2019

Neural Network Methods for Nonparametric Probabilistic Forecasting

Konstantinos Miltiadis Hatalis
Lehigh University

NEURAL NETWORK METHODS FOR NONPARAMETRIC PROBABILISTIC FORECASTING

by

KONSTANTINOS MILTIADIS HATALIS

Presented to the Graduate and Research Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy
in
Electrical Engineering

Lehigh University
May, 2019

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Date

Accepted Date

Dissertation Advisor

Committee Members:

Prof. Shalinee Kishore
(Committee Chair)

Prof. Parv Venkitasubramaniam

Prof. Wenxin Liu

Prof. Alberto J. Lamadrid

Acknowledgments

The completion of this Ph.D. is the culmination of immense hard work. Not only from myself, but also from everyone around me in my life in pushing and inspiring me to complete it. Although I just mention a few individuals here, I am deeply appreciative of everyone who has helped me along the way.

Firstly, I would like to express my enormous gratitude to my advisor Prof. Shalinee Kishore for the continuous support of my Ph.D. research, for her patience, motivation, and immense knowledge. I am grateful to her for giving me the chance to join her lab group and for allowing me to pursue my research passion in machine learning. I could not have imagined having a better advisor and mentor for my Ph.D. study. Along with my advisor, I would like to thank the rest of my dissertation committee: Prof. Parv Venkitasubramaniam, Prof. Wenxin Liu, and Prof. Alberto J. Lamadrid, for their insightful comments, reviews of my papers, and remarks. Most importantly they all pushed me with hard questions which motivated me to widen my research from various perspectives. I also want to give special thanks to all the anonymous reviewers, editors, and publishers of all my papers.

My sincere gratitude also goes to Prof. Rick S. Blum, Prof. Larry Snyder, Prof. Arindam Banerjee, and Prof. Katya Scheinberg, who at different times of my Ph.D. provided me with valuable feedback on my research. Thanks also go to the staff of Lehigh: Ruby Scott, Diane Hubinsky, Brie Lisk, and David Morrisette. I thank my fellow labmates, in Prof. Kishore's group, Dr. Parth Pradhan and Dr. Kwami Senam Sedzro. In all the years we spent together, in our shared office space in PA 401, I very much enjoyed our stimulating discussions and all the fun we had together. Kwami, your life stories continue to inspire me, and Parth your comradeship working

together side by side was remarkable. I want to give big thanks to my fellow doctoral student, and one of my closest friends, Dr. Dustin Dannenhauer. I'll never forget our amazing discussions and ideas on AI, our epic Go and Starcraft game nights, and all the veggie burgers we ate together.

I give the biggest thanks to my family who all have given me unconditional support. To my sisters Maria and Amalia, they have been my life long friends. To my mother, whom I am so gracious for everything you have done for me. And to my fiancée, future wife, best friend, and soulmate Radhika. Your understanding, encouragement, and love have kept me going through all the late nights and long hours. Finally, I would like to dedicate this dissertation to my father, Prof. Miltiadis Hatalis. My father is the sole inspiration to go into graduate school and pursue a Ph.D. in engineering. His guidance throughout high school, college, and graduate school have elevated me to success.

Contents

List of Tables

List of Figures

Abstract

The presented research investigates the use of neural networks for probabilistic forecasting in selected application areas. The topic of neural networks, also known as deep learning, has exploded as a research field, showing incredible results in image analysis and classification. But the application of neural networks to time series or regression-based forecasting is lesser known. Forecasting is the backbone of many industries and academic research areas. From predicting weather patterns to the stock market, to healthcare and energy, forecasting is vital to many operations of todays modern society. In the of evolution of the study of forecasting, the field of probabilistic forecasting has recently emerged. Unlike a deterministic forecast which only provides a single expected value, a probabilistic forecast provides information on the uncertainty of a prediction. We investigate how neural networks, which can automatically extract features via hidden layers, can be used to generate reliable and sharp probabilistic forecasts in the form of quantiles, prediction intervals, and full predictive densities. More specifically, we look at nonparametric probabilistic forecasting where we do not assume the underlying distribution of the forecasts. Our work seeks to evaluate these new methods in the application domains of renewable energies. In chapter 1 we provide a brief overview of probabilistic forecasting theory.

In our first study (chapter 2) of this thesis, we overview the basic theory of how neural networks can be used for deterministic forecasting. This presents as a foundation for our later work for probabilistic prediction. In this study, we propose the development of an adaptive particle swarm optimization (APSO) learning algorithm to train a non-linear autoregressive (NAR) neural network, which we call PSONAR, for short term time series prediction of ocean wave elevations. We also introduce

a new stochastic inertial weight to the APSO learning algorithm. Our work is motivated by the expected need for such predictions by wave energy farms. As such, we simulated noisy ocean wave heights for training and testing. We utilized our PSONAR to get results for 5, 10, 30, and 60-second multi-step predictions. Results show APSO can outperform backpropagation in training a NAR neural network.

In our second study (chapter 3) we study cyber-enabled demand-side management systems. DSM is a vital tool that can be used to ensure power system reliability and stability. In future smart grids, certain portions of a customers load usage could be under automatic control with a cyber-enabled DSM program which selectively schedules loads as a function of electricity prices to improve power balance and grid stability. In such a future, security of DSM cyberinfrastructure will be critical as advanced metering infrastructure, and communication systems are susceptible to hacking, cyber attacks. Such attacks, in the form of data injection, can manipulate customer load profiles and cause metering chaos and energy losses in the grid. These attacks are also exacerbated by the feedback mechanism between load management on the consumer side and dynamic price schemes by independent system operators. This work provides a novel methodology for modeling and simulating the nonlinear relationship between load management and real-time pricing. We then investigate the behavior of such a feedback loop under intentional cyber attacks using our feedback model. We simulate and examine load-price data under different levels of DSM participation with three types of additive attacks: ramp, sudden, and point attacks. We applied change point and supervised learning methods for the detection of DSM attacks.

Results conclude that higher amounts of DSM participation can exacerbate attacks but also lead to better detection of such attacks, point attacks are the hardest to detect, and supervised learning methods produce results on par or better than sequential detectors. This chapter serves as an example of how linear methods can often yield better results then nonlinear such as neural networks. The need for deep learning or advanced probabilistic forecasting is not warranted in this DSM domain when generation is constant. However, we hypothesize that when renewable energy generation is introduced into this problem, the detection of attacks can become much

more difficult. Due to the chaotic nature of renewable energies, there is a need to quantify the uncertainty in forecasting their power generation. As motivation and a prerequisite for future work to study DSM systems under renewable generation, in the next chapters we propose several new and advanced forecasting methods.

In our third study (chapter 4), we propose our first method to produce full predictive densities by examining how support vector machines (SVMs) can be used for quantile estimation. SVMs are one of the most efficient machine learning algorithms, which is mostly used for pattern recognition since its introduction in the 1990s. Uncertainty analysis in the form of probabilistic forecasting can provide significant improvements in decision-making processes in the smart power grid for better integrating renewable energies, particularly wind. This chapter analyzes the effectiveness of an approach for nonparametric probabilistic forecasting of wind power that combines support vector machines and nonlinear quantile regression with non-crossing constraints. A numerical case study is conducted using publicly available wind data from the Global Energy Forecasting Competition 2014. Multiple quantiles are estimated to form 20%, 40%, 60% and 80% prediction intervals which are evaluated using the pinball loss function and reliability measures. Three benchmark models are used for comparison where results demonstrate the proposed approach leads to significantly better performance while preventing the problem of overlapping quantile estimates.

In our fourth study (chapter 5) we analyze the effectiveness of a novel approach for nonparametric probabilistic forecasting of wind power that combines a smooth approximation of a pinball loss function with a deep neural network architecture and a smooth penalty scheme to prevent the quantile crossover problem. We call our model the smooth pinball neural network (SPNN). A numerical case study is conducted using publicly available wind data from the Global Energy Forecasting Competition 2014. Multiple quantiles are estimated to form 10%, to 90% prediction intervals which are evaluated using a quantile score and reliability measures. Benchmark models such as the persistence and climatology distributions, multiple quantile regression, and support vector quantile regression are used for comparison where results demonstrate the proposed approach leads to improved performance

while preventing the problem of overlapping quantile estimates.

In our fifth study (chapter 6) we radically extend SPNN to forecast time series. Point forecasting of univariate time series is a challenging problem with extensive work having been conducted. However, nonparametric probabilistic forecasting of time series, such as in the form of quantiles or prediction intervals is an even more challenging problem. To expand the possible forecasting paradigms we devise and explore an extrapolation-based approach that has not been applied before for probabilistic forecasting. We present a novel quantile Fourier neural network is for nonparametric probabilistic forecasting of univariate time series. Multi-step predictions are provided in the form of composite quantiles using time as the only input to the model. This effectively is a form of extrapolation based nonlinear quantile regression applied for forecasting. Experiments are conducted on eight real-world datasets that demonstrate a variety of periodic and aperiodic patterns. Nine simple and advanced methods are used as benchmarks including quantile regression neural network, support vector quantile regression, SARIMA, and exponential smoothing. The obtained empirical results validate the effectiveness of the proposed method of providing high quality and accurate probabilistic predictions. We then provide conclusions in our final chapter (chapter 7) as well as specific direction for future work.

Chapter 1

Introduction

In the last decade there have been two rising fields that have shown a lot of promise, unique results, and immense academic and industry attention. These are deep learning [1] and probabilistic forecasting [2]. Part of the success of deep learning, also known as the study of neural networks, is due to its ability to conduct automatic feature extraction at different levels of abstraction. Such feature extraction promotes by passes the need to provide manual feature engineering that is common among machine learning pipelines. Moreover, neural networks allow the representation of the nonlinearities in data, often associated with complex and real-world problems. Deep learning models have been used to achieve state-of-the art results in the field of computer vision [3] and have also been applied to the problem of forecasting [4]. However, the study of neural networks, and related methods, is limited when it comes to probabilistic forecasting. Thus, motivation for the presented research is to explore how neural networks could contribute to providing better probabilistic forecasts. Further motivation is provided by the new class of big data in regression and time series, which are vital in areas such as in the smart grid and renewable energy which are both used as application domains for our proposed forecasting frameworks.

Neural Networks

The study of neural networks has grown tremendously in the past decade. Neural networks are a set of algorithms, modeled very loosely after the human brain, that are designed to recognize patterns. They take in input data, which may come in different forms such as text, images, or time series, and they output either a class label for classification or a numeric value for regression. The layers of a network are made of computational nodes, again loosely modeled on a neuron in the human brain. These nodes take in data and perform some computation utilizing coefficients or weights which assign significance to inputs with regard to the task the algorithm is trying to learn. The weights can increase or decrease the significance of each data point. The input-weight products are fed into the node's activation function, usually some nonlinear function, which then determines to what extent that signal should progress further through the network. Through utilizing a finite amount of nodes a neural network can approximate any continuous function, this is also known as the universal approximation theorem.

There are many types of neural networks including feedfward networks, recurrent networks, convolutional networks, deep belief networks and autoencoders. Due to the popularity of neural networks in both academic research and industrial application, there is already a large body of work reviewing the algorithms, mathematics, methods, approaches and issues. We therefore refer to the following references on the background of deep learning [1,5–7], and in the next section we review in depth the field of probabilistic forecasting which has less literature.

Probabilistic Forecasting

Over the last several years there has been a large body of work conducted in nonparametric probabilistic forecasting. Recently the Global Energy Forecasting Competition in 2014 [8] and in 2017 [9] are further proof of the rising interest in probabilistic forecasting. Probabilistic forecasting is especially of large interest

to applications in renewable power such as wind. As such, the domain of wind forecasting is also part of the main focus of this dissertation. Wind forecasting models are either meteorological ensembles that are obtained by a weather model [10] or are statistical methods [11]. Under the statistical approach, we can estimate full predictive distributions in the form of quantiles or prediction intervals (PIs).

Some recent forecasting methods include extreme learning machines [13] where a direct quantile regression approach was presented to efficiently generate nonparametric probabilistic forecasting of wind power generation combining extreme learning machine and quantile regression. Hybrid intelligent methods have also been explored in [14] by feeding deterministic wind power forecasts made by a combination of wavelet transform and fuzzy ARTMAP network, optimized by using firefly optimization algorithm, in quantile regression. A PI estimation scheme is shown in [12] which uses a radial basis function neural network.Another approach to forecast the density of wind power is to take an ensemble of point forecasts and calculate the mean and variance of the combined forecasts. This has been studied in [15] where a wavelet transform and a convolutional neural network are used for ensemble point forecasting. Another ensemble approach can be seen in [16] where time series models such as ARMA and GARCH are combined to form density forecasts. One of the most prevalent approaches to probabilistic forecasting of wind power is to apply quantile regression (QR) which can be used to estimate different wind power quantiles [17].

Another alternative to nonparametric probabilistic wind forecasting is the application of the Lower Upper Bound Estimation (LUBE) method [18]. The LUBE method constructs a neural network with two outputs for estimating the prediction interval bounds. The coverage width-based criterion is used as the loss function for estimating PIs, and simulated annealing or particle swarm optimization [19] can be used to minimize that loss function. A complete review on probabilistic forecasting of wind power can be found in [17]. Other reviews on probabilistic forecasting methods can be found in [20] for solar power, [8] for load forecasting, and [21] for electricity price forecasting.

Next we highlight the underlying mathematics in nonparametric probabilistic

forecasting, overview linear quantile regression, and summarize the main evaluation metrics for density forecasts. Given a random variable Y_t such as wind power at time t, its probability density function is defined as f_t and its the cumulative distribution function as F_t. If F_t is strictly increasing, the quantile $q_t^{(\tau)}$ of the random variable Y_t with nominal proportion τ is uniquely defined on the value x such that $P(Y_t < x) = \tau$. It can also be defined as the inverse of the distribution function $q_t^{(\tau)} = F_t^{-1}(\tau)$. A quantile forecast $\hat{q}_{t+z}^{(\tau)}$ is an estimate of the true quantile $q_{t+z}^{(\tau)}$ for the lead time $t+z$, given a predictor values (such as numerical wind speed forecasts). Prediction intervals are another type of probabilistic forecast and give a range of possible values within which an observed value is expected to lie with a certain probability $\beta \in [0, 1]$. A prediction interval $\hat{I}_{t+z}^{(\beta)}$ is defined by its lower and upper bounds, which are the quantile forecasts $\hat{I}_{t+z}^{(\beta)} = \left[\hat{q}_{t+z}^{(\tau_l)}, \hat{q}_{t+z}^{(\tau_u)}\right] = \left[l_t^{(\beta)}, u_t^{(\beta)}\right]$ whose nominal proportions τ_l and τ_u are such that $\tau_u - \tau_l = 1 - \beta$.

In probabilistic forecasting, we are trying to predict one of two classes of density functions, either parametric or nonparametric. When the future density function is assumed to take a certain distribution, such as the Normal distribution, then this is called parametric probabilistic forecasting. For a nonlinear and bounded process such as wind generation, probability distributions of future wind power, for instance, may be skewed and heavy-tailed distributed [22]. Else if no assumption is made about the shape of the distribution, a nonparametric probabilistic forecast $\hat{f}_{t+z}$ [23] can be made of the density function by gathering a set of M quantiles forecasts such that $\hat{f}_{t+z} = \left\{\hat{q}_{t+z}^{(\tau_m)}, m = 1, ..., M | 0 \leq \tau_1 < ... < \tau_M \leq 1\right\}$ with chosen nominal proportions spread on the unit interval. As mentioned before, renewable power forecasting can be quite stochastic, thus making nonparametric forecasting more ideal then fitting a parametric density [17].

1.0.1 Quantile Regression

Quantile regression is a popular approach for nonparametric probabilistic forecasting. Koenker and Bassett [24] introduce it for estimating conditional quantiles and is closely related to models for the conditional median [25]. Minimizing the mean

absolute function leads to an estimate of the conditional median of a prediction. By applying asymmetric weights to errors through a tilted transformation of the absolute value function, we can compute the conditional quantiles of a predictive distribution. The selected transformation function is the pinball loss function as defined by

$$\rho_\tau(u) = \begin{cases} \tau u & \text{if } u \geq 0 \\ (\tau - 1)u & \text{if } u < 0 \end{cases}, \tag{1.1}$$

where $0 < \tau < 1$ is the tilting parameter. To better understand the pinball loss, we look at an example for estimating a single quantile. If an estimate falls above a reported quantile, such as the 0.05-quantile, the loss is its distance from the estimate multiplied by its probability of 0.05. Otherwise, the loss is its distance from the realization multiplied by one minus its probability (0.95 in the case of the 0.05-quantile). The pinball loss function penalizes low-probability quantiles more for overestimation than for underestimation and vice versa in the case of high-probability quantiles. Given a vector of predictors X_t where $t = 1, ..., N$, a vector of weights W and intercept b coefficient in a linear regression fashion, the conditional τ quantile $\hat{q}_\tau$ is given by $\hat{q}_t^{(\tau)} = W^\top X_t + b$. To determine estimates for the weights and intercept we solve the following minimization problem

$$\min_{W,b} \frac{1}{N} \sum_{t=1}^{N} \rho_\tau(y_t - \hat{q}_t^{(\tau)}), \tag{1.2}$$

where y_t is the observed value of the predictand. The formulation above in Eq. (1.2) can be minimized by a linear program.

There are many variations of QR which are traditionally solved using linear programming algorithms. In [26] local QR is applied to estimate different quantiles, while in [27] a spline-based QR is used to estimate quantiles of wind power. In [28] quantile loss gradient boosted machines are used to estimate many quantiles and in [29] multiple quantile regression is used to predict a full distribution with optimization achieved by using the alternating direction method of multipliers. Quantile regression forests [30] are another approach in forecasting which are an extension of regression forests based on classification and regression trees.

Due to their flexibility in modeling elaborate nonlinear data sets, artificial neural networks are another dominant class of machine learning algorithms that can be used to enhance QR. Taylor [31] is the first to propose a quantile regression neural network (QRNN) method, combining the advantages of both QR and a neural network. This method can reveal the conditional distribution of the response variable and can also model the nonlinearity of different systems. The author applies this method to estimate the conditional distribution of multi-period returns in financial systems, which avoids the need to specify the explanatory variables explicitly. However, the paper does not address how the network was optimized. The same QRNN was later used by [32] for credit portfolio data analysis where results showed that QRNN is more robust in fitting outliers compared to both local linear regression and spline regression. In [33] an autoregressive version of QRNN is used for applications to evaluating value at risk, and [34] implements the QRNN model in R as a statistical package.

1.0.2 Evaluation Metrics

In probabilistic forecasting it is essential to evaluate the quantile estimates and if desired also evaluate derived predictive intervals. Therefore, we reviews several important evaluation metrics here. To evaluate quantile estimates, one can use the pinball function directly as an assessment called the quantile score (QS). We choose QS as our main evaluation measure for most of our studies for the following reasons. When averaged across many quantiles it can evaluate full predictive densities; it is found to be a proper scoring rule [35]; it is related to the continuous rank probability score; and it is also the main evaluation criteria in the 2014 Global Energy Forecasting Competition (GEFCOM 2014), the source of our testing data. QS calculated overall N test observations and M quantiles is defined as

$$QS = \sum_{t=1}^{N} \sum_{m=1}^{M} \rho_{\tau_m}(y_t - \hat{q}_t^{(\tau_m)})$$

where y_t is an observation used to forecast evaluation such future wind power observations. To evaluate full predictive densities, QS is averaged across all target

quantiles for all look ahead time steps using equal weights. A lower QS indicates a better forecast.

With the QS calculated we can then also see what the relative performance of a proposed method is with respect to some benchmark method. We can assess relative performance between methods using the quantile verification skill score (QVSS) [36]

$$QVSS = 1 - \frac{QS^{for}}{QS^{ref}}$$

where QS^{for} is QS for the forecast method of interest, and QS^{ref} is the QS value for the reference forecast of a benchmark method, which we will assume to be linear quantile regression. If QVSS is positive then forecast of interest performs better than the reference forecast, and a QVSS = 1 means a perfect forecast. Negative QVSS values indicate that forecast of interest performs worse than the reference forecast.

In some applications, it may be needed to have wind forecasts in the form of prediction intervals (PIs) and as such, we look at two secondary evaluation measures: reliability and sharpness. Reliability is a measure which states that over an evaluation set the observed and nominal probabilities should be as close as possible, and the empirical coverage should ideally equal the preassigned probability. Sharpness is a measure of the width of prediction intervals, defined as the difference between the upper $u_t^{\beta_i}$ and lower $l_t^{\beta_i}$ interval values. For interval reliability we use the average coverage error (ACE) metric [17] and for measuring interval sharpness we use the interval score (IS) which can also be used to evaluate the overall skill of PIs [37]. For measuring reliability, PIs show where future wind power observations are expected to lie, with an assigned probability termed as the PI nominal confidence (PINC) $100(1-\beta_i)\%$. Here $i = 1...M/2$ indicates a specific coverage level. The coverage probability of estimated PIs is expected to eventually reach a nominal level of confidence over the test data. A measure of reliability which shows target coverage of the PIs is the PI coverage probability (PICP), which is defined by

$$PICP_i = \frac{1}{N}\sum_{t=1}^{N} \mathbb{1}\{y_t \in I_t^{\beta_i}(x_t)\}.$$

For reliable PIs, the examined PICP should be close to its corresponding PINC. A related and easier to visualize assessment index is the average coverage error (ACE), which is defined by

$$ACE = \sum_{i=1}^{M/2} |PICP_i - 100(1 - \beta_i)|.$$

This assumes calculation across all test data and coverage levels. To ensure PIs have high reliability, the ACE should be as close to zero as possible. A high reliability can be easily achieved by increasing or decreasing the distance between lower and upper interval bounds. Thus, the width of a PI can also influence its quality. For measuring the effective width of PIs we use the sharpness score proposed by [23] which measures how wide PIs are by focusing on the mean size of the intervals only. We define $\hat{q}_t^u - \hat{q}_t^l$ as the size of the central interval forecast with nominal coverage rate $(1 - \beta)$. For lead times $t = 1...N_{test}$, a measure of sharpness for PIs is then given by the mean size of the intervals

$$Sharpness = \frac{1}{N_{test}} \sum_{t=1}^{N_{test}} (\hat{q}_t^u - \hat{q}_t^l).$$

A lower sharpness score is considered more ideal, but too small and the PIs would not cover enough of the observed data. Thus sharpness is typically a measure to be considered along with reliability and a skill score. QS is a score that measures the skill of individual quantiles; to measure the skill of individual PIs we apply the interval score (IS) [37]. The IS - when evaluated with all test data and coverage levels - is defined by

$$\begin{aligned} IS = \frac{2}{NM} \sum_{t=1}^{N} \sum_{i=1}^{M/2} (u_t^{\beta_i} - l_t^{\beta_i}) + \\ \frac{2}{\beta_i}(l_t^{\beta_i} - y_t)\mathbb{1}\{y_t < l_t^{\beta_i}\} + \frac{2}{\beta_i}(y_t - u_t^{\beta_i})\mathbb{1}\{y_t > u_t^{\beta_i}\} \end{aligned}.$$

The prediction model is rewarded for narrow PIs and is penalized if the observation misses the interval. The size of the penalty depends on β_i. Including all aspects of PI evaluation, the IS can be used to compare the overall skill and sharpness of

interval forecasts. However, IS cannot identify the contributions of reliability and sharpness to the overall skill. Thus, ACE and sharpness are both used for evaluation of PIs along with QS for evaluation of quantile estimation.

Outline of the Dissertation

The rest of this dissertation is organized as follows. Chapter 2 showcases our work in a recurrent neural network trained by particle swarm optimization for multi-step deterministic forecasting. This work highlights the background theory of deep networks for prediction and provides a base for building probabilistic forecasting neural networks in later chapters. In chapter 3 we present our work on modeling and detecting cyber demand side management attacks. This problem domain shows how nonlinear methods are not always better than linear ones. However, we propose that the incorporation of renewable generation such as wind, solar, and wave energy can complicate the attack problem. Thus, we are motivated to provide solutions that can forecast renewables and other nonstationary time series. In chapter 4 we propose a shallow neural network called constrained support vector quantile regression method for providing probabilistic forecasts of wind power. Next, in chapter 5, we expand to deeper neural networks trained by stochastic gradient descent and introduce a novel loss function and architecture for composite quantile estimation. In chapter 6, we extend our neural network models from ideas in signal processing and propose a novel multiple quantile Fourier neural network that applies to probabilistic forecasting of nonstationary univariate time series data. We evaluate this model on multiple domains. Finally, Chapter 8 concludes the dissertation, and we provide a number of extensions for future work in quantile neural networks. Fig. 1.1 outlines the reading order for this dissertation. Chapter 3 in detecting cyber-DSM attacks, deterministic forecasting methods are used. As such, chapters 1 and 2 should be read first. Chapters 4 to 7 describe neural network methods for probabilistic forecasting which should be read in that order. But before reading these chapters, chapters 1 and 2 should be read first which lay the foundations for neural network based prediction.

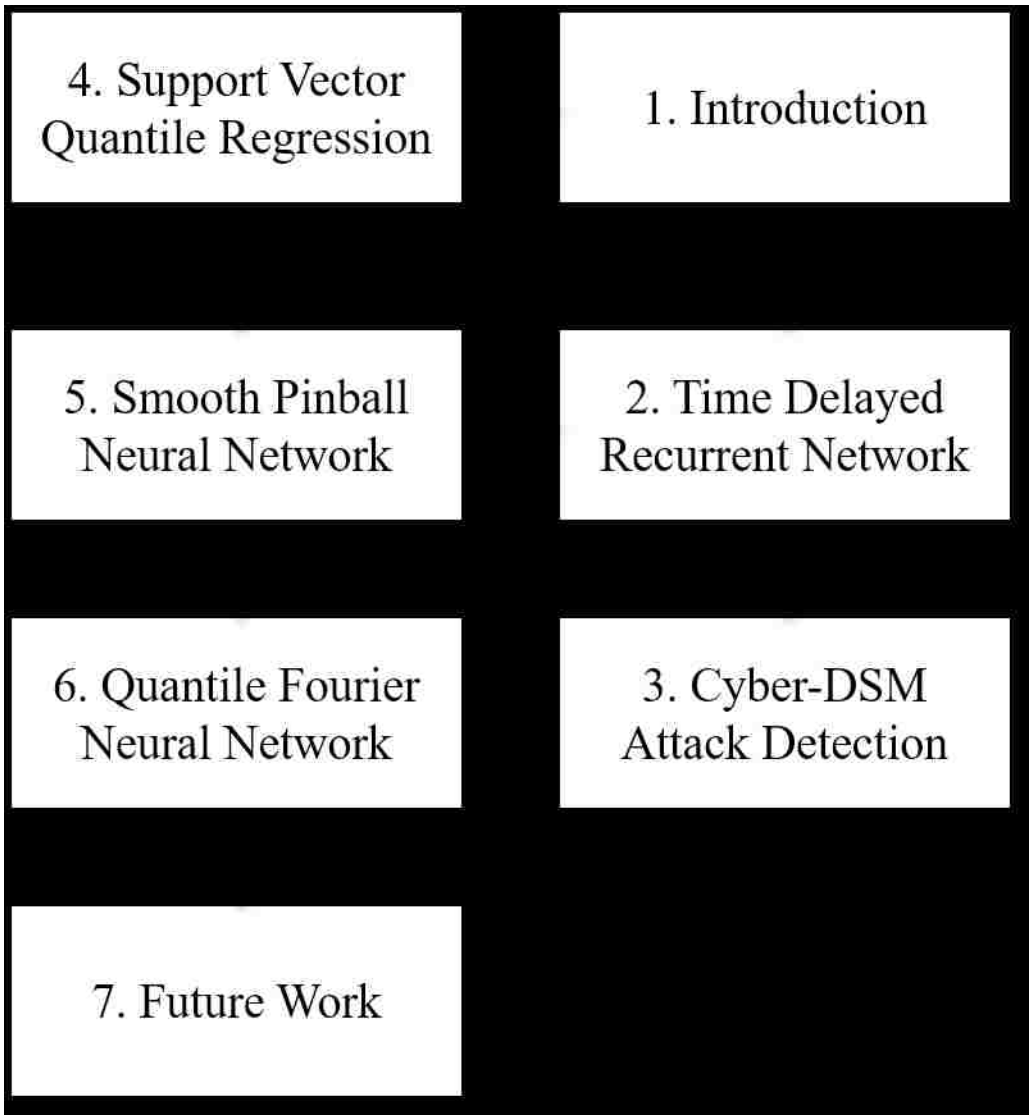

Figure 1.1: Reading order of the dissertation.

Chapter 2

Time Delayed Recurrent Neural Network for Multi-Step Prediction

---

Introduction

Time-series prediction is an active area of research and an important practical problem in a variety of disciplines such as in energy forecasting, economics, finance, signal processing, and many other fields [38]. In the last two decades, artificial neural networks (ANNs) have been extensively applied for complex time series processing tasks [39]. This is due to their strength in handling nonlinear functional dependencies between past time series values and the estimate of the value to be forecasted.

Our main motivating factor in developing an accurate non-linear time series prediction method is for use in short term forecasting of renewable energy sources (RES). Such predictions are vital for integrating RES into existing power grids. Because of the variability in the generation of power from renewables gaps are left in the supply which must be filled by dispatchable resources and this is where prediction plays a key role. This is especially true in the new and active research area of controlling ocean wave farms and individual wave energy converters (WECs). In a wave farm sensors are located near WECs to measure ocean waveforms and relay

the information to the WECs, resulting in enhanced control and better interactions with the grid and markets.

In this work, we focus on using an ANN for the prediction of future wave elevations at the measurement location. While waves are considered to be more predictable than other renewable energy sources such as wind or solar, the intermittency of wave energy is a big challenge in employing this resource as a reliable source of electric power in a generation portfolio. A demanding problem is the prediction of future levels of wave heights for operational scheduling and control in wave energy farms. The presented work is motivated by the fact that these predictions could allow new control methods to increase the efficiency of wave energy converter devices [40–42].

We have previously studied the use of a non-linear autoregressive (NAR) neural network trained using backpropagation which we called NARNET [43], to conduct multistep forecasts of wave power directly from observed wave heights. Long term predictions were made for several look ahead hours such as 3h, 6h, 12h, and 24h. The data used was significant wave height (SWH) which is the mean wave height of the highest third of the observed waves. This presents a challenge in testing accuracy of very short term predictions on the order of seconds. So in our current study we focus on shorter range forecasts which are obtained from simulated ocean wave height data rather than observed SWH data.

In this study we develop, test, and discuss a new method for nonlinear time series prediction. In enhancing the accuracy of our previous NARNET model while also using less memory in the training period, we used particle swarm optimization (PSO) for learning. We call this new model of using PSO to train a NAR neural network PSONAR. Since it is parametric it has all the advantages concerning estimation and testing connected with similar parametric methods. In addition, our neural network fulfills the requirements for the universal approximation theorem of neural networks [44] so it should be able to approximate any unknown nonlinear process. Further contribution comes in the form of our new stochastic inertia weight, a parameter used in our PSO learning algorithm, that emphasizes exploration more than exploitation in the search space.

Background

A variety of different prediction approaches have been applied to forecast signals vital to reliably integrating RES into the power grid. Such diverse domains include weather, wave characteristics, wind speed, and electric load. Looking specifically at ocean waves, it has been traditional to apply physical models to predict wave heights [45] but there is a recent effort to utilize machine learning algorithms to train on past time series data to predict future forms. Various learning methods being tested range from regression to neural network models. Prediction of wave elevation levels has been done before for real time control of wave energy converters where the wave elevation is treated as a univariate time series and it is predicted only on the basis of its past history [46].

Predictive time series models like the auto-regressive (AR), auto-regressive moving average (ARMA), and auto-regressive integrated moving average (ARIMA) have been found suitable for forecasting in a number of different domains. AR and ARMA models specifically are appropriate for stationary time series, while ARIMA and other models such as Kalman filters are aimed at non-stationary data and long-term series. These models therefore have also found widespread use in the simulation and estimation of the wave characteristics based on historical data [39]. Alternatively, artificial neural networks (ANN) have been found equally useful and even shown to outperform ARMA models [47]. Due to their high performance, we chose ANNs for prediction. In an ANN, forecasting is done by feeding a sequence of previous observations to the model as input so that it can recognize hidden patterns in the series and consequently estimate future values. Neural networks have been applied to study aspects of waves such as predicting the height of a wave at the time of its breaking and the depth of water at which it breaks [48].

An ANN can be trained by any number of learning algorithms, the most well known one being gradient descent backpropagation. Another lesser known alternative is particle swarm optimization (PSO). In 1995, Dr. Eberhart and Dr. Kennedy

developed PSO which is a population based stochastic optimization strategy, inspired by the social behavior of a flock of birds [49]. PSO is similar to Genetic Algorithms (GA) in terms of population initialization with random solutions and searching for global optima in successive generations [50]. But unlike GA, PSO does not undergo crossover and mutation, instead the particles move through the problem space following the current optimum particles. When using PSO, a possible solution to an underlying numeric optimization problem is represented by the position of a particle in the search space. PSO is suggested to be a powerful training algorithm for ANNs [51–54].

PSO is a powerful global optimization algorithm which has been successfully hybridized with genetic algorithms [55], fuzzy logic [56], and support vector machines [57] for application in various domains. In related work, using PSO to train ANNs for prediction has also been used in the past such as [58–62]. However, most of these cases only looked at single time step prediction and did not consider predicting hundreds of future data points instantaneously. Some of these papers also did not consider the effect of training using an adaptive inertial weight in the PSO algorithm which could dynamically change the velocity of a particle. Our contributions look at these factors.

Neural Network Model

In predicting a time series which exhibits volatile behavior, a number of different ANNs have been found particularly useful [63]. A popular ANN model for time series prediction is the Nonlinear Auto Regressive model with eXogenous inputs (NARX) neural network [64]. This is a network with tapped time delay inputs and has recurrent dynamic feedback connections enclosing the layers. The output of the network is fed back into the input of the model. It is a combination of a multilayered perceptron, a simple recurrent network, a time delayed network, and a feedfoward backpropagation network. Fig. 2.1 shows a NARX network with the feedback element and one hidden layer. Below we first describe the structure of the

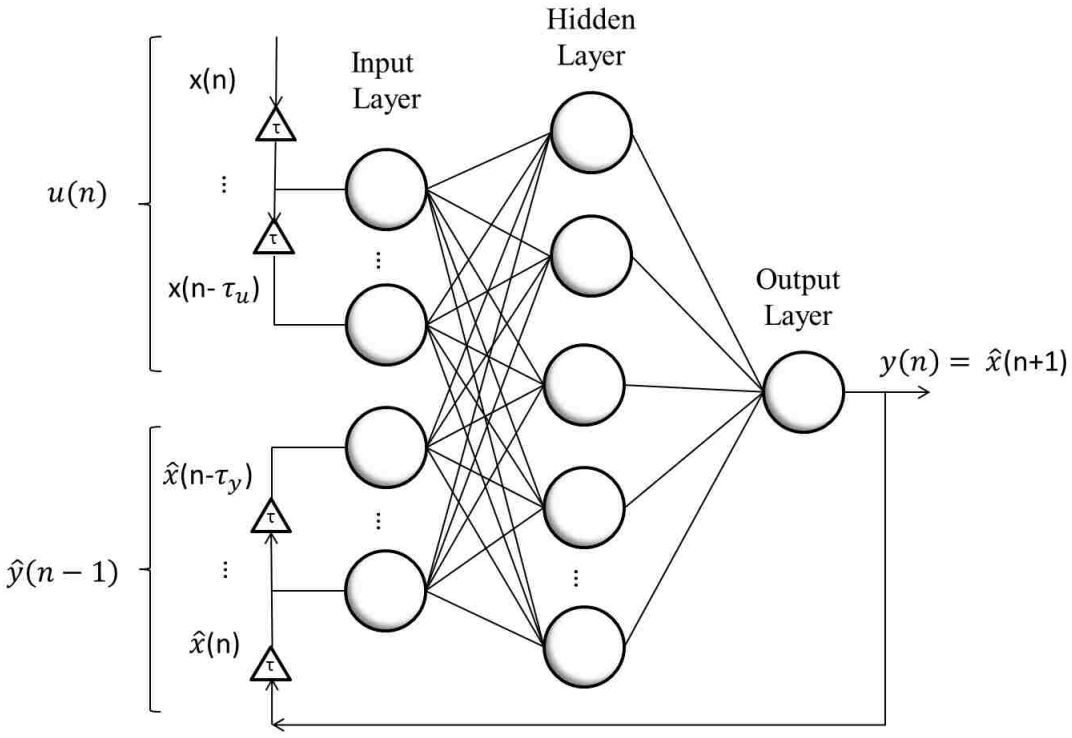

Figure 2.1: Example of a NARX network.

NARX model

$$y(n) =$$
$$f(x(n-1), x(n-2), \ldots, x(n-m_x), \ldots, \hat{x}(n-1),$$
$$\hat{x}(n-2), \ldots \hat{x}(n-m_{\hat{x}}))$$

where $x(n)$ and $\hat{x}(n)$ symbolize, respectively, the input and output functions of the neural network at time n, while m_x and $m_{\hat{x}}$ represent the input and output memory order, and f is a nonlinear function. The function f is approximated by a multilayered perceptron. The output at time-step n is dependent on its past $m_{\hat{x}}$ values and the past m_x input values. The network uses a time-delayed (TDL) architecture with a feedback connection from the output layer to the input layer of the network. States in the NARX network are updated as

$$x_i\left(n+1\right)$$

$$= \begin{cases} u(n) & i = m_x \\ y(n) & i = m_x + m_y \\ x_{i+1}(n) & 1 \leq i < m_x \text{ OR } m_x < i < m_x + m_y \end{cases}$$

so that at time n the tap delays correspond to the values

$$x\left(n\right) =$$

$$\left\{u\left(n-m_x\right), \ldots, u\left(n-1\right), y\left(n-m_y\right), \ldots, y(n-1)\right\}$$

The MLP approximation of the function f consists of a set of nodes organized into two layers. There are $i = 1...N$ inputs and $h = 1...q$ nodes in the hidden layer. Each hidden node performs the following function

$$z_h\left(n\right) = \sigma\left(\sum_{i=1}^{N} w_{ih} x_i\left(n\right) + b_h\right),$$

where σ is the nonlinear transfer function

$$\sigma = \frac{1}{1+e^{-x}}$$

and $w_{ih} \in W^H$ are the weights between the input and output layer and $b_h \in B^H$ are the weights of the bias term in the network, given by

$$W^H = \begin{bmatrix} w_{1,1}^H & \cdots & w_{1,i}^H \\ \vdots & \ddots & \vdots \\ w_{q,1}^H & \cdots & w_{q,i}^H \end{bmatrix}, B^H \begin{bmatrix} b_1^H \\ \vdots \\ b_q^H \end{bmatrix}.$$

The output layer consists of a single linear node

$$y\left(n\right) = \sum_{h=1}^{q} w_h z_h\left(n\right) + \theta_o,$$

where θ_o is the bias term at the output node and $w_h \in W^O$ real valued weights, given by

$$W^O = \begin{bmatrix} w_1^O \\ \vdots \\ w_q^O \end{bmatrix}.$$

An advantage of the NARX architecture is that it can converge faster and generalize better than other networks [63]. In our experiments we specifically use a derivative of the NARX network known as the non-linear autoregressive (NAR) neural network where we do not have exogenous inputs. During testing, predicted values at each time step are the only inputs fed to the network. Diving deeper into the training and testing of our network, we differentiate between two modes of operation similarly based on the NARX network modes of operation from [65].

We call the first one series mode, shown in Fig. 2.2; this is used during training. Here the network has a purely feedforward architecture and a learning algorithm such as backpropagation can be used for training. The second one is called parallel mode in which the output is fed back to the input of the feedforward neural network as part of the standard NARX architecture, as shown in Fig. 2.3, with the DELAY being the tapped delay line. The series mode NAR network is converted to the parallel mode to perform the prediction during testing. Our NAR network has an input layer of neurons, an output layer, and one hidden layer. We previously used the Lavenberg-Marquardt backpropagation algorithm as a training function to update the weight and bias values of the NAR model [43], we call this neural network NARNET.

Particle Swarm Optimization

PSO is a stochastic global optimization algorithm which is based on the simulation of the social behavior of a swarm. It exploits a population of candidate solutions in a search space. The particles in the swarm are initialized with a population of random solutions which will move through the N-dimensional problem space to search for new potential solutions. A fitness, f, is then calculated as a

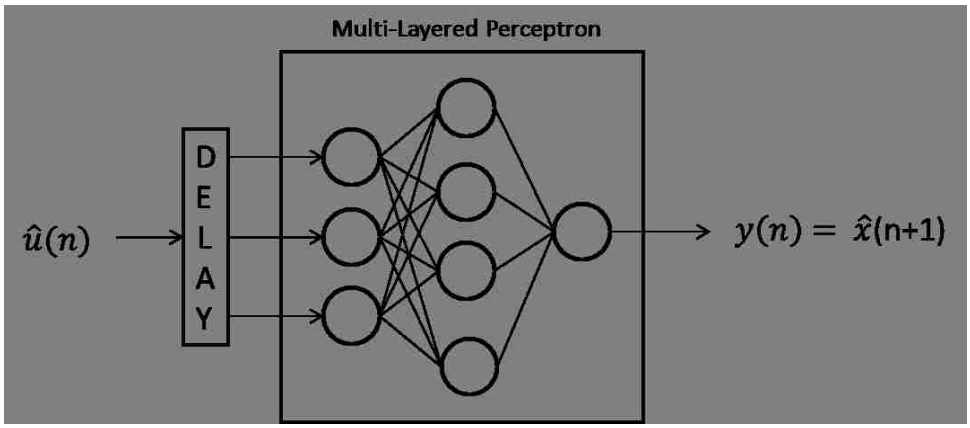

Figure 2.2: NAR network series mode.

