The Pennsylvania State University
The Graduate School

CONDITIONAL QUANTILE ESTIMATION WITH NEURAL NETWORK STRUCTURE

A Thesis in
Statistics
by
Yijia Feng

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2008

The thesis of Yijia Feng was reviewed and approved* by the following:

Runze Li
Professor of Statistics
Thesis Advisor and Graduate Study Chair

Bruce G. Lindsay
Professor of Statistics
Head of the Department of Statistics

Zhibiao Zhao
Assistant Professor of Statistics

*Signatures are on file in the Graduate School.

Abstract

In this thesis, we apply neural network method to estimate nonparametric conditional quantile under the quantile regression loss, i.e., the check loss function. The proposed robust neural network (RNN) method integrates quantile regression and neural network together, and is a useful modelling tool. We further apply an majorization-minization (MM) algorithm (Hunter & Lange, 2000) to deal with the minimization of RNN. Monte Carlo simulation study is conducted to examine the performance of the proposed robust neural network. From our simulation results, we found that the RNN method is promising. The proposed procedures are illustrated and compared with two popular nonparametric methods, local linear and regression splines, by a real data example in credit card portfolio analysis.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First of all, I am very grateful to my advisor, Dr. Runze Li, for his many helpful ideas and discussions on the contents of this thesis. Second, I deeply thank my thesis committee members, Dr. Bruce G. Lindsay and Dr. Zhibiao Zhao, for their precious time and valuable suggestions in improving the contents of this thesis. I also would like to thank my husband Yiyun Zhang. The discussions with him are great and helpful.

This thesis research was supported by grants from the National Science Foundation (DMS 0348869, CCF 0430349 and DMS 0722351) and a grant from the National Institute of Health (R21 DA024260).

At the end, I want to give my last thanks but not the least to my dear parents and parents in-law. Without their love and support, I cannot finish this paper by myself.

Chapter 1

Introduction

When some bivariate data are observed, their relationship is commonly studied via regression analysis. Very often a parametric model is not adequate for a general relationship, and therefore nonparametric regression is sought for its flexibility as an exclamatory and diagnostic tool. It can provide a good curve estimation to describe their relationship. There have been vast literatures developing various approaches for nonparametric regression, for example, kernel regression (Nadaraya, 1964; Watson, 1964), local polynomial regression (Fan, 1992, 1993), smoothing splines (Reinsch, 1967; Silverman, 1985), penalized splines (Ruppert, 2002), neural networks (Ripley, 1993; Haykin, 1994), just to name a few.

This thesis research was motivated by an analysis of credit card portfolio data. Figure 1.1 depicts the scatter plots of credit card portfolio data for two typical segments. The vertical axis stands for the charge-off rate, and horizontal axis is the month-on-book. See Section 4.2 for definitions of the charge-off rate and the month-on-book. It is believed that there is a smooth curve (relationship) between the charge-off rate and the month-on-book, known as maturation curve. Two common nonparametric smoothing methods: 1) local linear regression with a selected

optimal bandwidth, and 2) cubic regression splines with known selection knots, are used to estimate the maturation curve. The resulting estimates are also shown in Figure 1.1, from which it can be seen that they are wiggling over the month-on-book for both methods. This is an undesired feature because it is believed from the related business experience that the maturation curve is stable after a certain number of month-on-book. This motivates us to seek other nonparametric estimation procedure for estimation of the maturation curve. Thus, we further use neural network to estimate the maturation curve. The estimated curves are also displayed by dotted lines in Figure 1.1. Unlike the estimates of local linear and cubic regression splines, the neural network estimate does not wiggle around at all.

