006.

[MZR12] A. Motamedi, H. Zareipour, and W. Rosehart. Electricity price and demand forecasting in smart grids. *IEEE Transactions on Smart Grid*, 3(2):664–674, 2012.

[Nes83] Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[Ng04] A. Ng. Feature selection, l_1 vs. l_2 regularization, and rotational invariance. In *Proceedings of the 21st International Conference on Machine Learning*, pages 78–85. ACM, 2004.

[NH10] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814. ACM, 2010.

[oE12] U.S. Department of Energy. Smart grid. Technical report, Department of Energy, 2012.

[OO15] T. Osogami and M. Otsuka. Learning dynamic boltzmann machines with spike-timing dependent plasticity. *arXiv preprint arXiv:1509.08634*, 2015.

[PD11] P. Palensky and D. Dietrich. Demand side management: Demand response, intelligent energy systems, and smart loads. *IEEE Transactions on Industrial Informatics*, 7(3):381–388, 2011.

[PGCB13] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[PH90] A. Papalexopoulos and T. Hesterberg. A regression-based approach to short-term system load forecasting. *IEEE Transactions on Power Systems*, 5(4):1535–1547, 1990.

[Pin87] F. Pineda. Generalization of back-propagation to recurrent neural networks. *Physical review letters*, 59(19):2229, 1987.

[PKSTC13] M. Peters, W. Ketter, M. Saar-Tsechansky, and J. Collins. A reinforcement learning approach to autonomous decision-making in smart electricity markets. *Machine learning*, 92(1):5–39, 2013.

[PMB13] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of International Conference on Machine Learning*, pages 1310–1318. ACM, 2013.

[PSM09] M. Pedrasa, T. Spooner, and I. MacGill. Scheduling of demand side resources using binary particle swarm optimization. *IEEE Transactions on Power Systems*, 24(3):1173–1181, 2009.

[pt] http://www.powertac.org/wiki/index.php/form:broker.

[Put14] M. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley and Sons, 2014.

[QLZ$^+$09] T. Qin, T. Liu, X. Zhang, D. Wang, and H. Li. Global ranking using continuous conditional random fields. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1281–1288, 2009.

[RFFA12] M. Rastegar, M. Fotuhi-Firuzabad, and F. Aminifar. Load commitment in a smart home. *Applied Energy*, 96:45–54, 2012.

[RHW88] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[RN94] G. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report, Cambridge University Engineering Department, 1994.

[RVO10] V. Radosavljevic, S. Vucetic, and Z. Obradovic. Continuous conditional random fields for regression in remote sensing. In *Proceedings of European Conference on Artificial Intelligence*, pages 809–814, 2010.

[SB98] R. Sutton and A. Barto. *Reinforcement learning: an introduction*, volume 1. MIT press Cambridge, 1998.

[SBN01] S. Saab, E. Badr, and G. Nasr. Univariate modeling and forecasting of energy consumption: the case of electricity in lebanon. *Energy*, 26(1):1–14, 2001.

[Sch05] M. Schmidt. minfunc: unconstrained differentiable multivariate optimization in matlab. http://www.cs.ubc.ca/ schmidtm/Software/minFunc.html, 2005.

[SGH09] K. Sumer, O. Goktas, and A. Hepsag. The application of seasonal latent variable in forecasting electricity demand as an alternative method. *Energy policy*, 37(4):1317–1322, 2009.

[SHK$^+$14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[Sia14] P. Siano. Demand response and smart grids-a survey. *Renewable and Sustainable Energy Reviews*, 30:461–478, 2014.

[SLW$^+$85] R. Slutz, S. Luberker, S. Woodruff, R. Jenne, and P. Stuerer. Comprehensive ocean-atmosphere data set(coads), release 1. 1985.

[SMDH13] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1139–1147. ACM, 2013.

[SMG13] A. Saxe, J. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[Sri08] D. Srinivasan. Energy demand prediction using gmdh networks. *Neurocomputing*, 72(1):625–629, 2008.

[SS12] L. Suganthi and A. Samuel. Energy models for demand forecastin–a review. *Renewable and Sustainable Energy Reviews*, 16(2):1223–1240, 2012.