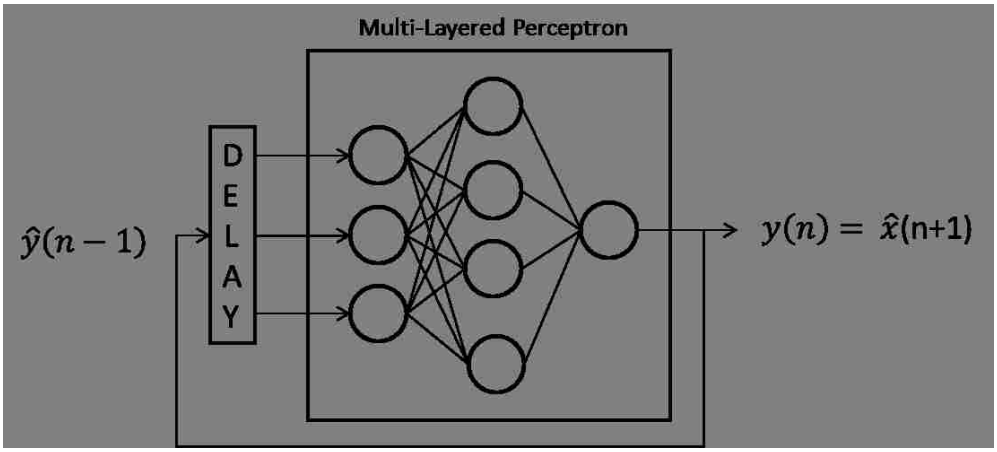

Figure 2.3: NAX network parallel mode.

certain measure of quality in reaching a target value. Each particle i is associated with two vectors, the velocity vector $\mathbf{V}_i = [v_i^1, v_i^2, ..., x_i^N]$ and the position vector $\mathbf{X}_i = [x_i^1, x_i^2, ..., x_i^N]$ and are updated as follows

$$\overrightarrow{v}_i(t+1) =$$

$$\overrightarrow{v}_i(t) + c_1\phi_1(\overrightarrow{p}_i(t) - \overrightarrow{x}_i(t)) + c_2\phi_2(\overrightarrow{p}_g(t) - \overrightarrow{x}_i(t))$$

$$\overrightarrow{x}_i(t+1) = \overrightarrow{x}_i(t) + \Delta t\,\overrightarrow{v}_i(t+1).$$

The first equation updates a particle's velocity which is a vector value with multiple components. The term $\overrightarrow{v}_i(t+1)$ is the new velocity at time $t+1$. The new velocity depends on three terms. The first term is $\overrightarrow{v}_i(t)$, the current velocity at time t. The second part is $c_1\phi_1(\overrightarrow{p}_i(t) - \overrightarrow{x}_i(t))$. The c_1 term is a positive constant called the coefficient of the self-recognition component. The ϕ_1 and ϕ_2 factors are uniformly distributed random numbers in [0,1]. The $\overrightarrow{p}_i(t)$ vector value is the particle's best position found so far. The $\overrightarrow{x}_i(t)$ vector value is the particle's current position. The third term in the velocity update equation is $c_2\phi_2(\overrightarrow{p}_g(t) - \overrightarrow{x}_i(t))$. The c_2 factor is a constant called the coefficient of the social component. The $\overrightarrow{p}_g(t)$ vector value is the best known position found by any particle in the swarm so far. Once the new velocity, $\overrightarrow{v}_i(t+1)$, has been determined, it is used to compute the new particle position $\overrightarrow{x}_i(t+1)$. The term $\overrightarrow{v}_i$ is limited to the range $\pm\overrightarrow{v}_{max}$. Δt is a time delay in updating a particles position, which is usually set to 1. The personal best position of a particle is calculated as:

$$\overrightarrow{p}_i(t+1) = \begin{cases} \overrightarrow{p}_i(t) & \text{if } f(\overrightarrow{x}_i(t+1)) \geq f(\overrightarrow{p}_i(t)) \\ \overrightarrow{x}_i(t+1) & \text{if } f(\overrightarrow{x}_i(t+1)) < f(\overrightarrow{p}_i(t)), \end{cases}$$

where f is the fitness function. PSO does not require a large number of parameters to be initialized. But the choice of PSO parameters can have a large impact on optimization performance and has been the subject of much research [66]. For most practical applications, a typical choice of the number of particles is in the range from 10 to 40. Usually, 20 particles is sufficient to get good results. In the case of more difficult problems, the choice can be increased to 100 - 200 particles. Parameters c_1 and c_2, (the coefficients of self-recognition and social components respectively) are not critical for the convergence of the PSO algorithm, but fine-tuning these learning vectors aids in faster convergence and alleviation of local minima. It is suggested to set these parameters to $c_1 = c_2 = 2$ [49]. An alternative could be to choose a larger self-recognition component, c_1, than the social component, c_2, such that it satisfies conditions such as $c_1 + c_2 = 4$ [67].

The particle dimension and range is determined based on the problem to be optimized. Various ranges can be chosen for different dimension of particles. Usually

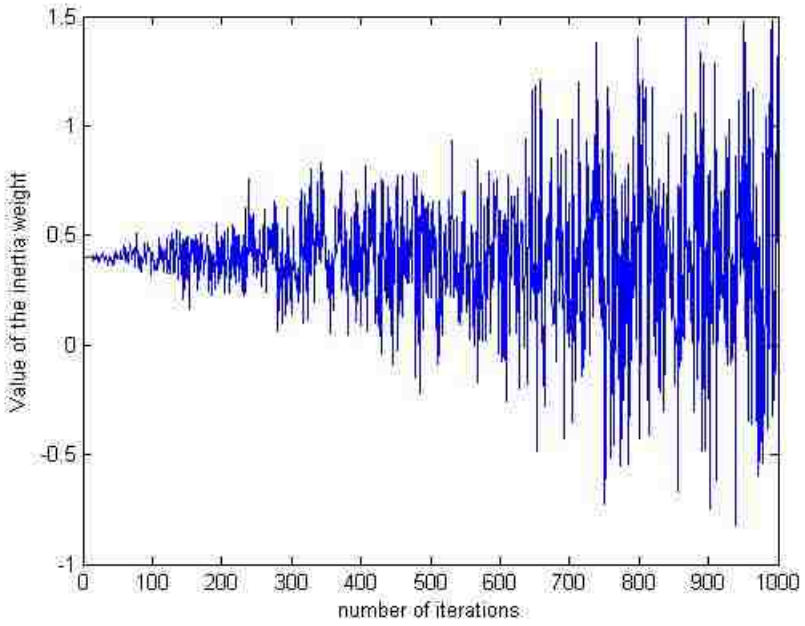

Figure 2.4: The scheme for value of the inertia weight w.

the range of particles is set as the maximum velocity. For instance, if a particle belongs to the numeric range -5 to 5, then the maximum velocity is 10. The stopping criteria may be any one of the following: the process can be terminated after a fixed number of training epochs or iterations such as 1000, or the process may be terminated when the error between the obtained objective function value and the best fitness value is less than a specified threshold.

2.0.1 Adaptive PSO

The adaptive particle swarm optimization (APSO) algorithm is based on the original PSO algorithm and is described below:

$$\overrightarrow{v}_i(t+1) =$$
$$\omega \overrightarrow{v}_i(t) + c_1\phi_1(\overrightarrow{p}_i(t) - \overrightarrow{x}_i(t)) + c_2\phi_2(\overrightarrow{p}_g(t) - \overrightarrow{x}_i(t)).$$

The ω factor is called the inertia weight. The inertia weight plays an important role in the convergence behavior of the PSO algorithm. The inertia weight is employed to control the impact of the previous history of velocities on the current one. Accordingly, the parameter ω regulates the trade-off between the global exploration and local exploitation abilities of the swarm. A large inertia weight aids in global exploration (searching wide ranging areas), while a small inertia weight aids in local exploitation (searching within the nearby areas). To obtain a balance between the global and local searches the number of iterations required to locate the optimum solution are reduced. The inertia weight is usually set as a constant initially and in order to promote global exploration of the search space, the parameter is gradually decreased to get more optimal solutions.

There are several variants of using an inertia weight. The inertia weight proposed by Shi and Eberhart [68] decreases with iterative generations as:

$$\omega = \omega_{max} - (\omega_{max} - \omega_{min})\frac{g}{G}$$

where g is the current iteration index, G is the predefined maximum number of training epochs, and ω_{max} and ω_{min} are the maximal and minimal weights. From empirical experimentation we have devised a slight variation of the inertia weight which we call a stochastic initial weight:

$$\omega = \omega_{min} + \phi(\omega_{max} - \omega_{min})\frac{g}{G}$$

where ϕ is a normally distributed random number at each iteration. Our measure is contradictory to the idea of having the inertia weight decrease over time. However, we believe the increased exploration allows the particles to search more broadly over time for new solutions in order to minimize the fitness function. Fig. 2.4 shows the change in ω over iteration where it increases stochastically over time.

PSO for Training of Neural Networks

In the case of training a neural network through PSO, a particle's position represents the values for the network's weights and biases. The goal is to find a position

so that the network generates computed outputs that match the outputs of the training data. PSO is initialized with a group of random particles. In one epoch or iteration, each particle is updated by following two best values. The first one is the best weight solution achieved so far by each individual particle's best position. Another best value tracked by the particle swarm optimizer, are the global best weight values, obtained by any particle in the population; so, the velocity and position of the obtained optimum solution is updated during an iterative process. The stop criteria are reaching the maximum iteration number or satisfaction of the minimum error condition.

Fig. 2.5 illustrates the flowchart of training a NAR neural network with PSO which we call our PSONAR network. The first step is initialization. The network is first constructed by choosing the number of input, hidden, and output node parameters. Then all the PSO variables such as c_1, c_2, dt, V_{max}, etc, and termination conditions are chosen. These parameters have a very important effect to promote the networks efficiency. The position vectors $\mathbf{X}_t$ and velocity vectors $\mathbf{V}_t$ are initialized randomly between 0 and 1. Here the training data can also be preprocessed.

In the training algorithm, the search space is $\mathbf{N}$-dimensional where $\mathbf{N}$ is set by the total number of weights and biases in the network between the input and hidden layer, and the hidden and output layer. The position coordinates of each particle $x_i(t)$ in the search space is then the following weights:

$$x_i(t) = \left\{ \begin{array}{cccc} w_{ih}^1, & w_{ih}^2, & ..., & w_{ih}^n, \\ w_{ho}^1, & w_{ho}^2, & ..., & w_{ho}^m \end{array} \right\},$$

where w_{IHn} represents all the weights between the input and hidden nodes and w_{HOm} represents all the weights between the hidden layered nodes and output node. During training, weights and biases of the NAR network are set to the calculated positions of the given particle. Then all training data is fed into the network for a given particle's weights to get the network's outputs. We then use the mean squared error (MSE) as our fitness function. If the MSE is below a threshold then training ends. If not then the pBest score of the given particle's best position $p_i(t)$ is compared to the MSE. If it is greater then the pBest score for the given particle

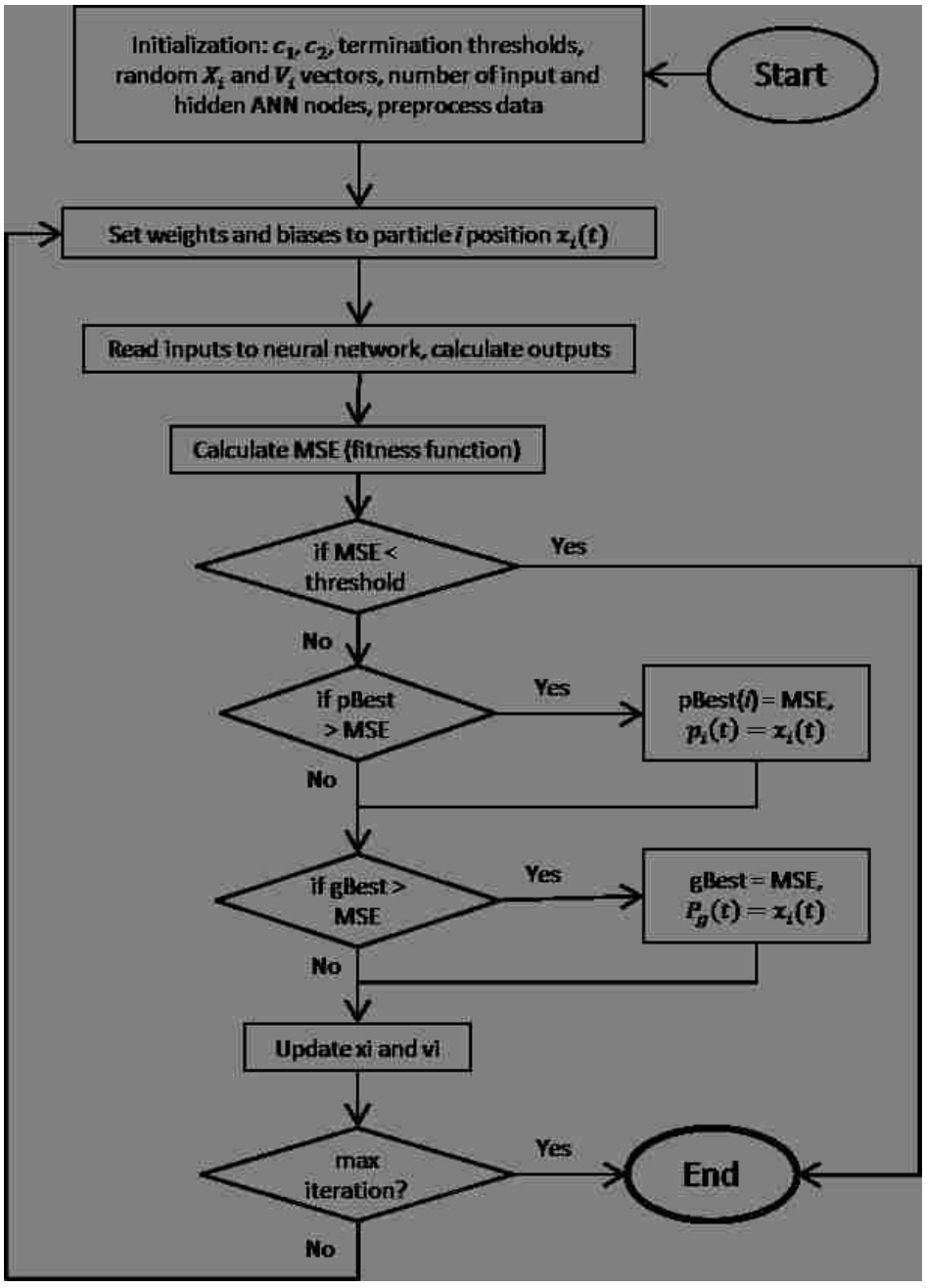

Figure 2.5: PSONAR learning algorithm.

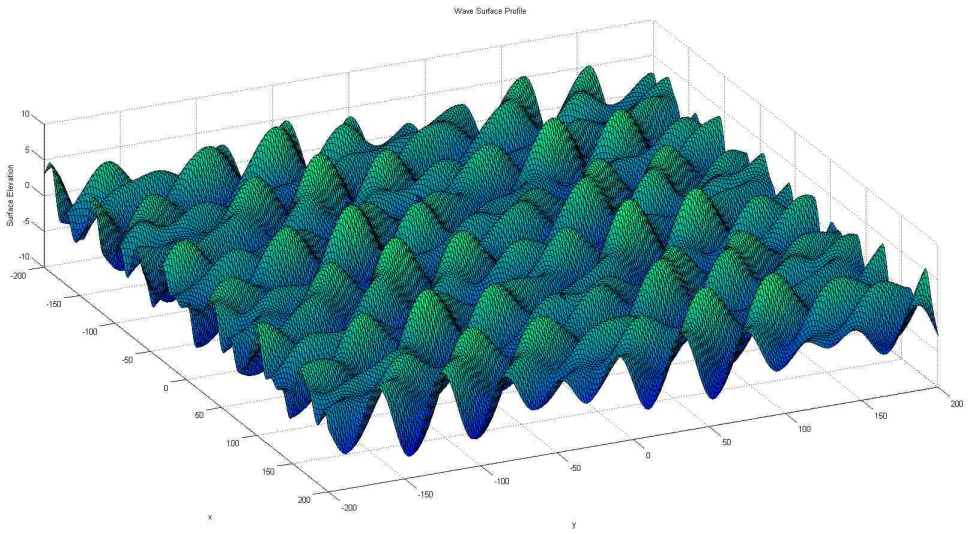

Figure 2.6: 3D plot of spatial distribution of simulated ocean waves.

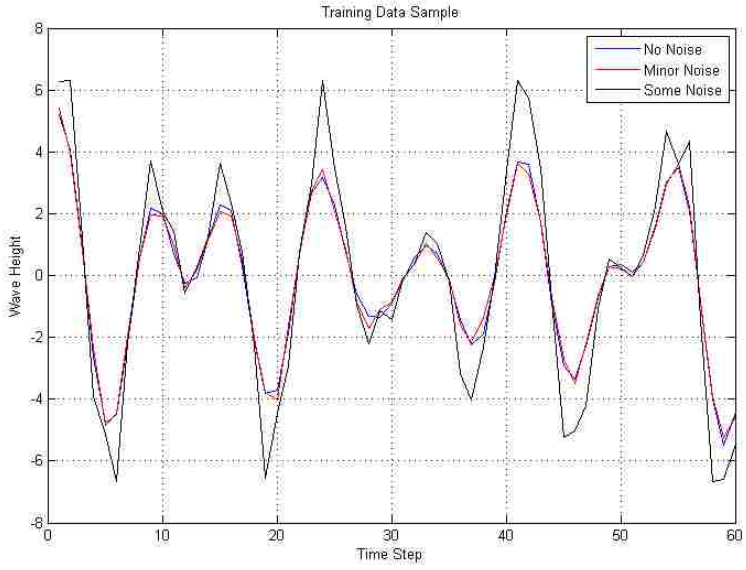

Figure 2.7: Training data window sample of 30 seconds.

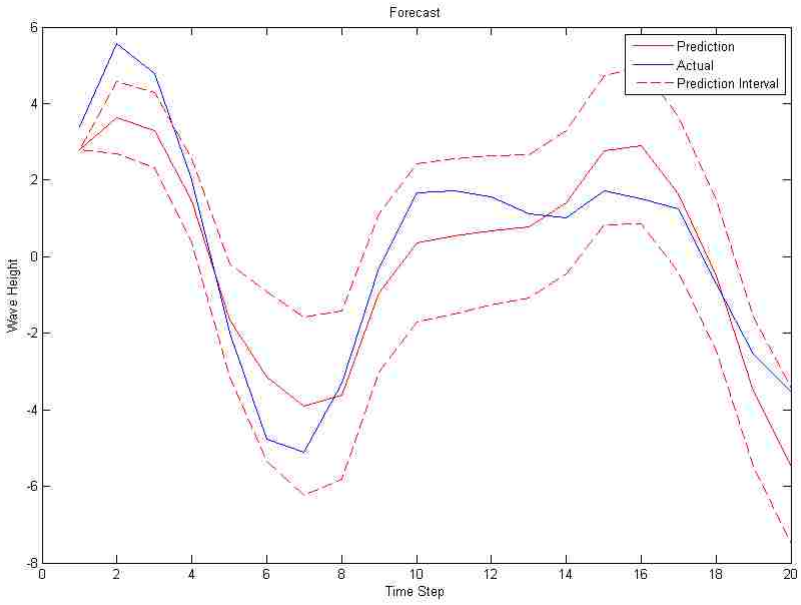

Figure 2.8: 10 second lookahead (20 steps) PSONAR prediction example.

is set to the MSE and $p_i(t)$ is set to the current position $x_i(t)$. The global best of all the particles is compared to the fitness MSE. If it is greater, then the gBest score is set to the MSE and $P_g(t)$ is set to the current position $x_i(t)$. Next the position and velocity vectors of the current particle are updated. If at this point the maximum number of epochs has been reached then training stops. Otherwise the next particle in the swarm is evaluated in the network. While we use one hidden layer in our PSONAR, the training algorithm can easily be scaled to include multiple hidden layers.

2.0.2 Data Source

For our analysis, we use the well accepted standard wave equation model from [69] where we assume the ocean is an ideal incompressible fluid with no loss of mechanical energy. We also adopt the common assumptions that the fluid motion is irrotational and that the wave amplitudes are small enough so that linear theory is applicable. Moreover, the deployment area in the ocean is assumed to be of sufficient depth such that finite depth effects, other than dispersion, are small. Finally, we assume that the waves were created by forcing functions, distant storms for example, that were applied at sufficient distances away resulting in the observation of fully developed ocean waves. Under the assumptions just described, the solutions for the simplified differential wave equation are plane waves consisting of a sum of sinusoids with different amplitudes, frequencies, directions, and phases. Therefore, we assume that wave elevations under a local wave field are described by plane waves having, in total, M directions and frequencies. Then, for all two dimensional sensor locations $(x, y)^T$ on the surface, all times t of interest, the exact phase-resolved (wave-by-wave) wave elevation $\eta(x, y, t)$ which would be observed at a particular location in the deployment area is described by

$$\eta(x, y, t) =$$
$$\sum_{i=1}^{M} A_i \cos\left((\frac{w_i^2}{g})(x\cos(\beta_i) + y\sin(\beta_i)) - tw_i + \phi_i\right),$$

where A_i is the amplitude in meters, ω_i is the frequency in radians per second, β_i is the angular direction in radians measured relative to the x-axis, ϕ_i is the phase in radians, and g is the acceleration due to gravity in $\frac{m}{s^2}$.

Using the just described model for the wave elevation, we assumed a local wave field that is described by five component plane waves, each having its own amplitude, frequency, direction, and phase. The sets used to describe the wave field were $A = \{2, 2, 1, 1, 1.5\}$, $w = 2\pi\{0.2, 0.25, 0.3, 0.22, 0.15\}$, $\beta(degrees) = \{50, 40, 25, 50, 0\}$,

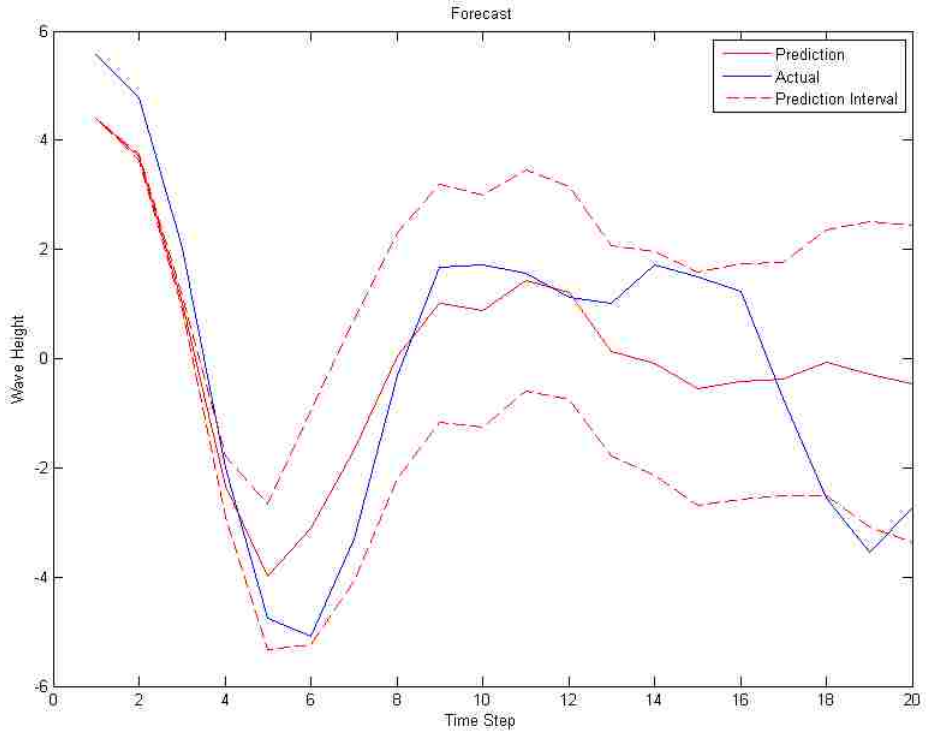

Figure 2.9: 10 second lookahead (20 steps) NARNET prediction example.

$\phi = \{2, 1, 3, 0.3, 0\}$ where $A_i, w_i, \beta_i, \phi_i$ are the amplitude, frequency, direction, and phase of the i^{th} component wave, where $i = 1, 2, \ldots, 5$. These values are chosen based on typical values reported in experiments and observations found in [69–71]. A snapshot of the assumed wave field is given in Fig. 2.6.

The training data is then taken to be the wave elevation measurements taken at the origin for 250 seconds, starting at time equal to zero, sampled at 2 Hz or one sample every half a second observed under independent additive white Gaussian noise with zero mean and 0.01 variance (minor noise) and second set with zero mean and 0.25 variance (some noise). A sample window of 30s of data with noise is shown in Fig. 2.7.

2.0.3 Error Measurements

Error measurement statistics play a critical role in analyzing forecast accuracy, observing exceptions, and benchmarking methods. In our results we use five error statistics, so that we can prove a broad comparison metrics for future studies, and we provide confidence intervals for predicted values. We first use the mean absolute percentage error (MAPE) which measures the size of the error in percentage terms and is a common estimate of error in forecasting problems. MAPE is defined as

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{x_i - y_i}{x_i}\right|,$$

where N, is the total number of predicted values, x_i is the actual observed value, and y_i is the forecasted value by the neural network. We also calculate the mean square error (MSE) and root mean squared error (RMSE) to evaluate accuracy

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2, RMSE = \sqrt{MSE}.$$

We then use the mean absolute deviation (MAD) which measures the size of the error in units. MAD takes the absolute value of forecast errors and averages them over the entirety of the forecast time periods. It is calculated as the mean of the unsigned errors

$$MAD = \frac{1}{N}\sum_{i=1}^{N}|x_i - y_i|.$$

Lastly we use the Pearson's correlation coefficient (CC) which is a measure of the linear dependency between two variables, in our case the observed and forecasted values, and is calculated as

$$CC = \frac{\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \overline{x})^2 \sum_{i=1}^{N}(y_i - \overline{y})^2}}.$$

A confidence bound for a model or variable is an interval of values within which we expect the true value of the population parameter to be contained. For calculating

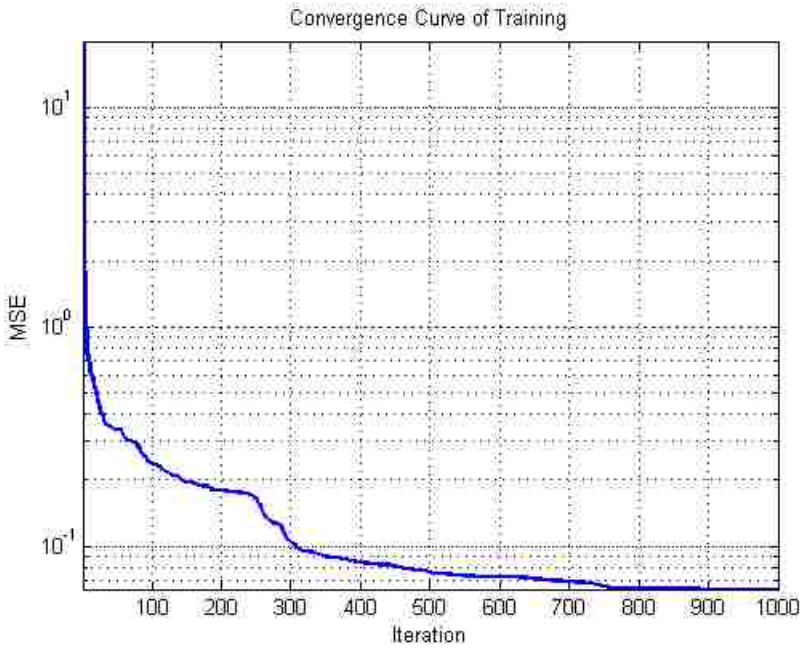

Figure 2.10: Convergence plot of PSONAR in Fig. 2.8.

bounds for our model we use the bootstrapping method [72] which resamples residuals for estimating the variance of forecasted values at each time step. The upper and lower endpoints are specified as

$$\overline{Y} \pm z\frac{\sigma}{\sqrt{n}},$$

with $\overline{Y}$ representing the sample mean (center of the confidence interval), $\sigma/\sqrt{n}$ being the standard deviation of the sampling distribution, and z is a constant multiplier set at 1.96 for a 95% confidence interval.

Results

The PSONAR neural network is used to predict results of up to 5, 10, 30, and 60 second forecasts where each forecast is composed of 10, 20, 60, and 120 prediction steps. The max number of training epochs was set to 1000. The NAR

network for both PSONAR and NARNET has 1 hidden layer and empirically we chose them to have 10 input nodes and 14 hidden node each. For the PSONAR network we chose the coefficient of self-recognition to have a weight of 2.5 and the social component to have a weight of 1.5. For the stochastic inertial weight we set $\omega_{max} = 0.9$ and $\omega_{min} = 0.4$. Fig. 2.10 shows an MSE convergence plot for an example prediction using PSONAR whose results are showcased in Fig. 2.8. Fig. 2.9 shows the same sample experiment run with the NARNET. Error statistics of the plots are summarized in Table II.

Table 2.1: Table of error statistics with minor noisy data.

PSONAR	**5s**	**10s**	**30s**	**60s**
MSE	0.1626	0.7284	6.3961	9.8831
RMSE	0.4033	0.8535	2.5290	3.1437
MAPE	0.3428	0.4302	1.5131	4.5228
MAD	0.3437	0.7399	2.0099	2.5102
CC	0.9972	0.9643	0.4944	0.0103
NARNET	**5s**	**10s**	**30s**	**60s**
MSE	0.1531	1.3392	7.2867	11.4703
RMSE	0.3912	1.1572	2.6993	3.3867
MAPE	0.2033	0.7033	1.3948	3.4318
MAD	0.3161	0.8971	2.0568	2.6227
CC	0.9971	0.9332	0.4980	0.2076

Table I shows the error statistics for multistep prediction on data with minor additive white Gaussian noise. Table II shows the error statistics for data with more noise. As seen in the two tables, for 5s lookahead prediction the NARNET using gradient descent backpropagation for learning performed better. Looking further into the future PSONAR outperformed the NARNET network to a small extent.

Table 2.2: Table of error statistics with higher noisy data.

PSONAR	**5s**	**10s**	**30s**	**60s**
MSE	0.2097	0.9153	8.9066	13.4073
RMSE	0.4579	0.9567	2.9843	3.6615
MAPE	0.1734	0.2964	2.0758	3.2429
MAD	0.3470	0.6783	2.4161	2.8644
CC	0.9974	0.9693	0.3743	0.0902
NARNET	**5s**	**10s**	**30s**	**60s**
MSE	0.1603	1.4119	10.8491	14.0818
RMSE	0.4003	1.1882	3.2937	3.7525
MAPE	0.2531	0.7421	1.3229	4.7647
MAD	0.3698	0.8938	2.0769	3.0842
CC	0.9971	0.9165	0.31433	0.12301

Conclusion

This work proposed a method for using particle swarm optimization to train a non-linear autoregressive neural network. The accuracy of our proposed PSONAR is tested using ocean wave heights for the purpose of aiding the integration of wave energy into the power grid. As such, we believe our scheme will be helpful in multi-step prediction as needed in integrating other stochastic renewable resources, such as wind or solar. Moreover, compared to existing methodologies, the PSONAR can be applied to other applications where predictions are needed for multiple time steps. Our method of using a stochastic inertial weight in the PSO learning algorithm to train our NAR neural network with simulated data showed successful results in predicting short term ocean wave levels. Results show PSO can outperform backpropagation.

In this study we ran experiments on one test set and used only a single set of parameters for initializing the neural networks and the PSO learning algorithm. Further work is planned to compare results of the stochastic inertia weight with a decreasing inertia weight, expand the number of testing sets, test the effect of varying the number of neural nodes and hidden layers, and provide a deeper computational

analysis such as usage of memory and error analysis. We plan to also test a number of PSO variants such as Clan PSO [58, 73], Trelea PSO [74], hybrid backpropagation PSO [51], and other swarm methods such as ant colony optimization [75]. We would then evaluate their effectiveness as training algorithms for training a NAR network for time series prediction by running a Monte Carlo method to obtain a distribution of multiple runs with each comparative method. In these studies we plan to incorporate both simulated data and actual data from buoy sensors for wave height elevations for studies in wave energy farm and wind speeds for studies in wind energy farms.

Chapter 3

Detection of Cyber-DSM Attacks using Forecasting and Supervised Learning

---

Introduction

In this chapter we examine a new domain of cyber enabled demand side management (DSM). We study this domain in depth, provide a mathematical framework for it, and showcase it's vulnerabilities to attacks with the main goal to detect such attacks. We introduce this problem to determine if its structure requires the need of deep learning and forecasting. A number of statistical and machine learning methods are used for the detection of attacks and we provide an analysis if linear or nonlinear methods in this case warrant better or worse results.

DSM is an essential component in smart grids for planning, monitoring, and modification of consumer loads levels. Furthermore, future cyber-enabled DSM will allow smart grids even higher levels of automated decision-making capabilities to selectively schedule loads on local grids to improve power balance and grid stability. Such a cyber approach relies heavily on real-time, two-way communication capabilities between a central controller and various adaptable loads. Research into security

and reliability of the cyberinfrastructure that enables DSM is therefore vital. The main concerns in ensuring DSM security and safety lay in the feedback mechanism of real-time electricity pricing and distributed DSM controllable loads. Particularly in residential grids, each load contributes only a small amount of power and its compromising might not cause a noticeable impact on the power grid. However, a carefully planned or even chaotic cyber attack might impact other loads not under attack or not under DSM control by taking advantage of the feedback mechanism of load management.

Two-way communication capabilities of advanced metering infrastructure (AMI) enables a utility or independent system operator (ISO), in the retail power markets, to collect high-resolution energy usage from consumers and enable dynamic pricing to adapt to consumer demand. Thus, AMIs provide an efficient way for ISOs to schedule prices and to then communicate those prices to consumers for automatic DSM control of certain portions of a load. AMIs can also provide practical ways for ISOs to set DSM goals such reducing peak or decreasing aggregate load levels through price influences. However, there are several vulnerabilities in AMIs that present noteworthy security issues since they are directly accessible by users. Additionally, due to the large scale deployments of AMIs, ISOs encourage the utilization of marginally cheaper hardware which results in constrained computational resources to allow for robust security capacities, for example, intrusion monitoring.

DSM programs utilize demand response, which is a specific tariff or program to motivate customers to respond to changes in price or availability of electricity over time by altering their regular electricity use habits. We take this a step further and envision that cyber-enabled DSM programs will be able to autonomously control household loads such as water heaters and HVAC units based on RTP. As part of the reliable implementation of this future cyber-DSM, it is crucial to be able to understand the dependency between dynamic pricing and automatic demand response as well as the risks. Cyber-DSM programs can be particularly vulnerable to cyber attacks such as false pricing information or direct load manipulation.

Our work is thus motivated to study this vulnerability in DSM. In the absence of appropriate simulation methodologies, we provide a mathematical formulation of

the feedback between utilities and DSM systems, and then simulate, analyze, and test different detection methods for attacks on such feedback. This relationship between load and price is shown visually in Fig. 3.1. As prices go up, demand naturally responds by decreasing. However, if AMIs are hijacked, and false lower prices are reported to DSM systems, then there will be an inappropriate increase in demand. A similar effect happens if an attacker directly controls user loads. Then a higher load usage by the attacker may inadvertently lead to higher prices for the rest of the grid. We present how attackers can exploit such a dependency. We propose a mathematical framework of the feedback between price setting and DSM systems to study how attacks take place and how to detect them. The main contributions of our approach can be summarized as follows.

First, we show the simple application of block bootstrap to simulate load data for analyzing the relationship between load and price instead of relying on cumbersome grid system simulations. We formalizing the price to load relationship using an elastic demand model to achieve DSM goals. We combine the first two points to generate load and pricing data in a DSM system; in particular under a strategic conservation scheme as an example of a DSM goal. We propose two modes of attacks on DSM systems: false pricing data injection and direct load manipulation. We prove their equivalence and highlight three types of attacks that could be undertaken by each mode. We empirically show how a high use of DSM can exacerbate attacks. We simulate these attacks and review sequential change-point and machine learning methods for detecting DSM attacks. We show that despite the complexity of the domain, linear detection methods yield better results then nonlinear.

In section 3 we provide a literature review on DSM, and important DSM strategies real-time pricing and load forecasting. In section 3.0.4 we apply the block bootstrap technique for simulating the non-DSM load distribution of a micro-grid of N homes from template residential load time series. Dependency models for the feedback nature of load and prices are proposed in section 3.0.6 where we showcase simulations of residential load and electricity prices when an automatic DSM program controls certain portions of consumer demand as a function of price. In section 3.0.10 we present two modes of cyber attacks, direct load manipulation attacks and

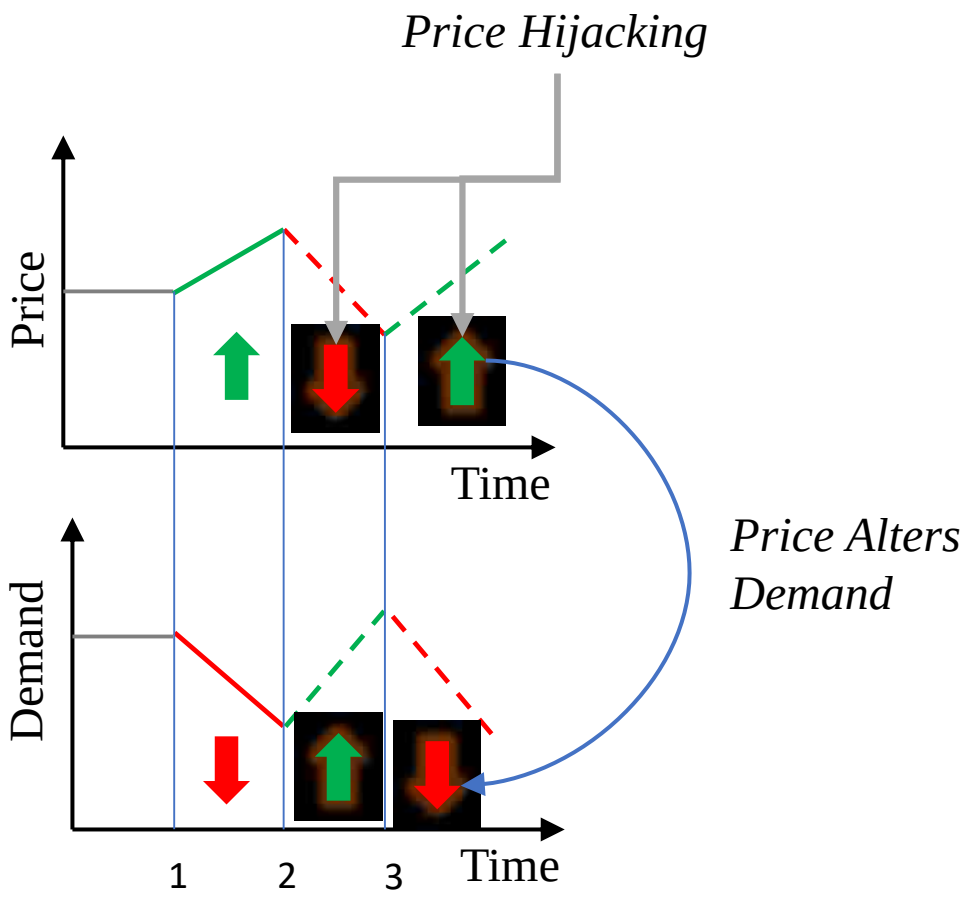

Figure 3.1: Feedback effect between price and DSM demand. As prices go up, demand decreases. But if prices are hijacked and false prices are fed to DSM systems then a false low price increases demand, and a false high price can decrease demand. The same is true if demand was altered by an attack. If load usage is increased by an attacker then prices would increase and vice versa.

price data injection attacks that can have a significant influence on the feedback of load and price. We prove these two attacks are equivalent. We conclude in section 3.0.13 with possible directions of future work.

Background

There are multiple strategies to accomplish DSM like load forecasting and real-time pricing which are used for load management on in-home energy management systems. So before introducing our models on the relationship between price setting and demand response by cyber DSM systems, it is crucial to review foundational material on DSM. In this section, we provide a brief overview of DSM goals and approaches, real-time pricing, and load forecasting. We also review different pricing simulation schemes which will play a role in modeling RTP.

3.0.1 Demand Side Management

DSM is an active and voluntary approach for reducing electricity use through activities or programs that promote electric energy efficiency, conservation, or more efficient management of electric energy loads [76]. Very often financial incentives and educational programs are used to modify consumer demand. More specifically, the main goals of DSM are peak clipping, valley filling, load shifting, strategic load growth, flexible load shaping, and strategic conservation. These goals are summarized in Fig. 3.2. In these goals, consumers are encouraged to use less energy during peak hours, or to move the time of energy use to off-peak times such as nighttime, or reduce overall consumption. Other applications for DSM is to aid grid operators in balancing intermittent generation from wind and solar farms due to their volatility nature which may not coincide with energy demand at different times of the day.

In our study, we focus on modeling and simulating the DSM goal of strategic conservation, due to its simplicity and essential use in the smart grid. This goal also makes it easier to study attacks on DSM by modeling strategic conservation as a general reduction in load. Attacks then could stand out more versus goals like flexible load shaping. More specifically, this DSM goal aims at reducing aggregate load demand through directed reduction of electricity consumption. The successful implementation of strategic conservation programs usually requires some combination of financial incentives to customers, the promotion of energy-efficient building

standards, and appliance efficiency improvement. Strategic conservation also requires a more excellent knowledge on the part of the utility concerning customer behavior. We envision in the future that AMI and smart appliances in residential DSM programs will automatically control specific portions of consumer load as a function of real-time electricity prices to achieve the goal of strategic conservation.

Most DSM programs are formulated as an optimization problem as follows

$$\min_{P_t} \sum_{t=1}^{T} (L_t - L'_t)^2,$$

where P_t is RTP at time t, L_t is actual load, and L'_t is the target load level the ISO is interested in achieving via DSM. The aim is to choose a price P_t for each time step such that the actual load would reach as close as possible to the target level. In [77], a DSM strategy was proposed based on heuristic optimization to shape the load curve close to the desired shape. A heuristic-based evolutionary algorithm was used to solve the above minimization problem. A multi-agent game theoretical DSM approach is proposed in [78]. The authors use game theory and formulate an energy consumption scheduling game, where the players are the users, and their strategies are the daily schedules of their household appliances and loads. In [79], the minimization problem is solved by utilizing a feedforward neural network to map the nonlinear relationship between price and load. Recently, the DSM problem is addressed in [80] as a multi-objective optimization problem that also seeks to balance other merit functions such as energy production cost, costumers preferences, and other constraints. Unlike these approaches, we propose a non-optimization based solution utilizing the formulation of the price elasticity of demand. Our approach is thus simpler, tractable, and more interpretable. Our solution also provides a framework to simulate and study the effects of DSM attacks easily.

3.0.2 Load Forecasting

Load forecasting (LF) techniques are an essential component for RTP and other ISO operations by predicting future energy requirements of a system from previous data and weather conditions. It is recognized as the initial building block of utility

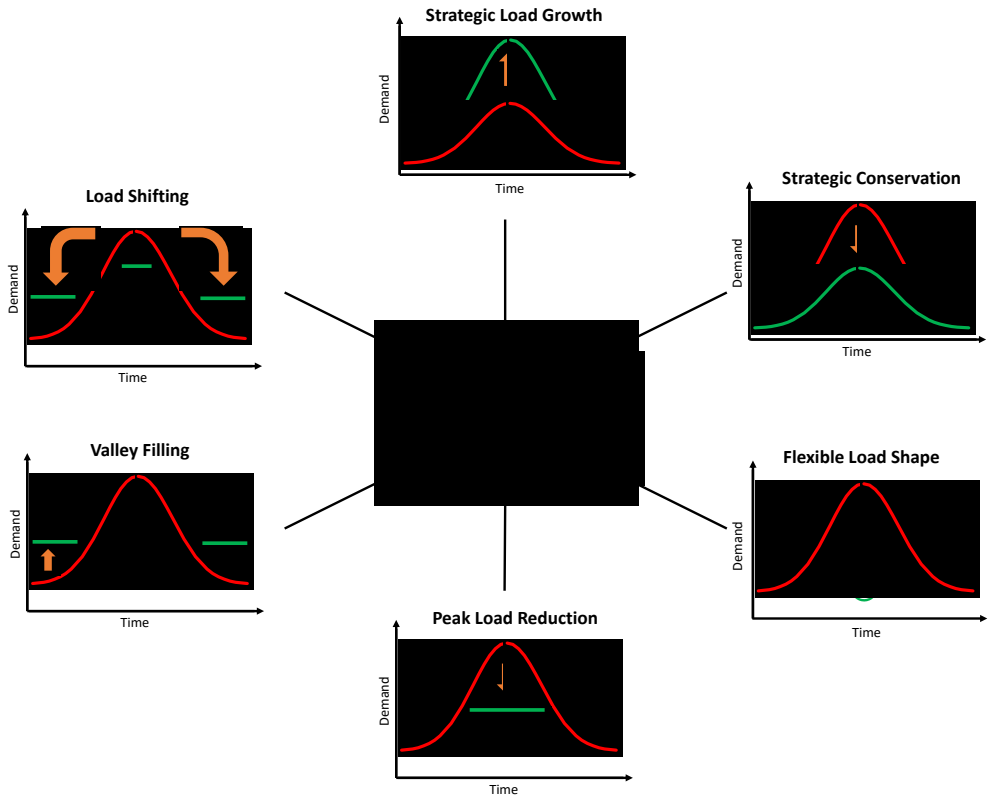

Figure 3.2: Various demand side management goals.

planning efforts and ensures the balance between supply and demand of energy. For a given system and requirements, LF provides predictions for specific periods. These periods are divided into short, medium, and long term forecasts. Short term LF is used to predict load on an hourly basis up to 1 week for daily operations and cost minimization. Medium-term LF typically predicts load on weekly, monthly, or yearly basis for efficient operational preparations. Long term LF is used to predict load up decades ahead to facilitate grid and generation expansion planning. In this work, we look at short term LF on the resolution of one hour up to a week.