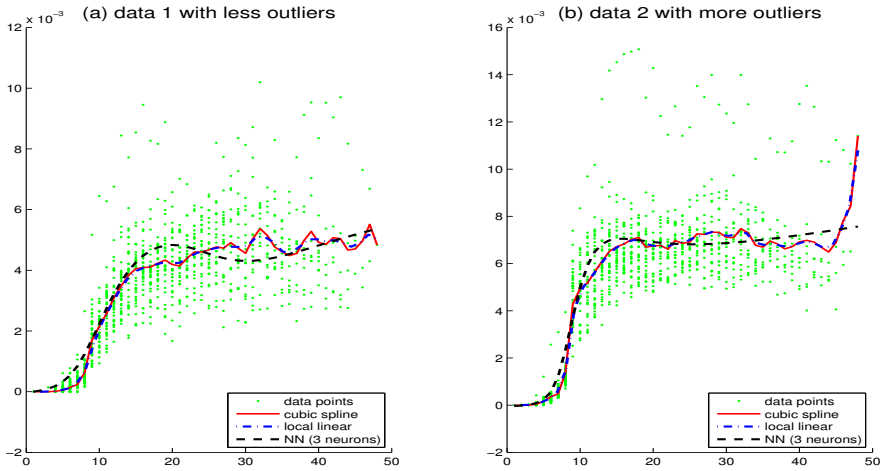

Figure 1.1. Credit Card Portfolio Data and Its Nonparametric Estimation

Neural network (NN, first mentioned by Rosenblatte, 1958) has been applied to many scientific fields as well as industrial areas (Widrow, Rumehart, & Lehr, 1994). As a nonparametric regression method, the NN has its special structure with fewer assumptions, which leads to the capability of seizing the obscure relations. Also, the features of data-driven and self-adaptive make NN a very attractive method for estimation and forecasting. There are many literatures that treat NN as a nonparametric regression estimation method (White, 1989a; Ripley, 1993; Cheng

& Titterington, 1994; Stern, 1996; Warner & Misra, 1996).

The scatter plots in Figure 1.1 clearly show that there are some noticeable outliers in both data sets, although the second segment data seems to contain more outliers than the first one. This leads us to further develop nonparametric quantile regression. As usual, data are usually not scattered well around the center, and it is very often to obtain some extreme observations, i.e. outliers. Classical regression methods give the mean estimates for the conditional distribution of the response variable by minimize the L_2 loss. As a result, these methods are sensitive to outliers by nature. Hence it is not a surprise that the estimation and interpretation of usual nonparametric estimation methods can be largely jeopardized by those extreme values. Quantile regression was introduced by Koenker and Bassett (1978) in order to obtain a robust estimator which is not sensitive to outliers. Furthermore, not like ordinary regression methods that only give a single mean estimate which only provide limited information, quantile regression extends the mean estimates to quantile estimates of distribution conditional on the covariates. As a result, it provides a complete picture of conditional distribution, which is useful for prediction. In this thesis, we aim to develop nonparametric quantile regression using neural network.

In this thesis, we propose robust neural network (RNN) procedure. The proposed procedure remain the merits of both neural network and quantile regression. It is challenging in optimizing the objective function involved in the RNN method because the objective function is not first order differentiable. The existing commonly used algorithms for neural network, such as backpropagation algorithm (Rumelhart, Hinton, & Williams, 1986), which is a gradient-descent related algorithm, cannot be directly applicable for RNN because of the singularity in the loss function. To handle the computational issue, we develop an MM algorithm

(Hunter & Lange, 2000) for the proposed RNN procedure. We further test the proposed algorithm by Monte Carlo simulation. We also address other issues related to practical implementation of the proposed procedure. From our simulation studies, we find that the proposed algorithm performs quite well with reasonable sample sizes. We further demonstrate the proposed methodology by an analysis of credit portfolio data. Numerical comparison with local linear regression and cubic regression splines are conducted in the real data analysis.

The rest of this thesis is organized as follows. Neural network is reviewed in Chapter 2, along with some other nonparametric smoothing methods including local polynomial approximation, polynomial regression splines and smoothing splines. Quantile regression and its application with nonparametric settings are also reviewed. Chapter 3 developed our methodology of neural network regression and provided discussions of some related issues. In Chapter 4, simulation studies of several different curve fittings are illustrated. The performance of RNN is very promising in our simulations. Also, one real data example is presented in Chapter 4, where we apply the proposed RNN methodology to credit card portfolio data for the estimation of the mature curve. Discussion and possible future work are given in Chapter 5.