[TB02] G. Tesauro and J. Bredin. Strategic sequential bidding in auctions using dynamic programming. In *Proceedings of the 1st International Conference on Autonomous Agents and Multiagent Systems*, pages 591–598. ACM, 2002.

[TB03] J. Taylor and R. Buizza. Using weather ensemble predictions in electricity demand forecasting. *International Journal of Forecasting*, 19(1):57–70, 2003.

[TEL$^+$92] J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J. Farmer. Testing for nonlinearity in time series: the method of surrogate data. *Physica D: Nonlinear Phenomena*, 58(1-4):77–94, 1992.

[The16] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[THMT00] B. Tang, W. Hsieh, A. Monahan, and F. Tangang. Skill comparisons between neural networks and canonical correlation analysis in predicting the equatorial pacific sea surface temperatures. *Journal of Climate*, 13(1):287–293, 2000.

[Tib96] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[TTMH98] F. Tangang, B. Tang, A. Monahan, and W. Hsieh. Forecasting enso events: A neural network–extended eof approach. *Journal of Climate*, 11(1):29–41, 1998.

[Ünl08] A. Ünler. Improvement of energy demand forecasts using swarm intelligence: The case of turkey with projections to 2025o. *Energy Policy*, 36(6):1937–1944, 2008.

[US14] D. Urieli and P. Stone. Tactex'13: a champion adaptive power trading agent. In *Proceedings of International Conference on Autonomous Agents and Multi-agent Systems*, pages 1447–1448. ACM, 2014.

[VB98] J. Vermaak and E. Botha. Recurrent neural networks for short-term load forecasting. *IEEE Transactions on Power Systems*, 13(1):126–132, 1998.

[VCJ08] P. Vytelingum, D. Cliff, and N. Jennings. Strategic bidding in continuous double auctions. *Artificial Intelligence*, 172(14):1700–1729, 2008.

[VSP$^+$17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017.

[VVL07] D. Vail, M. M. Veloso, and J. Lafferty. Conditional random fields for activity recognition. In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, pages 235–242. ACM, 2007.

[WK13] M. Wytock and J Z. Kolter. Sparse gaussian conditional random fields: Algorithms, theory, and application to energy forecasting. In *Proceed-*

ings of International Conference on Machine Learning, pages 1265–1273. ACM, 2013.

[WZR16] X. Wang, M. Zhang, and F. Ren. Load forecasting in a smart grid through customer behaviour learning using l_1-regularized continuous conditional random fields. In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, pages 817–826. ACM, 2016.

[WZRI15] X. Wang, M. Zhang, F. Ren, and T. Ito. Gongbroker: A broker model for power trading in smart grid markets. In *Proceedings of International Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 21–24. IEEE, 2015.

[WZZS09] J. Wang, W. Zhu, W. Zhang, and D. Sun. A trend fixed on firstly and seasonal adjustment model combined with the ε-svr for short-term forecasting of electricity demand. *Energy Policy*, 37(11):4901–4909, 2009.

[XKDL09] X. Xin, I. King, H. Deng, and M. Lyu. A social recommendation framework based on multi-scale continuous conditional random fields. In *Proceedings of International Conference on Information and Knowledge Management*, pages 1247–1256. ACM, 2009.

[XWCL15] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[XWM10] C. Xia, J. Wang, and K. McMenemy. Short, medium and long term load forecasting model and virtual load forecaster based on radial basis function neural networks. *International Journal of Electrical Power & Energy Systems*, 32(7):743–750, 2010.

[XYZS09] Z. Xiao, S. Ye, B. Zhong, and C. Sun. Bp neural network with rough set for short term load forecasting. *Expert Systems with Applications*, 36(1):273–279, 2009.

[YVGS10] J. Yu, S. Vishwanathan, S. Günter, and N. Schraudolph. A quasi-newton approach to nonsmooth convex optimization problems in machine learning. *The Journal of Machine Learning Research*, 11:1145–1200, 2010.

[Zei12] M. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[ZSV14] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.