LF models can be divided into two approaches [81], the first being statistical based modeling and the second being machine learning. The statistical approach

can be further broken down into regression and time series models. Multiple linear regression can be used with the weighted least squares estimation technique to form a relationship between different independent covariates that load depends on such as weather conditions. Regression models have been applied in LF in different works such as in [82]. Time series models are also prevalent to apply to LF. The most common model is the autoregressive moving average (ARMA) model and its variants that include components such as integration (I), fractional integration (FI), multivariate series (V), seasonality (S), exogenous (X) data, conditional heteroskedasticity (CH), and nonlinearity (N).

Hyperparameters of ARMA models can be solved using Box-Jenkins decomposition or grid search with an Akaike information criterion. Various studies have looked at all the different ARMA models for LF [83]. Other time series methods for LF include simple exponential smoothing [84] and the Holt-Winters seasonal method [85]. Time series analysis and regression analysis share many models and ideas, but they are theoretically different. Time series analysis first deals with time indexed stationary data and account for the autocorrelation between time events. In regression we assume there is no autocorrelation, and that all observations are independent and identically distributed. Furthermore, we also assume in regression the data is homeostatic and does not exhibit multicollinearity.

Most recently, machine learning methods have seen a huge spike in LF research. Machine learning models are data-driven, typically providing a nonlinear fit to input covariate data to predict load. Advantages of this approach include not needing preconditions for data such as stationarity (a requirement for most time series methods), excels at modeling nonlinear dependences, and can fit large data sets. Disadvantages for most machine learning models are that most hyperparameters are continuous (difficult to tune), they require extensive feature engineer, and may get stuck in local minimums. Models for LF include support vector machines [86], feedforward neural networks [87], recurrent neural networks [88], random forests [89], and ensemble learning [90].

In all the various LF approaches, benchmarks are required to compare the prediction performance of our models. The most common benchmarks are the naive

persistence and seasonal methods. In the naive persistence method a prediction of the load for time t is equal to the value from the previous time step $t-1$ and in the seasonal persistence method a prediction of the load for time t is equal the same hourly value from the previous day $t-24$.

3.0.3 Real Time Pricing

Every consumer of electricity is charged with a certain amount per kilowatt hour (kWh) of energy. Such a charge is done to cover the costs associated with the generation, transmission, and distribution of electricity. The two main types of costs are operational and fixed costs. During the 20th century, tariffs have been used to recover costs. Lately, clever pricing schemes have been developed to meet the requirements of modern power systems [91], such as, real-time pricing (RTP) where consumers are charged with a price nearest to the real price of generation at a specific interval in time. RTP plays an integral part in time-based DSM programs that makes consumers choose the time of consumption of power as a response to prices [92]. Cyber-enabled DSM programs are an automated form of time-based DSM programs.

There are two types of RTP schemes, hourly pricing and day ahead pricing. In the first type, the price of electricity is released on an hourly basis for the next hour. In day ahead pricing, prices are announced for the next 24 hours based on predicting the load demand and the cost of generation. RTP signals combined with DSM automation at the consumer level provides benefits to both consumers and utility. A properly designed RTP scheme increases the reliability of the grid, reduces associated costs with generation, and lowers electricity bills of consumers. Further review of RTP and other dynamic pricing schemes can be found in [81].

3.0.4 Models for Pricing Simulation

Most work that model and simulate the relationship between load demand to electricity prices are those that study how to price financial derivatives in electricity

markets. Most models for a spot market employ one or two factors: one factor capturing the short-term hourly price dynamics characterized by mean reversion and very high volatility, and another factor representing seasonal price behavior. The work in [93] provides a good overview of stochastic electricity pricing models which we outline below.

In the last few years there has been a rapid increase in literature on stochastic models for prices of electricity and other commodities. The simplest model for spot prices takes into account mean-reverting behavior and is given by an Ornstein-Uhlenbeck process, where spot prices S_t at time t follow a diffusion process satisfying the stochastic differential equation

$$dS_t = -\lambda(S_t - \alpha)dt + \sigma dW_t$$

where W_t is a standard Brownian motion, σ is the volatility of the process, and λ is the velocity with which the process reverts to its long term mean α. In electricity markets, prices show strongly mean reverting behavior so estimates for λ are set quite high. Alternatively, a two factor model can be used of the form

$$dS_t = -\lambda(S_t - Y_t)dt + \sigma dW_t$$

where Y_t is a Brownian motion. Several other approaches describe prices in the form

$$S_t = f(t) + X_t \text{ or } S_t = \exp(f(t) + X_t) \text{ or } S_t = \exp(Y_t + X_t)$$

where X_t is an Ornstein-Uhlenbeck process responsible for the short-term variation, Y_t is a Brownian motion describing the long-term dynamics, and $f(t)$ is an arbitrary deterministic function to model seasonality based on time or a load forecast. These approaches are able to realistically simulate electricity prices to help price derivative contracts such as options and futures but they do not help describe the nonlinear feedback relationship between load and price, nor do they incorporate elements to model DSM in this feedback relationship. Therefore, we provide in this work a feedback framework between load and price that includes DSM dynamics.

Microgrid Simulation

Before modeling the load-price dependency of a DSM system, we need first to obtain some ground truth data of what load data from a residential micro-grid looks like without the presence of DSM, where we assume the elasticity of demand to price is very low. To do this, we use the power time series from several homes as templates for our grid and then generate artificial N household datasets. There are alternative ways to generate artificially residential load data, such as by using power grid simulators such as MATPOWER [94] or GridLAB-D [95]. Such software is cumbersome and time-consuming in the simulation. Our object is to create unlimited but plausible univariate load data to serve as the base demand for sample households before the application of a DSM system. Thus, time series processes are more suitable for such a task.

3.0.5 Data Source

We use the UMass Smart* [96] dataset, 2017 release, for the simulation of micro-grid load time series. The Smart* project built a data collection infrastructure that records data from a variety of sensors deployed in real homes. Their infrastructure supports both pulling data by querying individual sensors and pushing data from sensors to a gateway server, which ran on their software tools. The 2017 Home dataset release is comprised of electrical power readings from seven homes from 2014 to 2016 at a minute resolution. It includes readings from individual appliance sensors as well as total power usage of each home. We chose to use these seven datasets as template homes in simulating the power usage of a micro-grid due to the breadth of the data collected. For DSM attack research, these datasets can help model an attacker compromising individual appliances. For each home the power consumption is given in kW for every minute. We convert this time series to kWh with a resolution of one hour which is common for smart meter readings and real time price modeling [81]. We do this by obtaining the average power consumption

within an hour and multiplying it by the time period as such

$$E_{(kWh)} = t_{(hr)} \times \frac{1}{60} \sum_{i=1}^{60} P^{i}_{(kW)} \tag{3.1}$$

3.0.6 Block Bootstrap Simulation

In the generation of new time series from sample data, several approaches can be applied depending on the statistical properties of the series. Data that is stationary can be modeled and generated using an ARMA process [97]. An ARMA model is fitted to the data, and then future data is sampled from the ARMA distribution. If there is no serial correlation, then the distribution of some sample data can be modeled using Markov Chain Monte Carlo [98], and new data can be sampled from this estimated distribution. However, in the case that data exhibited autocorrelation and non-stationarity in the presence of a periodic seasonal pattern, then a natural choice is to use the block bootstrap method [99].

Bootstrap is used in simulation statistics for estimating the distribution of a statistic such as mean or variance. This is particularly useful when there is no analytical form to estimate the density of our underlying statistics. A bootstrap analysis is conducted by using the Monte Carlo algorithms with replacement. Data is sampled with replacement until a new set is formed and then statistics are calculated from that new set. The process can be repeated to get a more precise estimate of the Bootstrap distribution and to form confidence intervals for those statistics. The block bootstrap is used when the data, or the errors in a model, are correlated. The block bootstrap attempts to replicate the serial correlation by resampling blocks of data instead of individual observations. This is why the block bootstrap is used primarily with correlated time series. In block bootstrap, blocks sampled can overlap or be non-overlapping. For load time series simulation we use block bootstrap with non-overlapping blocks to preserve the daily seasonal pattern of power consumption. The process of block bootstrap simulation of a new home power usage from a template home is shown in Fig. 3.3.

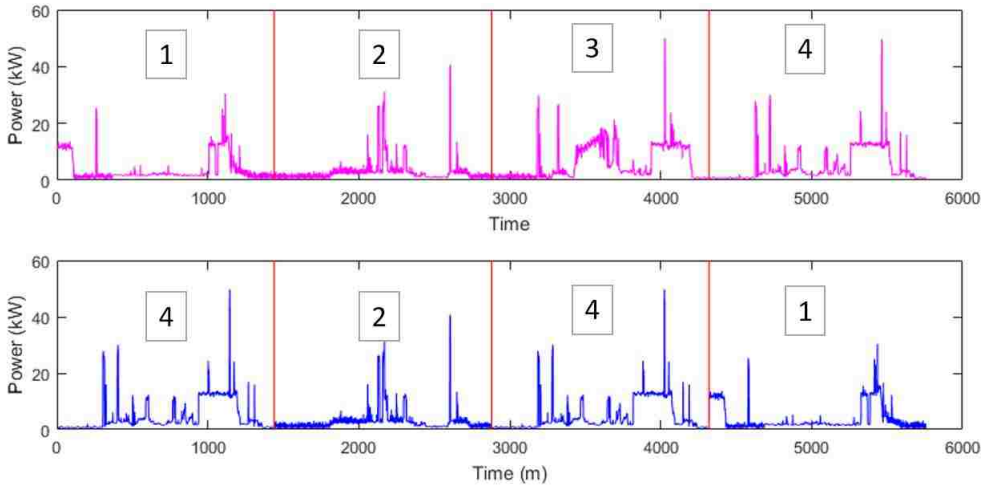

Figure 3.3: Process of block bootstrap simulation of a new home power usage (bottom) from a template home (top). Example simulation samples are taken from four days from the template series with replacement.

Dependency Model

3.0.7 Modeling Elastic Demand

For analyzing the feedback dependency between load and price in a DSM setting it is first required to define a supply and demand relationship of electricity. To do so we utilize the well known measure in economics, the Price Elasticity of Demand (PED) [100] which can be given by

$$\epsilon_d = \frac{dL}{dP} \cdot \frac{P}{L}. \tag{3.2}$$

PED shows the responsiveness of the Load (L) demanded of electricity to a change in its Price (P). An absolute value of PED = 1 shows unitary elasticity. For instance, when $\epsilon_d = -1$ then a 1% change in the price will have a 1% change in the load demanded. As prices increase, load will decrease. When absolute PED falls between 0 and 1, this signifies that the demand for load is inelastic, while a value greater then 1 says that the demand is elastic. When $|\epsilon_d| = 0$ the demand is perfectly inelastic. A change in price has no affect on the load. While $\epsilon_d = \infty$ represents

perfect elasticity. If ϵ_d is constant the whole demand curve then,

$$\begin{aligned}
\frac{1}{L}dL &= \epsilon_d \frac{1}{P}dP \\
\int \frac{1}{L}dL &= \epsilon_d \int \frac{1}{P}dP \\
\ln(L) &= \epsilon_d \ln(P) + c \\
e^{\ln(L)} &= e^{\epsilon_d \ln(P)+c} \\
L &= e^{\epsilon_d \ln(P)} \cdot e^c \\
L &= \left(e^{\ln(P)}\right)^{\epsilon_d} \cdot e^c \\
L &= P^{\epsilon_d} \cdot e^c \\
\text{substitute } a &= e^c \\
\Rightarrow L &= a \cdot P^{\epsilon_d}
\end{aligned} \tag{3.3}$$

where a is a scaling constant. An example demand curve estimated from Eq. 3.3 can be seen in Fig. 3.4. The figure also showcases the nonlinear relationship between load and price where as the price, the independent variable, increases the load demanded, the dependent variable, decreases.

3.0.8 Modeling Consumer DSM

In modeling the relationship between load and price under a DSM program, it is important to define the individual loads of each customer in determining aggregate load. For a customer who does not participate in a DSM program their load is determined by the stochastic demand process of users actions such as watching tv, using the AC, etc. Demand is impacted by multiple factors such as user preferences, weather, and time of day. In this process, electricity prices have a small influence on demand - individual customer demand is fairly inelastic to price. Following the derivation in Eq. 3.3 we define the load usage of an individual customer i for time t as

$$\phi_{t,i} = \theta_{t,i}(P_t + P_c)^{\epsilon^d_{t,i}},$$

where P_t is the RTP for time t, P_c is constant of the retailer's market costs which does not vary with RTP, $\theta_{t,i} \geq 0$ is a scaling factor representing the stochastic

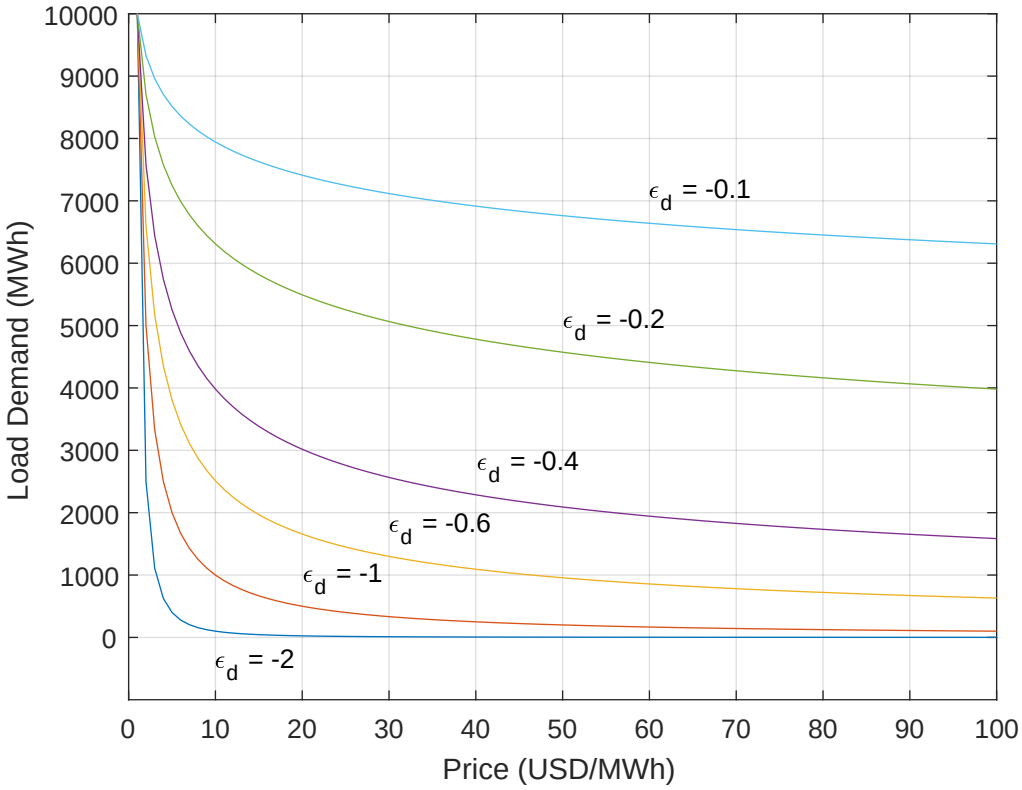

Figure 3.4: Load as a function of price with arbitrary price range \$1-100, $\alpha = 10,000$, and ϵ_d = -0.1,-0.2,-0.4,-0.6,-1, -2.

process that determines the user load, and $\epsilon^d_{t,i}$ is the elasticity coefficient for the individual customers sensitivity to price changes. It can vary over time but without DSM incentives most users have a fairly inelastic PED. For experimental purposes, in modeling individual user loads we set $\phi_{t,i}$ equal to simulated bootstrapped user load profiles defined in Section 3.0.6. We assume that prices, user preferences $\theta_{t,i}$, and $\epsilon^d_{t,i}$ have been absorbed in the calculation of the simulated load series. Thus we use $\phi_{t,i}$ as a reference point to how much electricity a user wants to consume without the influence of a DSM program. We model the task of the ISO as modifying $\phi_{t,i}$ to some desired load levels.

The goal of DSM is to motivate the consumer to use less energy typically during peak hours. A DSM program could have a number of goals all aimed at reducing load

usage. These may be peak clipping where peak load is reduced, load shifting where times of higher load are reduced and times of low load are increased, and many more strategies. There are multiple ways a DSM program can achieve these goals such as financial incentives, education, and regulation. In this work we look at the near term future where in a home with a smart meter and smart appliances, users can participate in a DSM program that automatically adjusts the load of interconnected appliances (such as the water heater) as a function of price and a DSM elasticity term that determines how fast or slow a users load under DSM control.

Realistically, only a certain portion $\kappa_{t,i} \in [0, 1)$ of customer i's power usage will be under control of a cyber DSM program. There will always be some stochastic component of power usage such as using a microwave oven or electric hairdryer. An ISO can signal a users DSM program by setting prices where smart appliances adopt to price changes with certain elasticity. We do no model direct load control. If prices are set too high it will signal the DSM component of a users demand to start using less load. If prices are set low then it signals the DSM program to increase load usage. We model this DSM component as such

$$l_{t,i} = (\kappa_{t,i}\phi_{t,i})P_t^{\epsilon_{t,i}^{dsm}} + (1 - \kappa_{t,i})\phi_{t,i}, \tag{3.4}$$

where the first part $(\kappa_{t,i}\phi_{t,i})P_t^{\epsilon_{t,i}^{dsm}}$ is the load level customer i allows the DSM program to determine as a function of price P_t and the DSM's elasticity to price $\epsilon_{t,i}^{dsm}$. Price elasticity of DSM $\epsilon_{t,i}^{dsm}$ may very over time and customer and affects how much power usage should be affected by price. The second portion $(1 - \kappa_{t,i})\phi_{t,i}$ of a customers load is the stochastic component. The DSM portion parameter $\kappa_{t,i}$ is defined as a function of time where users can add or remove house loads under DSM control over time. The term $\kappa_{t,i}$ can be modeled as a random variable (e.g. Uniform or Gaussian) or as a fixed constant for users. Once each individual customers load has been determined then total load (modified by a cyber-DSM program) for time t for N customers is calculated by

$$L_t = \sum_{i=1}^{N} l_{t,i}. \tag{3.5}$$

We also define the aggregate base load as

$$\Phi_t = \sum_{i=1}^{N} \phi_{t,i}, \tag{3.6}$$

which represents the total demand had there not been a DSM program for a time period t (ie $\kappa_{t,i} = 0, \forall_i$).

3.0.9 Modeling DSM Goals

The ISO has different DSM goals for reducing a customers load profile as mentioned before these goals can be to reduce peak load or increase load in periods of low demand. Two ways for a DSM program to control load are to make it a function of electricity prices or for an ISO to directly control a homes energy usage by programming its smart appliance. Given the privacy issues of having an ISO directly control a homes energy usage, we model the indirect approach of achieving DSM goals through setting electricity prices.

We introduce an approach how an ISO can set RTP, on an hourly basis, as a function of aggregate load to achieve a desired load level L'_t. The approach takes as input a forecast $\hat{\Phi}_t$ of the base aggregate load to calculate price P_t. This prediction can be defined as

$$\hat{\Phi}_t = f_{pred}\left(\Phi_{t-k:t-1}, P_{t-k:t-1}, X\right), \tag{3.7}$$

where inputs to the prediction model are past base load Φ and price P values from time $t-k$ to $t-1$, and other predictor variables X such as time-of-day and weather information. Various types of prediction models can be used for f_{pred} such as neural networks as reviewed in section 3.0.2. We now define RTP based on the formulation in Eq. 3.3 as such

$$P_t = \left(\frac{L^*}{\hat{\Phi}_t}\right)^{1/\epsilon^{dsm}}. \tag{3.8}$$

Where,

$$\begin{aligned} &\text{Goal 1: } L^* = L'_t \\ &\text{Goal 2: } L^* = L'_t + (L'_{t-1} - L_{t-1}) \end{aligned}. \tag{3.9}$$

The component L^* adjusts the RTP based on two goals the ISO may have. The first goal is to adjust the price to push power usage directly to the target level L'_t with the assumption that there is near 100% DSM participation by all customers. The expectation is that if demand for time t is $\hat{\Phi}_t$, then a price point is set to push load usage to L'_t. Of course,if participation is less then 100%, which is more likely, then the target level L'_t will not be reached by P_t. So, to push the aggregate load, from all users, as close as possible to the target load level, with an unknown amount of participation, then a penalty would need to be added to P_t. We model this as Goal 2 where the idea is to affect the power usage of those under cyber-DSM control even more than goal 1 to compensate for users who are not participating in DSM.

Some users will not be participating, or only have a small portion of their power usage under control by the cyber-DSM program. We model their remaining power usage as inelastic to RTP. Thus, to push aggregate load to a target level, taking into account some load usage is inelastic, we need to push RTP much higher or lower to have a bigger effect in pushing DSM controlled load closer to the target load. This is what Goal 2 attempts to do, with the component $L^* = L'_t + (L'_{t-1} - L_{t-1})$ taking the target load level for time t and adding the difference from the previous target load L'_{t-1} and realized load L_{t-1} as a penalty to adjust RTP to compensate for the difference. If $L^* < 0$ then we set $L^* = 10$ or to some arbitrary small target value. By subtracting the difference between the previous load and target level, we make up for users not participating in DSM by forcing DSM users a higher price to push their load even lower.

The term $\hat{\epsilon}^{dsm}$ in Eq. 3.8 is an estimate of the price elasticity of DSM of the whole grid; if individual user coefficients $\epsilon^{dsm}_{t,i}$ are unknown then $\hat{\epsilon}^{dsm}$ can be estimated from observing past values of price and load under different levels of DSM control. Alternatively, the ISO can define ϵ^{dsm} for all household cyber DSM programs. The formulation in Eq. 3.8 sets prices by comparing the adjusted target load for time t to the forecasted base demand $\hat{\Phi}_t$ for the same time. This demand $\hat{\Phi}_t$ would be the level if no load was under DSM influence, thus to influence and alter it, Φ_t needs to be estimated as accurate as possible.

In our approach, if the aggregate load is above the target load, RTP is set higher

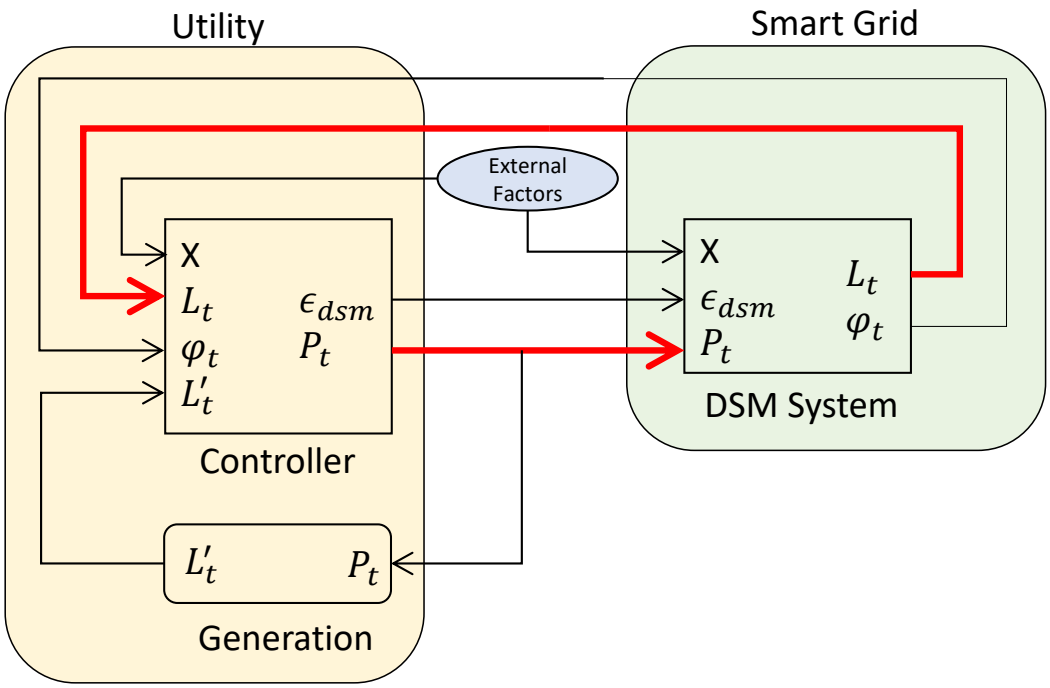

Figure 3.5: Block diagram highlighting the feedback between the utility and grid.

to decrease demand. If the aggregate load is lower than the target load, then the price is decreased to increase demand. Thus, as also can be observed, in Eq. 3.8, there is direct feedback between price and load Eq. 3.4. The block diagram in Fig. 3.5 also outlines this feedback that showcases the relationship between the utility and grid. Generation sets the target load based, on the price and supply of power, and the controller sets the price signal and the elasticity of demand coefficient for DSM systems. The price is then fed into the grid into DSM systems which adjust load usage appropriately. The bold red lines in Fig. 3.5 highlight the feedback relationship between price and demand. The scope of work is in the mathematical modeling of the controller and DSM system relationship. With such a model we present in the next section how attackers can exploit it.

Simulation Parameters	
N	200 homes
ϵ_{dsm}	-1
L_{target}	200 kWh

Table 3.1: Simulation parameters used in case studies.

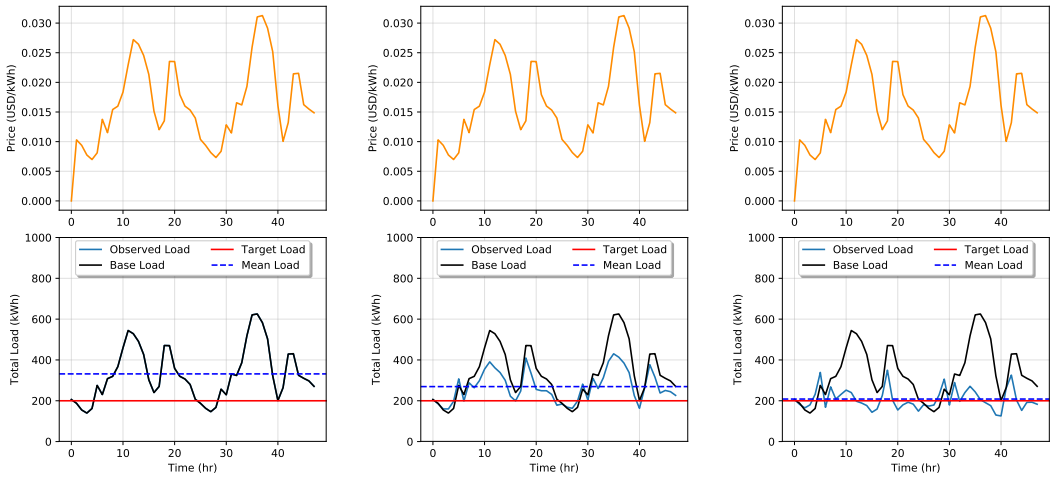

Figure 3.6: Simulation of price-load interaction with Goal 1 with DSM when $\kappa_i = 0, \forall_i$ (a), $\kappa_i = 0.5, \forall_i$ (b), and $\kappa_i = 0.99, \forall_i$ in (c).

3.0.10 Simulation Parameters and Assumptions

For experimental purposes, in modeling individual user loads we set $\phi_{t,i}$ equal to simulated bootstrapped user load profiles defined in Section 3.0.6. We assume that prices, user preferences $\theta_{t,i}$, and $\epsilon^d_{t,i}$ have been absorbed in the calculation of the simulated load series. Thus we use $\phi_{t,i}$ as a reference point to how much electricity a user wants to consume without the influence of a DSM program. We model the task of the ISO as modifying $\phi_{t,i}$ to some desired load levels. Furthermore, we include the following assumptions in our modeling and simulation. We assume the ISO can define ϵ^{dsm} for each household and we set it as a constant for all customers and time.

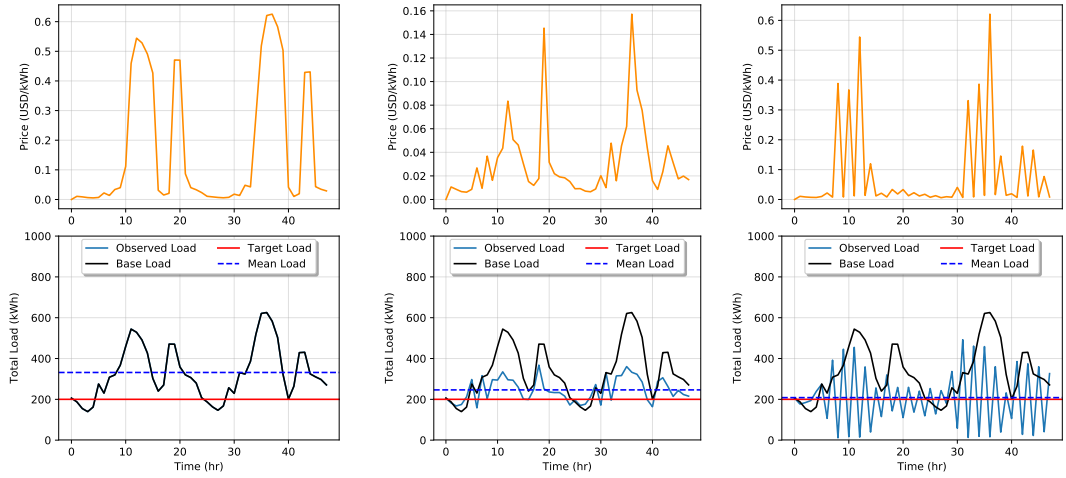

Figure 3.7: Simulation of price-load interaction with Goal 2 with DSM when $\kappa_i = 0, \forall_i$ (a), $\kappa_i = 0.5, \forall_i$ (b), and $\kappa_i = 0.99, \forall_i$ in (c).

For additional simplicity, we model κ_i also invariant in time, but it may vary per customer. Lastly, through AMI, we assume ISO the can obtain an estimated reading of Φ_{t-1} but not of individual user $\phi_{t-1,i}$ to preserve privacy. This way the ISO has a time series of estimated non-DSM load demand in order to provide predictions $\hat{\Phi}_t$.

For all case studies in the rest of our paper, we use the UMass Smart* dataset (2017 release) to bootstrap simulate residential load as described in Section 3.0.4. We simulate a micro-grid of $N = 200$ residential homes for a time period of T to obtain a *phi* distribution that defines a base load profile for each home i at each time t. We set the DSM demand elasticity for each user to $\epsilon^{dsm} = -1, \forall_i$ to allow the DSM component of customer load to be sensitive enough to price changes. For all case studies we set a simple flat target load of $L_{target} = 200$ kWh. We note, however, that with our pricing formulation in Eq. 3.8 we can model any DSM goal from peak load reduction to flexible load shaping. These simulation parameters are summarized in Table 3.1. For forecasting $\hat{\Phi}_t$ we use the naive persistence prediction method.

In Figures 3.6 and 3.7 we demonstrate a price-load feedback simulation for a period of $T = 48$ hrs under Goals 1 and 2 with parameters defined in Table 3.1. Under the Goal 1 scenario when $\kappa_i = 0, \forall_i$, shown in Fig. 3.6(a), we see that prices range from \$0.01 to \$0.03 per kWh. As expected with no DSM program to influence demand, the observed load is equal to the base load. The same is observed in Fig. 3.7(a) when the simulation is run with the Goal 2 RTP model. The only difference under this scenario is the price range which is exceptionally high ranging from \$0.01 to \$0.6 per kWh. This is expected in this scenario since the ISO is attempting to set prices to maximize the effect on DSM customers, which there are none, but this is unknown to the utility. With no DSM the mean observed load is 332 kWh.

Next, we rerun the simulation setting $\kappa_i = 0.5, \forall_i$. In the Goal 1 scenario in Fig. 3.6(b) we see that the base load was reduced with a mean observed load of 269 kWh. In Fig. 3.7(b) the same simulation is run with Goal 2. Here the mean observed load was further reduced to 246 kWh, but large price spikes occur at peak load times. Finally, we run simulations for $\kappa_i = 0.99, \forall_i$ to study the effects of high penetration of DSM. In simulating both goals, shown in Figures 3.6(c) and 3.7(c), the mean observed load is reduced to 208 kWh, which is very close to the target load. However, in Goal 2 we see great resonating feedback affect occur when prices spike very high. RTP increases as a response to large values in observed load. Then when load decreases to low levels prices decrease cause load to spike more during the next time step. While under Goal 1 this is not observed. The higher prices set in Goal 2 would see a large cost to DSM participating customers.

DSM Attack Models

An attacker can exploit the feedback between the customer and utility in determining RTP and load usage by cyber DSM programs by injecting false price or corrupted load data into the feedback loop. The attack exploitations we study here are different from the false data injection attacks studied in other smart grid papers. Most false data injection attack works [101] study the compromise in energy

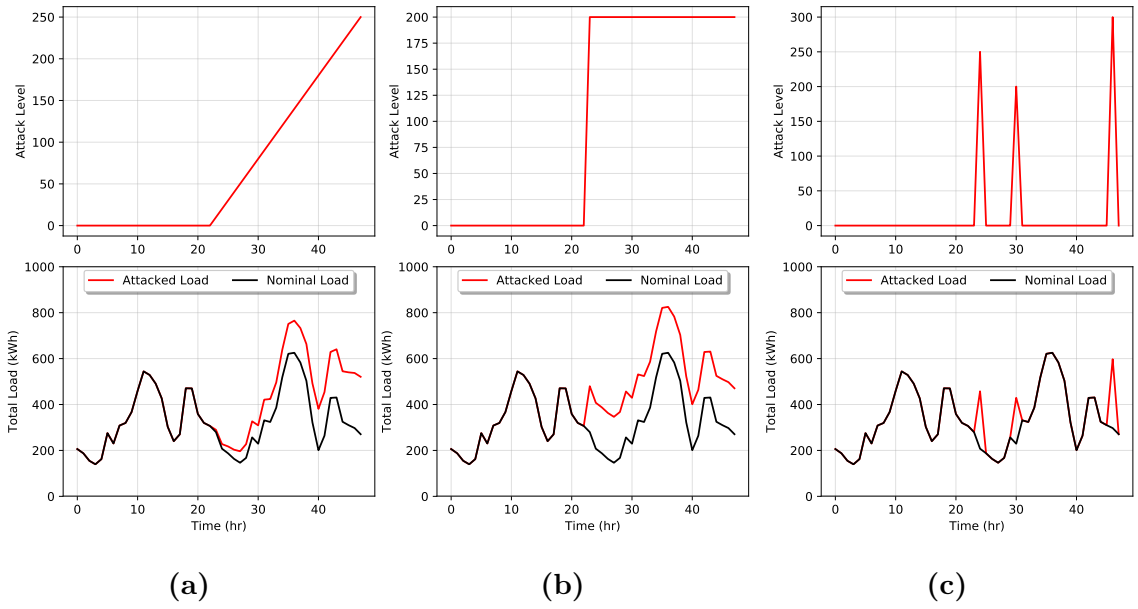

Figure 3.8: Examples of different type of DSM attacks: (a) ramp attack, (b) sudden attack, and (c) point attack.

management systems to alter power state estimates by the utility operator. In our case, we study attacks that aim to alter a users load profile by exploiting cyber DSM vulnerabilities.

For modeling attacks on a cyber DSM managed micro-grid we assume that the attacker compromises a subset of all the N customers, we denote this subset as $\mathcal{A}$, for an attack period $t \in T_a$. We study two modes of attacks: false pricing data injection attacks in which a compromised user receives manipulated pricing information, and a direct load manipulation attack in which the appliances of the compromised customer are under the control of the attacker. When communication encryption is broken with an AMI, then a pricing data injection attack can occur. By hacking into a cyber DSM load controller, or directly hacking into smart appliances, then a direct load manipulation can occur by altering a users load profile. The two modes of attacks are outlined below.

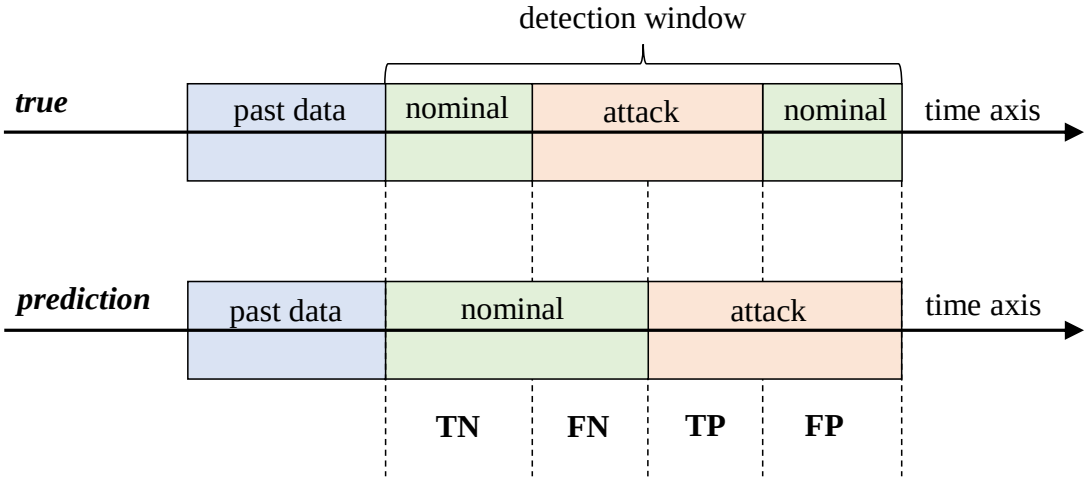

Figure 3.9: Confusion matrix imposed on a time axis of attack predictions vs true observations.

- **False Pricing Data Injection**: The attacker can manipulate prices P_t received by each compromised costumer $i \in \mathcal{A}$, and the received price P_t^i can be different for various customers in order to achieve the attacker's desired effect:

$$P_{t,i}^a = P_{t,i} + a_{t,i}^P, \forall_i \in \mathcal{A}, t \in T_a$$

  This has the affect of compromising the demand response of a customer in the following way:

$$l_{t,i}^{a^P} = (\kappa_i \phi_{t,i})(a_{t,i}^P)^{\epsilon^{dsm}} + (1 - \kappa_i)\phi_{t,i}$$

- **Direct Load Manipulation**: The attacker can manipulate the load of each compromised customer $l_{t,i}, i \in D$ directly:

$$l_{t,i}^{a^L} = l_{t,i} + a_{t,i}^L, \forall_i \in \mathcal{A}, t \in T_a$$

  Under both attack modes, we would get a compromised aggregate load which may include one or both attacks occurring simultaneously

$$L_t^a = \sum_{i=1}^{N} l_{t,i}^{a^P} + l_{t,i}^{a^L}.$$

These two modes of attack are equivalent as they both affect a customers load response as long a part of the load is under cyber DSM control that is sensitive to price changes.

Theorem 1. Given a set of customers $\mathcal{A}$ compromised by the attacker, there always exist a direct load manipulation attack such that all customers behave the same as a pricing data injection attack and vice versa for $\kappa_i > 0, \forall_i, t \in T_a$.

Proof. Setting both attacks to have the same load, then the attack load is set as follows

$$a^L_{t,i} = (\kappa_i \phi_{t,i})(a^P_{t,i})^{\epsilon^{dsm}} + (1 - \kappa_i)\phi_{t,i},$$

and the equivalent attacked price is

$$a^P_{t,i} = \left(\frac{a^L_{t,i} - (1 - \kappa_i)\phi_{t,i}}{\kappa_i \phi_{t,i}} \right)^{1/\epsilon^{dsm}}$$

If $\kappa_i = 0$ then the two attack modes are not equivalent since a price attack will have no affect on customer load. □

Since false pricing data injection and direct load manipulation attacks are equivalent, we focus only on direct load manipulation attack analysis.

There are different goals an attacker can have to harm the power grid or exploit it. For example, an attacker can cause chaotic metering by messing the metering data transmission, efficiency loss of the energy provided by causing greater load volatility, or the energy system failure by overloading the power lines or devices. The focus of this work is efficiency loss by increasing user loads through direct load manipulation. In this scenario, we introduce three possible types of load attacks. A ramp attack, sudden attack, and point attack. These type of attacks are shown in Fig. 3.8 wherein plot (a) an attacker gradually increases a users load over time. In plot (b) an attacker suddenly ramps up the power usage to a specified level, and in plot (c) we demonstrate a point attack where the attacker increases loads only for specific hours.

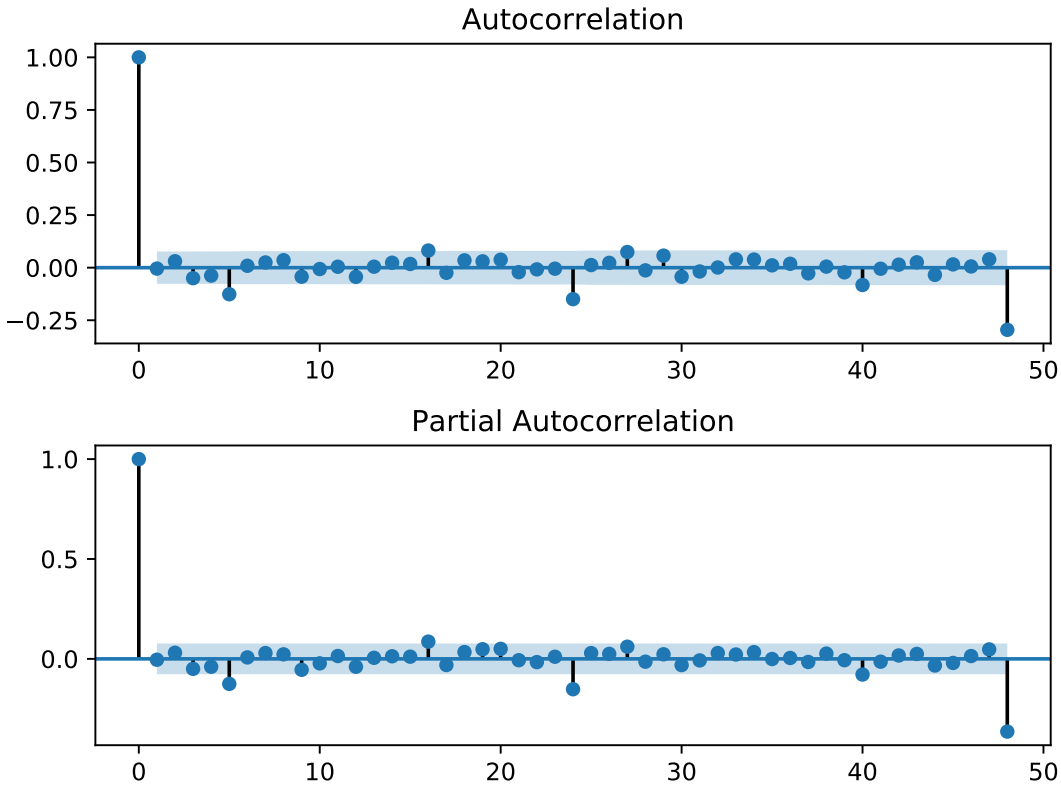

Figure 3.10: ACF and PACF fplots of the residual series between the SARIMA fit tp training data. From the plots we observe that the residuals are stationary.