Chapter 2

Literature Review

Suppose that n data points $\{(x_i, y_i)\}_{i=1}^n$ is a random sample from distribution (X, Y), where X is the explanatory variable and Y is the response variable. Consider the simple linear regression model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \qquad i = 1, \ldots, n, \tag{2.1}$$

where $\{\epsilon_i\}_{i=1}^n$ are i.i.d. random errors with mean zero and variance σ^2. Note that we assume both X and Y are scalar. Multivariate covariates are not discussed in this paper.

Model (2.1) may be extended in two directions. First, a simple linear model is often not adequate to capture the relationship between X and Y. The first possible extension is to relax the constraint of linearity to an unknown function $m(\cdot)$ with nonparametric structure. Secondly, the estimation of the conditional mean function of $Y|X$ often are not robust to outliers. Instead of modelling the conditional mean, we might model the conditional median.

The rest of this chapter is organize as follows. Section 2.1.2 is devoted to the first extension, i.e. nonparametric modelling of the condition mean function. In

Section 2.2, we briefly discuss the second extension in quantile regression. Section 2.3 reviews several attempts in combining nonparametric regression with quantile regression.

2.1 Nonparametric Smoothing Methods

We model the mean function in (2.1) as $m(x)$, then the model can be rewritten as

$$y_i = m(x_i) + \epsilon_i, \qquad i = 1, \ldots, n. \tag{2.2}$$

To estimate the unknown curve $m(x)$, there are many nonparametric approaches in literature, such as kernel regression (Nadaraya, 1964; Watson, 1964), local polynomial regression (Fan, 1992, 1993), smoothing splines (Reinsch, 1967; Silverman, 1985), regression penalized splines (Ruppert, 2002), neural networks (Ripley, 1993; Haykin, 1994), etc. In this section, we focus on neural networks as a nonparametric smoothing tool, along with some other popular methods, local polynomial, regression penalized splines and smoothing splines.

2.1.1 Neural Networks

Neural Network (NN) was first introduced by Rosenblatte (1958) and has been developed since then. It is often viewed as a powerful modelling tool. It had been applied to a wide variety of scientific fields such as

1) Estimation and Forecasting. E.g. predicting dynamic nonlinear systems (Lapedes & Farber, 1988), estimating the pricing formula (Hutchinson, Lo, & Poggio, 1994), etc.

2) Classification Problems. E.g. identifying underwater sonar contacts (Gorman & Sejnowski, 1988), predicting heart problems in patients (Baxt, 1990), etc.

3) Pattern Recognition. E.g. speech recognizing (Lippmann, 1988), etc.

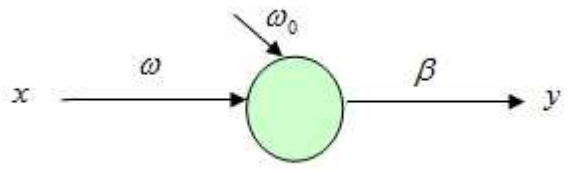

Figure 2.1. A Single Processing Neuron

Neural networks mimic the architecture of biological neural networks in human or animal brain and involve several simple processing elements (artificial neurons) which work in parallel. Figure 2.1 shows a simple version of neural network named *single neuron* or *perceptron*, where ω and β are connecting weights, which are essentially parameters. The relationship between response y and explanatory variable x is modelled as

$$y = \omega_0 + x\omega. \tag{2.3}$$

Notice that if x is multi-dimensional as $\mathbf{x} = (x_1, \ldots, x_p)^T$, and then the weights ω can be changed to p-dimensional accordingly.

In general, neural networks are equipped with a collection of such single neurons. These neurons are interlinked by a system of connecting weights, which results in different structures of neural networks. Two types are particular popular: 1) feed-forward networks, often known as multi-layer perceptron (MLP), and 2) symmetric recurrent networks, known as Hopfield nets (Hopfield, 1982). The

neurons in feed-forward networks do not form a directed cycle and keep information propagated forward. In consequence, feed-forward networks are usually utilized in classification, regression and function approximation problems due to the input-output model. In symmetric recurrent networks, the visit of a neuron may be recursive, which is qualified for *associate memory* (a type of memory search). In this paper, the main focus is on regression, therefore we focus on feed-forward network, especially MLP network.