Attack Detection

Here we outline sequential and supervised learning-based methods for attack detection. The sequential detections methods are the one-sided cumulative sum (CUSUM) test [102] and windowed generalized likelihood ratio test (GLRT) [103]. Both these methods take as an input a residual time series that is the output of applying a SARIMA filter to load observations. If enough past observations of load data that is labeled as nominal or under attack are collected, then detection of attacks can be made by training supervised learning classifiers. Supervised learning

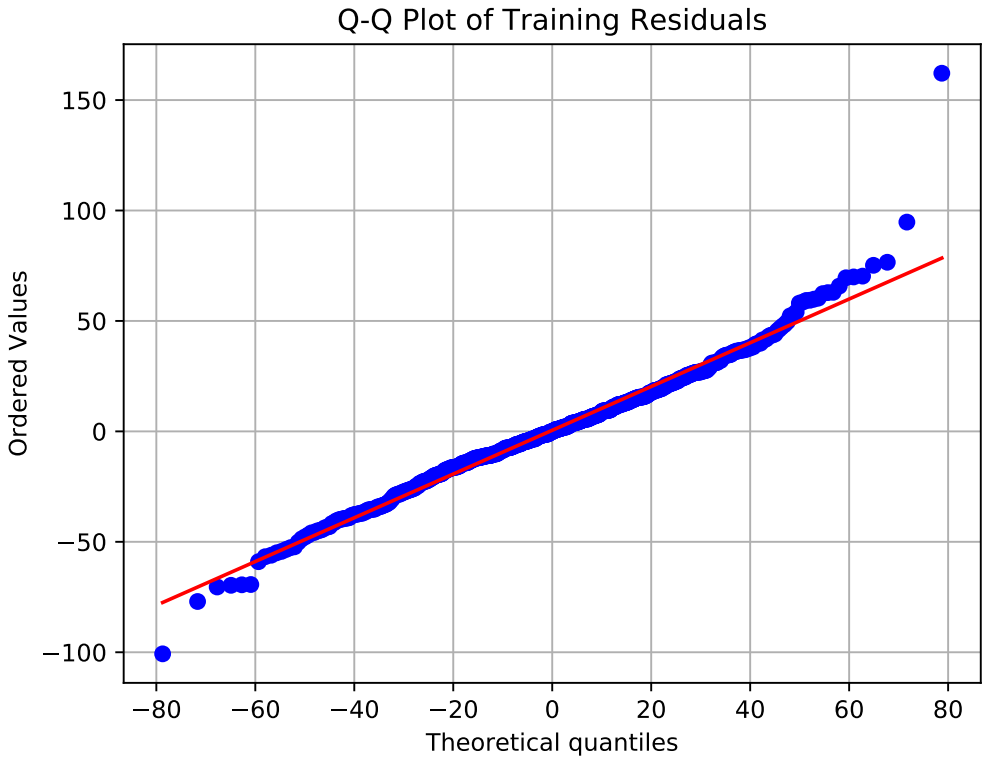

Figure 3.11: Q-Q plot of the residual series between the SARIMA fit tp training data. From the plot we observe that for extreme quantiles the distribution is not Gaussian.

algorithms have been broadly adopted to the smart grid literature for monitoring and detecting cyber attacks on power systems [104–106]. Here we employ several supervised learning methods for detecting attacks on cyber-DSM systems. It is unlikely that an ISO can collect high-quality attack data due to the lack of such attacks occurring. However, such data can be simulated. Using past load data we simulate direct load manipulation attacks by creating different types of attacks as shown in Fig. 3.8.

3.0.11 Sequential Detection Methods

In sequential change point detection, a series does not have a fixed length. Instead, observations are received and processed sequentially over time. When an observation has been received, a decision is made about whether a change has occurred in the state based only on the observations which have been received so far or within a fixed past window size. If no change is detected, then the next observation in the sequence is processed. The sequential formulation allows sequences containing multiple change points to be easily handled. Sequential change point detection can be applied in the case of attack detection to identify if a load time series has been compromised. If an attack is flagged, then an ISO can take appropriate actions to prevent further damage to the grid.

Under the Sequential Detection paradigm we collect observations an apply a whitening filter to produce a residual series with the assumption that it is white Gaussian noise. If an additive attack $A_t > 0$ is present for observation t, this will cause a definite shift in the mean of the residual series. This detection problem can thus be stated as deciding if a null hypothesis $\mathcal{H}_0$ is true, where the residual series has zero mean and known variance (invariant in time and estimated from a sample population), or if the alternative hypothesis $\mathcal{H}_1$ is true which states that the examined series has some mean not equal to zero thus being under attack. This can be modeled as a hypothesis test, and for the GLRT and CUSUM detectors this translates to

$$\mathcal{H}_0 : x_t \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2), t = 1, ..., N$$
$$\mathcal{H}_1 : x_t \overset{\text{iid}}{\sim} \mathcal{N}(A_t, \sigma^2), A_t > 0, t = 1, ..., N$$

For the GLRT detector, to simplify implementation, we model the attack as if it were constant $A > 0$ but unknown. To produce a residual series a SARIMA multi-step forecast for $t = 1, ..., N$ is made before the detection period. This forecast is conducted using a past window of training data that was not under attack. The forecast is made for time t to $t + k$. These predictions are then subtracted from the incoming observations to produce a residual time series which is then fed as input into the GLRT and CUSUM detectors.

The generalized likelihood ratio with a threshold to decide $\mathcal{H}_1$ is defined as

$$\frac{p(\mathbf{x}; A, \mathcal{H}_1)}{p(\mathbf{x}; \mathcal{H}_0)} = \frac{\frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{1}{2\sigma^2}\sum_{t=1}^{N}(x_t - A)^2}}{\frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{1}{2\sigma^2}\sum_{t=1}^{N} x_t^2}} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma$$

Taking the log of both sides simplifies the results to

$$T(\mathbf{x}) = \frac{1}{N}\sum_{t=1}^{N} x_t \underset{H_0}{\overset{H_1}{\gtrless}} \frac{\sigma^2}{NA}\ln\gamma + \frac{A}{2} = \gamma'$$

The threshold is then found by

$$P_{FA} = P(T(\mathbf{x}) > \gamma'; \mathcal{H}_0) = Q\left(\frac{\gamma'}{\sqrt{\sigma^2/N}}\right)$$
$$\Rightarrow \gamma' = \sqrt{\frac{\sigma^2}{N}} Q^{-1}(P_{FA})$$

where N is the size of our window and σ is estimated from past training data used to produce the SARIMA forecasts.

A CUSUM test is a control chart, that is used to monitor the mean of a process based on samples taken from past data at specific time intervals. It is a class of non-linear stopping rules for structural changes. Given information of current and previous samples, a CUSUM test relies on the specification of a target value h and a known or reliable estimate of the standard deviation σ the process. The CUSUM test typically signals an out of control or anomalous process by an upward or downward drift of the cumulative sum until it crosses the target threshold. For attack detection, if the mean of the load series shifts above the target threshold, we then assume the grid is under attack.

We define the CUSUM detector as follows. Taking the residual series $x_t = y_t - E_{t-1}[y_t]$, again defined by a SARIMA forecast $E_{t-1}[y_t]$, we define a one-sided CUSUM detector as

$$g_t = \max(0, g_{t-1} + x_t - k)$$

where k is called the reference value (sometimes also called drift) set priori to values such as 0, 0.5, or $A/2$ if the size of A is known in advance. When $g_t = 0$ then we

define the change time as $t_c = t$, and when $g_t > h > 0$ we reset $g_t = 0$ and flag an alarm at time $t_a = T$. The alarm threshold is also set priori to some value based on the sample population standard deviation such as $h = 2\sigma$ where σ is estimated from past training data used to produce the SARIMA forecasts.

3.0.12 Supervised Learning Methods

Changepoint detection could alternatively be treated as a supervised learning binary classification problem. Under this scheme, all of the change point sequences, or in our case attacks, represent one class, and all of the nominal sequences represents a second class. Supervised learning methods are machine learning algorithms that learn a mapping from input data to a target class label. Given a set of samples $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ and a set of labels $\mathcal{Y} = \{y_i\}_{i=1}^N$, then the supervised learning detection problem is defined as a hypothesis function that captures the relationship between samples and labels $f : \mathcal{X} \longrightarrow \mathcal{Y}$. A sliding window moves through the data, considering each difference between two data points as a possible change point.

An advantage of treating attack detection as a supervised learning approach is a more straightforward training phase. However, a sufficient amount and diversity of training data need to be provided to represent all of the classes. To ensure enough training data, and to prevent class label imbalance, we simulate all attack data to train our algorithms. Machine learning methods have successfully been applied several times in data injection attacks in power systems [105, 106], so we analyze here their ability in detecting data attacks on cyber DSM systems. The binary classification problem for attack detection can be defined as

$$y_i = \begin{cases} 1, \text{ if } A_i > 0 \\ 0, \text{ if } A_i = 0 \end{cases}$$

where $y_i = 1$ if the i-th observation is under attack, and $y_i = 0$ if there is no attack. A variety of classifiers can be used for this learning problem. For detection of DSM attacks, we examine logistic regression (LR), random forests (RF), Gaussian Naive Bayes (GNB), gradient boosting classifier (GBC), and artificial neural network (ANN). We chose these classifiers because they are all very powerful and have

widespread use in both industry and academia; model descriptions of these methods can be found in [107, 108].

3.0.13 Performance Analysis

For the security of cyber DSM systems, the major concern is not just the detection of attacks, but also the detection of nominal data with high reliability. That is, we want a detection system that can predict not only with high accuracy but also with high precision and recall to avoid false alarms. Therefore, we measure the true positives (TP), the true negatives (TN), the false positives (FP), and the false negatives (FN). Definitions of these measures are visually shown in Fig. 3.9. We use these measures to calculate several main performance indicators of accuracy, precision, and recall.

We calculate accuracy as the ratio of correctly classified data points to total data points

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

This measure provides the total classification success of the models. But alone, accuracy is not enough to get a full picture of performance. Precision is calculated as the ratio of true positive data points (attacks) to total points classified as attacks

$$Precision = \frac{TP}{TP + FP}$$

On the other hand, recall, also known as the true positive rate (TPR), refers to the portion of attacks that were recognized correctly

$$Recall = \frac{TP}{TP + FN}$$

Precision values give information about the prediction performance of the algorithms, whereas recall values measure the degree of attack retrieval. For instance, a recall value equal to 1 signifies that none of the attacked measurements were misclassified as nominal.

We use one more final measure of total performance of our detectors, the receiver operating characteristics (ROC) curve. The ROC curve is an assessment that enables

visual analysis of the trade-off between TPR and false positive rate (FPR). This can also be seen as the trade-off between the probability of detection and the probability of false alarm. FPR is defined as follows

$$FPR = \frac{FP}{FP + TN}.$$

The ROC curve is constructed by plotting a two-dimensional graph with FPR on the x-axis and TPR on the y-axis at various threshold settings. A detection algorithm produces a (TPR, FPR) pair that corresponds to a single point in the ROC space. The best possible detection method would produce a point in the upper left corner, coordinate (0,1) of the ROC space. A random prediction would give a point along a diagonal line from (0,0) to (1,1). Points above this line are considered to have performance, while points below are considered with performance worse than guessing.

Detection Experiments

For attack simulation and detection experiments we use all the same parameters from Table 3.1 but we varied the levels of κ and attacks a_t. We simulate each of the three types of attacks as visualized in Fig. 3.8, each in the form of direct load manipulation attack, under DSM participation levels $\kappa = 0.1$ and 0.9. This creates a total of six experiment scenarios. For each scenario, we simulate 28 days of training data (672 observations) and 2 proceeding days of test data (48 observations), both at a resolution of 1 hr. All training data is created nominally with no attacks. In the test data sets, the first 24 hrs of the test data are not under attack. The last 24 hours of the test data we add one of the three type of attacks. For ramp attacks, for each time step we add an attack $a_t = a_{t-1} + 5$ where the first 24 hours of the test set $a_{t=0\ldots23} = 0$. The sudden attacks are similar except the attack is an additive constant $a_t = 150$. The last type of attack, point attacks, are $a_{t=24} = 250, a_{t=29} = 200, a_{t=34} = 300, a_{t=37} = 100, a_{t=46} = 150$, and $a_t = 0$ everywhere else.

For the sequential detectors, we use multi-step SARIMA forecasts to predict the

next 48 hrs, and then use those forecasts to filter incoming test data for detection. Training was conducted on the training time series of 28 days of nominal data. SARIMA hyperparameters were chosen by examining lag one differenced autocorrelation function (ACF) and partial autocorrelation function (PACF) plots of the training data. These forecasts where then subtracted from the incoming test data to obtain a residual series that is input to the sequential detectors. The assumption for both detectors is that the residual series is Gaussian white noise, where the series has zero mean, and each observation is independent identically distributed from a Gaussian distribution.

To ensure the residuals are white noise we apply the augmented Dickey-Fuller (ADF) test to check for stationarity, we examine the ACF/PACF plots to check for independence and run a Jarque-Bera test, and we examine a Q-Q plot to check if the residual series has a Gaussian probability distribution. We conduct these checks on all scenario training datasets, where we first train a SARIMA fit on them and then subtract that fit to produce the residual series. In all the training data sets, the residuals were proven to be stationary from the ADF test and independent from ACF/PACF plots. However, the Jarque-Bera test and Q-Q plots showed that the observations did not come from a Gaussian distribution. Example, ACF/PACF and Q-Q plots for $\kappa = 0.1$ are shown in Fig. 3.10 and Fig. 3.11. Despite the residuals not being Gaussian, we still run the sequential detectors and examine their performance.

For training the supervised learning methods, for each of the six scenarios, we simulate more data to ensure proper class learning. We keep the test sets the same, but we extend each training set doubling its size. We keep the original training set, labeling it as nominal, and then make a copy of it. In the copy, we split it into three parts, each part we add one of the types of attacks at random levels. We label each observation of this set as under attack. We add the nominal and attacked training sets together to form a new training set with a total of 1,344 observations. With our training and test time series, we then create a set $\mathcal{X}$ features and $\mathcal{Y}$ labels that could be fed into the supervised learning classifiers. Each training sample $x_i \in \mathcal{X}$ is composed of 24 hours of lagged data, each hour is one feature, and each label $y_i \in \mathcal{Y}$ corresponds to a class 0 (not under attack) and class 1 (under attack). Together we

get $N = 1,298$ pairs of (x_i, y_i) training samples.

We run our experiments on a computer with an Intel i7 6700 2.6 GHz, and 16 GB of RAM. Implementation of the simulations, experiments, and sequential detectors were done in Python 3.6. Implementation for SARIMA forecasts was done using the Python package Statsmodels [109], the supervised learning methods and confusion evaluation metrics were implemented using the Scikit-Learn Python package [110], and ROC curves were created using the Matplotlib Python package [111]. All classifiers used default hyperparameters from Scikit-Learn. We compiled all code and data used in our work, into a Python package titled LehighDSM which is publicly available on GitHub [112].

Experimental results from the six scenarios are reported in Table 3.2, in the form of accuracy, recall, and precision metrics, and Fig. **??** which showcases six ROC curves, one for each scenario. All performance measures in Table 3.2, were calculated using the best thresholds found in the ROC curves by searching for the shortest distance from each curve to the corner (0,1). As seen in Fig. **??**(c,f) type 3 attacks, under any DSM level, had the lowest detection rates across thresholds with ANN being the worst performer. In Fig. **??**(a,d) LR yielded the best performance with GNB, and again ANN being the worst. Type 2 attacks in Fig. **??**(b,e) resulted in the best detection with CUSUM, LR, GNB and GBC having perfect performance.

With the results from Table 3.2, we have the following conclusions; demonstrated findings imply that detection of attacks has a higher accuracy with higher levels of DSM participation. This occurs since higher DSM penetration results in more aggregated attacks. However, we note that an attacker, if having perfect knowledge of the grid could decrease their intensity and make detection more difficult. Furthermore, supervised learning classifiers performance on average was on par or better than sequential detection methods. LR detector had the highest accuracy for a lower level of DSM usage, while ANNs performed the worst in all scenarios. This highlights the power of linear detection methods over nonlinear. Point attacks resulted in the poorest detection with the CUSUM detector having the best performance. Type 2 attacks had higher recall and precision across all detectors for both levels of DSM. This is because a sudden attack shifts the mean of the data as a constant over time,

Table 3.2: Evaluation metrics for attack detection.

	κ	0.1			0.9		
	Attack Type	1	2	3	1	2	3
Accuracy	LR	**89.6**	**100.0**	87.5	87.5	**100.0**	75.0
	RF	77.1	97.9	64.6	77.1	93.8	60.4
	GNB	70.8	77.1	75.0	79.2	**100.0**	72.9
	GBC	85.4	97.9	66.7	85.4	**100.0**	64.6
	ANN	56.3	95.8	20.8	75.0	79.2	14.6
	GLRT	79.2	95.8	58.3	**91.7**	97.9	58.3
	CUSUM	83.3	95.8	**95.8**	87.5	**100.0**	**100.0**
Recall	LR	**92.0**	**100.0**	80.0	**89.0**	**100.0**	80.0
	RF	56.0	96.0	80.0	56.0	88.0	80.0
	GNB	88.0	56.0	80.0	60.0	**100.0**	80.0
	GBC	76.0	96.0	**100.0**	88.0	**100.0**	80.0
	ANN	68.0	92.0	60.0	76.0	68.0	80.0
	GLRT	76.0	92.0	80.0	84.0	96.0	80.0
	CUSUM	80.0	96.0	**100.0**	84.0	**100.0**	**100.0**
Precision	LR	88.5	**100.0**	44.4	88.0	**100.0**	26.7
	RF	**100.0**	**100.0**	20.0	**100.0**	**100.0**	18.2
	GNB	66.7	**100.0**	26.7	**100.0**	**100.0**	25.0
	GBC	95.0	**100.0**	23.8	84.0	**100.0**	20.0
	ANN	56.7	**100.0**	7.7	76.0	89.5	9.1
	GLRT	82.6	**100.0**	17.4	**100.0**	**100.0**	17.4
	CUSUM	87.0	96.0	**71.4**	91.3	**100.0**	**100.0**

which is more identifiable, yielding fewer false positives and more true positives.

Conclusion

In this work, we study the exploitation of the hypothetical premise of the feedback between future cyber-enabled DSM programs, on the consumer side, and dynamic RTP on the utility side. An attacker with exploitive economic or nefarious

intentions, such as causing efficiency loss of energy provision, can take advantage of the dependency between dynamic pricing and DSM load control. The utility modifies prices in response to forecasted demand in order to push realized load up or down. This is done to achieve some target load level with the goal to reduce peak load or achieve some other DSM objective. On the user side, cyber DSM programs then autonomously respond to prices to adjust certain portions of a users load up or down with some given elasticity.

We propose two modes of attacks, false price data injections, and direct load manipulation. Under a false price data injection, an attacker modifies the RTP that users receive to alter their demand. Through a direct load manipulation attack, an attacker hacks and alters a users load profile directly. In both these attacks, aggregate load from the grid is modified which then can alter future prices or demand. We showcase how these two modes of attacks are equivalent and introduce three ways an attack can occur. The first type is a ramp attack, the second is a sudden attack, and the third is a point attack. We simulate these type of attacks and review several methods to detect them.

We simulate and examine load-price data under different levels of DSM participation with three types of additive attacks: ramp, sudden, and point attacks. We applied sequential change point and supervised learning methods for detection of DSM attacks. Results conclude that higher amounts of DSM participation can exacerbate attacks but also lead to better detection of such attacks, point attacks are the hardest to detect, and supervised learning methods produce results on par or better than sequential detectors. Examining these detection methods, we conclude that linear methods such as logistic regression resulted in better detection of attacks then nonlinear methods such as deep learning. Additionally, linear SARIMA forecasts as part of the change point detectors also yielded acceptable results.

This cyber-enabled DSM domain is an excellent example of a problem where linear methods are better than nonlinear ones thus the need for deep learning or advanced probabilistic forecasting is not warranted. However, we hypothesize that when renewable energy generation is introduced into this problem, the detection of attacks can become much more difficult. Renewable generation can bring a lot of

uncertainty, and an attacker can exploit this uncertainty and amplify in the DSM problem.

The following chapters thus examine various new probabilistic forecasting methods hybridized with deep learning to predict highly chaotic and nonstationary renewables such as wind energy. We propose several novel architectures including support vector machines and neural networks for the forecasting problem. In future work, we plan to expand the DSM problem to include renewables and utilize probabilistic forecasting methods for the detection of attacks.

Chapter 4

Constrained Support Vector Quantile Regression

Introduction

Predicting and managing uncertainty in the production of wind power is one of the biggest challenges facing its integration into the smart grid. Forecasting uncertainty in wind is needed for many operational applications in a wind farm from turbine and storage control to bidding and trading in energy markets. Forecasting horizons can be categorized into three main time scales: short-term looking out several hours or days, long-term looking out to weeks or a month, and seasonal. Traditionally wind power prediction is based on deterministic point forecasts where they provide an expected output for a given look-ahead time. These forecasts however lack uncertainty information. As such a large research effort has been taken recently by the renewables forecasting community [8] to produce full probabilistic predictions which derive quantitative information on the associated uncertainty of power output. Although various methods have been proposed, it is still a challenge to make accurate and robust probabilistic predictions for highly nonlinear and complex data, such as wind.

Probabilistic wind models are based on either meteorological ensembles that are

obtained by a weather model [10] or on statistical learning methods [11]. Focusing on statistical learning, these methods can be applied to forecast full predictive distributions in the form of quantiles or prediction intervals. For instance, in [113] prediction intervals are estimated by adaptive re-sampling which is a common probabilistic forecasting strategy. Quantile regression (QR) is another very popular approach. In [26] local QR is applied to estimate different quantiles while In [27] spline based QR is used to estimate quantiles of wind power. In [28] quantile loss gradient boosted machines are used to estimate 99 quantiles and in [29] multiple quantile regression is used to predict a full distribution with optimization done using the alternating direction method of multipliers. Quantile regression forests [30] have also been applied in forecasting which are an extension of regression forests based on classification and regression trees. A thorough overview of probabilistic wind power forecasting is provided in [17].

In most of these approaches, estimation of each quantile is conducted independently. This could lead to the quantile cross over problem where a lower quantile overlaps a higher one. This is undesirable as it violates the principle of distribution functions where their associated inverse functions should be monotone increasing. A way to prevent this issue is to utilize a simple heuristic of reordering estimated quantiles, however this does not have much theoretical basis and may lead to inappropriate quantiles.

The solution then is to optimize quantiles together with non-crossing constraints. In [114] a constrained support vector quantile regression (CSVQR) method was developed with non-crossing constraints where it was used to fit quantiles on static data. This formulation is re-purposed here for probabilistic forecasting. Other machine learning frameworks have been used before for uncertainty prediction of renewables such as nearest neighbors [115], neural networks [12], and extreme learning machines [116] but support vector machines (SVMs) have yet to be examined for wind uncertainty forecasting. We propose that SVMs are not only effective in long term prediction due to their ability to handle nonlinear data via kernels but can be easily extended with constraints to ensure non-overlapping quantile estimates. Our study is the first to showcase the use of CSVQR with a sliding window of

training data as well as showcase the effectiveness of constraints to ensure monotonically increasing quantiles for probabilistic prediction. We provide the derivation of CSVQR and analysis of experimental results on publicly available wind data. Several common benchmark methods are used for comparison.

Support Vector Quantile Regression

The proposed approach of support vector quantile regression for nonparametric probabilistic forecasting is directly related to the derivation of support vector regression (SVR). The goal with SVR is to find

$$f(x_i) = w \cdot x_i + b \tag{4.1}$$

that has at most ε deviation from the target y_i for all training data. To do this and ensure w is small the Euclidean norm is minimized

$$\min_{w} \frac{1}{2}\|w\|^2 \tag{4.2}$$

$$\text{subject to } \begin{cases} y_i - f(x_i) \leq \varepsilon \\ f(x_i) - y_i \leq \varepsilon \end{cases}$$

The assumption in (4.2) is that such a function f actually exists and can approximate all pairs (x_i, y_i) with ε precision, or in other words that the convex optimization problem is feasible. Sometimes this may not be the case or one may also want to allow for some errors. In that instance slack variables ξ_i^- and ξ_i^+ can be introduced to deal with infeasible constraints of the optimization problem

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i^- + \xi_i^+) \tag{4.3}$$

$$\text{subject to } \begin{cases} y_i - f(x_i) \leq \varepsilon + \xi_i^- & \forall_i \\ f(x_i) - y_i \leq \varepsilon + \xi_i^+ & \forall_i \\ \xi_i^-, \xi_i^+ \geq 0 & \forall_i \end{cases}$$

The constant $C > 0$ determines the trade off between the flatness of f and the amount up to which deviations larger than ε are tolerated.

4.0.1 Nonlinear Quantile Regression

To fit the nonlinearity of wind data, nonlinear quantile regression (NQR) can be utilized. NQR is implemented by projecting an input vector x into a potentially higher dimensional feature space $\mathcal{F}$ using a nonlinear mapping function $\phi(\cdot)$ implicitly defined by a kernel K. This gives the functional form of

$$f_\tau(x) = w_\tau^\top \phi(x)$$

where f_τ is the τ-th quantile of the distribution of y conditional on the values of x, w_τ is a vector of parameters. The NQR simplifies into linear quantile regression if $\phi(x) = x$. To solve the NQR problem it can be expressed by the following formulation with added L_2 penalty to prevent overfitting

$$\min_{w_\tau} \frac{1}{2}\|w_\tau\|^2 + C\sum_{i=1}^{N} \rho_\tau(y_i - f_\tau(x_i))$$

By introducing slack variables ξ_i^- and $\xi_i^+$the problem can be re-written as a support vector quantile regression problem

$$\min_{w,b,\xi^-,\xi^+} \frac{1}{2}\|w_\tau\|^2 + C\sum_{i=1}^{N}(\tau\xi_i^+ + (1-\tau)\xi_i^-) \tag{4.4}$$

$$\text{s.t.} \begin{cases} y_i - w_\tau^\top \phi(x_i) - \xi_i^+ \leq 0 \\ -y_i + w_\tau^\top \phi(x_i) - \xi_i^- \leq 0 \\ \xi_i^-, \xi_i^+ \geq 0 \end{cases}$$

This form is the support vector quantile regression primal.

4.0.2 SVQR Dual Formulation

The optimization problem in (4.4) can be solved more easily in its dual form. The dual also provides the key for extending support vector machines to nonlinear functions, and is done by using the standard dualization method utilizing Lagrange

multipliers. The main idea is to construct a Lagrange function from both the primal formulation and the corresponding constraints by introducing a dual set of variables. The Lagrangian is then defined as

$$\begin{aligned} L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\tau\xi_i^+ + (1-\tau)\xi_i^-) \\ + \sum_{i=1}^{N}\alpha_i^+(u_i - \xi_i^+) + \sum_{i=1}^{N}\alpha_i^-(-u_i - \xi_i^-) \\ - \sum_{i=1}^{N}(\eta_i^+\xi_i^+ + \eta_i^-\xi_i^-) \end{aligned} \tag{4.5}$$

where $\alpha_i^+, \alpha_i^-, \eta_i^+$, and η_i^1, $\forall_i$ are the Lagrange multipliers (dual variables) having positivity constraint $\alpha_i^+, \alpha_i^-, \eta_i^+, \eta_i^1 \geq 0$. It follows from the saddle point condition that the partial derivatives of L with respect to the primal variables (w, b, ξ_i^+, ξ_i^-) have to be zero for optimality

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-)\phi(x_i) = 0 \tag{4.6}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-) = 0 \tag{4.7}$$

$$\frac{\partial L}{\partial \xi_i^+} = \tau C - \alpha_i^+ - \eta_i^+ = 0 \tag{4.8}$$

$$\frac{\partial L}{\partial \xi_i^-} = (1-\tau)C - \alpha_i^- - \eta_i^- = 0 \tag{4.9}$$

Substituting Eq. (4.6) to Eq. (4.9) into Eq. (4.5) yields the dual minimization

optimization problem

$$\min_{\alpha_i^+,\alpha_i^-} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)K(x_i,x_j) - \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-)y_i$$
$$\text{subject to} \begin{cases} \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-) = 0 \\ \alpha_i^+ \in [0, \tau C], \forall_i \\ \alpha_i^- \in [0, (1-\tau)C], \forall_i \end{cases} \tag{4.10}$$

where $K(x_i, x_j)$ is a kernel function in the input space and equal to the inner product of vector x_i and x_j in the feature space, i.e. $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$. Eq. (**??**) can then be rewritten as

$$f(x) = \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-)K(x, x_i) \tag{4.11}$$

4.0.3 Non-crossing Quantile Constraints

In Eq. (4.4) a single quantile is estimated. To estimate multiple quantiles this formulation could be run to solve for different τ's independently. However in doing so quantiles may cross each other which is not desirable since it violates the principle of monotone increasing inverse density functions. To prevent this, constraints need to be introduced [114]. $0 < \tau_1 < ... < \tau_M$ are defined as the orders of M conditional quantiles to be estimated. To ensure these quantiles do not cross each other the following constraint is needed $f_1(x_i) \leq ... \leq f_M(x_i), \forall_i$. With this constraint the primal problem of the non-crossing conditional quantile estimator is given by

$$\min_{w,\xi^-,\xi^+} \sum_{m=1}^{M}\left(\frac{1}{2}\|w_m\|^2 + C\sum_{i=1}^{N}(\tau_m\xi_{mi}^+ + (1-\tau_m)\xi_{mi}^-)\right) \tag{4.12}$$

$$\text{s.t.} \begin{cases} y_i - w_m^\top \phi(x_i) - \xi_{mi}^+ \leq 0, & \forall_m, \forall_i \\ -y_i + w_m^\top \phi(x_i) - \xi_{mi}^- \leq 0, & \forall_m, \forall_i \\ \xi_{mi}^-, \xi_{mi}^+ \geq 0, & \forall_m, \forall_i \\ w_m^\top \phi(x_i) - w_{m+1}^\top \phi(x_i) \leq 0, & \forall_m, \forall_i \end{cases}$$

The Lagrangian for the problem is then defined by

$$\begin{aligned} L = \sum_{m=1}^{M} \Bigg(& \frac{1}{2}\|w_m\|^2 + C \sum_{i=1}^{N} (\tau_m \xi_{mi}^+ + (1-\tau_m)\xi_{mi}^-) \\ & + \sum_{i=1}^{N} \alpha_{mi}^+ (y_i - w_m^\top \phi(x_i) - \xi_{mi}^+) \\ & + \sum_{i=1}^{N} \alpha_i^- (-y_i + w_m^\top \phi(x_i) - \xi_{mi}^-) \\ & - \sum_{i=1}^{N} (\eta_{mi}^+ \xi_{mi}^+ + \eta_{mi}^- \xi_{mi}^-) \Bigg) \\ & + \sum_{m=1}^{M} \sum_{i=1}^{N} \lambda_{mi} \left(w_m^\top \phi(x_i) - w_{m+1}^\top \phi(x_i) \right) \end{aligned} \tag{4.13}$$

where a Lagrange multiplier $\lambda_{mi} \geq 0$ is introduced for $m = 1, ..., M-1$, $\forall_i$, and $\lambda_{0i} = \lambda_{Mi} = 0$. By letting the partial derivatives of L with respect to w_m be zero, the following is obtained

$$\begin{aligned} \frac{\partial L}{\partial w_m} &= w_m - \sum_{i=1}^{N} (\alpha_{mi}^+ - \alpha_{mi}^-)\phi(x_i) \\ &+ \sum_{i=1}^{N} (\lambda_{mi} - \lambda_{m-1i})\phi(x_i) = 0 \end{aligned} \tag{4.14}$$

Partial derivatives of the other primal variables ξ_{mi}^+ and ξ_{mi}^- are

$$\frac{\partial L}{\partial \xi_{mi}^+} = \tau_m C - \alpha_{mi}^+ - \eta_{mi}^+ = 0 \tag{4.15}$$

$$\frac{\partial L}{\partial \xi_{mi}^-} = (1-\tau_m) C - \alpha_{mi}^- - \eta_{mi}^- = 0 \tag{4.16}$$

Plugging these equalities back into Eq. (4.13) the following dual minimization problem can be obtained

$$\begin{aligned}
&\min_{\alpha^+,\alpha^-,\lambda} \sum_{m=1}^{M} \Bigg(-\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_{mi}^+ - \alpha_{mi}^-)(\alpha_{mj}^+ - \alpha_{mj}^-)... \\
&K(x_i, x_j) + \sum_{i=1}^{N} (\alpha_{mi}^+ - \alpha_{mi}^-) y_i \\
&- \frac{1}{2} \sum_{i=1}^{N} \sum_{i=j}^{N} (\lambda_{mi} - \lambda_{m-1i})(\lambda_{mj} - \lambda_{m-1j}) K(x_i, x_j) \\
&+ \sum_{i=1}^{N} \sum_{i=j}^{N} (\alpha_{mi}^+ - \alpha_{mi}^-)(\lambda_{mj} - \lambda_{m-1j}) K(x_i, x_j) \Bigg)
\end{aligned} \tag{4.17}$$

$$\text{subject to} \begin{cases} \lambda_{mi} \geq 0, \forall_m \forall_i \\ \alpha_{mi}^+ \in [0, \tau_m C], \forall_m \forall_i \\ \alpha_{mi}^- \in [0, (1 - \tau_m) C], \forall_m \forall_i \end{cases}$$

From this dual formulation the conditional quantile τ_m can then be given by

$$\begin{aligned}
f_{\tau_m}(x) &= \sum_{i=1}^{N} (\alpha_{mi}^+ - \alpha_{mi}^-) K(x, x_i) \\
&- \sum_{i=1}^{N} (\lambda_{mi} - \lambda_{m-1i}) K(x, x_i)
\end{aligned} \tag{4.18}$$

Since the dual form is a Quadratic Programming (QP) problem it can be solved by a number of QP methods. For testing the Constrained SVQR (CSVQR) method the Radial Basis Function (RBF) kernel is utilized as it is a popular kernel function choice for support vector machines. Other kernels were tested on the case data sets described in the next section but resulted in poor results. The RBF kernel, given two samples $\mathbf{x}$ and $\mathbf{x}'$ which are represented as feature vectors, is calculated as

$$K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2} \right)$$

An advantage of a RBF kernel is that it can project vectors into an infinite dimensional feature space. In order to quickly solve for conditional quantile estimates sequential minimization optimization [117] is applied to Eq. (4.17).

Application To The GEFCom2014 Dataset

Data for this case study comes from the publicly available Global Energy Forecasting Competition 2014 [8]. The goal of the competition was to design parametric or nonparametric forecasting methods that would allow conditional predictive densities of the wind power generation to be described as a function of input data which were future weather forecasts and/or past wind power. Data is provided for the years of 2012 and 2013 from 10 wind farms titled Zone 1 to Zone 10. The predictors are numerical weather predictions (NWPs) in the form of wind speeds at an hourly resolution at two heights, 10m and 100m above ground level. These forecasts are for the zonal and meridional wind components (denoted U and V). It was up to users to deduce exact wind speed, direction, and other wind features if necessary. These NWPs were provided for the exact locations of the wind farms. Additionally, power measurements at the various wind farms, with an hourly resolution, are also provided. All power measurements are normalized by the nominal capacity of their wind farm. The goal in forecasting was to learn to associate the provided NWPs (or derived features) with wind power. Then NWPs are provided for the forecasting horizon of one month and it is up to a learning model to use those NWPs as input to a learning model to predict quantiles at each future time step. Fig. 4.1 showcases an example month worth of data where Fig. 4.1.a shows the four NWP given and Fig. 4.1.b shows their corresponding normalized wind power output.

In our analysis of CSVQR we used the summer months of June 2013 to August 2013 and fall months of September 2013 to November 2013 for testing from Zone 1. Training was done using a sliding window of three previous months to forecast the fourth month. For instance to predict June training was done on observed data from March to May, then to predict July training was done from April to June, etc. Thirteen features were derived from the raw data for training the CSVQR model. Features used are derived wind speeds at 10m and 100m, wind direction at 10m and 100m, wind energy at 10m and 100m, wind shear, wind energy difference

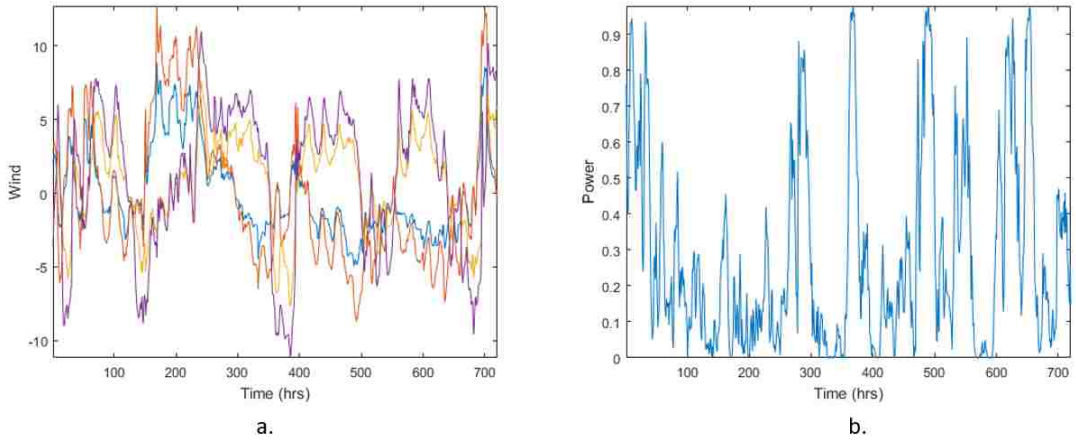

Figure 4.1: (a) Example plot of numerical wind predictions at 10m and 100m for U and V directions used as inputs to forecast wind power. (b) Observed wind power corresponding to the same time stamps.

(between 10m and 100m), wind direction difference (between 10m and 100m), and included in training are also the four raw wind speeds at 10m and 100m for U and V directions. All features were normalized between 0 and 1. Denoting u and v as the wind components and d as the energy density (we used $d = 1$), the equations used to compute wind speed (ws), wind direction (wd), wind energy (we), and wind shear (wsh) are

$$\begin{aligned} \text{ws} &= \sqrt{u^2 + v^2} \\ \text{wd} &= \frac{180}{\pi} \times \arctan(u, v) \\ \text{we} &= \frac{1}{2} \times d \times \text{ws}^3 \\ \text{wsh} &= \sqrt{\text{ws10}^2 + \text{ws100}^2} \end{aligned}$$

To empirically analyze the CSVQR model as an appropriate method for wind forecasting it is compared with two industry models and a naive model that are used for benchmarking in probabilistic wind forecasting applications [12, 23, 118]. The first is called the persistence method which is the most common benchmark and is considered difficult to outperform for short-term forecasting. This method

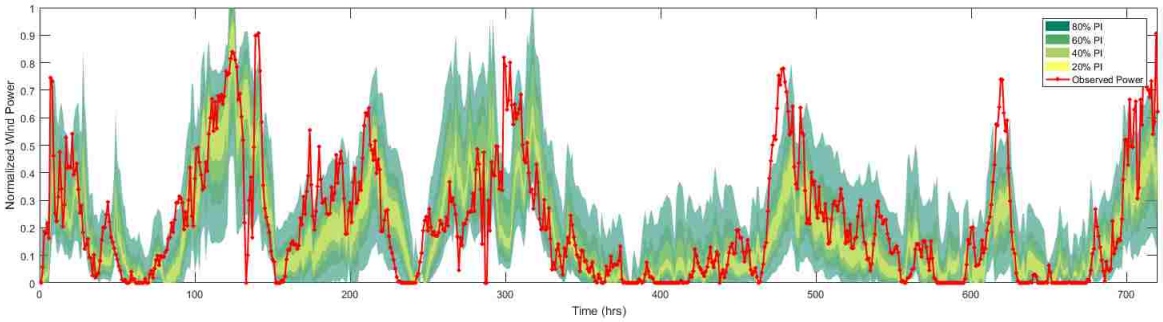

Figure 4.2: Example plot of estimated 80%, 60%, 40%, and 20% prediction intervals along with observed wind power in red for the month of July 2013.

corresponds to the persistence distribution and is formed by the most recent observations. For this case study, the past 12 hours of wind power observations were used to form the persistence distribution. Second method is the climatology approach where its predictive distribution is unconditional and based on all available past wind power observations. It is considered harder to beat in long-term forecasting. Lastly, the uniform distribution is used for a naive benchmark method where it assumes all wind power values at each time step occur with equal probability.

4.0.4 Results

To visualize a probabilistic forecast Fig. 4.2 shows an example prediction for 80%, 60% 40%, and 20% prediction intervals for the month of July 2013. Observed wind power is shown in red. From such probabilistic forecasts it is then possible to derive full predictive density functions following that the estimated conditional quantiles are nondecreasing [119]. Evaluation results for reliability of probabilistic forecasts in the form of prediction intervals of wind power over the months of June 2013 to November 2013 is shown in Table 1. Results are shown for the CSVQR method and for the climatology, persistence, and uniform benchmark methods. Evaluation metrics for the PINC are the PICP and ACE.

For the month of June and October, the climatology method was slightly better but this was due to the fact that this model can yield wide intervals to cover more

data. However in all other months CSVQR outperformed all three benchmarks by several magnitudes. To further fully evaluate the forecasts it is also important to look at the quantile score to measure the coverage of the estimated quantiles. Table 2 shows the summary of Q-scores averaged across all quantiles from all lookahead periods for every forecast month. Their standard deviation is also provided to quantify the amount of variation among the quantiles. The Q-scores of the proposed approach was very low and gave excellent probabilistic forecasts across all different months.

Conclusion

Wind power forecasting is crucial for many decision making problems in power systems operations, and is a vital component in integrating more wind into the power grid. Due to the chaotic nature of the wind it is often difficult to forecast. Uncertainty analysis in the form of probabilistic wind prediction can provide a better picture of future wind coverage. This work studies a framework for probabilistic forecasting using support vector quantile regression with non-crossing constraints to ensure multiple quantiles can be predicted without overlapping each other. Effectiveness of the CSVQR approach is validated with the real world dataset of the Global Energy Forecasting Competition 2014. Forecasts are compared to common benchmarks and are evaluated using the quantile score and reliability metrics. Results show adequate reliability and low quantile scores across the prediction horizon, which verify effectiveness of the model for forecasting while preventing estimated quantiles from overlapping. Furthermore, this approach has the potential to be applied across a variety of domains. Future work will look into applying CSVQR to forecast electricity pricing and load demand for smart grid applications.

Month	PINC	CSVQR		Climatology		Persistence		Uniform	
		PICP	ACE	PICP	ACE	PICP	ACE	PICP	ACE
June 13	80%	**85.00**	**5.00**	95.28	15.28	46.11	33.89	60.97	19.03
	60%	66.25	6.25	**62.50**	**2.50**	37.64	22.36	40.97	19.03
	40%	45.56	5.56	**42.92**	**2.92**	30.56	9.44	23.47	16.53
	20%	25.42	5.42	**22.64**	**2.64**	26.30	6.31	10.69	9.31
July 13	80%	**78.49**	**1.50**	76.08	3.92	12.77	67.23	59.27	20.73
	60%	**56.04**	**3.95**	55.38	4.62	6.72	53.28	36.96	23.04
	40%	**38.70**	**1.29**	35.08	4.92	5.24	34.76	21.91	18.09
	20%	**20.96**	**0.96**	16.80	3.20	2.55	17.45	10.08	9.92
August 13	80%	**78.36**	**1.64**	65.73	14.27	22.04	57.96	61.83	18.17
	60%	**59.27**	**0.73**	42.61	17.39	13.44	46.56	44.49	15.51
	40%	**40.46**	**0.46**	25.94	14.06	7.80	32.20	30.11	9.89
	20%	**19.89**	**0.11**	9.95	10.05	4.57	15.43	15.05	4.95
September 13	80%	**79.03**	**0.97**	81.81	1.81	31.53	48.47	60.69	19.31
	60%	**60.69**	**0.69**	59.30	0.70	23.75	36.25	35.56	24.44
	40%	**42.92**	**2.92**	34.31	5.69	14.86	25.14	20.97	19.03
	20%	**22.36**	**2.36**	15.83	4.17	5.97	14.03	9.31	10.69
October 13	80%	83.20	3.20	**81.85**	**1.85**	52.82	27.18	62.77	17.23
	60%	68.15	8.15	**62.77**	**2.77**	23.92	36.08	45.70	14.30
	40%	52.55	12.55	**46.24**	**6.24**	6.85	33.15	28.76	11.24
	20%	**24.36**	**4.36**	25.27	5.27	1.88	18.12	16.67	3.33
November 13	80%	**80.42**	**0.42**	90.14	10.14	25.83	54.17	72.36	7.64
	60%	**59.31**	**0.69**	75.00	15.00	15.14	44.86	48.75	11.25
	40%	**36.11**	**3.89**	55.69	15.69	11.94	28.06	29.17	10.83
	20%	**16.53**	**3.47**	29.03	9.03	10.42	9.58	13.19	6.81

Table 4.1: Results of prediction interval reliability in different months.

Month	Method	Q-Score	SD
June 13	CSVQR	**0.0404**	**0.0119**
	Climatology	0.0628	0.0230
	Persistence	0.0880	0.0406
	Uniform	0.1105	0.0434
July 13	CSVQR	**0.0546**	**0.0169**
	Climatology	0.1038	0.0401
	Persistence	0.1799	0.0681
	Uniform	0.1112	0.0428
August 13	CSVQR	**0.0677**	**0.0199**
	Climatology	0.1374	0.0555
	Persistence	0.1734	0.0738
	Uniform	0.1033	0.0380
September 13	CSVQR	**0.0590**	**0.0172**
	Climatology	0.0992	0.0401
	Persistence	0.1659	0.0582
	Uniform	0.1107	0.0429
October 13	CSVQR	**0.0561**	**0.0159**
	Climatology	0.0971	0.0366
	Persistence	0.1807	0.0977
	Uniform	0.1033	0.0382
November 13	CSVQR	**0.0557**	**0.0186**
	Climatology	0.0844	0.0396
	Persistence	0.1089	0.0533
	Uniform	0.0978	0.0406
All	CSVQR	**0.0556**	**0.0167**
	Climatology	0.0974	0.0391
	Persistence	0.1494	0.1261
	Uniform	0.1061	0.0409

Table 4.2: Summary of the mean Q-score across all quantiles for a given method and month and their standard deviation.

Chapter 5

Smooth Pinball based Composite Quantile Neural Network

Introduction

In the last thirty years wind power has experienced rapid global growth, and in some countries, it is the most used form of renewable energy. However, due to the chaotic nature of the weather, variable and uncertain wind power production poses planning and operational challenges unseen in conventional generation. From the grid operator's perspective, uncertainty in wind production could cause inefficiencies in the power flow, operating reserve requirements, stochastic unit commitment, and electricity market settlements [120–122]. From the wind generator's perspective, reliable wind forecasts are needed for several operations at a wind farm, ranging from energy storage control to bidding and trading in energy markets. Thus, to ensure both stable grid operations and continued growth and increased penetration of wind power, highly reliable forecasting of wind power production is needed.

Traditionally wind power prediction has focused on developing point forecasts which provide a single expected output for a given look-ahead time. Point forecasting horizons fall into several scales: very short-term (seconds or minutes ahead),

short-term (hours to days ahead), long-term (weeks or months ahead), and seasonal. A thorough review in wind forecasting can be found in [123]. However, point forecasting can result in certain errors which can be significant and they also lack information on uncertainty. Therefore, a significant research effort has begun recently by the renewables forecasting community [8] to produce fully probabilistic predictions which derive quantitative information on the associated uncertainty of power output. For example, to capture the uncertainty of wind power, forecasting errors can be statistically analyzed and modeled by the Beta distribution. However, such assumption may not be applicable for short-term forecasting, and thus researchers are looking at different approaches for probabilistic wind power forecasts by quantifying prediction uncertainty. Although there are various methods proposed, it is still a challenge to make accurate and reliable probabilistic predictions for volatile renewables, such as wind.

Probabilistic forecast can play a key role in integrating and managing wind farms. For instance, in [124] the optimal level of generation reserves is estimated using the uncertainty of wind power predictions, and in [125, 126] the optimization of wind energy production is investigated taking into account the forecasts of a probabilistic prediction method. Additionally, increased revenues can be obtained using bidding strategies built on predictive densities, as shown in [127, 128]. Wind power density forecasting can be used for analysis of probabilistic load flow, as in [129]. Machine learning frameworks such as nearest neighbors [115], neural networks [12], and extreme learning machines [116] are also noted approaches for uncertainty prediction.

Our work is motivated by exploring a direct and nonparametric probabilistic forecasting approach for wind power. To address the problem of dealing with nonlinearity in wind data [130], we propose a novel neural network model which we call the smooth pinball neural network (SPNN). This network is able to provide probabilistic forecasts in the form of multiple monotonically increasing quantiles estimated simultaneously.

The main contributions of our approach can be summarized as follows. First, we propose and investigate a new objective function which is a logistic based smooth

approximation of the pinball loss function for multiple quantile regression. We introduce a smooth penalty scheme to prevent the quantile crossover problem. We showcase how a multiple quantile based neural network can be used for probabilistic forecasting of wind. We design experiments to validate our model using publicly available data from 10 wind farms from the Global Energy Forecasting Competition 2014 and benchmark performance with common and advanced methods. And finally, we show our method improves the skill, reliability, and sharpness of forecasts over various benchmarks.

Related Work

With QR being a comprehensive strategy for providing the conditional distribution of a response y given x, we highlight several of its variants. In a generalization of QR [131, 132] introduces the censored QR model, which consistently estimates conditional quantiles when observations on the dependent variable are censored. Yu and Jones [133] propose a nonparametric version of QR estimation by using a kernel-weighted local linear fitting. Chen et al. [134] propose a copula-based nonlinear quantile autoregression, addressing the possibility of deriving nonlinear parametric models for different conditional quantile functions. QR can also be hybridized with machine learning methods to form powerful nonlinear models. For instance, support vector regression is introduced for QR in [135], yielding support vector quantile regression (SVQR). SVQR extends the QR model to non-linear and high dimensional spaces, but it requires solving a quadratic programming problem.

Due to their flexibility in modeling elaborate nonlinear data sets, artificial neural networks are another dominant class of machine learning algorithms that can be used to enhance QR. Taylor [31] is the first to propose a quantile regression neural network (QRNN) method, combining the advantages of both QR and a neural network. This method can reveal the conditional distribution of the response variable and can also model the nonlinearity of different systems. The author applies this method to estimate the conditional distribution of multi-period returns in financial systems,

which avoids the need to specify the explanatory variables explicitly. However, the work does not address how the network was optimized. The same QRNN was later used by [32] for credit portfolio data analysis where results showed that QRNN is more robust in fitting outliers compared to both local linear regression and spline regression. In [33] an autoregressive version of QRNN is used for applications to evaluating value at risk, and [34] implements the QRNN model in R as a statistical package.

In all the QR approaches mentioned, only a single quantile is estimated at a time. In the case of estimating multiple quantiles, this could lead to what is known as the quantile crossover problem, where a lower quantile overlaps a higher quantile. Equivalently, a prediction interval for a lower probability (e.g., range in which 10% of future values are predicted to lie) exceeds that of a higher probability (e.g., the range in which 20% of the future values are predicted to lie). Crossing quantiles are undesirable as it violates the principle of cumulative distribution functions where their associated inverse functions should be monotonically increasing. A possible way to prevent this issue is to utilize simple heuristics of reordering estimated quantiles. However, this approach does not have a strong theoretical foundation and may lead to inappropriate quantiles. The solution then is to optimize quantiles together with non-crossing constraints. In [114, 136] a constrained support vector quantile regression (CSVQR) method is developed with non-crossing constraints where it was used to fit quantiles on static data. However, CSVQR is computationally very expensive and slow to train. In Section 5 we review approaches for preventing the quantile crossover problem in neural networks and we also propose a novel way to prevent this problem using a smooth penalty function.

Smooth Pinball Network Model

We propose to use a feedforward neural networks for probabilistic forecasting due to their flexibility and strength in dealing with nonlinear and nonstationary data. We can use the pinball loss in the objective function of such a neural network

to estimate conditional quantiles. However, the pinball function ρ employed by the original linear quantile regression model in Eq. (1.1) is not differentiable at the origin, $x = 0$. The non-differentiability of ρ makes it difficult to apply gradient-based optimization methods in fitting the quantile regression model. Gradient-based methods are preferable for training neural networks since they are time efficient, easy to implement and yield a local optimum. Therefore, we need a smooth approximation of the pinball function that allows for the direct application of gradient-based optimization. We call our new model the smooth pinball neural network (SPNN).

We are not the first to apply a smooth approximation to the pinball function for a quantile regression based neural network. [34] used the Huber norm to construct smooth approximations of the pinball loss function, following the work in [137], to form a QRNN. Using the same Huber norm approximation, a composite QRNN is proposed in [138] to estimate multiple quantiles. The Huber norm requires multiple optimization runs with a fixed schedule of a decreasing smoothing constant to from the final weights and biases. Chen et al. [139] introduced another class of smooth functions for nonlinear optimization problems and applied this idea to support vector machines [140]. Emulating the work of Chen, a study by Zheng [141] presents an approximation to the pinball loss function by a smooth logistic function; this then allows the application of gradient descent for optimization. Zheng called the resulting algorithm the gradient descent smooth quantile regression model. We extend that model here for the case of a neural network. Based on our knowledge, we are the first to investigate the usage of a smooth logistic loss function to estimate multiple quantile using a neural network.

5.0.1 Smooth Quantile Regression

The smooth approximation [141] of the pinball function in Eq. (1.1) is given by

$$S_{\tau,\alpha}(u) = \tau u + \alpha \log\left(1 + \exp\left(-\frac{u}{\alpha}\right)\right), \tag{5.1}$$

where $\alpha > 0$ is a smoothing parameter and $\tau \in [0, 1]$ is the quantile level we are trying to estimate. In Fig. 6.1 we see the pinball function with $\tau = 0.5$ as the

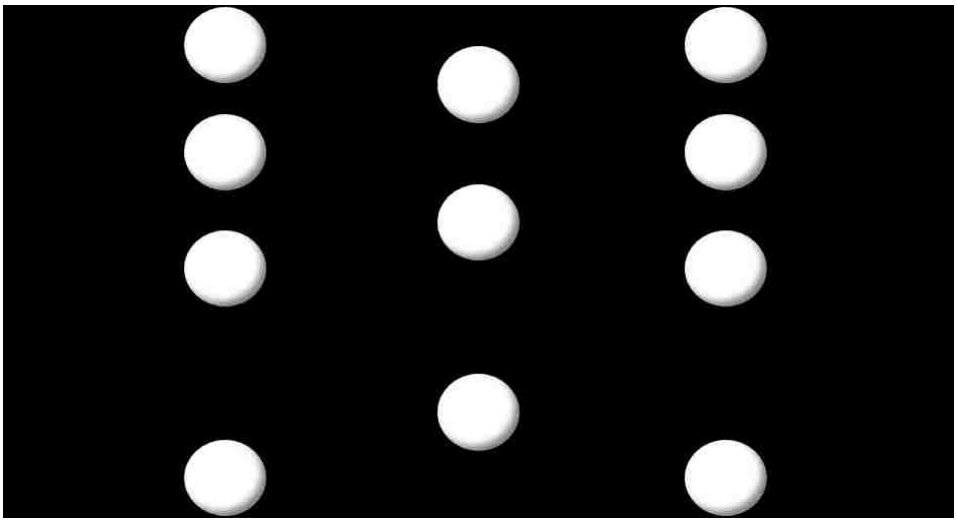

Figure 5.1: Schematic diagram of the smooth pinball neural network.

red line and the a smooth approximation as the blue line with $\alpha = 0.2$. Zheng proves [141] that in the limit as $\alpha \to 0^+$ that $S_{\tau,\alpha}(u) = \rho_\tau(u)$. He also derives and discusses several other properties of the smooth pinball function. The smooth quantile regression optimization problem then becomes

$$\min_{W,b} \frac{1}{N} \sum_{t=1}^{N} S_{\tau,\alpha}(y_t - \hat{q}_t^{(\tau)}), \tag{5.2}$$

where N is the number of training examples and $\hat{q}_t^{(\tau)} = WX_t + b$ where W, b are the model parameters and X_t is a vector of features at time t. This form conveniently allows gradient based algorithms to be used for optimization.

5.0.2 Smooth Pinball Neural Network

For simplicity we describe here the construction of a single hidden layered SPNN for nonlinear multiple quantile regression, but SPNN can easily be extended to multiple hidden layers. In a single hidden layered SPNN the input layer consists of n_x number of input nodes and takes vector X_t of input features at time t. The hidden layer consists of n_h number of hidden neurons and the output layer consists of M number

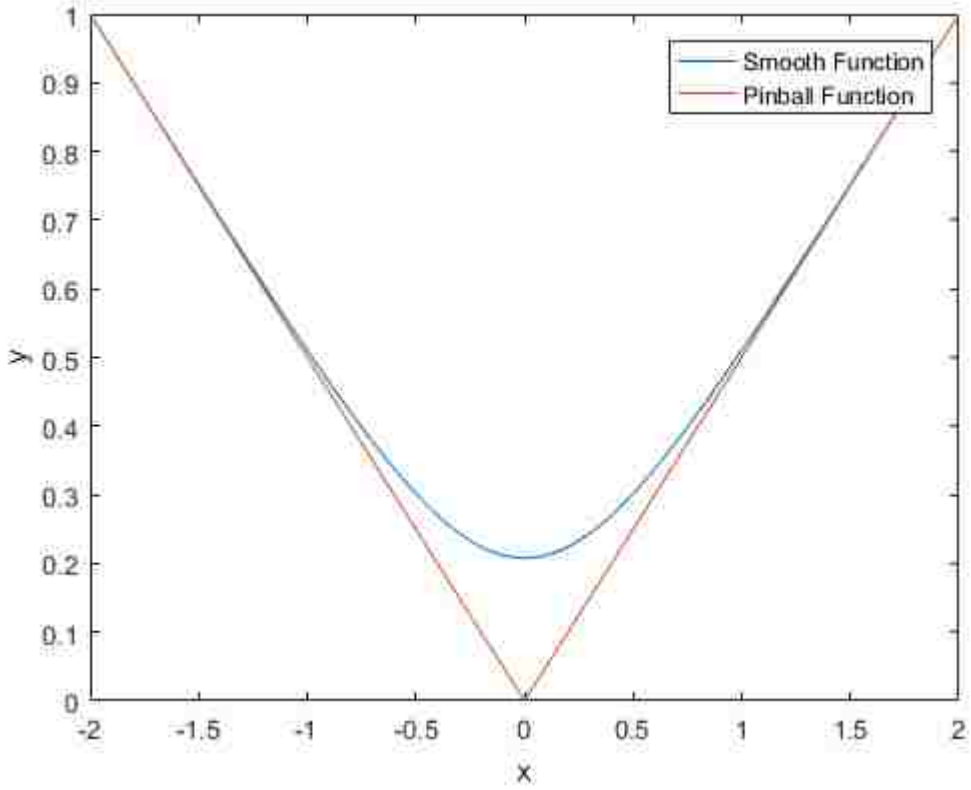

Figure 5.2: Pinball ball function versus the smooth pinball neural network with smoothing parameter $\alpha = 0.2$.

of output nodes corresponding to the estimated quantiles $\hat{Q}_t = [\hat{q}_t^{(\tau_1)}, ..., \hat{q}_t^{(\tau_M)}]^\top$ where $\hat{q}_t^{(\tau_m)}$ is the τ_m quantile level we want to estimate at time t. Every element in the first layer is connected to hidden neurons with the weight matrix $W^{[1]}$ of size $(n_x \times n_h)$ and bias vector $b^{[1]}$of size $(n_h \times 1)$. A similar connection structure is present in the second layer in the network between the hidden and output layers with $W^{[2]}$ the output weight matrix of size $(n_h \times M)$ and bias vector $b^{[2]}$ of size $(M \times 1)$.

The input to hidden neurons is calculated, in vectorization notation, by $Z_t^{[1]} = W^{[1]}X_t + b^{[1]}$, the output of the hidden layer then uses the logistic activation function $H_t = \tanh\left(Z_t^{[1]}\right)$. The input to output neurons is then calculated by $Z_t^{[2]} = W^{[2]}H_t + b^{[2]}$, and the output layer uses the identity activation function $\hat{Q}_t = Z_t^{[2]}$.

The objective function for our SPNN model is then the smooth pinball approximation summed over M number of τ's we are trying to estimate in the output layer. We also use L2 regularization on the network weights in the objective function to prevent over-fitting during training. The objective function for SPNN is then given by

$$E = \frac{\lambda_1}{2NM}\|W^{[1]}\|_F^2 + \frac{\lambda_2}{2NM}\|W^{[2]}\|_F^2 + \frac{1}{NM}\sum_{t=1}^{N}\sum_{m=1}^{M}... \\ \left[\tau_m(y_t - \hat{q}_t^{(\tau_m)}) + \alpha \log\left(1 + \exp\left(-\frac{y_t - \hat{q}_t^{(\tau_m)}}{\alpha}\right)\right)\right]. \tag{5.3}$$

where $\|.\|_F$ is the Frobenius norm. Fig. 6.2 shows a schematic diagram of our SPNN model with n_x number of input features and M number of quantile outputs.

Standard gradient descent with backpropagation can be used to train SPNN. Through this process we compute the gradient of the objective function E_t at each data point at time t with respect to $W^{[1]}, b^{[1]}, W^{[2]}$ and $b^{[2]}$. We start with the gradient with respect to the hidden-to-output weights $W^{[2]}$. In order to compute the gradient at time t, we apply the chain rule in vector notation as follows

$$\begin{aligned}\frac{\partial E_t}{\partial W^{[2]}} &= \frac{\lambda_2}{M}W^{[2]} + \frac{\partial E_t}{\partial \hat{Q}_t}\cdot\frac{\partial \hat{Q}_t}{\partial Z_t^{[2]}}\cdot\frac{\partial Z_t^{[2]}}{\partial W^{[2]}} \\ &= \frac{\lambda_2}{M}W^{[2]} + \frac{1}{M}\left(\frac{1}{1+\exp\left(\frac{y_t-\hat{Q}_t}{\alpha}\right)} - T\right)H_t,\end{aligned}$$

where $T = [\tau_1, ..., \tau_m]^\top$ is a vector of all our τ's. The gradient of $b^{[2]}$ can be calculated similarly. Next we calculate the gradient of the objective function with respect to the weights of the first layer $W_{[1]}$ as follows

$$\begin{aligned}&\frac{\partial E_t}{\partial W_{[1]}} = \\ &\frac{\lambda_1}{M}W^{[1]} + \left(\frac{\partial E_t}{\partial \hat{Q}_t}\cdot\frac{\partial \hat{Q}_t}{\partial Z_t^{[2]}}\cdot\frac{\partial Z_t^{[2]}}{\partial H_t}\right)\cdot\frac{\partial H_t}{\partial Z_t^{[1]}}\cdot\frac{\partial Z_t^{[1]}}{\partial W^{[1]}} \\ &= \frac{\lambda_1}{M}W^{[1]} + \frac{1}{M}\left(\frac{1}{1+\exp\left(\frac{y_t-\hat{Q}_t}{\alpha}\right)} - T\right)W^{[2]} \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \left(1 - H_t^2\right)X_t\end{aligned}.$$

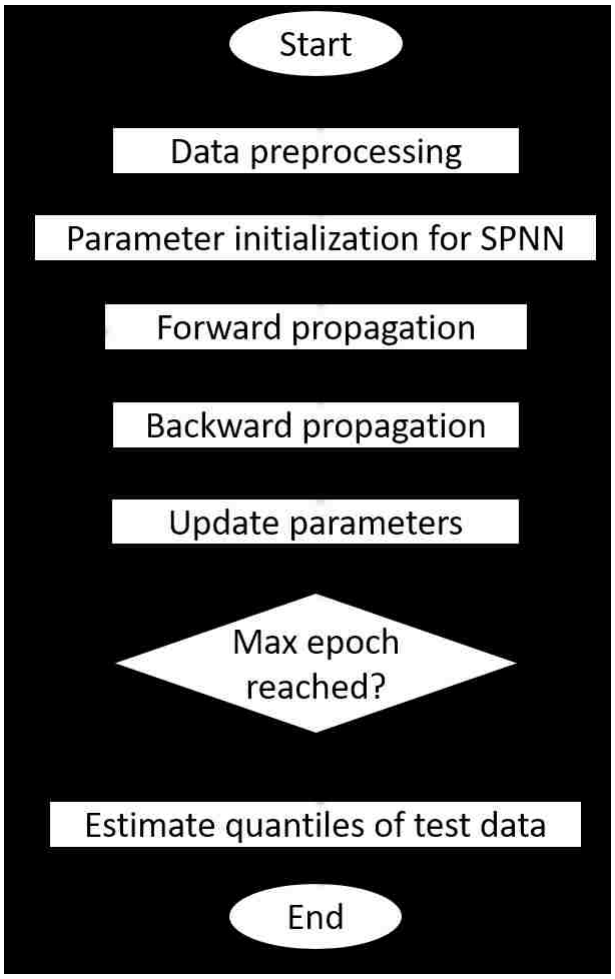

Figure 5.3: Flowchart of the steps taken when conducting a probabilistic forecast with SPNN.

The gradient of $b^{[1]}$ can be calculated similarly. These gradients can then be directly used in many other gradient descent based optimization schemes. As such, we apply the Adam optimizer [142], an algorithm for first-order gradient-based optimization, to learn the parameters of SPNN. Adam has been shown [142] to yield superior results compared to other gradient-based optimizers.

5.0.3 Noncrossing Quantiles

In quantile regression normally a single quantile is estimated. To estimate multiple quantiles, one could be run QR to solve for different τ's independently. However, in doing so, quantiles may cross each other which is not desirable since it violates the principle of monotonically increasing inverse density functions. To prevent this, we need to introduce constraints as per [114]. The condition $0 < \tau_1 < ... < \tau_M$ are defined as the orders of M conditional quantiles to be estimated. To ensure these quantiles do not cross each other the following constraint is needed $q_t^{(\tau_1)} \leq ... \leq q_t^{(\tau_M)}, \forall_t$.

However, it is not easy to solve the neural network optimization problem with such constraints using gradient descent methods. One possible solution is proposed in [143] where a monotonic composite QRNN is presented that applies partial monotonicity constraints to the weights of the network and uses a stacked input matrix of covariates of size $N \times M$ with an added covariate τ_m. This can add additional complexity to the network, by adding more parameters, so we propose a simpler alternative of applying a penalty term [144] directly into the cost function. We define the non-crossing quantile penalty term p as follows

$$p = c \sum_{t=1}^{N} \sum_{m=1}^{M} \left[\max \left(0, \epsilon - \left(\hat{q}_t^{(\tau_{m-1})} - \hat{q}_t^{(\tau_m)} \right) \right) \right]^2 \tag{5.4}$$

where $\hat{q}_t^{(\tau_0)} = 0$, ϵ is the least amount that the two quantile should differ by, and c is the penalty parameter with a high value. This penalty p is added to the cost function in Eq. 6.4. If the constraints are not violated no penalty is added to the cost function. If a lower quantile exceeds the value of a higher one, the squared difference of these two quantiles is added to the cost function as a penalty. A full model implementation flowchart is shown in Fig. 6.3. First the data is preprocessed which includes deriving different input features, feature standardization, and partitioning the data into training and testing sets. Training of the model is conducted using gradient descent optimization method. After the max number of training epochs is reached the model is ready to be used on testing data for multiple quantile estimation.

Results and Discussions

To validate our model for probabilistic forecasting of wind power we utilize wind data from the publicly available Global Energy Forecasting Competition 2014 (GEFCom2014) [8]. The goal of the wind component of GEFCom2014 was to design parametric or nonparametric forecasting methods that would allow conditional predictive densities of the wind power generation to be a function of input data which are numerical weather predictions (NWPs). Evaluation of predicted densities was done using the quantile score. Data is from the years of 2012 and 2013 from 10 wind farms titled Zone 1 to Zone 10. The predictors are NWPs in the form of wind speeds at an hourly resolution at two heights, 10m and 100m above ground level. These forecasts are for the zonal and meridional wind components (denoted U and V). It was up to the contestants to deduce exact wind speed, direction, and other wind features if necessary. These NWPs are from the exact locations of the wind farms. Additionally, power measurements at the various wind farms, with an hourly resolution, are also provided. All power measurements are normalized by the nominal capacity of their wind farm. The goal in forecasting is to learn to associate the provided NWPs (or derived features) with wind power. NWPs are provided for the forecasting horizon of one month, and it is up to a forecasting model to use those NWPs as input to predict quantiles at each future time step.

5.0.4 Benchmark Methods

We use three standard [12] and two advanced benchmark methods for density forecasting of wind power. The standard methods are the persistence model that corresponds to the normal distribution and is formed by the last 24 hours of observations, the climatology model that is based on all past wind power, and the uniform distribution that assumes all observations occur with equal probability. For our advanced benchmarks, we use a linear and nonlinear version of QR. The linear version is multiple quantile regression (QR) with L2 regularization, and nonlinear version is support

vector quantile regression (SVQR) [135] with a radial basis function kernel.

5.0.5 Case Study Descriptions

In the analysis of SPNN for forecasting wind power quantiles, we conduct studies with SPNN having one and two hidden layers denoted as SPNN1 and SPNN2. We study if the addition of a second hidden layer improves performance. Our SPNN model is a fully connected feedforward neural network, with rectified linear units for hidden activation functions, and it uses Adam for weight optimization [142]. Default Adam parameters follow those provided in the original paper. The quality of the quantile estimates is sensitive to the hyperparameters of the network. SPNN has several hyperparameters that need to be chosen before training. Through empirical testing on training data, we found the following values as adequate for our model hyperparameters: 2000 training iterations, 200 batch size, 40 hidden nodes for SPNN1, 20 and 40 hidden nodes for SPNN2, 0.01 for the smoothing rate, 0.01 for each of the weight regularization terms, 1000 for the cross-over penalty term, and 0 for the cross-over margin.

For testing we conduct two case studies using the GEFCom2014 wind datasets. To ensure that our study is unbiased, we use for assessment the whole year of 2013. This dataset gives a total of $365 \times 24 = 8760$ test samples for wind power forecasting per wind farm. The first case study uses wind data from Zone 1 and 2. We estimate quantile to produce prediction intervals with nominal coverage from 10% to 90% in increments of 10%. The goal of this study is to evaluate the quantile and prediction interval estimates from SPNN in detail for reliability and sharpness. We also look at QVSS to see improvements between SPNN1 and SPNN2 use QR as the reference model. We also compare results to SVQR as it is the only other nonlinear quantile regression benchmark model.

In the second case study, we estimate 99 quantiles on par with GEFCom2014. Results are derived for all ten wind farms in total, where we have 87,600 total test observations. For each test month, we are estimating 99 quantiles for 720 look ahead hours across ten farms. Results are derived across all Zones for QS, IS, ACE, and

Sharpness. Given so much data we need a way to summarize results. Thus for every farm, we take the mean of all the evaluation scores across all Zones/months. In both case studies training is done using a sliding window of the previous twelve months to forecast the whole next month. Data from 2013 are used for hold out test sets. For instance, we start with predicting January 2013 using the past 12 months of 2012. After a month is predicted, the training window moves to incorporate new data and the prediction model is retrained to get a new prediction.

We run our case study on a computer with an Intel i7 6700 2.6 GHz, and 16 GB of RAM. For both studies, we use as input features the raw wind speed data at 10m and 100m for U and V directions. The only engineered features are four time features based on the hour of the day and day of the year

$$\cos(2\pi\frac{hour}{24}), \sin(2\pi\frac{hour}{24}), \cos(2\pi\frac{day}{365}), \sin(2\pi\frac{day}{365}).$$

This is contrast with the winning teams from GEFCom2014 who each used dozens of engineered features including lagged data, data from neighboring wind farms, and more complex features such as derived wind speeds, wind direction, wind energy, wind shear, direction differences between 10m and 100m, etc. Most of the winning teams in GEFCom2014 conducted heavy manual feature engineering to reduce the quantile score throughout the competition. The goal of our study is not custom feature engineering, which might result in better scores, but to highlight the effectiveness of SPNN in creating its own latent features via its hidden layers, and to showcase the feasibility of our method as a robust probabilistic forecasting model.

5.0.6 Case Study 1

For this first case study, quantiles are computed to form predictive intervals. Each prediction interval is estimated to have a future observation of wind power within a lower and upper bound for a given probability or nominal coverage rate. As previously mentioned, we estimate quantile to produce prediction intervals with nominal coverage from 10% to 90% in increments of 10%. We estimate intervals

for SPNN1, SPNN2, QR, and SVQR. The difference between the nominal coverage rates and the observations for Zone 1 are shown in Fig. 5.4. This reliability diagram showcases results similar to the ACE score. It can be see that SPNN2 has the lowest deviation from the nominal coverage with SPNN1 and QR coming second and third with result magnitudes ranging from -3% to -0.3%. SVQR has a very poor coverage with deviations as high as -40%. This can be attributed to having too tight intervals and over-fitting. In Fig. 5.7 we showcase reliability results from Zone 2. Similarly to Zone 1, SPNN2 yields intervals with a deviation close to 0, while SVQR continued to have a poor coverage.

Sharpness is the other important statistic that we look at for individual predictive intervals which is calculated independent of observations. Measured as the mean interval size as described in Eq. 1.0.2, it demonstrates the usefulness of predictions. Ideally, we would like to have intervals as small as possible but too small and observations may fall outside the intervals. Thus, too wide and too narrow intervals providing poor forecasts. Sharpness needs to be analyzed together with reliability to ensure robust predictions. In Fig. 5.5, we see the mean interval sizes for each coverage level for Zone 1. QR resulted in having the widest intervals and SVQR having the narrowest intervals. With such narrow intervals SVQR was not able to capture the observations which indicated in its reliability diagram. In Fig. 5.8 we see similar results for Zone 2. Our proposed method, SPNN1 and SPNN2, were able to estimate effective sized intervals that resulted in high reliability with good sharpness.

As a last evaluation, we look at the performance of the individual quantiles that formed the prediction intervals of this case study. We do this using QVSS to analyze relative performance gain relative to a reference benchmark model. Here we use quantile regression for the reference model and we study if the nonlinear quantile regression models, SPNN and SVQR provide any improvements over QR. In Fig. 5.6, we report the QVSS across the 18 quantiles for the three nonlinear methods. SPNN1 and SPNN2 provide a clear performance increase with respect to QR. For quantiles with a nominal probability less then 0.7, we see SPNN1 having a small lead over SPNN2. While SPNN2 shows a small lead for quantiles with $\tau > 0.7$. Not

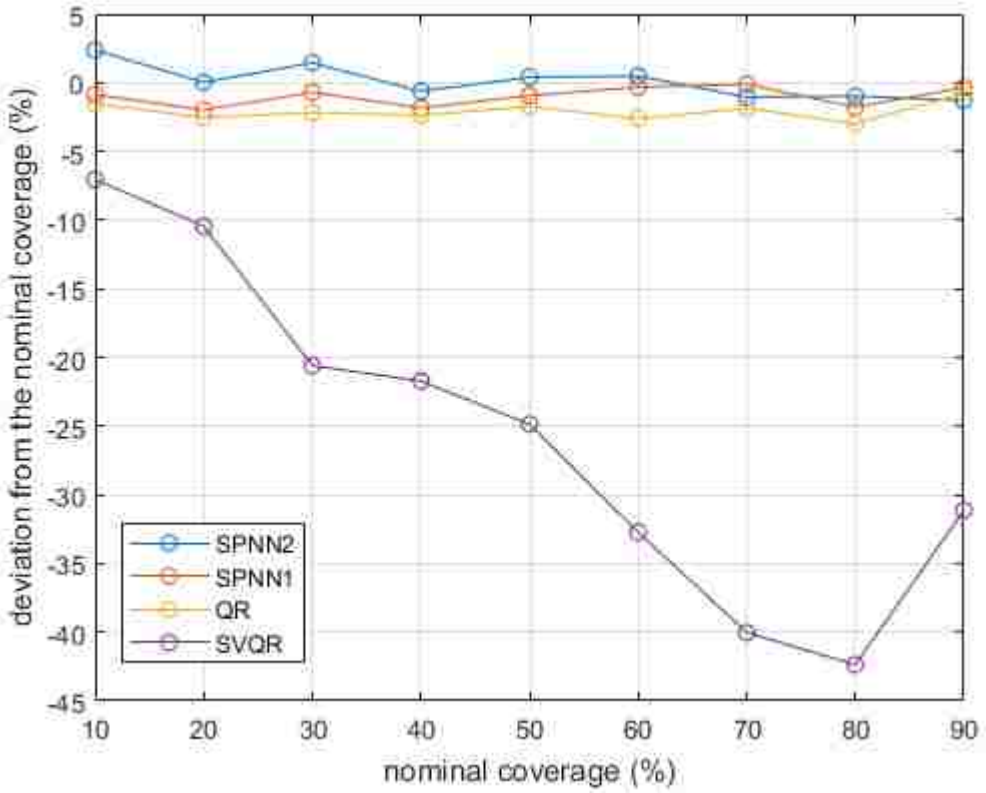

Figure 5.4: Reliability of prediction intervals from Zone 1 measured by the frequency of observation falling with each interval.

surprisingly, SVQR shows a decreased negative performance over QR, indicating its inability to extract meaningful features from the raw data for Zones 1 and 2. In Fig. 5.9, we see similar QVSS results for Zone 2, but with SPNN2 showing a small lead over SPNN1 for quantiles with $\tau < 0.7$.

5.0.7 Case Study 2

In our second case study we analyze a higher number of estimated quantile (99) across all wind farms for all 12 test months to ensure an unbiased assessment of SPNN relative to the benchmark models. Due to the large number of quantiles and wind farms, instead of forming reliability or sharpness diagrams for individual PIs

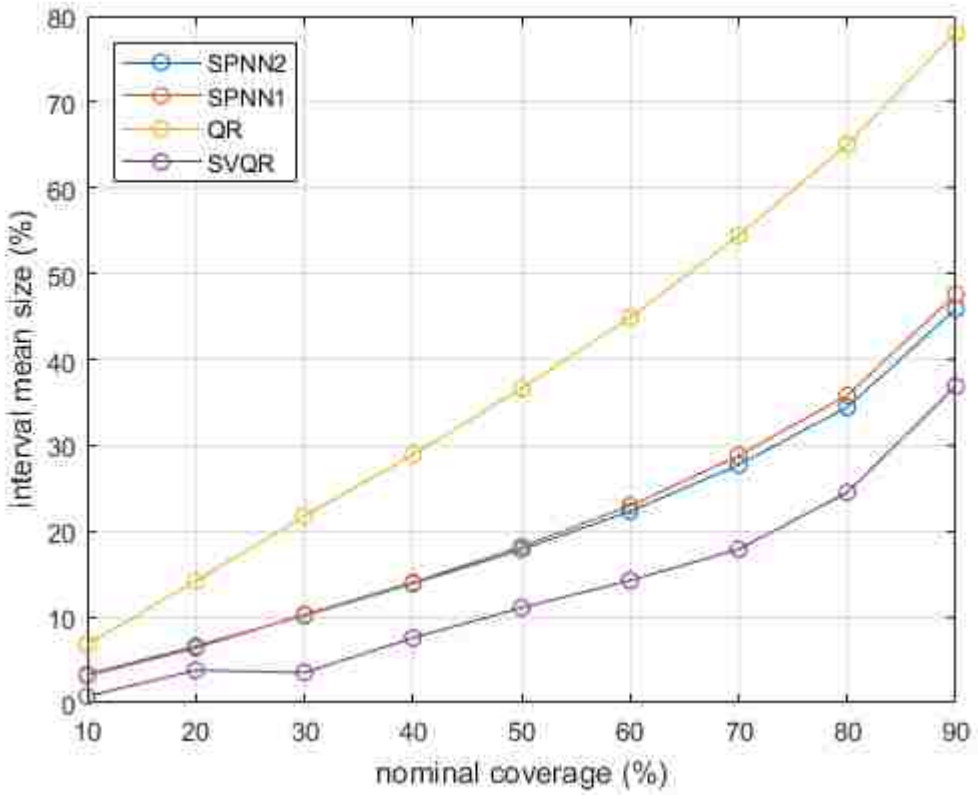

Figure 5.5: Sharpness of prediction intervals for Zone 1 measured by the interval mean size.

and QVSS diagrams for individual quantiles, we instead look at box plots and report the distribution of evaluation results including QS, IS, ACE, and Sharpness.

In Fig. 5.10 we report the QS metric for SPNN and the five benchmark methods. We see that SPNN2 had the lowest QS range from 0.036 to 0.047 with SPNN1 being a close second. The other benchmarks had a QS in the range of 0.075 to 0.011. Inspecting the coverage analysis of our prediction intervals with the ACE score in Fig. 5.11, we see that SPNN overall has the lowest ACE with SPNN2 having a median value lower then SPNN1. The uniform benchmark produced a wide range for the ACE score due to having fixed size intervals across all zones and months, while SPNN2 had the narrowest range of ACE scores. Looking at the sharpness of

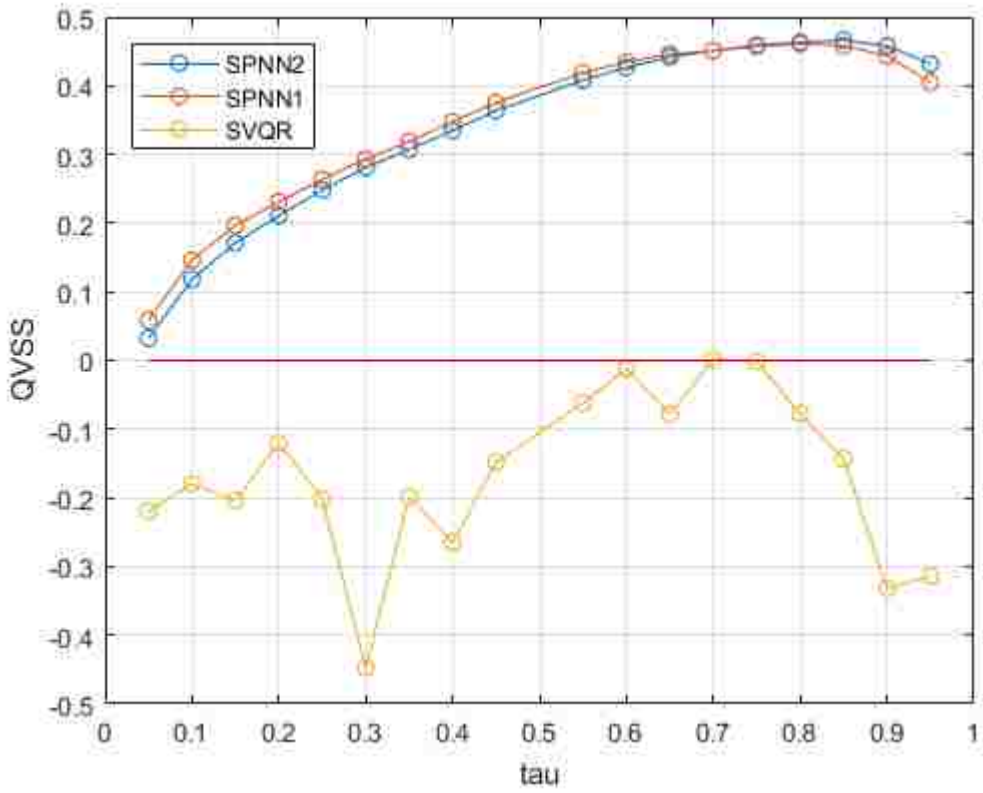

Figure 5.6: QVSS measured relative performance of SPNN2, SPNN1, and SVQR to QR on Zone 1 dataset.

PIs with the interval score in Fig. 5.12 and general sharpness score in Fig. 5.13, we see that SPNN has the sharpest intervals across all farms. The persistence and climatology methods yielded a wide distribution for the interval score but narrow one for sharpness. SVQR in contrast to the first case study did not calculate narrow intervals when estimating 99 quantiles.

Since both QS and IS also measure skill, we can say that SPNN was able to produce the highest quality estimates from all methods. An interesting observation is the SPNN is designed to produce optimal quantile estimates and that indirectly it also produces adequate interval forecasts. If the primary goal is to reduce ACE and IS as best as possible, alternative loss functions that incorporate prediction interval

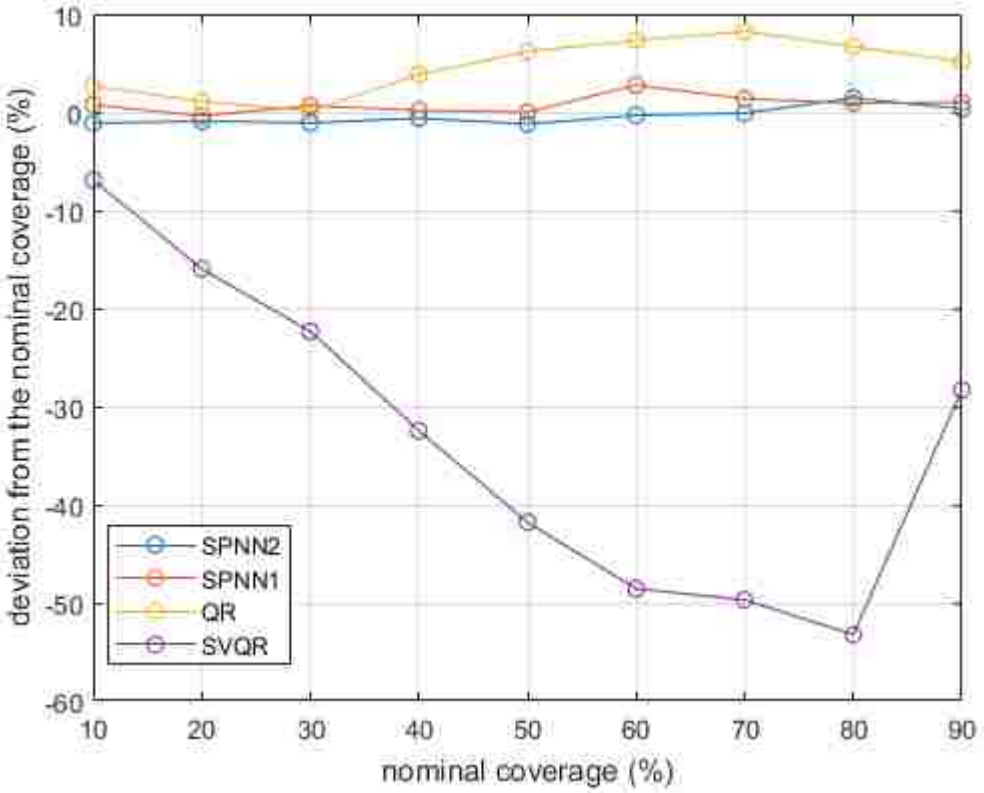

Figure 5.7: Reliability of prediction intervals from Zone 2 measured by the frequency of observation falling with each interval.

coverage and width functions can be used. However, while not directly optimizing for coverage or sharpness, SPNN does produce superior results from the advanced benchmarks multiple quantile regression and support vector quantile regression.

Lastly, we compare the mean QS of our proposed method to the final quantile scores for the top teams in the GEFCom2014 as originally reported in [8]. We note again that the top teams used a wide range of engineered features while we used raw wind speed data along with time as input to our model. The winning team in GEFCom2014 was kPower with a mean QS of 0.038. Our method SPNN2 has a close mean QS of 0.042 which would qualify SPNN to be in the top winning teams. Comparing the results from the four box plots, we see the robust prediction ability of

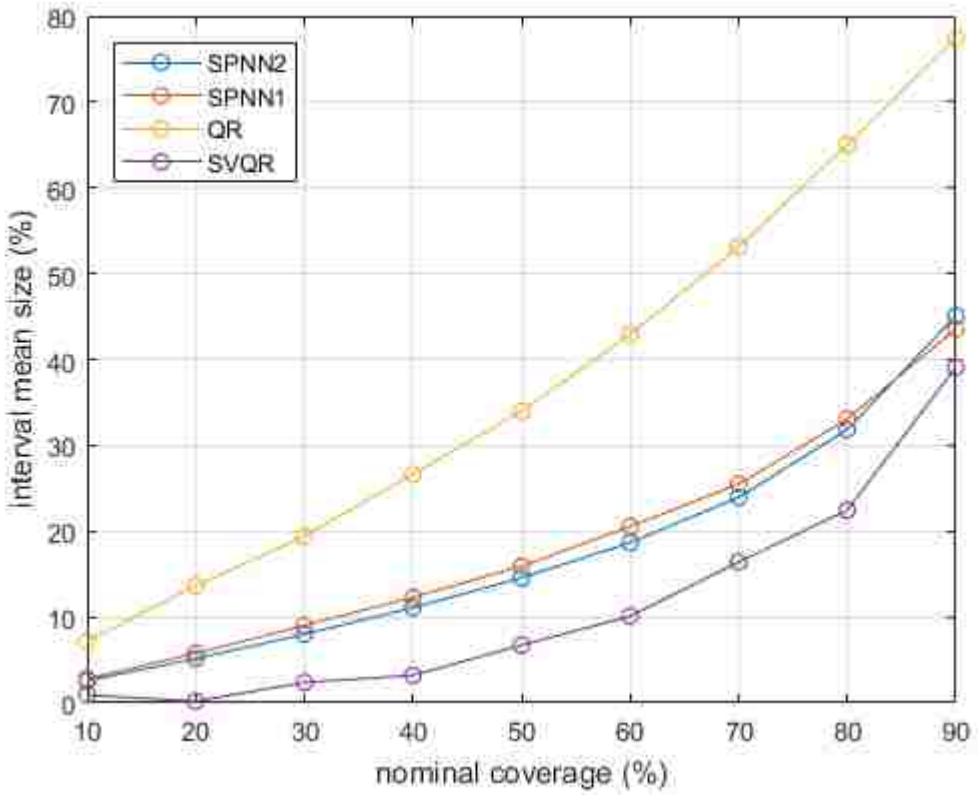

Figure 5.8: Sharpness of prediction intervals for Zone 2 measured by the interval mean size.

the proposed SPNN prediction method. Additionally, for all the runs across months and farms, the preassigned PI coverage levels are satisfied which implies that the constructed PIs cover the target values with a high probability and with the lowest QS and IS.

Conclusion

Wind power forecasting is crucial for many decision-making problems in power systems operations and is a vital component in integrating more wind into the power

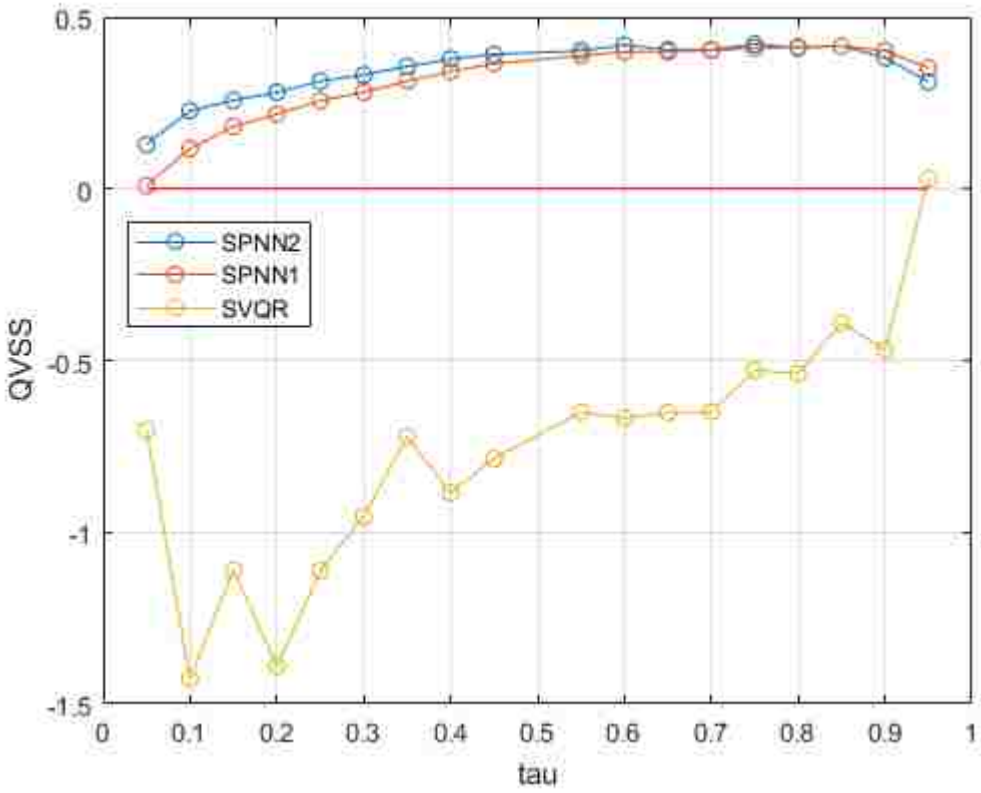

Figure 5.9: QVSS measured relative performance of SPNN2, SPNN1, and SVQR to QR on Zone 1 dataset.

grid. Due to the chaotic nature of the wind, it is often difficult to forecast. Uncertainty analysis in the form of probabilistic wind prediction can provide a better picture of future wind coverage. This work proposes a novel approach we call SPNN for probabilistic wind forecasting using a neural network with a smooth approximation to the pinball ball loss function in estimating multiple quantiles.

We also introduce non-crossing constraints in the form of a smooth penalty in the loss function. This is done to ensure multiple quantiles can be estimated simultaneously without overlapping each other. We verify the effectiveness of our SPNN model with the dataset of the Global Energy Forecasting Competition 2014. We compare forecasts to standard and advanced benchmarks and employ standard

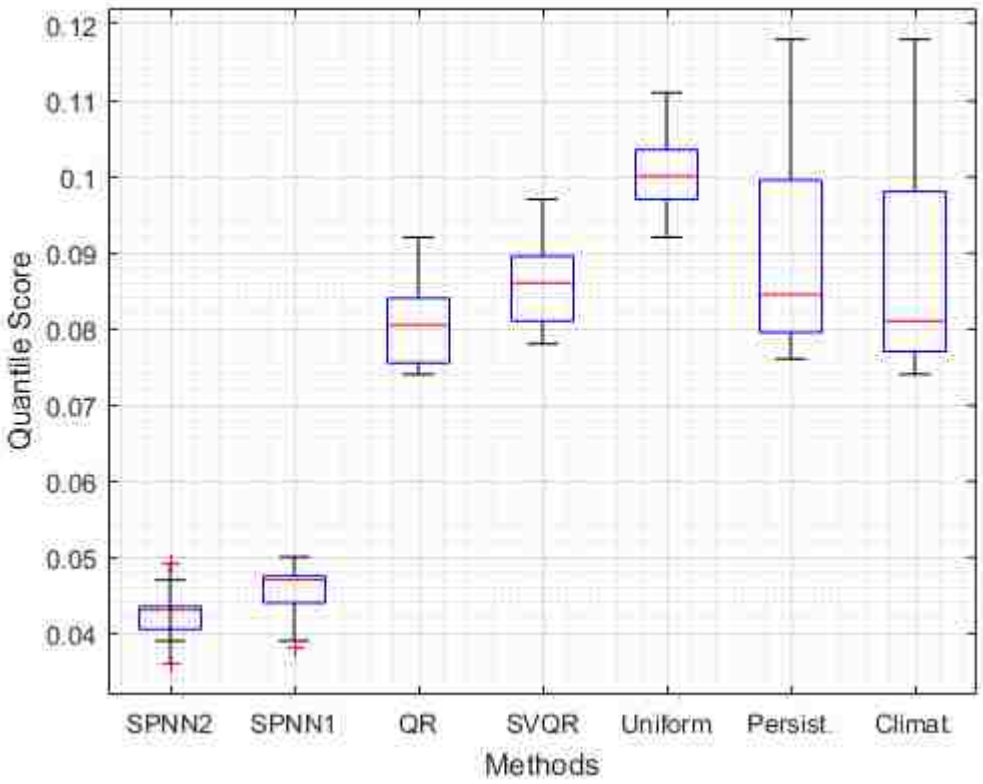

Figure 5.10: Box plot of quantile score evaluation across all datasets.

quantile score, reliability, and sharpness metrics. Our results show superior performance across the prediction horizons, which verify the effectiveness of the model for forecasting while preventing estimated quantiles from overlapping.

Our SPNN method has the potential to be applied to a variety of domains for probabilistic forecasting or multiple quantile estimations. Future work will look into applying SPNN to forecast solar and ocean wave power, to test its effectiveness across different renewable energies, and on electricity pricing and load demand for smart grid applications. In this study, we trained our model using NWP data. Another problem to study is very short-term probabilistic forecasting using only past wind power data. Future work can also then look into expanding the SPNN model for providing full predictive densities given lagged past data of power only.

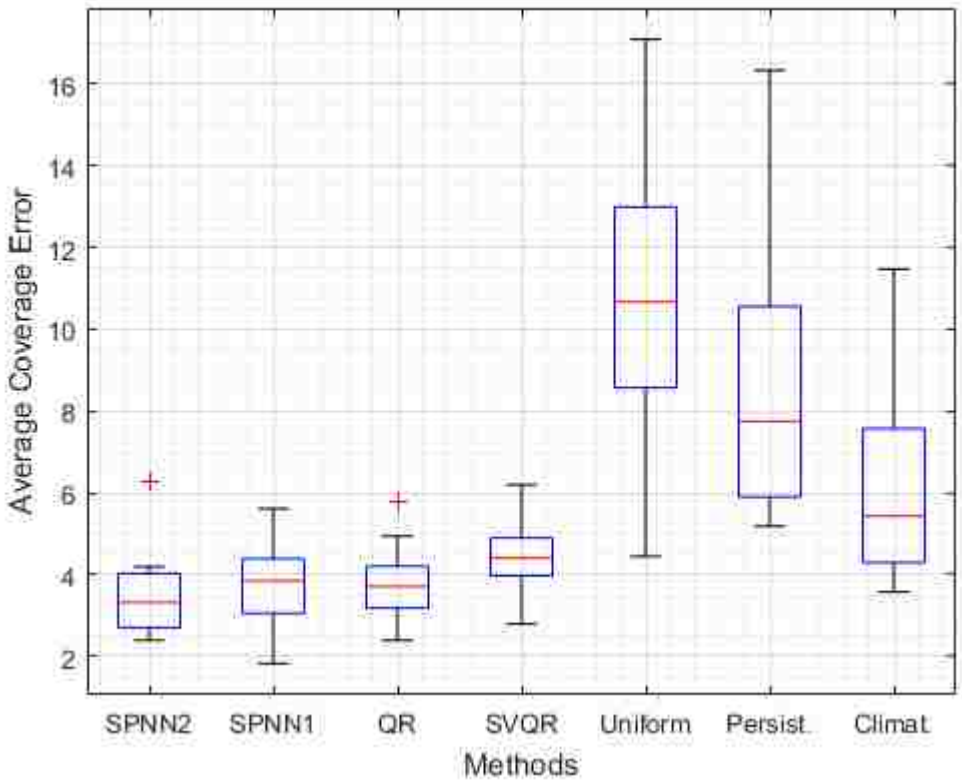

Figure 5.11: Box plot of average coverage error evaluation across all datasets.

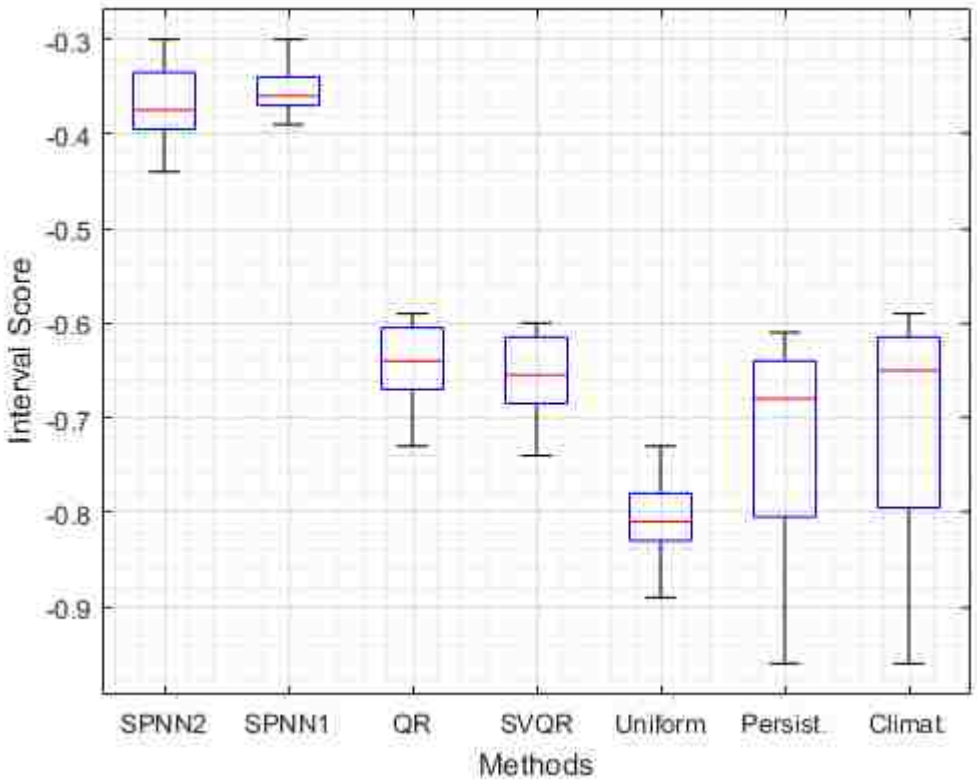

Figure 5.12: Box plot of interval score evaluation across all datasets.

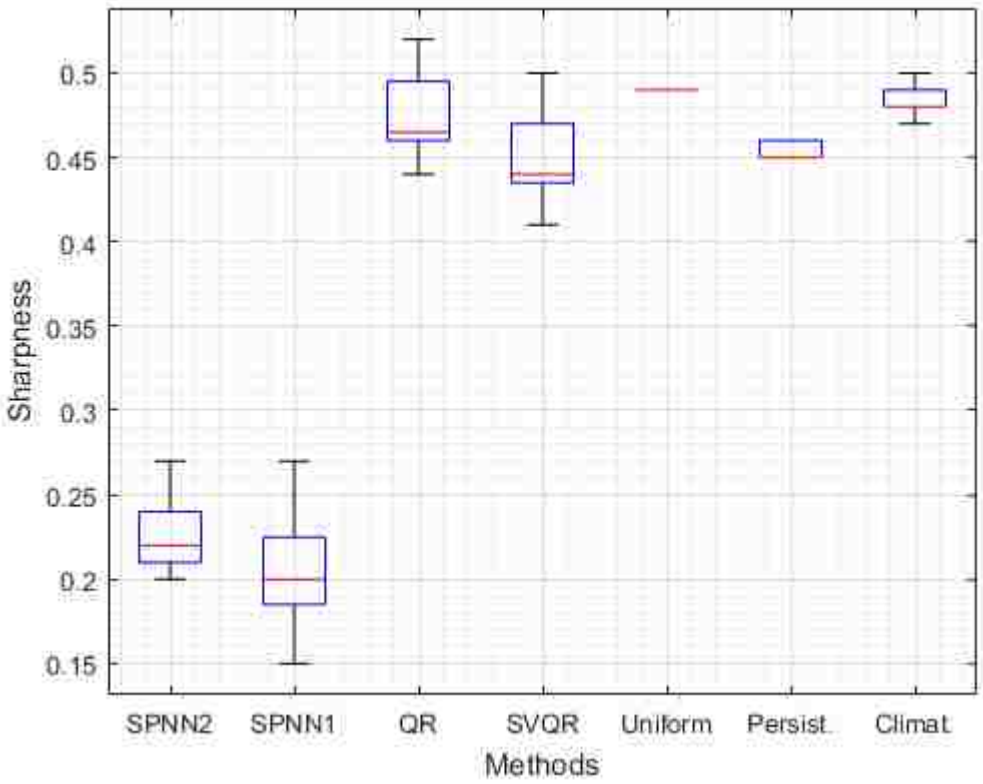

Figure 5.13: Box plot of sharpness evaluation across all datasets.

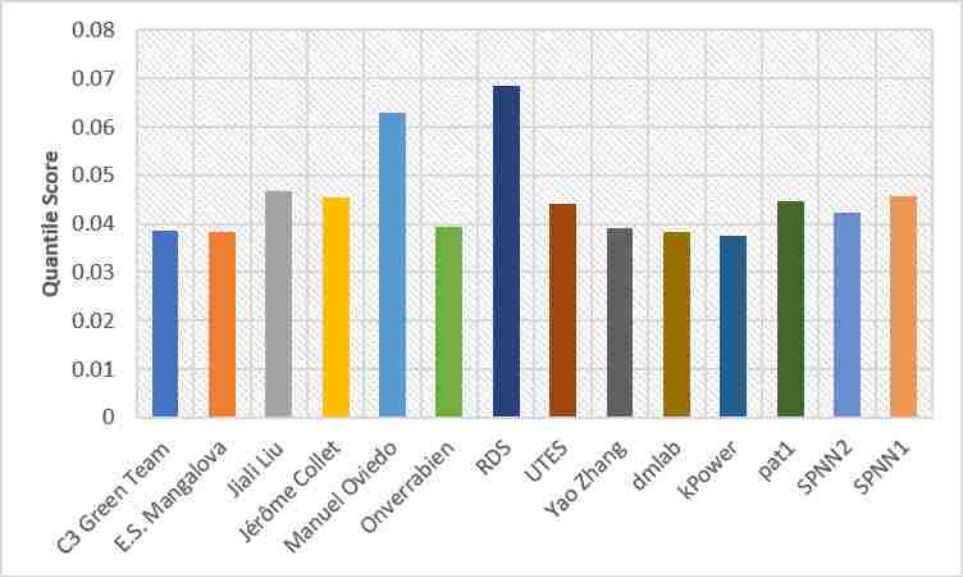

Figure 5.14: Bar plot of SPNN2 and SPNN1 mean quantile score across all wind data compared to the performance of the top teams in GEFCom2014 Wind Track.

Chapter 6

Multiple Quantile Fourier Neural Network

Introduction

Univariate time series based deterministic or point forecasting is a well-studied field that has numerous applications. Examples of such applications include finance [145], topic behavior [146], traffic flow, [147], and renewable power [148]. There are several approaches to forecasting with different classes of methods. Approaches include having a sliding window of past data to predict future data, recurrent models, and extrapolation based regression such as signal approximation [149]. In all the approaches, methods can are divided into two classes as linear or nonlinear. In the first class, methods include linear regression, autoregression (AR), autoregressivemoving-average (ARMA) models, and exponential smoothing. The second class of methods is nonlinear models which are predominantly machine learning based such as support vector regression, nonlinear autoregression neural networks, and recurrent neural networks. Deterministic forecasts which provide a single expected output for a given look-ahead time have been successfully applied for multiple domains such as renewable energy prediction for solar, wind, and wave

power. Other applications include agriculture, economics, finance, and manufacturing. A thorough overview of time series and machine learning based deterministic forecasting can be found in [4, 150].

Despite the popularity of deterministic forecasting, it does have a significant disadvantage in that it can result in individual errors which can be significant. Additionally, deterministic forecasting lacks information on associated uncertainty. A solution to these problems is probabilistic forecasting (PF) where the goal is to produce fully probabilistic predictions that derive quantitative information on the uncertainty. A PF takes the form of a predictive probability distribution over future time horizons and aims to maximize the sharpness of predictive densities while subject to reliability. Sharpness refers to the concentration of the predictive distribution and reliability refers to the accuracy of the forecasted probability in conveying the actual probability of events.

A popular application of PF is in the fields of renewable energies and power systems. A probabilistic forecast is vital, for different operations to renewable energy farms. This includes managing the optimal level of generation reserves [124], optimizing production [126], and bidding strategies for electricity markets [128]. Applications to the power grid include load analysis [151], smart meters [152], scheduling [153], system planning [154], unit commitment [155], and energy trading [156]. A thorough overview of probabilistic wind and solar power forecasting is provided in [17] and [20].

There are several essential classes in the type of PF models which include if they are parametric or nonparametric, direct or indirect, and the type of inputs they use for prediction. In PF, we are first trying to predict one of two types of density functions, either parametric or nonparametric. When the future density function is assumed to take a specific distribution, such as the Normal distribution, then this is known as parametric probabilistic forecasting. For processes where no assumption is made about the shape of the distribution, a nonparametric probabilistic forecast can be made. Nonparametric predictions can be made in the form of quantiles, prediction intervals, or full density functions. For example, nonlinear and nonstationary data, such as wind speeds or stocks, may not correspond to fixed or

known distributions. When in need of forecasting such data it can be more beneficial to apply a nonparametric probabilistic forecast to estimate the distribution instead then assume it is shaped.

The second classification of PF models is whether they are combined with point forecasts or not. For instance in [157] a deterministic and probabilistic forecast for wind power is combined. This approach is known as an indirect PF method. First, a point forecast is made such as with support vector regression, and then prediction intervals for point forecast values are obtained with a PF method such as quantile regression. On the other hand, when a PF method estimates future quantiles or prediction intervals without using as input point forecasts, this is known as direct forecasting. The last distinction to be made with PF models is if past lagged data are used as inputs to the forecast model or if future exogenous variables are used too. For instance in renewable power PF, if numerical weather predictions (NWP) exist for each forecasting horizon that we are interested in, then those exogenous NWPs are used as inputs to provide a PF in that prediction horizon. When NWPs are not given then lagged past time values of renewable power can be used for prediction.

We introduce a new approach to developing a nonparametric direct PF where the input to the model is neither exogenous variables, such as NWPs in the case of renewable forecasting nor past data but instead treats the series as a signal. This approach is motivated by Fourier extrapolation which is the process by which a Fourier transform is applied to a data set to decompose it into a sum of sinusoidal components thus interpreting it as a signal. In time series analysis this is related to Harmonic regression. In accounting for periodic and non-periodic aspects of a signal such as a trend, Fourier neural networks (FNN) have been proposed. FNNs are feedforward neural networks with sinusoidal activation functions that model the Fourier transform. Most recently, a new FNN called neural decomposition (ND) was proposed in [149] that can decompose a signal into a sum of its constituent parts, model trend, and reconstruct a signal beyond the training samples. ND can provide a prediction by having time as its only input similarly to an inverse Fourier transform. We propose a PF model motivated by the ND model.

Several works have explored Fourier extrapolation based deterministic forecasting with sinusoidal neural networks [149, 158, 159], but none have yet explored it for probabilistic forecasting. We are the first to introduce an FNN for forecasting composite quantiles that we dub the quantile Fourier neural network. Contributions of our approach can be summarized as follows:

1. We demonstrate how this extrapolation based quantile forecasting is able to model periodic and non-periodic components of nonstationary time series.

2. We demonstrate an initialization process that fixes parameters to none random values and train the model with gradient descent backpropagation.

3. We design experiments to validate our approach for direct probabilistic forecasting and provide insight how this method can generalize modeling uncertainty on real-world datasets.

The contents of the paper are: in Section 6 we review existing architectures of FNNs, go over our model, its architecture, training, and weighting initialization scheme. Results and discussion of validating our method are presented in Section 6.0.4. We conclude the paper and review future research directions in Section 6.0.7.

Proposed Methodology

Fourier analysis examines the approximation of functions through their decomposition as a sum or product of trigonometric functions, while Fourier synthesis focuses on the reconstruction of a signal from its decomposed oscillatory components. These well-studied processes have a large utility in time series analysis. By decomposing a time series into its frequencies one could then interpolate missing time values by reconstructing the original signal. Further applications include modeling seasonality and even prediction of a time series through extrapolation of an approximated signal. In the application of Fourier analysis for time series analysis, an important

method is the discrete Fourier transform (DFT), which converts a series into its frequency domain representation, and the inverse discrete Fourier transform (iDFT) which maps the frequency representation back to the time domain. The transforms can be expressed as either a summation of complex exponentials or sines and cosines by Eulers formula. In this section we explore existing works on Fourier networks that directly use iDFT in their operation or mimic it, then we describe our proposed FNN methodology for quantile forecasting.

6.0.1 Fourier Neural Networks

Neural networks with sine as an activation function are difficult to train in theory and when initialized randomly yield poor results [160]. Thus, few works have attempted to explore Fourier analysis with sinusoidal neural networks. We highlight most of the works here. One of the first FNNs was introduced by Adrian Silvescu [161, 162] who developed Fourier-like neurons for learning boolean functions. The FNN model used the units of the network to approximate a DFT in its output. Similar in spirit to an FNN was a Fourier transform neural network introduced in [163] that uses the Fourier transform of the data as input to an artificial neural network. FNNs have since been used for stock prediction [164], aircraft engine fault diagnostics [165], harmonic analysis [159], and extensions include a single input multiple outputs based FNNs that can turn nonlinear optimization problems into linear ones [164], FNNs for output feedback learning control schemes [166], and deep FNNs for lane departure prediction [167].

There are two recent works that study FNNs for time series prediction that use the Fourier transform of the data as weights. The first is an FNN presented by Gashler and Ashmore in [168]. Their technique uses the fast Fourier transform (FFT) to approximate the DFT and then uses the obtained values to initialize the weights of the neural network. Their model uses a combination of sinusoid, linear, and softplus activation units for modeling periodic and non-periodic components of a time series. However, their trained models were slightly out of phase with their validation data. The second study on FNNs for time series prediction is presented

by Godfrey and Gashler [149] who proposed a similar model to [168] called neural decomposition (ND), except that they do not use the Fourier transform to directly initialize any weights.

The ND model is inspired by the inverse discreet Fourier transform where given time t as input it attempts to model the signal $x(t)$. However, there are some distinctions between ND and iDFT. First ND allows sinusoid frequencies to be trained and second ND can also model non-periodic components in a signal such as trend. With the ability to train the frequencies, ND learns the actual period of a signal whereas iDFT assumes that the underlying function always has a period equal to the size of the samples it represents. ND is a feedforward neural network with a single hidden layer with N nodes and has one input and one output node. Hidden nodes are composed of sinusoid units for capturing the periodic component in an underlying signal and other activation functions, such as linear or sigmoid units, for capturing the non-period component. Parameters of ND are initialized in such a way so as to mimic the iDFT. ND is then trained with stochastic gradient descent with backpropagation. ND was applied to time series deterministic forecasting and showed very promising results across different data sets, often beating state-of-the-art methods such as LSTM, SVR, and SARIMA.

6.0.2 Quantile Fourier Neural Networks

Inspired by the ND model we propose a new forecasting method which we call the quantile Fourier neural network (QFNN). Unlike ND and other FNNs our QFNN model is trained to extrapolate composite quantiles of an underlying time series. The use of sinusoid activation functions allows the model to fit periodic data, and coupled with an augmentation function QFNN is able to probabilistically forecast a time series that is made up of non-periodic components too. The model is defined as follows. Let each a_{jk} represent an amplitude, each ω_k represents a frequency, each ϕ_k represents a phase shift, and $b_\tau^{[2]}$ and $b_k^{[1]}$ represent bias terms for the quantile signal representation. Let $f(t)$ be an augmentation function that represents the

non-periodic components of the signal. QFNN then can be defined by

$$q_t^\tau = f(t) + b_\tau^{[2]} + \sum_{k=1}^{N} \left(a_{\tau,k} \cdot \cos\left(\omega_k t + \phi_k\right) + b_k^{[1]} \right) \tag{6.1}$$

where given time t as the input, it attempts to predict the τ-level quantile. QFNN is loosely modeling a time series as a partial Fourier cosine series

$$x(t) = A_0 + \sum_{n=1}^{N} A_n \cos(n\omega_0 t + \phi_n) \tag{6.2}$$

where $\omega_0 = \frac{2\pi}{T}$, T is the period of the signal $x(t)$, and A_0, A_n, and ϕ_n are real numbers. The main difference between Eq. 6.1 and Eq. 6.2 is that QFNN does not fix the period of the signal to a predetermined size T, it allows for bias terms, it has an augmentation function to represent non-periodic components of the signal, and it learns the frequencies ω_k versus keeping them at a fixed size. The bias terms in the output layer of the network are important because it allows shifting the level of each quantile appropriately.

The hidden layer of QFNN is composed of N units with a sinusoid activation function and an arbitrary number of units with other activation functions to calculate $f(t)$. The output layer is composed of M number of linear units that represent quantiles. The parameters a_{jk}, being the weights between the hidden and output layers allows us to model different amplitudes for composite quantiles while simultaneously learning the frequency and phases for all quantiles in the hidden layer. Utilizing conventional neural network notation $W^{[1]}$ is a matrix of the $f(t)$ unit parameters and the frequency parameters in Eq. 3. The $b^{[1]}$ vector represents the phases of the sinusoidal components, $W^{[2]}$ is a parameter matrix of the amplitudes, and we also add additional bias terms to the output nodes for each quantile with the $b^{[2]}$ vector.

To estimate quantiles we need to solve the minimization problem described in Eq. 1.2. However, the pinball function ρ in Eq. 1.2. is not differentiable at the origin, $x = 0$. The non-differentiability of ρ makes it difficult to apply gradient-based optimization methods in fitting the quantile regression model. Gradient-based methods are preferable for training neural networks since they are time efficient, easy to

implement and can yield a local optimum. Therefore, we need a smooth approximation of the pinball function that allows for the direct application of gradient-based optimization. A smooth approximation of the pinball function in Eq. (1.1) can be given by Zheng in [141] as

$$S_{\tau,\alpha}(u) = \tau u + \alpha \log\left(1 + \exp\left(-\frac{u}{\alpha}\right)\right), \tag{6.3}$$

where $\alpha > 0$ is a smoothing parameter and $\tau \in [0, 1]$ is the quantile level we're attempting to estimate. In Fig. 6.1 we see the pinball function with $\tau = 0.5$ as the red line and the a smooth approximation as the blue line with $\alpha = 0.2$. Zheng proves [141] that in the limit as $\alpha \to 0^+$ then $S_{\tau,\alpha}(u) = \rho_\tau(u)$. With this smooth approximation we can then define the cost minimization problem for QFNN as

$$E = \frac{1}{NM} \sum_{t=1}^{N} \sum_{m=1}^{M} \left[\tau_m (y_t - \hat{q}_t^{(\tau_m)}) + \alpha \log\left(1 + \exp\left(-\frac{y_t - \hat{q}_t^{(\tau_m)}}{\alpha}\right)\right)\right]. \tag{6.4}$$

where M number of τ's we are trying to estimate in the output layer. The input to hidden neurons is calculated, in vectorization notation, by $Z_t^{[1]} = W^{[1]}t + b^{[1]}$, the output of the hidden layer then uses the logistic activation function $H_t = \cos\left(Z_t^{[1]}\right), f\left(Z_t^{[1]}\right)$. The input to output neurons is then calculated by $Z_t^{[2]} = W^{[2]}H_t + b^{[2]}$, and the output layer uses the identity activation function $\hat{Q}_t = Z_t^{[2]}$ where $\hat{Q}_t$ is a vector output of the estimated composite quantiles. An architectural view of the QFNN is shown in Fig. 6.2.

6.0.3 Monotone Constraints

With QFNN estimating composite quantiles, this could lead to what is known as the quantile crossover problem, where a lower quantile overlaps a higher quantile. For instance, a quantile with $\tau = 0.5$ could be estimated at a higher level than a quantile with $\tau = 0.4$. The quantile crossover problem violates the principle of cumulative distribution functions where their associated inverse functions should be monotonically increasing. One way to prevent this problem is to enforce non-crossing constraints where $\hat{q}_t^{\tau_1} >= \hat{q}_t^{\tau_2}$ where $\tau_1 >= \tau_2$. For neural networks trained

with gradient descent backpropagation, it is not straightforward to directly apply inequality constraints to the cost function. The easiest is to utilize a simple heuristic of reordering predicted quantiles which is what we apply in this work.

6.0.4 Implementation Details

The proposed QFNN model is trained using gradient descent with backpropagation. The training process allows the model to learn better frequencies and phase shifts so that the sinusoid units more accurately represent the seasonality of an underlying time series. Since frequencies and phase shifts can change, the model learns a more reliable periodicity of the underlying series rather than assuming the period is of a predetermined size. Training also tunes the weights of the augmentation function which estimates the non-periodic component of the time series. Additionally, the cost function of the model uses L1 regularization on the output weights only to promote sparsity and shrink the less essential cosine components of each quantile.

There is a considerable distinction in how QFNN is initialized compared to other FNNs. Instead of randomly setting parameters or initializing them to mimic the iDFT we set all bias terms to 0, the output weights $W^{[2]}$ which represent the amplitudes are initialized near 1, and the input weights $W^{[1]}$ which represent the frequencies are set to multiples of πk where k is a specific hidden node. The input weight parameters of the augmentation function $f(t)$ are initialized to 1 and its bias terms set to 0. By configuring all the parameters in such a fixed fashion, we eliminate the randomness associated with neural network initialization. QFNN, therefore, yields the same results on the same set of data every time after training.

Before training starts, the input data is preprocessed in the same fashion as in [149] to improve learning. First, the time associated with each training sample is normalized between 0 (inclusive) and 1 (exclusive) on the time axis. By doing this normalization testing data points will have a time value greater than or equal to 1. With this normalization the 1/N term in the frequencies is taken into account by transforming t into t/N. Next, if the max value in the training set is greater then 10, then the training set is scaled between 0 and 10. Both these preprocessing steps

expedite learning and help prevent the model from falling into local optimums.

We present the full proposed QFNN methodology architecture in Fig. 6.3. Summarizing our methodology, we first partition a given time series into training and testing sets. Preprocessing of the training set is then conducted which includes applying a logarithmic filter if multiplicative trend or seasonality is present. If the training data has points above 10, the data is normalized between 0 and 10. Next, the training and testing times are normalized so that training times are between 0 and 1. Parameters of the QFNN are then initialized as described in the previous paragraphs. Training of the model is conducted using batch gradient descent. After the max number of training epochs is reached the QFNN model is ready to be used for testing. Forecasts can be provided for a multi-step period of indefinite time steps. After a test set prediction is made, preprocessing steps are reversed if any were conducted. Preprocessing steps may include scaling a time series back to its original scale or removing the logarithmic filter from the outputs by exponentiating the predictions.

Validation

In this section, we evaluate the effectiveness of the QFNN method for the estimation of quantiles and prediction intervals. We first describe the different time series datasets we use for experiments and the different benchmark methods that we compare QFNN to. Then we conduct an assessment of the predictive power of QFNN based on the quantile and interval score metrics. In each of the experiments, we use one linear activation function for the augmentation function of QFNN to capture the trend. We found that using more than one augmentation function or using other activation functions such as tanh, sigmoid, and rectified linear units did not provide any significant improvements in trend estimation. An L1 regularization term of 10 is used in all case studies except for the Air Passengers dataset where a regularization term of 1 is used based on the QS fit of QFNN on the training data. For all experiments, we use a maximum training iteration of 10000 for QFNN,

a learning rate of 0.1, and a smoothing rate of 0.05. No form of hyper-parameter tuning was used for the max iteration, learning rate, or smoothing rate.

6.0.5 Case Studies

We carry out experiments on eight nonstationary univariate time series datasets, seven being real-world case studies and one synthetic case. These datasets were explicitly picked because they display a diverse set of periodic and aperiodic patterns such as trend, additive and multiplicative seasonality, multiple seasonality, cycles, and irregular patterns. Table 6.1 the characteristics of all the datasets. The first case study is the classical Air Passengers time series [169] which is composed of 144 samples of the number of passengers flying each month from January 1949 to December 1960. It has a positive linear trend and multiplicative seasonality.

The second case study is the yearly mean and monthly smoothed total sunspot numbers from 1700 to 2017 [170]. It consists of 318 samples with a time granularity of one year. This time series includes an unstable (non-constant) seasonal patterns over time. Case study 3 is the load demand from ISO New England [171]. Its time series is composed of 744 samples for January 2017. Target values represent real-time demand in MW for wholesale market settlement from revenue quality metering. This case study displays seasonal and cyclical patterns. Internet traffic data in bits from a private ISP with centers in 11 European cities is used for the 4rth case study [172], which exhibits multiple seasonality. We use the data that corresponds to a transatlantic link and was collected from on June 18 to July 16, 2005.

The highly random movements of the stock market are almost impossible to predict but some stocks may exhibit unseen cycles or trends over more extended periods of time [173]. To examine such possible patterns we use the closing stock prices of Apple Inc. over five years from 2012 to the beginning of April 2018 [174]. The next two case studies are normalized solar and wind power for September 2012 and January 2012. These two datasets come from the Global Energy Forecasting Competition of 2014 [8]. Solar power forecasting is fairly accurate when training on data from sunny days but is trickier when training data contains non-sunny days

too. Wind power, on the other hand, is highly chaotic and is very difficult to forecast from univariate time series.

The last case study looks at ocean wave elevation, the main motivation for using such data is the irregular sinusoidal nature of waves. Due to the difficulty of finding high resolution deep ocean wave elevation measurements we construct a synthetic dataset. For simulating of ocean waves we focus on vertical sensors for predicting irregular wave formations. Under generally well accepted assumptions [175], the wave elevation for sensor locations (x, y) on the ocean surface for all times t the exact time waveform which would be observed at a particular point in the ocean can be described by

$$H(x, y, t) = \sum_{i=1}^{L} A_i \cos\left(\frac{\omega_i^2}{g}\big(x\cos(\beta_i) + y\sin(\beta_i)\big) - \omega_i t + \phi_i\right), \tag{6.5}$$

which has the parameters A for the amplitude, ω for the frequency measured in radians per second ($rads/s$), β for the wave angular direction in radians measured relative to the x-axis, and ϕ for the phase in radians. We chose to estimate waves at the origin with $L = 2$, and for each parameter we arbitrarily chose the values $A = [1, 1.5]$, $\omega = [0.5\pi, 0.1\pi]$, and $\phi = [1.2, 1.4]$. To each observation we also include additive white Gaussian noise which we assume come from the sensors.

6.0.6 Benchmark Methods

To thoroughly examine the forecasting accuracy of our QFNN method we compare it to nine simple and state-of-the-art probabilistic forecasting methods. These include two naive approaches, the uniform and persistence methods. Three-time series models which are the autoregressive integrated moving average model, the seasonal autoregressive integrated moving average model, and exponential smoothing with trend and seasonality. Lastly, we use four advanced PF methods: linear quantile regression, polynomial quantile regression, composite support vector quantile regression, and a composite quantile regression neural network.

The uniform method (UM), commonly used in wind power PF studies [12], is the simplest of all the methods. UM assumes that any observation in the time series has

equal probability to occur at any time step. The support of the UM is defined by the parameters a and b which are the minimum and maximum values of the training set for each case study. Quantiles are then defined by $F^{-1}(\tau) = (1-\tau)a + \tau b$ for $\tau \in [0, 1]$. For deterministic forecasting, the persistence forecast method is a very popular benchmark and is known to be hard to outperform for single point or short look-ahead forecasts. We use the persistence method (PM) [176] for PF as a benchmark where the forecast error is assumed to be random and normally distributed, it's mean and variance are computed by the latest observations. For our experiments, we use the last S observations from the training set to calculate the moments of the PM distribution where S corresponds to the size of the seasonality derived from the autocorrelation function (ACF). To ensure that UM and PM can estimate appropriate multi-step forecasts we extend both of them by adding an estimated linear trend component from the training data. We implement both UM and PM in Matlab R2017a.

The next three benchmark methods are well established time series models. We use the autoregressive integrated moving average (ARIMA) model, seasonal ARIMA (SARIMA) model, and exponential smoothing with trend and seasonality (ETS) model, also known as the Holt-Winters seasonal method. We choose ARIMA because it can eliminate non-stationarity through an initial differencing step to better fit time series for prediction, and we select SARIMA and ETS to capture periodic patterns better. Parameters of ARIMA and SARIMA are selected using grid search with the Akaike information criterion and the application of the parsimony principle to prevent over-fitting. The seasonal parameter S for SARIMA and ETS is chosen using the ACF. Quantiles are estimated for ARIMA, SARIMA, and ETS assuming the normality assumption [177, 178]. ARIMA and SARIMA are implemented in Python using the sarimax function from the statsmodels package [179], and ETS is implemented in Python from a holtwinters package [180].

Assuming a normal distribution for quantile prediction with ARIMA, SARIMA, and ETS is a parametric PF approach and can be somewhat restrictive and may not appropriately estimate the forecast distribution. Therefore, we use four advanced nonparametric PF methods linear quantile regression (QR), polynomial QR (PQR),

composite support vector quantile regression (SVQR) [181], and a relatively new forecasting method of composite quantile regression neural network (QRNN) [182]. All four methods have one input node for time, similar to QFNN. QR and PQR methods are implemented in Python using the quantreg function from the statsmodels package [183].

We implement composite SVQR in Matlab R2017a. It is common for support vector machines to use a Gaussian kernel function. From initial experiments, we found that using a Gaussian kernel in SVQR produced quantile forecasts that are flat or suddenly drop. A similar effect was reported in [149] when using support vector regression for extrapolation based forecasting. To alleviate this problem we propose a novel approach of combining Fourier [184] and linear kernels in an attempt to capture periodic and aperiodic patterns when forecasting. This new kernel takes the following form

$$F(x, y) = \frac{1 - q^2}{2(1 - 2q\cos(x - y) + q^2)} + xy \tag{6.6}$$

where $0 < q < 1$. For our SVQR benchmark we choose q = 0.5 and set the C regularization parameter as the max value in the training set, we found that these choices worked the best when fitting the training sets of the case studies. Our last benchmark, QRNN, is similar in structure to our QFNN model where we use one trend component hidden node (otherwise we found it could not predict trend), and uses a smooth approximation of the pinball loss function. The main differences in QRNN are that L2 regularization is used, the sigmoid tanh activation function in the hidden nodes is applied, and all parameters are randomized during initialization. For QRNN we use a maximum training iteration of 10000, a learning rate of 0.1, and a smoothing rate of 0.1, and a regularization rate of 10 for all experiments.

6.0.7 Results and Discussion

For experimentation in each case study, we use 50% of the data for training and the other 50% for testing. All models only saw observations in the training sets. Test data were never presented to the models and were used just to calculate QS

and IS metrics. The use of 50% of the time series for testing was done to achieve the goal of long-term multi-step PF. For each case study, we estimate 100 quantiles whose nominal values are equally spaced between 0 and 1. These 100 quantiles can be combined to form upper and lower bounds of 50 prediction intervals. We use the QS metric to evaluate the quality of the 100 quantiles and the IS metric to evaluate the 50 prediction intervals. In all experiments, QFNN is used for PF. Median values with nominal level $\tau = 0.5$ were estimated separately for visual comparison of QFNN forecasts with those of the benchmarks.

For figures **??** to 6.8, red dots represent the underlying case study time series. The colored curves represent median forecasts of the top four methods that were able to capture the most information of periodic and aperiodic patterns visually. We found that only the benchmark methods SARIMA, ETS, and SVQR for case studies 1 to 3 and 5 to 6 in Table 6.1 were able to fit meaningful periodic patterns and thus we only display these three benchmarks. All benchmark methods yielded poor fits for case studies 4, 7, and 8; therefore we do not show these plots. Figures 6.9 to 6.16 showcase QFNN estimation of 50 prediction intervals across test and training data from all 8 case studies. The red line in these figures represents the time series observations. Evaluation metrics are reported in Tables 6.3 and 6.4.

In our first experiment, we use the air passengers dataset, where the first six years of data (72 samples) are used for training QFNN and each benchmark method. The next six years were used for prediction. An ACF evaluation finds that the time series has a season of $S = 12$, and grid search found ARIMA(2,1,3) and SARIMA(1,0,0)(1,0,1)[12] to be the best hyperparameters for ARIMA and SARIMA respectively. We see in Fig. 6.8 SVQR estimates the trend but not the seasonality very well. ETS and SARIMA estimate both trend and seasonality well, but the median forecasts fall below and above the test data. QFNN learns the shape of the data better and captures the median appropriately. Fig. 6.9 shows prediction intervals fitting the test data well, and Tables 6.3 and 6.4 report that QFNN without constraints yielded the best scores with QFNN coming second. For the remaining experiments, ARIMA and SARIMA hyperparameters are displayed in Table 6.2.

Our second experiment demonstrates the power of QFNN in modeling non-constant seasonal patterns. The sunspots case study is used which has seasonal patterns of varying amplitudes. The years from 1700 to 1858 were used for training, and the years 1859 to 2017 were used for testing. We see in Fig. 6.4 that SVQR fails to capture any meaningful pattern in its prediction. SRIMA captures a seasonal pattern that is out of phase with the sunspot test series, and ETS shots off in the test set with a positive trend. QFNN captures a seasonal pattern that is a bit more in phase with the number of sunspots over the years and is also able to learn multiple patterns of the sunspot time series. Fig. 6.14 shows prediction intervals fitting the test data surprising well, QFNN captures higher peaks around 1943 and lower peaks around 1903. In Tables 6.3 and 6.4 we see that QFNN has the best score.

The third experiment uses the real-time load demand case study. We use the first 372 hours in the time series for training. In the median plot of Fig. 6.6 SVQR is able to capture a poor and small seasonal pattern. SRIMA captures the daily seasonality but fails to capture any cycles in the test set, and ETS shots off in the test set with a positive trend. QFNN learns both the seasonal and cyclical pattern of the load demand. The capture of the cyclical pattern in the load data by QFNN is better presented in Fig. 6.11 where we see a tight fit of the daily seasonal and weekly cyclical pattern. The only deviation being around January 21, 2017, which shows a lower observed demand in load possibly due to a warmer weekend and less power needed for heating. Tables 6.3 and 6.4 report that QFNN has the best scores with QFNN second best. The fourth experiment uses the solar power case study and uses the first 380 hours of training. The training set includes samples from both sunny and non-sunny days where solar power is lower than average. In Fig. 6.7 SVQR poorly estimates the daily seasonality. SRIMA has a seasonal pattern reducing over time in the test set, and while ETS is able to capture the daily seasonality, we see a slight negative trend. QFNN learns a constant daily quantile pattern for its median estimate and for all its prediction intervals in Fig. 6.12. These consistent patterns can be associated with sunny days and in Tables 6.3 and 6.4 QFNN has the top results.

The fifth experiment using the closing stock prices of the Apple corporation is considered a fascinating case study due to the highly random nature of stock movements. For training, the first 2.5 years of closing prices are used, and testing is composed of closing prices up until the start of April 2018. In Fig. 6.5 we see a long-term positive trend and possibly a cycle in the stock price of Apple across the five years. In the plot, SVQR fits the linear trend of the stock series but nothing else. ETS learns a non-existing seasonal pattern, and SARIMA doesn't seem to capture any meaningful pattern. With QFNN we see that it learns the cyclic and positive trend of the stock price which follows the test data better than any other method. This is also demonstrated in Fig. 6.13 that up until the end of 2017 QFNN follows the trend and cycle, but then in 2018, the price of Apple jumps higher than the prediction. Despite the visually good fit of QFNN in the figures, the UM outperforms all the other methods in Tables 6.3 and 6.4. This is not a surprise as it's prediction intervals would be centered entirely on the trend.

The remaining experiments are the internet, wind, and wave case studies. Median plots of these experiments are not shown since the benchmark methods performed poorly in capturing the multiple or irregular seasonal patterns in these time series. We present the prediction intervals by QFNN in Fig. 6.10 where the first 343 hours are used for training. We see that QFNN can learn the multiple seasonal patterns of internet traffic data. The prediction intervals forecasted by QFNN for the simulated wave elevation case study is shown in Fig. 6.15. It is not a surprise that since ocean waves can be modeled by a sum of sinusoids that QFNN can estimate well the amplitudes, frequencies, and phases of the irregular periodic patterns.

The last experiment conducted is on the wind power dataset. In Fig. 6.16 we see no identifiable periodic or aperiodic patterns in the wind time series training set. This explains why QFNN has a hard time modeling the test set. We do see a few peaks predicted by QFNN such as on January 21 and the 27th, but overall the PF is very poor. Results from all the benchmark methods on the wind case study are even worse then QFNN. The wind experiment demonstrates that not all nonstationary time series can be predicted by QFNN. Tables 6.3 and 6.4 report that QFNN has the lowest QS and IS metrics for the internet, wind, and wave experiments.

Conclusion

Probabilistic predictions can provide a much better analysis of uncertainty then point forecasting. In this paper, a novel approach for probabilistic forecasting is presented called the quantile Fourier neural network. The proposed approach uses a smooth approximation to the pinball ball loss function for estimating composite quantiles. Furthermore, the proposed model provides forecasts using extrapolation based regression instead of autoregression. Extrapolation based regression has not been studied before for probabilistic forecasting. Empirical results on real world univariate time series showcase that our model is able to appropriately capture periodic and aperiodic components to provide high-quality probabilistic predictions.

Given the novelty of our approach, more research needs to be conducted to assess its application to more domains and under different scenarios. Further studies could also look at other cost functions such as using the interval score directly. This could provide appropriate prediction intervals that may have even higher sharpness and reliability. Furthermore, the influence of exogenous variables as additional inputs to the quantile Fourier neural network could be explored.

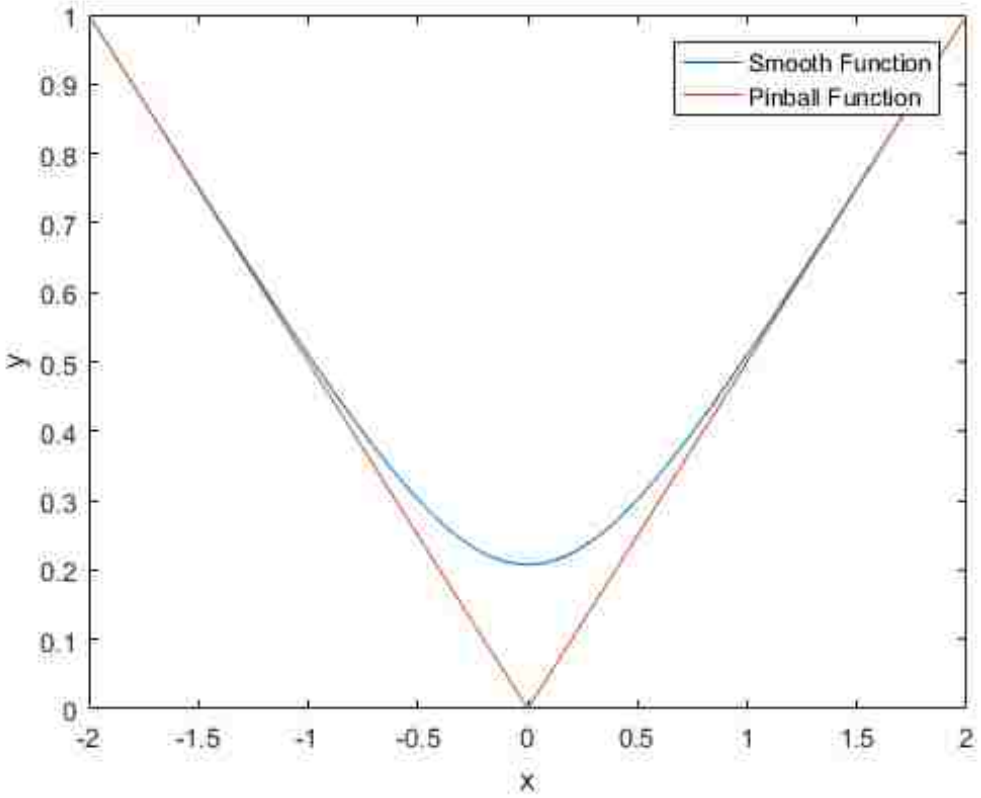

Figure 6.1: Pinball ball function versus the smooth pinball neural network with smoothing parameter $\alpha = 0.2$.

Table 6.1: Datasets used in the experiments.

Case Study	Target	Samples	Time Granularity	Reference
1	Air Passengers	144	Month	[169]
2	Sunspots	318	Year	[170]
3	Real-Time Load Demand	744	Hour	[171]
4	Internet Traffic Data (in bits)	686	Hour	[172]
5	Apple Closing Stock Price	1581	Day	[174]
6	Solar Power	760	Hour	[8]
7	Wind Power	744	Hour	[8]
8	Ocean Wave Elevation	400	Second	(simulated)

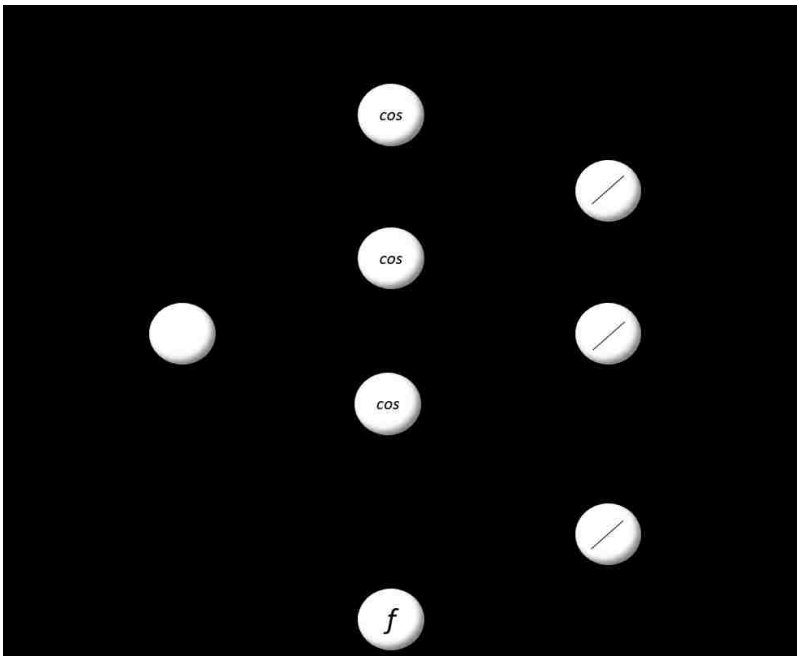

Figure 6.2: Architecture of the quantile Fourier neural network.

Table 6.2: Hyperparameters estimated by grid search for ARIMA and SARIMA for each case study. The seasonal term S is estimated using the ACF plot.

Case Study	ARIMA(p,d,q)	SARIMA(p,d,q)(P,D,Q)	S
1	(2, 1, 3)	(1, 0, 0)(1, 0, 1)	12
2	(3, 1, 2)	(1, 0, 1)(0, 1, 1)	10
3	(2, 1, 3)	(1, 1, 1)(1, 1, 1)	24
4	(2, 2, 2)	(1, 1, 1)(1, 1, 1)	24
5	(2, 2, 2)	(1, 1, 1)(0, 1, 1)	149
6	(2, 1, 2)	(1, 0, 1)(1, 0, 1)	24
7	(2, 0, 1)	(1, 1, 1)(0, 0, 0)	24
8	(2, 0, 2)	(1, 0, 1)(0, 1, 1)	39

Table 6.3: Quantiles scores from QFNN and benchmark methods.

Series	PM	UM	SVQR	PQR	QR	QRNN	ETS	SARIMA	ARIMA	QFNN
Passengers	0.032	0.029	0.053	0.050	0.032	0.029	0.022	0.035	0.549	0.015
Sunspots	0.078	0.083	0.069	0.200	0.068	0.071	0.172	0.068	0.069	0.058
Load	0.037	0.034	0.035	0.265	0.034	0.054	0.154	0.055	0.042	0.028
Internet	0.093	0.084	0.088	0.210	0.079	0.071	0.121	0.513	7.481	0.055
Stock	0.041	0.031	0.048	0.348	0.047	0.052	0.047	0.103	0.206	0.063
Solar	0.083	0.118	0.078	0.077	0.078	0.076	0.102	0.045	0.082	0.030
Wind	0.095	0.092	0.093	0.545	0.099	0.280	0.293	0.192	0.137	0.089
Wave	0.100	0.149	0.145	0.217	0.159	0.207	0.223	0.187	0.138	0.056

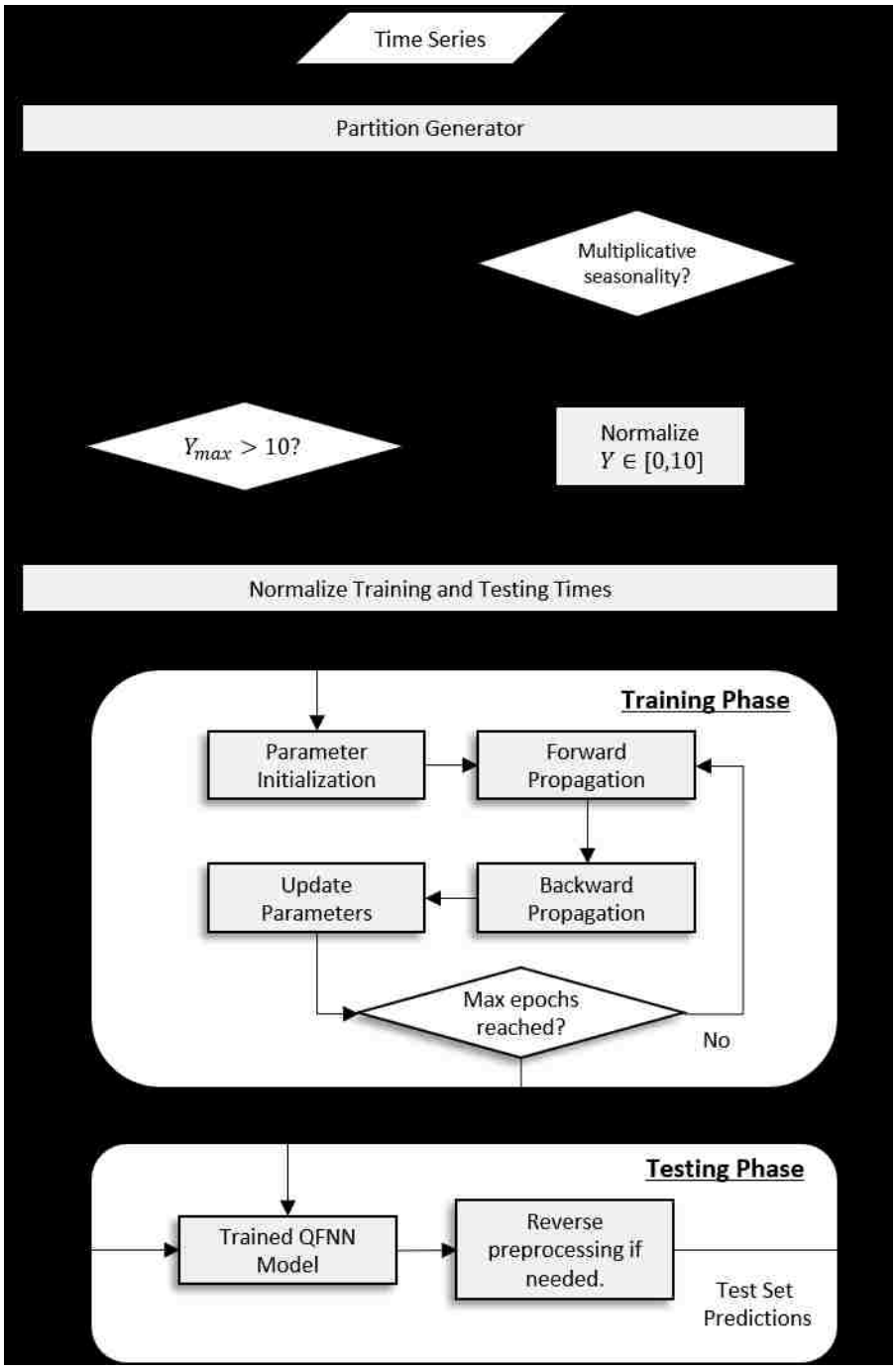

Figure 6.3: Flowchart of proposed methodology using QFNN.

Table 6.4: Interval scores from QFNN and benchmark methods.

Series	PM	UM	SVQR	PQR	QR	QRNN	ETS	SARIMA	ARIMA	QFNN
Passengers	-0.26	-0.23	-0.42	-0.39	-0.25	-0.23	-0.17	-0.28	-0.43	-0.12
Sunspots	-0.62	-0.67	-0.55	-1.60	-0.55	-0.57	-1.37	-0.54	-0.55	-0.46
Load	-0.29	-0.27	-0.28	-2.12	-0.27	-0.43	-1.23	-0.44	-0.34	-0.22
Internet	-0.72	-0.67	-0.70	-1.32	-0.63	-0.57	-0.97	-9.80	-4.10	-0.44
Stock	-0.33	-0.30	-0.38	-2.78	-0.38	-0.41	-0.37	-0.82	-1.65	-0.50
Solar	-0.67	-0.94	-0.63	-0.62	-0.61	-0.61	-0.82	-0.36	-0.66	-0.24
Wind	-0.76	-0.75	-0.77	-2.63	-0.73	-2.24	-2.35	-1.54	-1.10	-0.71
Wave	-0.80	-1.19	-1.16	-1.70	-1.27	-1.66	-1.78	-1.49	-1.10	-0.45

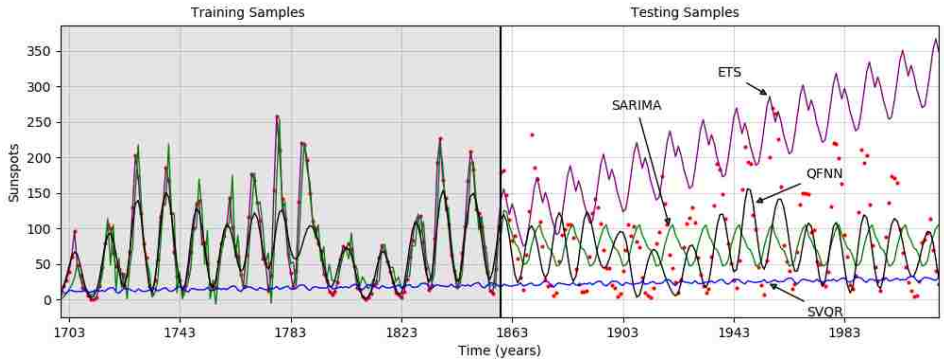

Figure 6.4: Forecast comparison of the median quantile for the Sunspots time series (red dots) by QFNN (shown in black), SVQR (shown in blue), SARIMA (shown in green), and ETS (shown in purple). SVQR fails to capture any meaningful pattern in its prediction. SRIMA captures a seasonal pattern that is out of phase with the sunspot test series, and ETS shots off in the test set with a positive trend. QFNN captured a seasonal pattern that is a bit more in phase with the number of sunspots over the years and is also able to learn multiple seasons of sunspots thus providing the most accurate quantile forecast of all the methods.

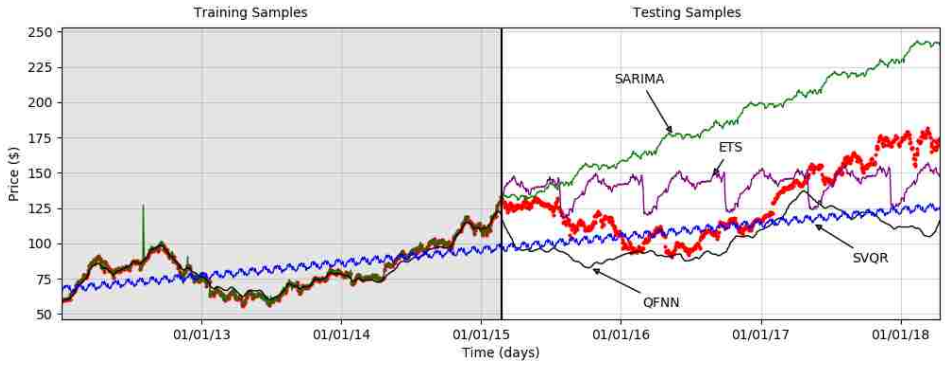

Figure 6.5: Forecast comparison of the median quantile for the Apple Closing Stock Price time series (red dots) by QFNN (shown in black), SVQR (shown in blue), SARIMA (shown in green), and ETS (shown in purple). SVQR can learn the linear trend of the stock series but nothing else. ETS learns a non-existing seasonal pattern, and SARIMA does not seem to capture any meaningful pattern. While QFNN does not have the highest accuracy regarding the QS and IS metrics we can see in the plot that it learns a cyclic trend of the stock price which follows the test set better than any other method.

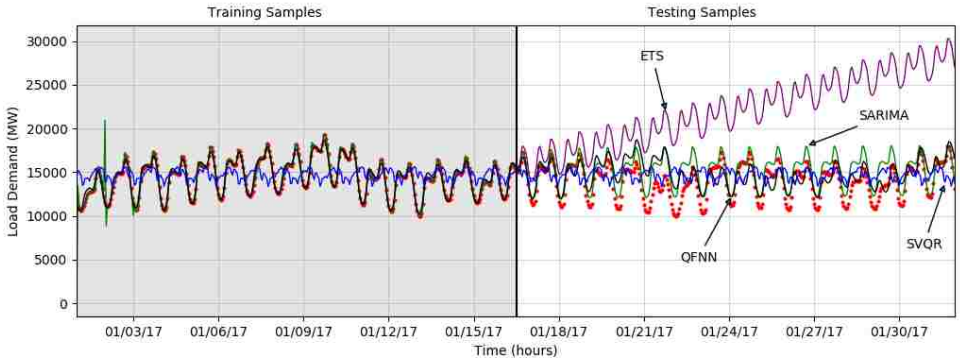

Figure 6.6: Forecast comparison of the median quantile for the Load Demand time series (red dots) by QFNN (shown in black), SVQR (shown in blue), SARIMA (shown in green), and ETS (shown in purple). SVQR captures a poor and small seasonal pattern. SRIMA captures the seasonality but fails to capture any cycles in the test set, and ETS shots off in the test set with a positive trend. QFNN learns both the seasonal and cyclical pattern of the load demand.

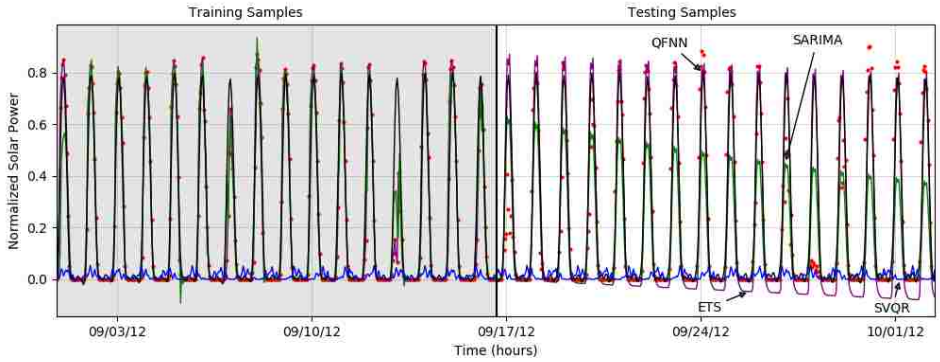

Figure 6.7: Forecast comparison of the median quantile for the Solar Power time series (red dots) by QFNN (shown in black), SVQR (shown in blue), SARIMA (shown in green), and ETS (shown in purple). SVQR is barely able to estimate the seasonality. SRIMA has a seasonal pattern reducing overtime in the test set, and while ETS captures the seasonality we see a negative trend. QFNN learns a constant seasonal quantile pattern that can be attributed to sunny days.

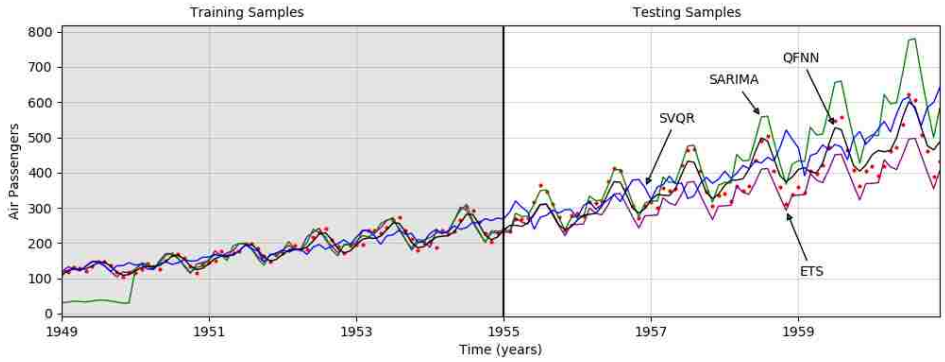

Figure 6.8: Forecast comparison of the median quantile for the Air Passengers time series (red dots) by QFNN (shown in black), SVQR (shown in blue), SARIMA (shown in green), and ETS (shown in purple). SVQR estimates the trend but not the seasonality so well. ETS and SARIMA estimate both trend and seasonality well, but the median forecasts fall below and above the test data. QFNN learns the shape of the data better and appropriately captures the median.

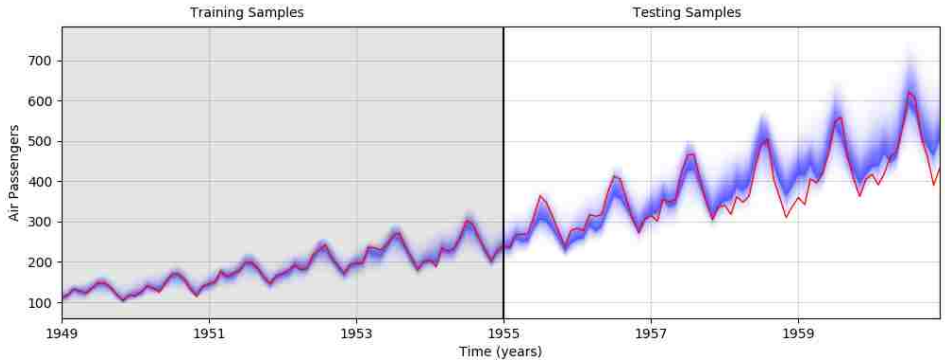

Figure 6.9: Probabilistic forecasting of 50 prediction intervals for the Air Passengers series.

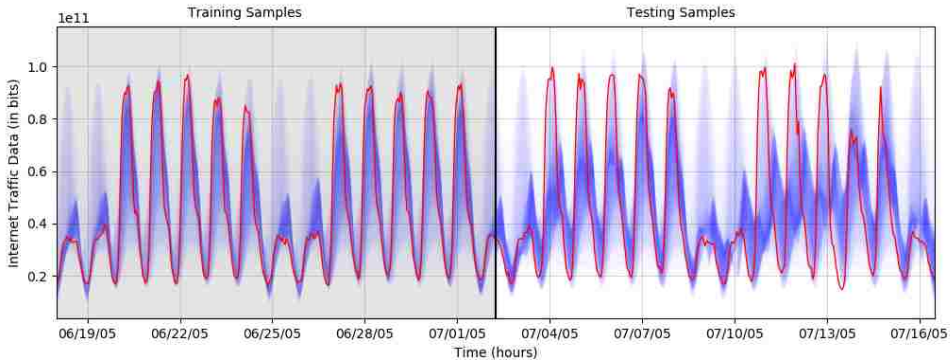

Figure 6.10: Probabilistic forecasting of 50 prediction intervals for the Internet Traffic series.

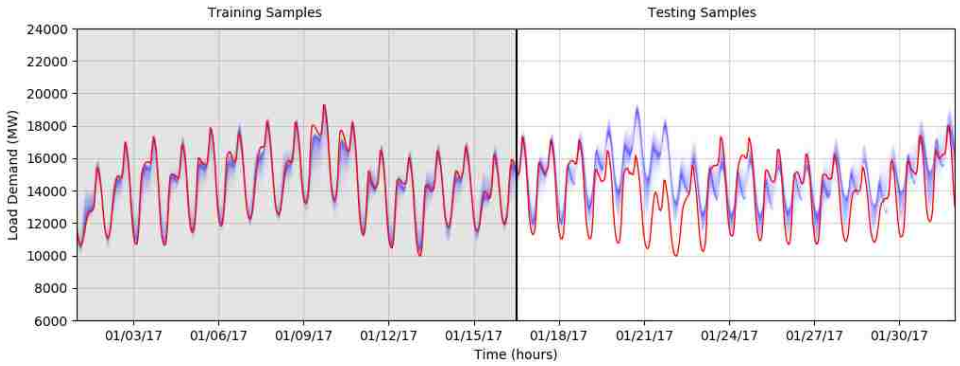

Figure 6.11: Probabilistic forecasting of 50 prediction intervals for the Load Demand series.

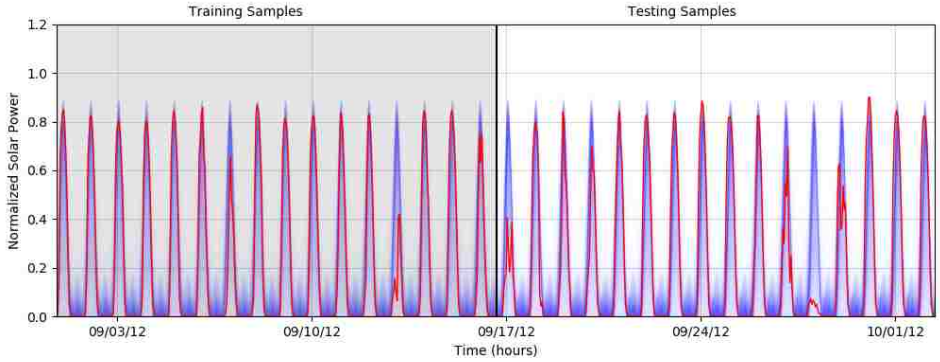

Figure 6.12: Probabilistic forecasting of 50 prediction intervals for the Solar Power series.

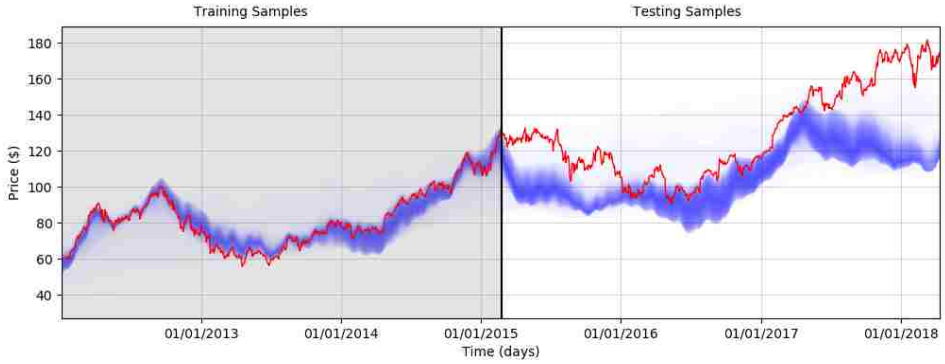

Figure 6.13: Probabilistic forecasting of 50 prediction intervals for the Apple Closing Stock Prices time series.

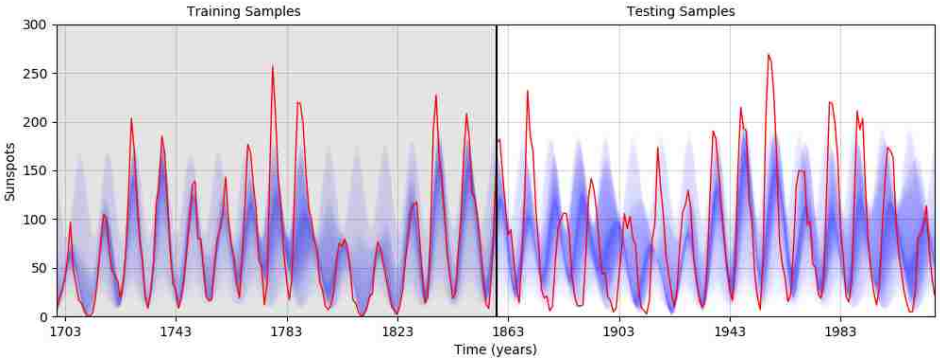

Figure 6.14: Probabilistic forecasting of 50 prediction intervals for the Sunspots time series.

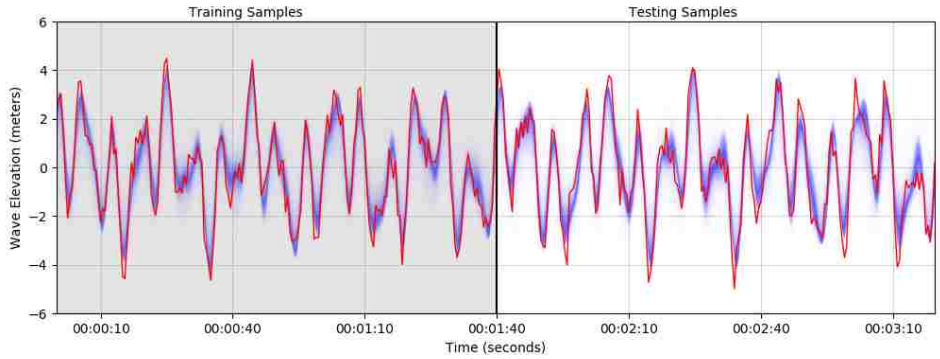

Figure 6.15: Probabilistic forecasting of 50 prediction intervals for the simulated Ocean Wave Elevation time series.

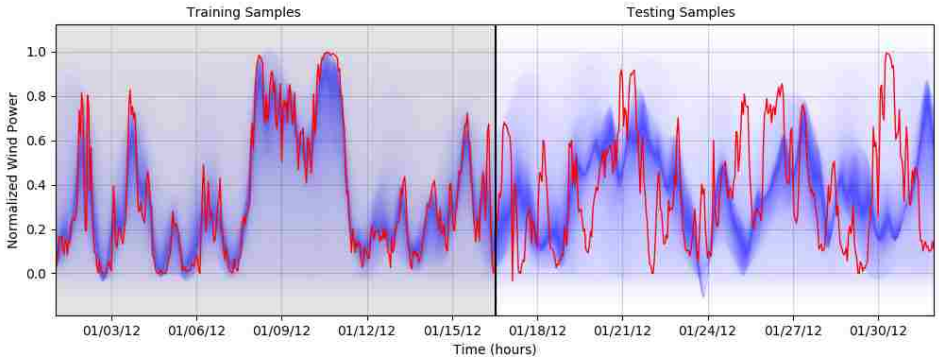

Figure 6.16: Probabilistic forecasting of 50 prediction intervals for the wind power time series.

Chapter 7

Conclusion

A paradigm shift is occurring in the world of prediction. Just like in the world of artificial intelligence where a shift is being made from shallow to deep learning methods, the world of forecasting is also changing. Where for decades the forecasting community was concerned in estimating single-valued or deterministic forecasts, we are now seeing a disciplinary transition to conducting full probabilistic forecasts. Such projections allow us to quantify the uncertainty in a prediction and therefore provide more information for optimal decision making in any field requiring forecasts.

From smart grid operations to the integration of renewable energies, probabilistic forecasts can significantly improve knowledge of uncertainty. In this dissertation, we set out to study the utilization of neural network architectures for nonparametric probabilistic forecasting. Neural network methods have shown incredible results in time series and regression problems. We study how they can be successfully applied for obtaining probabilistic forecasts in the form of quantiles or prediction intervals. We show the ability of neural networks to automatically learn features from raw data and how their capabilities yield significantly higher forecasting performance.

Before proposing probabilistic forecasting methods, we highlight the use of neural networks for deterministic forecasting in chapter 2. We showcase this chapter as a background to the fields of both neural network theory and time series forecasting. In chapter 2 we propose a novel method for using particle swarm optimization to train a non-linear autoregressive neural network. The accuracy of our approach

is tested using ocean wave heights to aid the integration of wave energy into the power grid. We believe our scheme improves multi-step prediction as needed in integrating stochastic renewable resources. Compared to existing methodologies, our method can be applied to other applications where forecasts are required for multiple time steps. Through our use of a stochastic inertial weight, in the PSO learning algorithm to train our NAR network, we show with simulated data successful results in predicting short term ocean wave levels.

In chapter 3 we presented an approach for modeling and analyzing the price to load feedback relationship in a cyber-enabled demand side management system. We showed vulnerabilities in this feedback and how an attack can launch load or price data attacks. We then presented change paint and supervised learning methods and discovered how linear techniques such as logistic regression were quite useful for attack detection. This DSM domain is a perfect example of a problem where linear methods are better than nonlinear ones. Thus the need for deep learning or advanced probabilistic forecasting is not warranted. However, we hypothesize that when renewable energy generation is introduced into this problem, the detection of attacks can become much more difficult. Renewable generation can bring a lot of uncertainty, and an attacker can exploit this uncertainty and amplify in the DSM problem. We are then motivated to conduct advanced probabilistic forecasting of renewables to help with integration, smart grid operations, and detection of DSM attacks in a stochastic environment. In the next several chapters we propose several novel probabilistic forecasting approaches.

In chapter 4 the issues associated with the generation and evaluation of wind power forecasts are first introduced in the form of quantiles and prediction intervals. Due to the stochastic nature of the wind, it is often difficult to forecast. Uncertainty analysis in the form of probabilistic wind prediction can provide a better picture of future wind coverage. In this chapter, we propose a framework for probabilistic forecasting using support vector quantile regression with non-crossing constraints to ensure multiple quantiles can be predicted without overlapping each other. Support vector machines are considered a shallow neural network. The effectiveness of our approach is validated with the real world dataset of the Global Energy Forecasting

Competition 2014. The relationship between estimation and scoring of quantiles are closely tied, and empirical results of our methods show reliability and low quantile scores across the prediction horizon.

Utilizing the same public data set, the Global Energy Forecasting Competition 2014, in chapter 5 we create a new deep feedforward neural network which we call SPNN. We use a novel smooth approximation to the pinball ball loss function in estimating multiple quantiles, and also incorporate non-crossing constraints in the form of a smooth penalty in the loss function. Such a neural network with this loss function and penalty has not been studied before. We compare forecasts to standard and advanced benchmarks and employ standard quantile score, reliability, and sharpness metrics. Our results show superior performance across the prediction horizons, which verify the effectiveness of the model for forecasting while preventing estimated quantiles from overlapping. Our deep network can automatically learn latent features in its hidden layers from raw data. Comparing with state-of-the-art benchmark methods, we show that no other method can generalize on raw data without feature engineering.

Our SPNN method has the potential to be applied to a variety of domains for probabilistic forecasting. However, it is limited in that it has no mechanisms to capture long term periodic patterns. Thus, in chapter 6 we radically extend SPNN to decompose univariate time series into a cosine Fourier series. The proposed model we dub the quantile Fourier neural network provides forecasts using extrapolation based regression instead of autoregression. Extrapolation based regression has not been studied before for probabilistic forecasting. We conduct empirical case studies not only on wind but on eight real world (and public) datasets. We show that such a Fourier decomposition of time series with our model is able to appropriately capture periodic and aperiodic components to provide high-quality probabilistic predictions.

Given the novelty of all our approaches, more research needs to be conducted to assess their application to more domains and under different scenarios. New studies could also look at other cost functions such as using the interval score directly. This could provide appropriate prediction intervals that may have even higher sharpness and reliability. Overall, the field of probabilistic neural forecasting is young. There

is still a vast forest of deep learning methods that could be explored, hybridized and evaluated for quantile and prediction interval estimation. In the next section, we quickly introduce several possible extensions for future work.

Future Work

From our research on the application of novel neural network frameworks for probabilistic forecasting, we plan several extensions for future work. The first of these is a stepwise quantile network as an alternative approach for possibly more efficiently utilized non-crossing constraints. Next, in our study of DSM attacks, we looked at various machine learning methods for detection which resulted in high accuracy of detection. We hypothesize this will not be the case when incorporating stochastic renewables into the problem and therefore provided several methods in this dissertation to accurately forecast the uncertainty of renewables. An alternative approach to forecasting renewables to is to forecasting demand in the presence of renewables. Solving such a problem, we propose the use of a quantile autoregressive neural network to estimate multi-step-ahead prediction intervals for anomaly or attack detection. Lastly, we propose extending our feedforward neural network approaches to study recurrent and convolutional architectures for quantile forecasting. Both deep architectures have shown promise in time series prediction [185, 186]. In the next sections, we briefly introduce all these proposed architectures.

7.0.1 Stepwise Quantile Networks for Non-crossing Constraints

One of the core tenants of this dissertation is the nonlinear estimation of multiple quantile regression functions. However, as we point out in early chapters, a phenomenon of quantile crossing may occur which violates the basic principle of distribution functions where their associated inverse functions should be monotone increasing. We introduce in chapter 5 our SPNN model with non-crossing constraints

in the form of a smooth penalty function in the loss. The equivalent number of constraints is equal to $M+1$ where we have M constraints plus an additional constraint to bound the upper or lower quantile between 0 and 1. In summary these constraints are $0 < \tau_1 < \tau_2 < ... < \tau_M < 1$. Most of these constraints are superfluous, in that not all quantiles will be crossing each other at all times. To improve the constrained estimation scheme, we propose a more efficient iterative method for SPNN which is based on stepwise quantile regression [187].

As an extension to SPNN, we propose a new method to estimate multiple quantiles without crossing. This estimation scheme would be a stepwise formulation to ensure the non-crossing of the nonlinear regression functions. The stepwise procedure is relatively simple. We start with one hidden layered network, with an input and output layer of parameters. We first train the network to estimate a single quantile such as the median $\tau = 0.5$. Then keeping the weights of the input layer and the output weights for $\tau = 0.5$ both fixed, we learn another set of output parameters to estimate the quantile $\tau = 0.6$. If the output violates the constraint where $\tau_{0.6} < \tau_{0.5}$, we re-estimate the weights for $\tau = 0.6$ but this time incorporating the smooth penalty in the loss function to prevent the cross over

$$p = c \sum_{t=1}^{N} \left[\max \left(0, \epsilon - \left(\hat{q}_t^{(\tau_{0.5})} - \hat{q}_t^{(\tau_{0.6})} \right) \right) \right]^2 \tag{7.1}$$

Effectively, we are only adding constraints in the estimation procedure when the next quantile regression function does not cross the current one. This procedure continues until SPNN outputs all desired τ levels.

7.0.2 Quantile Autoregressive Network for Detection

The detection of anomalous time series through multi-step ahead forecasts is a difficult task. In chapter 3 we have shown that linear methods could accurately detect attacks on DSM systems by estimating daily load usage with constant power generation. However, the problem can become significantly complicated when generation is stochastic such as by wind, wave, or solar power. In such cases, forecasting of

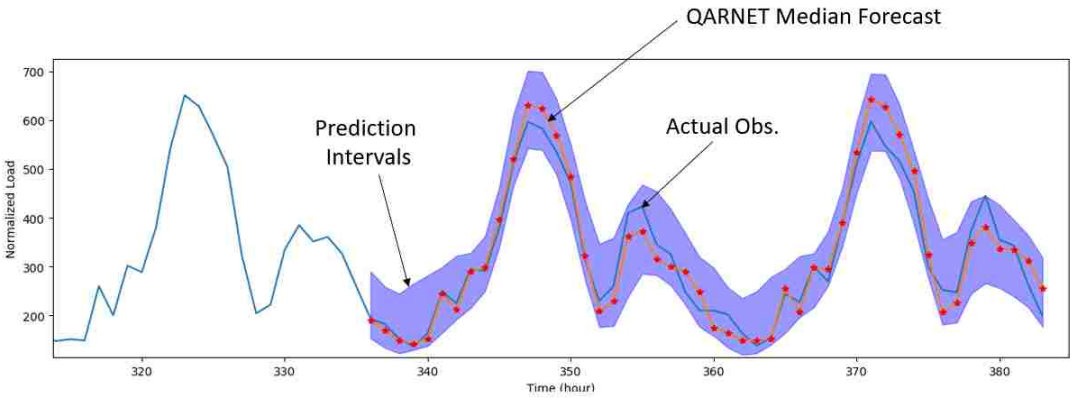

Figure 7.1: Example multi-step forecast from proposed QARNET model for load demand. Anomalous data is flagged when above or below the prediction intervals with a certain nominal coverage rate.

generation in relation to demand is considerably more difficult. Therefore, we propose a nonlinear quantile regression in the context of time series data and propose a quantile autoregression neural network (QARNET) model by adding an SPNN structure to the quantile autoregression (QAR) model. The linear QAR model with exogenous covariates is defined as

$$Q_{y_t}(\tau) = \alpha_0^\tau + \sum_{i=1}^{p} \alpha_i^\tau y_{t-i} + \sum_{j=1}^{p} \beta_j x_{t-j} \tag{7.2}$$

This model is easily extended to a neural network by using embedded nonlinear functions. The extended QARNET model could be more flexible as it can implement a quantile autoregression for time series data for probabilistic forecasting and estimate nonlinear relationships with lagged data or covariates.

Detection of anomalous time series through multi-step ahead forecasts is then conducted by forecasting $t + K$ step ahead prediction interval estimates. The exact nominal coverage rate of such intervals could be estimated a posteriori. Tens or hundreds of prediction intervals can be estimated during training. Then the intervals with the highest reliability and sharpness could be taken and then estimated for the testing data. With these intervals, if an observed future data point is above or below

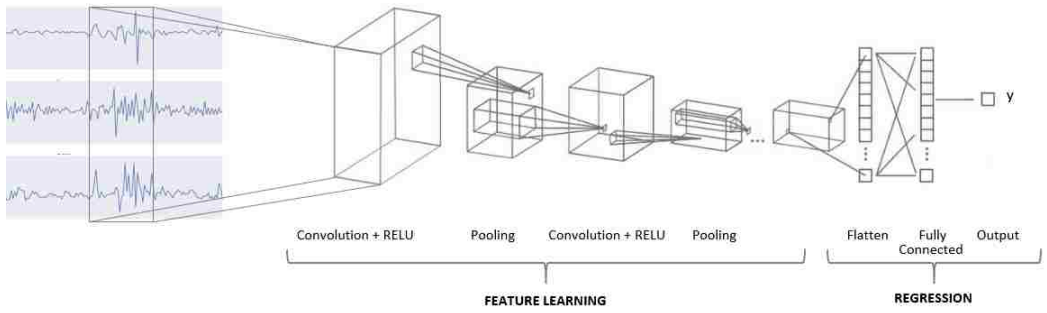

Figure 7.2: Example architecture of a convolutional neural network for regression.

the bounds, it is flagged as anomalous. This method may be considered as a deep nonparametric change point detector.

7.0.3 Convolutional and Recurrent Quantile Networks

Convolutional neural networks (CNN) [188] are an extension of feedforward neural networks. Their architecture, loosely inspired by the biological visual system, possess two key properties that make them very valuable. The first is spatially shared weights and second is spatial pooling. This kind of network automatically learns features that are shift-invariant (stationary). The pooling layers are responsible for reducing the sensitivity of the output to a slight input shift and distortions. Most common CNN architectures are composed of multiple stages, as shown in Fig. 7.2. The output of each stage is made of a set of two-dimensional arrays called feature maps. Each feature map is the outcome of one convolutional and pooling filter applied over the full input data set. A non-linear activation function, such as a rectified linear unit (RELU) follows a pooling layer.

This type of neural network has been shown to be extremely efficient in many image and computer vision applications, such as object recognition [189], segmentation [190], and classification [3]. Recently there have been several advances in CNN's for time series forecasting. In [191] the authors propose a CNN for time series modeling based on the wavelet transform. In [192] the authors propose to use

an autoregressive-type weighting system for forecasting financial time series, where the weights are allowed to be data-dependent by learning them through a CNN. In [15] an ensemble of CNN's is used to generate multiple point forecasts of wind power from which a density function could be estimated.

Despite recent works, there is still little literature on convolutional and deep neural networks for time-series prediction, and even fewer works on their application to probabilistic forecasting. For future work, we plan to study extensions of SPNN into a CNN architecture and evaluate its efficacy with multivariate inputs and outputs. We also plan similar studies of extending our SPNN and QFNN models into recurrent architectures. A recurrent neural network (RNN) is a class of methods where connections between nodes form a directed graph along a sequence which allows it model temporal dynamic behavior for a time series. RNNs can thus use their internal state to process sequences of inputs and learn hidden or nonlinear correlations between sequences. RNN models have shown tremendous success for time series prediction, particularly variants that utilize long short-term memory (LSTM) units [193]. For probabilistic forecasting, recently [194] shows how to use ensemble RNNs to make point forecasts and then derive distributional information, and in [195] prediction intervals are estimated using a lower upper bound estimation method. Like CNNs, there is overall limited literature on the subject of probabilistic forecasting and RNNs, particularly in conditional quantile estimation.

Bibliography

[1] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[2] T. Gneiting and M. Katzfuss, "Probabilistic forecasting," *Annual Review of Statistics and Its Application*, vol. 1, pp. 125–151, 2014.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[4] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[6] L. Deng, D. Yu *et al.*, "Deep learning: methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[7] M. A. Nielsen, *Neural networks and deep learning*. Determination press USA, 2015, vol. 25.

[8] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman, "Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond," 2016.

[9] "Gefcom2017: Hierarchical probabilistic load forecasting," Oct 2016. [Online]. Available: http://blog.drhongtao.com/2016/10/gefcom2017-hierarchical-probabilistic-load-forecasting.html

[10] G. Giebel, L. Landberg, J. Badger, K. Sattler, H. Feddersen, T. S. Nielsen, H. A. Nielsen, and H. Madsen, "Using ensemble forecasting for wind power," *Proceedings Cd-rom. Cd 2*, 2003.

[11] A. M. Foley, P. G. Leahy, A. Marvuglia, and E. J. McKeogh, "Current methods and advances in forecasting of wind power generation," *Renewable Energy*, vol. 37, no. 1, pp. 1–8, 2012.

[12] G. Sideratos and N. D. Hatziargyriou, "Probabilistic wind power forecasting using radial basis function neural networks," *IEEE Transactions on Power Systems*, vol. 27, no. 4, pp. 1788–1796, 2012.

[13] C. Wan, J. Lin, J. Wang, Y. Song, and Z. Y. Dong, "Direct quantile regression for nonparametric probabilistic forecasting of wind power generation," *IEEE Transactions on Power Systems*, vol. 32, no. 4, pp. 2767–2778, 2017.

[14] A. U. Haque, M. H. Nehrir, and P. Mandal, "A hybrid intelligent model for deterministic and quantile regression approach for probabilistic wind power forecasting," *IEEE Transactions on Power Systems*, vol. 29, no. 4, pp. 1663–1672, 2014.

[15] H.-z. Wang, G.-q. Li, G.-b. Wang, J.-c. Peng, H. Jiang, and Y.-t. Liu, "Deep learning based ensemble approach for probabilistic wind power forecasting," *Applied energy*, vol. 188, pp. 56–70, 2017.

[16] J. W. Taylor, P. E. McSharry, and R. Buizza, "Wind power density forecasting using ensemble predictions and time series models," *IEEE Transactions on Energy Conversion*, vol. 24, no. 3, pp. 775–782, 2009.

[17] Y. Zhang, J. Wang, and X. Wang, "Review on probabilistic forecasting of wind power generation," *Renewable and Sustainable Energy Reviews*, vol. 32, pp. 255–270, 2014.

[18] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Lower upper bound estimation method for construction of neural network-based prediction intervals," *IEEE Transactions on Neural Networks*, vol. 22, no. 3, pp. 337–346, 2011.

[19] H. Quan, D. Srinivasan, and A. Khosravi, "Short-term load and wind power forecasting using neural network-based prediction intervals," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 2, pp. 303–315, 2014.

[20] D. van der Meer, J. Widén, and J. Munkhammar, "Review on probabilistic forecasting of photovoltaic power production and electricity consumption," *Renewable and Sustainable Energy Reviews*, 2017.

[21] J. Nowotarski and R. Weron, "Recent advances in electricity price forecasting: A review of probabilistic forecasting," *Renewable and Sustainable Energy Reviews*, 2017.

[22] A. S. Dorvlo, "Estimating wind speed distribution," *Energy Conversion and Management*, vol. 43, no. 17, pp. 2311–2318, 2002.

[23] P. Pinson, H. A. Nielsen, J. K. Møller, H. Madsen, and G. N. Kariniotakis, "Non-parametric probabilistic forecasts of wind power: required properties and evaluation," *Wind Energy*, vol. 10, no. 6, pp. 497–516, 2007.

[24] R. Koenker and G. Bassett Jr, "Regression quantiles," *Econometrica: journal of the Econometric Society*, pp. 33–50, 1978.

[25] R. Koenker, *Quantile regression*. Cambridge university press, 2005, no. 38.

[26] J. B. Bremnes, "Probabilistic wind power forecasts using local quantile regression," *Wind Energy*, vol. 7, no. 1, pp. 47–54, 2004.

[27] H. A. Nielsen, H. Madsen, and T. S. Nielsen, "Using quantile regression to extend an existing wind power forecasting system with probabilistic forecasts," *Wind Energy*, vol. 9, no. 1-2, pp. 95–108, 2006.

[28] M. Landry, T. P. Erlinger, D. Patschke, and C. Varrichio, "Probabilistic gradient boosting machines for gefcom2014 wind forecasting," *International Journal of Forecasting*, vol. 32, no. 3, pp. 1061–1066, 2016.

[29] R. Juban, H. Ohlsson, M. Maasoumy, L. Poirier, and J. Z. Kolter, "A multiple quantile regression approach to the wind, solar, and price tracks of gefcom2014," *International Journal of Forecasting*, vol. 32, no. 3, pp. 1094–1102, 2016.

[30] J. Juban, L. Fugon, and G. Kariniotakis, "Uncertainty estimation of wind power forecasts: Comparison of probabilistic modelling approaches," in *European Wind Energy Conference & Exhibition EWEC 2008*. EWEC, 2008, pp. 10–pages.

[31] J. W. Taylor, "A quantile regression neural network approach to estimating the conditional density of multiperiod returns," *Journal of Forecasting*, vol. 19, no. 4, pp. 299–311, 2000.

[32] Y. Feng, R. Li, A. Sudjianto, and Y. Zhang, "Robust neural network with applications to credit portfolio data analysis," *Statistics and its Interface*, vol. 3, no. 4, p. 437, 2010.

[33] Q. Xu, X. Liu, C. Jiang, and K. Yu, "Quantile autoregression neural network model with applications to evaluating value at risk," *Applied Soft Computing*, vol. 49, pp. 1–12, 2016.

[34] A. J. Cannon, "Quantile regression neural networks: Implementation in r and application to precipitation downscaling," *Computers & Geosciences*, vol. 37, no. 9, pp. 1277–1284, 2011.

[35] Y. Grushka-Cockayne, K. C. Lichtendahl, V. R. R. Jose, and R. L. Winkler, "Quantile evaluation, sensitivity to bracketing, and sharing business payoffs," 2016.

[36] P. Friederichs and A. Hense, "Statistical downscaling of extreme precipitation events using censored quantile regression," *Monthly weather review*, vol. 135, no. 6, pp. 2365–2378, 2007.

[37] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[38] G. E. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis: forecasting and control*. John Wiley & Sons, 2013.

[39] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International journal of forecasting*, vol. 22, no. 3, pp. 443–473, 2006.

[40] M. Belmont, "Increases in the average power output of wave energy converters using quiescent period predictive control," *Renewable Energy*, vol. 35, no. 12, pp. 2812–2820, 2010.

[41] J. Hals, J. Falnes, and T. Moan, "A comparison of selected strategies for adaptive control of wave energy converters," *Journal of Offshore Mechanics and Arctic Engineering*, vol. 133, no. 3, p. 031101, 2011.

[42] G. Li, G. Weiss, M. Mueller, S. Townley, and M. R. Belmont, "Wave energy converter control by wave prediction and dynamic programming," *Renewable Energy*, vol. 48, pp. 392–403, 2012.

[43] K. Hatalis, P. Pradhan, S. Kishore, R. Blum, and A. Lamadrid, "Multi-step forecasting of wave power using a nonlinear recurrent neural network," *Power and Energy Society General Meeting, 2014 IEEE*, 2014.

[44] K. Hornik, "Some new results on neural network approximation," *Neural Networks*, vol. 6, no. 8, pp. 1069–1072, 1993.

[45] P. A. Janssen, "Progress in ocean wave forecasting," *Journal of Computational Physics*, vol. 227, no. 7, pp. 3572–3594, 2008.

[46] F. Fusco and J. V. Ringwood, "Short-term wave forecasting for real-time control of wave energy converters," *Sustainable Energy, IEEE Transactions on*, vol. 1, no. 2, pp. 99–106, 2010.

[47] H. B. Hwarng, "Insights into neural-network forecasting of time series corresponding to arma structures," *Omega*, vol. 29, no. 3, pp. 273–289, 2001.

[48] M. Deo and S. Jagdale, "Prediction of breaking waves with neural networks," *Ocean Engineering*, vol. 30, no. 9, pp. 1163–1178, 2003.

[49] J. Kennedy, R. Eberhart *et al.*, "Particle swarm optimization," in *Proceedings of IEEE international conference on neural networks*, vol. 4, no. 2. Perth, Australia, 1995, pp. 1942–1948.

[50] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Evolutionary Programming VII*. Springer, 1998, pp. 611–616.

[51] J.-R. Zhang, J. Zhang, T.-M. Lok, and M. R. Lyu, "A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training," *Applied Mathematics and Computation*, vol. 185, no. 2, pp. 1026–1037, 2007.

[52] R. Mendes, P. Cortez, M. Rocha, and J. Neves, "Particle swarms for feedforward neural network training," *learning*, vol. 6, no. 1, 2002.

[53] V. G. Gudise and G. K. Venayagamoorthy, "Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks," in *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*. IEEE, 2003, pp. 110–117.

[54] M. Carvalho and T. B. Ludermir, "An analysis of pso hybrid algorithms for feed-forward neural networks training," in *Neural Networks, 2006. SBRN'06. Ninth Brazilian Symposium on.* IEEE, 2006, pp. 6–11.

[55] F. Valdez, P. Melin, and O. Castillo, "Modular neural networks architecture optimization with a new nature inspired method using a fuzzy combination of particle swarm optimization and genetic algorithms," *Information Sciences*, vol. 270, pp. 143–153, 2014.

[56] P. Melin, F. Olivas, O. Castillo, F. Valdez, J. Soria, and M. Valdez, "Optimal design of fuzzy classification systems using pso with dynamic parameter adaptation through fuzzy logic," *Expert Systems with Applications*, vol. 40, no. 8, pp. 3196–3206, 2013.

[57] Q. Shen, W.-M. Shi, W. Kong, and B.-X. Ye, "A combination of modified particle swarm optimization algorithm and support vector machine for gene selection and tumor classification," *Talanta*, vol. 71, no. 4, pp. 1679–1683, 2007.

[58] S. M. Santos, M. J. Valença, and C. J. Bastos-Filho, "Comparing particle swarm optimization approaches for training multi-layer perceptron neural networks for forecasting," in *Intelligent Data Engineering and Automated Learning-IDEAL 2012.* Springer, 2012, pp. 344–351.

[59] G. K. Jha, P. Thulasiraman, and R. K. Thulasiram, "Pso based neural network for time series forecasting," in *Neural Networks, 2009. IJCNN 2009. International Joint Conference on.* IEEE, 2009, pp. 1422–1427.

[60] Z. Bashir and M. El-Hawary, "Applying wavelets to short-term load forecasting using pso-based neural networks," *Power Systems, IEEE Transactions on*, vol. 24, no. 1, pp. 20–27, 2009.

[61] L. Zhao and Y. Yang, "Pso-based single multiplicative neuron model for time series prediction," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2805–2812, 2009.

[62] P. S. de M Neto, G. G. Petry, R. Aranildo, and T. A. Ferreira, "Combining artificial neural network and particle swarm system for time series forecasting," in *Neural Networks, 2009. IJCNN 2009. International Joint Conference on.* IEEE, 2009, pp. 2230–2237.

[63] R. J. Duro and J. S. Reyes, "Discrete-time backpropagation for training synaptic delay-based artificial neural networks," *Neural Networks, IEEE Transactions on*, vol. 10, no. 4, pp. 779–789, 1999.

[64] H. T. Siegelmann, B. G. Horne, and C. L. Giles, "Computational capabilities of recurrent narx neural networks," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 27, no. 2, pp. 208–215, 1997.

[65] H. Xie, H. Tang, and Y.-H. Liao, "Time series prediction based on narx neural networks: An advanced approach," in *Machine Learning and Cybernetics, 2009 International Conference on*, vol. 3. IEEE, 2009, pp. 1275–1279.

[66] A. El-Gallad, M. El-Hawary, A. Sallam, and A. Kalas, "Enhancing the particle swarm optimizer via proper parameters selection," in *Electrical and Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on*, vol. 2. IEEE, 2002, pp. 792–797.

[67] S. Sumathi and S. Paneerselvam, *Computational intelligence paradigms: theory & applications using MATLAB.* CRC Press, 2010.

[68] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on.* IEEE, 1998, pp. 69–73.

[69] J. Falnes, *Ocean waves and oscillating systems.* Cambridge University Press, 2004.

[70] Y. Goda, *Random seas and design of maritime structures*, 2010.

[71] D. Hasselmann, M. Dunckel, and J. Ewing, "Directional wave spectra observed during jonswap 1973," *Journal of physical oceanography*, vol. 10, no. 8, pp. 1264–1280, 1980.

[72] R. Tibshirani, "A comparison of some error estimates for neural network models," *Neural Computation*, vol. 8, no. 1, pp. 152–163, 1996.

[73] C. J. Bastos-Filho, D. F. Carvalho, E. M. Figueiredo, and P. B. de Miranda, "Dynamic clan particle swarm optimization," in *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on.* IEEE, 2009, pp. 249–254.

[74] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information processing letters*, vol. 85, no. 6, pp. 317–325, 2003.

[75] K. Socha and C. Blum, "An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training," *Neural Computing and Applications*, vol. 16, no. 3, pp. 235–247, 2007.

[76] P. Palensky and D. Dietrich, "Demand side management: Demand response, intelligent energy systems, and smart loads," *IEEE transactions on industrial informatics*, vol. 7, no. 3, pp. 381–388, 2011.

[77] T. Logenthiran, D. Srinivasan, and T. Z. Shun, "Demand side management in smart grid using heuristic optimization," *IEEE transactions on smart grid*, vol. 3, no. 3, pp. 1244–1252, 2012.

[78] A.-H. Mohsenian-Rad, V. W. 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*, vol. 1, no. 3, pp. 320–331, 2010.

[79] L. Gelazanskas and K. A. Gamage, "Demand side management in smart grid: A review and proposals for future direction," *Sustainable Cities and Society*, vol. 11, pp. 22–30, 2014.

[80] A. C. Batista and L. S. Batista, "Demand side management using a multi-criteria -constraint based exact approach," *Expert Systems with Applications*, vol. 99, pp. 180–192, 2018.

[81] A. R. Khan, A. Mahmood, A. Safdar, Z. A. Khan, and N. A. Khan, "Load forecasting, dynamic pricing and dsm in smart grid: A review," *Renewable and Sustainable Energy Reviews*, vol. 54, pp. 1311–1322, 2016.

[82] G. Dudek, "Pattern-based local linear regression models for short-term load forecasting," *Electric Power Systems Research*, vol. 130, pp. 139–147, 2016.

[83] S. S. Pappas, L. Ekonomou, P. Karampelas, D. Karamousantas, S. Katsikas, G. Chatzarakis, and P. Skafidas, "Electricity demand load forecasting of the hellenic power system using an arma model," *Electric Power Systems Research*, vol. 80, no. 3, pp. 256–264, 2010.

[84] W. Christiaanse, "Short-term load forecasting using general exponential smoothing," *IEEE Transactions on Power Apparatus and Systems*, no. 2, pp. 900–911, 1971.

[85] J. W. Taylor, "Triple seasonal methods for short-term electricity demand forecasting," *European Journal of Operational Research*, vol. 204, no. 1, pp. 139–152, 2010.

[86] A. Kavousi-Fard, H. Samet, and F. Marzbani, "A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting," *Expert systems with applications*, vol. 41, no. 13, pp. 6047–6056, 2014.

[87] A. Baliyan, K. Gaurav, and S. K. Mishra, "A review of short term load forecasting using artificial neural network models," *Procedia Computer Science*, vol. 48, pp. 121–125, 2015.

[88] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on lstm recurrent neural network," *IEEE Transactions on Smart Grid*, 2017.

[89] G. Dudek, "Short-term load forecasting using random forests," in *Intelligent Systems' 2014*. Springer, 2015, pp. 821–828.

[90] S. Li, L. Goel, and P. Wang, "An ensemble approach for short-term load forecasting by extreme learning machine," *Applied Energy*, vol. 170, pp. 22–29, 2016.

[91] S. Chakraborty, T. Ito, and T. Senjyu, "Smart pricing scheme: A multi-layered scoring rule application," *Expert Systems with Applications*, vol. 41, no. 8, pp. 3726–3735, 2014.

[92] N. Nazar, M. Abdullah, M. Hassan, and F. Hussin, "Time-based electricity pricing for demand response implementation in monopolized electricity market," in *Research and Development (SCOReD), 2012 IEEE Student Conference on*. IEEE, 2012, pp. 178–181.

[93] M. Burger, B. Klar, A. Miller, and G. Schindlmayr, "A spot market model for pricing derivatives in electricity markets," *Quantitative finance*, vol. 4, no. 1, pp. 109–122, 2004.

[94] R. D. Zimmerman, C. E. Murillo-Sánchez, and D. Gan, "Matpower: A matlab power system simulation package," *Manual, Power Systems Engineering Research Center, Ithaca NY*, vol. 1, 1997.

[95] D. P. Chassin, K. Schneider, and C. Gerkensmeyer, "Gridlab-d: An open-source power systems modeling and simulation environment," in *Transmission*

and distribution conference and exposition, 2008. t&d. IEEE/PES. IEEE, 2008, pp. 1–5.

[96] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht, "Smart*: An open data set and tools for enabling research in sustainable homes," *SustKDD, August*, vol. 111, no. 112, p. 108, 2012.

[97] P. J. Brockwell, R. A. Davis, and M. V. Calder, *Introduction to time series and forecasting*. Springer, 2002, vol. 2.

[98] D. Gamerman and H. F. Lopes, *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. Chapman and Hall/CRC, 2006.

[99] D. N. Politis and H. White, "Automatic block-length selection for the dependent bootstrap," *Econometric Reviews*, vol. 23, no. 1, pp. 53–70, 2004.

[100] P. R. Thimmapuram, J. Kim, A. Botterud, and Y. Nam, "Modeling and simulation of price elasticity of demand using an agent-based model," in *Innovative Smart Grid Technologies (ISGT), 2010*. IEEE, 2010, pp. 1–8.

[101] G. Liang, J. Zhao, F. Luo, S. R. Weller, and Z. Y. Dong, "A review of false data injection attacks against modern power systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 4, pp. 1630–1638, 2017.

[102] A. Tartakovsky, I. Nikiforov, and M. Basseville, *Sequential analysis: Hypothesis testing and changepoint detection*. Chapman and Hall/CRC, 2014.

[103] S. Key, "Fundamentals of statistical signal processing, volume ii: Detection theory," 1993.

[104] R. C. B. Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," in *2014 7th International symposium on resilient control systems (ISRCS)*, 2014, pp. 1–8.

[105] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor, "Machine learning methods for attack detection in the smart grid," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 8, pp. 1773–1786, 2016.

[106] M. Esmalifalak, L. Liu, N. Nguyen, R. Zheng, and Z. Han, "Detecting stealthy false data injection using machine learning in smart grid," *IEEE Systems Journal*, vol. 11, no. 3, pp. 1644–1652, 2017.

[107] C. Bishop, "Pattern recognition and machine learning," *Pattern Recognition and Machine Learning*, 2006.

[108] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[109] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with python," in *Proceedings of the 9th Python in Science Conference*, vol. 57. SciPy society Austin, 2010, p. 61.

[110] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[111] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[112] K. Hatalis, "Lehighdsm python package," https://github.com/, 2019.

[113] P. Pinson and G. Kariniotakis, "On-line assessment of prediction risk for wind power production forecasts," *Wind Energy*, vol. 7, no. 2, pp. 119–132, 2004.

[114] I. Takeuchi, Q. V. Le, T. D. Sears, and A. J. Smola, "Nonparametric quantile estimation," *Journal of Machine Learning Research*, vol. 7, no. Jul, pp. 1231–1264, 2006.

[115] E. Mangalova and O. Shesterneva, "K-nearest neighbors for gefcom2014 probabilistic wind power forecasting," *International Journal of Forecasting*, vol. 32, no. 3, pp. 1067–1073, 2016.

[116] C. Wan, Z. Xu, P. Pinson, Z. Y. Dong, and K. P. Wong, "Probabilistic forecasting of wind power generation using extreme learning machine," *IEEE Transactions on Power Systems*, vol. 29, no. 3, pp. 1033–1044, 2014.

[117] J. Platt *et al.*, "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.

[118] P. Pinson and G. Kariniotakis, "Conditional prediction intervals of wind power generation," *IEEE Transactions on Power Systems*, vol. 25, no. 4, pp. 1845–1856, 2010.

[119] J. Quinonero-Candela, C. E. Rasmussen, F. Sinz, O. Bousquet, and B. Schölkopf, "Evaluating predictive uncertainty challenge," pp. 1–27, 2006.

[120] Y. V. Makarov, C. Loutan, J. Ma, and P. De Mello, "Operational impacts of wind generation on california power systems," *IEEE Transactions on Power Systems*, vol. 24, no. 2, pp. 1039–1050, 2009.

[121] W. A. Bukhsh, C. Zhang, and P. Pinson, "An integrated multiperiod opf model with demand response and renewable generation uncertainty," *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1495–1503, 2016.

[122] A. Botterud, Z. Zhou, J. Wang, R. J. Bessa, H. Keko, J. Sumaili, and V. Miranda, "Wind power trading under uncertainty in lmp markets," *IEEE Transactions on power systems*, vol. 27, no. 2, pp. 894–903, 2012.

[123] C. Monteiro, R. Bessa, V. Miranda, A. Botterud, J. Wang, G. Conzelmann *et al.*, "Wind power forecasting: state-of-the-art 2009." Argonne National Laboratory (ANL), Tech. Rep., 2009.

[124] R. Doherty and M. O'Malley, "A new approach to quantify reserve demand in systems with significant installed wind capacity," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 587–595, 2005.

[125] J. Usaola, O. Ravelo, G. González, F. Soto, M. C. Dávila, and B. Díaz-Guerra, "Benefits for wind energy in electricity markets from using short term wind power prediction tools; a simulation study," *Wind Engineering*, vol. 28, no. 1, pp. 119–127, 2004.

[126] E. D. Castronuovo and J. P. Lopes, "On the optimization of the daily operation of a wind-hydro power plant," *IEEE Transactions on Power Systems*, vol. 19, no. 3, pp. 1599–1606, 2004.

[127] G. N. Bathurst, J. Weatherill, and G. Strbac, "Trading wind generation in short term energy markets," *IEEE Transactions on Power Systems*, vol. 17, no. 3, pp. 782–789, 2002.

[128] P. Pinson, C. Chevallier, and G. N. Kariniotakis, "Trading wind generation from short-term probabilistic forecasts of wind power," *IEEE Transactions on Power Systems*, vol. 22, no. 3, pp. 1148–1156, 2007.

[129] T. Karakatsanis and N. Hatziargyriou, "Probabilistic constrained load flow based on sensitivity analysis," *IEEE Transactions on Power Systems*, vol. 9, no. 4, pp. 1853–1860, 1994.

[130] M. Lange, "On the uncertainty of wind power predictionsanalysis of the forecast accuracy and statistical distribution of errors," *Journal of Solar Energy Engineering*, vol. 127, no. 2, pp. 177–184, 2005.

[131] J. L. Powell, "Least absolute deviations estimation for the censored regression model," *Journal of Econometrics*, vol. 25, no. 3, pp. 303–325, 1984.

[132] ——, "Censored regression quantiles," *Journal of econometrics*, vol. 32, no. 1, pp. 143–155, 1986.

[133] K. Yu and M. Jones, "Local linear quantile regression," *Journal of the American statistical Association*, vol. 93, no. 441, pp. 228–237, 1998.

[134] X. Chen, R. Koenker, and Z. Xiao, "Copula-based nonlinear quantile autoregression," *The Econometrics Journal*, vol. 12, no. s1, pp. S50–S67, 2009.

[135] C. Hwang and J. Shim, "A simple quantile regression via support vector machine," *Advances in natural computation*, pp. 418–418, 2005.

[136] I. Takeuchi and T. Furuhashi, "Non-crossing quantile regressions by svm," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 1. IEEE, 2004, pp. 401–406.

[137] C. Chen, "A finite smoothing algorithm for quantile regression," *Journal of Computational and Graphical Statistics*, vol. 16, no. 1, pp. 136–164, 2007.

[138] Q. Xu, K. Deng, C. Jiang, F. Sun, and X. Huang, "Composite quantile regression neural network with applications," *Expert Systems with Applications*, vol. 76, pp. 129–139, 2017.

[139] C. Chen and O. L. Mangasarian, "A class of smoothing functions for nonlinear and mixed complementarity problems," *Computational Optimization and applications*, vol. 5, no. 2, pp. 97–138, 1996.

[140] Y.-J. Lee and O. L. Mangasarian, "Ssvm: A smooth support vector machine for classification," *Computational optimization and Applications*, vol. 20, no. 1, pp. 5–22, 2001.

[141] S. Zheng, "Gradient descent algorithms for quantile regression with smooth approximation," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 3, p. 191, 2011.

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

[143] A. J. Cannon, "Non-crossing nonlinear regression quantiles by monotone composite quantile regression neural network, with application to rainfall extremes," 2017.

[144] R. M. Freund, "Penalty and barrier methods for constrained optimization," 2004.

[145] J. Xue, S. Zhou, Q. Liu, X. Liu, and J. Yin, "Financial time series prediction using 2, 1rf-elm," *Neurocomputing*, 2017.

[146] Y. Hu, C. Hu, S. Fu, P. Shi, and B. Ning, "Predicting the popularity of viral topics based on time series forecasting," *Neurocomputing*, vol. 210, pp. 55–65, 2016.

[147] T. Zhou, G. Han, X. Xu, Z. Lin, C. Han, Y. Huang, and J. Qin, "δ-agree adaboost stacked autoencoder for short-term traffic flow forecasting," *Neurocomputing*, vol. 247, pp. 31–38, 2017.

[148] J. Yan, K. Li, E. Bai, Z. Yang, and A. Foley, "Time series wind power forecasting based on variant gaussian process and tlbo," *Neurocomputing*, vol. 189, pp. 135–144, 2016.

[149] L. B. Godfrey and M. S. Gashler, "Neural decomposition of time-series data for effective generalization," *arXiv preprint arXiv:1705.09137*, 2017.

[150] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[151] Y. Yang, S. Li, W. Li, and M. Qu, "Power load probability density forecasting using gaussian process quantile regression," *Applied Energy*, 2017.

[152] S. B. Taieb, R. Huser, R. J. Hyndman, and M. G. Genton, "Forecasting uncertainty in electricity smart meter data by boosting additive quantile regression," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2448–2455, 2016.

[153] W. Su, J. Wang, and J. Roh, "Stochastic energy scheduling in microgrids with intermittent renewable energy resources," *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 1876–1883, 2014.

[154] T. Hong, J. Wilson, and J. Xie, "Long term probabilistic load forecasting and normalization with hourly information," *IEEE Transactions on Smart Grid*, vol. 5, no. 1, pp. 456–462, 2014.

[155] A. Botterud, Z. Zhou, J. Wang, J. Sumaili, H. Keko, J. Mendes, R. J. Bessa, and V. Miranda, "Demand dispatch and probabilistic wind power forecasting in unit commitment and economic dispatch: A case study of illinois," *IEEE Transactions on Sustainable Energy*, vol. 4, no. 1, pp. 250–261, 2013.

[156] J. R. Andrade, J. Filipe, M. Reis, and R. J. Bessa, "Probabilistic price forecasting for day-ahead and intraday markets: Beyond the statistical model," *Sustainability*, vol. 9, no. 11, p. 1990, 2017.

[157] C.-M. Huang, Y.-C. Huang, K.-Y. Huang, S.-J. Chen, and S.-P. Yang, "Deterministic and probabilistic wind power forecasting using a hybrid method," in *Industrial Technology (ICIT), 2017 IEEE International Conference on*. IEEE, 2017, pp. 400–405.

[158] M. S. Gashler and S. C. Ashmore, "Modeling time series data with deep fourier neural networks," *Neurocomputing*, vol. 188, pp. 3–11, 2016.

[159] K. E. Germec, "Fourier neural networks for real-time harmonic analysis," in *Signal Processing and Communications Applications Conference, 2009. SIU 2009. IEEE 17th.* IEEE, 2009, pp. 333–336.

[160] G. Parascandolo, H. Huttunen, and T. Virtanen, "Taming the waves: sine as activation function in deep neural networks," 2016.

[161] A. Silvescu, "A new kind of neural networks," 1997.

[162] ——, "Fourier neural networks," in *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, vol. 1. IEEE, 1999, pp. 488–491.

[163] K.-i. Minami, H. Nakajima, and T. Toyoshima, "Real-time discrimination of ventricular tachyarrhythmia with fourier-transform neural network," *IEEE transactions on Biomedical Engineering*, vol. 46, no. 2, pp. 179–185, 1999.

[164] L. Mingo, L. Aslanyan, J. Castellanos, M. Diaz, and V. Riazanov, "Fourier neural networks: An approach with sinusoidal activation functions," 2004.

[165] H. Tan, "Fourier neural networks and generalized single hidden layer networks in aircraft engine fault diagnostics," *Journal of engineering for gas turbines and power*, vol. 128, no. 4, pp. 773–782, 2006.

[166] W. Zuo, Y. Zhu, and L. Cai, "Fourier-neural-network-based learning control for a class of nonlinear systems with flexible components," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 139–151, 2009.

[167] D. Tan, W. Chen, and H. Wang, "On the use of monte-carlo simulation and deep fourier neural network in lane departure warning," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 4, pp. 76–90, 2017.

[168] M. S. Gashler and S. C. Ashmore, "Training deep fourier neural networks to fit time-series data," in *International Conference on Intelligent Computing*. Springer, 2014, pp. 48–55.

[169] Box and Jenkins, "International airline passengers: monthly totals in thousands. jan 49 to dec 60," *Time Series Data Library*, 1976, (Date last accessed 5-April-2018). [Online]. Available: http://datamarket.com/data/list/?q=provider:tsdl

[170] S. W. D. Center, "Sunspot data from the world data center silso, royal observatory of belgium, brussels," *International Sunspot Number Monthly Bulletin and online catalogue*, 1770-2017, (Date last accessed 5-April-2018). [Online]. Available: http://www.sidc.be/silso/datafiles

[171] "Iso ne ca, rt demand, from 1-jan-2017 to 1-feb-2017," *ISO New England Public*, 2017, (Date last accessed 5-April-2018). [Online]. Available: https://www.iso-ne.com/isoexpress/web/reports/load-and-demand/-/tree/zone-info

[172] "Internet traffic data (in bits) from a private isp with centres in 11 european cities. from 18-june-2005 to 16-july-2005," *Time Series Data Library*, 2012, (Date last accessed 5-April-2018). [Online]. Available: https://datamarket.com/data/list/?q=provider:tsdl

[173] E. F. Fama and K. R. French, "Business cycles and the behavior of metals prices," *The Journal of Finance*, vol. 43, no. 5, pp. 1075–1093, 1988.

[174] "Apple inc. (aapl) closing stock price. from 3-january-2012 to 13-april-2018," *Yahoo Finance*, 2018, (Date last accessed 13-April-2018). [Online]. Available: https://finance.yahoo.com/quote/AAPL?p=AAPL

[175] J. Falnes, *Ocean waves and oscillating systems: linear interactions including wave-energy extraction*. Cambridge university press, 2002.

[176] C. Wan, Z. Xu, P. Pinson, Z. Y. Dong, and K. P. Wong, "Optimal prediction intervals of wind power generation," *IEEE Transactions on Power Systems*, vol. 29, no. 3, pp. 1166–1174, 2014.

[177] "Generating quantile forecasts in r," https://robjhyndman.com/hyndsight/quantile-forecasts-in-r/, accessed: 20-April-2018.

[178] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose, "A state space framework for automatic forecasting using exponential smoothing methods," *International Journal of forecasting*, vol. 18, no. 3, pp. 439–454, 2002.

[179] "Seasonal autoregressive integrated moving average with exogenous regressors model in python," http://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html, accessed: 20-April-2018.

[180] "Python implementation of holt-winters seasonal methods," https://gist.github.com/andrequeiroz/5888967, accessed: 20-April-2018.

[181] K. Hatalis, S. Kishore, K. Scheinberg, and A. Lamadrid, "An empirical analysis of constrained support vector quantile regression for nonparametric probabilistic forecasting of wind power," *arXiv preprint arXiv:1803.10888*, 2018.

[182] K. Hatalis, A. J. Lamadrid, K. Scheinberg, and S. Kishore, "Smooth pinball neural network for probabilistic forecasting of wind power," *arXiv preprint arXiv:1710.01720*, 2017.

[183] "Quantile regression in python," http://www.statsmodels.org, accessed: 20-April-2018.

[184] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[185] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, "Ensemble deep learning for regression and time series forecasting," in *Computational Intelligence in Ensemble Learning (CIEL), 2014 IEEE Symposium on*. IEEE, 2014, pp. 1–6.

[186] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, p. 6085, 2018.

[187] Y. Liu and Y. Wu, "Stepwise multiple quantile regression estimation using non-crossing constraints," *Statistics and its Interface*, vol. 2, no. 3, pp. 299–310, 2009.

[188] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, pp. 143–155, 1989.

[189] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 922–928.

[190] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[191] R. Mittelman, "Time-series modeling with undecimated fully convolutional neural networks," *arXiv preprint arXiv:1508.00317*, 2015.

[192] M. Bińkowski, G. Marti, and P. Donnat, "Autoregressive convolutional neural networks for asynchronous time series," *arXiv preprint arXiv:1703.04122*, 2017.

[193] F. A. Gers, D. Eck, and J. Schmidhuber, "Applying lstm to time series predictable through time-window approaches," in *Neural Nets WIRN Vietri-01*. Springer, 2002, pp. 193–200.

[194] L. Cheng, H. Zang, T. Ding, R. Sun, M. Wang, Z. Wei, and G. Sun, "Ensemble recurrent neural network based probabilistic wind speed forecasting approach," *Energies*, vol. 11, no. 8, p. 1958, 2018.

[195] Z. Shi, H. Liang, and V. Dinavahi, "Direct interval forecast of uncertain wind power based on recurrent neural networks," *IEEE Transactions on Sustainable Energy*, vol. 9, no. 3, pp. 1177–1187, 2018.

Vita

Personal Profile

Konstantinos Hatalis is an Electrical Engineering Ph.D. major at Lehigh University where his research focus is in converging deep learning, time series analysis, statistical signal processing, and evolutionary optimization methods for nonparametric probabilistic forecasting and pattern recognition. From Lehigh he also holds a MSc degree in computer science, with a thesis published analyzing neural networks for semantic interference. He is experienced in data science and is an ardent advocate of artificial intelligence technologies.

Education

2013-2018 Ph.D. in Electrical Engineering - Lehigh University
Advisor - Prof. Shalinee Kishore
Awards - P.C. Rossin Doctoral Fellow, Packard Fellowship

2012-2013 M.Sc. in Computer Science - Lehigh University
Advisor - Prof. Hector Munoz-Avila
Awards - Sherman Fairchild Fellowship

2011-2012 B.Sc. in Computer Science and Business - Lehigh University
Advisor - Prof. Sharon Kalafut

2007-2011 B.Sc. in Computer Engineering - Lehigh University
Advisor - Prof. Ed Kay

Professional Experience

Sep 2013 - May 2019 Integrated Networks for Electricity Research Cluster
Research Assistant

Sep 2013 - May 2019 Power from Ocean, Rivers, and Tides (PORT) Lab
Research Assistant

May 2012 - Aug 2013 Intelligent Decision Systems and Technologies Lab
Research Assistant

Publications

Journal Publications

1. K. Hatalis, P. Venkitasubramaniam, and K. Shalinee, "Modeling and Detection of Future Cyber-Enabled DSM Data Attacks using Supervised Learning", In Preparation.

2. K. Hatalis and K. Shalinee, "A Composite Quantile Fourier Neural Network for Multi-Horizon Probabilistic Forecasting", *Neurocomputing*, In-Submission.

3. K. Hatalis, A.J. Lamadrid, K. Scheinberg, and K. Shalinee, "Multi Quantile Estimation based Neural Network for Probabilistic Forecasting of Wind

Power", *Transactions on Transactions on Neural Networks and Learning Systems*, In-Submission.

Conference & Poster Publications

1. C. Zhao, Y. Zhuo, K. Hatalis, P. Venkitasubramaniam, L. Snyder, R. Blum, "Misuse of Demand Side Management", *Secure Evolvable Energy Delivery Systems Industry Meeting*, 2018.

2. K. Hatalis, K. Shalinee, K. Scheinberg, and A.J. Lamadrid, "An Empirical Analysis of Constrained Support Vector Quantile Regression for Nonparametric Probabilistic Forecasting of Wind Power", *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

3. P. Pradhan, K. Hatalis, and K. Shalinee, "Protecting Power System Operations Against Cyber Attacks in the Presence of Renewables", *Secure Evolvable Energy Delivery Systems Industry Meeting*, 2016.

4. K. Hatalis, B. Alnajjab, K. Shalinee, R. Blum, and A.J. Lamadrid, "Particle swarm based model exploitation for parameter estimation of wave realizations", *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016.

5. K. Hatalis and K. Shalinee, "On Enabling Autonomous Operation and Control of Ocean Wave Energy Farms", *Proceedings of the 3rd Marine Energy Technology Symposium*, 2015.

6. K. Hatalis, and R. Chandramouli, "Simulations on the Effect of External Semantic Interference in Lexical Retrieval and Priming in Memory", *Neuroscience*, 2015.

7. K. Hatalis, B. Alnajjab, K. Shalinee, and R. Blum, "Swarm Based Parameter Estimation of Wave Characteristics for Control in Ocean Energy Farms", *IEEE PES General Conference & Exposition*, 2015.

8. K. Hatalis and K. Shalinee, "A Multi-Agent System Architecture for Operational Autonomy in Ocean Wave Energy Farms", *Proceedings of the 28th International Flairs Conference (FLAIRS-28)*, 2015.

9. K. Hatalis, B. Alnajjab, K. Shalinee, and A.J. Lamadrid, "Adaptive particle swarm optimization learning in a time delayed recurrent neural network for multi-step prediction", *IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, 2014.

10. K. Hatalis, P. Pradhan, K. Shalinee, R. Blum, and A.J. Lamadrid, "Multi-step forecasting of wave power using a nonlinear recurrent neural network", *IEEE PES General Conference & Exposition*, 2014.

11. P. Pradhan, K. Hatalis, K. Shalinee, R. Blum, and A.J. Lamadrid, "Prospects of wave power grid integration", *IEEE PES General Conference & Exposition*, 2014.

12. P.G. OSéaghdha, D. Packer, A.K. Frazer, K. Preusse, K. Hatalis, H. Munoz-Avila, A. Hupbach, "Does mere co-activation drive semantic interference", *54th annual meeting of the Psychonomic Society, Toronto, Ontario, Canada*, 2013.