2.1.1.1 Multi-Layer Perceptron Networks

Each neuron in a multi-layer perceptron (MLP) includes a nonlinear activation function. This feature along with the input-output mapping largely enhances the smoothness. A general MLP usually consists of three components:

- *Input layer* - used to input information from the explanatory variable $\{x_i\}_{i=1}^n$;
- *Hidden layers* - used to process the summation from input layer via some nonlinear transformation and each layer contains ceratin amount of neurons;
- *Output layer* - used to output the estimation of response variable $\{y_i\}_{i=1}^n$.

Before putting more feature in a neural network, we have to decide the number of hidden layers. Once the structure of NN is decided, the network is determined given

- *A nonlinear activation function* - used on each neuron and denoted g;
- *Connecting weights* - denoted by $\boldsymbol{\omega}$ for connection between input and hidden layers, $\boldsymbol{\beta}$ for hidden and output layers;
- *Bias weights* - denoted $\boldsymbol{\omega}_0$, β_0 and used to adjust the bias on neurons and output respectively.

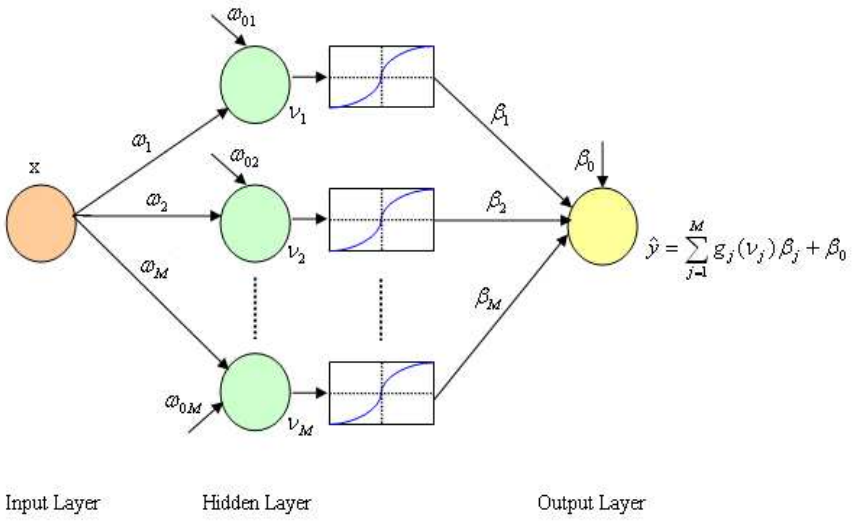

Figure 2.2. 3-Layer Perceptron (adopted from Figure 5.12, 5.13 of Fang et al., 2005)

Though a MLP network can have multiple hidden layers, more than one hidden layer increases the complexity and usually a 3-layer perceptron (one input, one hidden and one output layers, see Figure 2.2) network is enough to be a universal function approximator (Cybenko, 1989; Hornik, 1993). In the rest of this paper, we will focus on the 3-layer perceptron network with M parallel neurons in the hidden layer.

MLP follows a supervised learning procedure, which "learns" information from observations and then infer the object function. The estimated output for i^{th} observation is as follows:

$$\hat{y}_i = \sum_{j=1}^{M} \beta_j g(\nu_{ij}) + \beta_0, \tag{2.4}$$

where β_j is the weight connecting j^{th} neuron and the output, β_0 is a bias adjusted for the output, and $g(\cdot)$ is the activation function. It is often chosen to be a

logistic-type function, such as

$$g(\nu_{ij}) = \frac{1}{1+e^{-\nu_{ij}}} \qquad \text{or} \qquad g(\nu_{ij}) = \tanh(\nu_{ij}), \tag{2.5}$$

where each ν_{ij} is the weighted summation of each input

$$\nu_{ij} = x_i\omega_j + \omega_{0j}, \quad j = 1, \ldots, M, \tag{2.6}$$

where ω_j is the connection weight between input and j^{th} neuron, while ω_{0j} is a bias adjusted at neuron j.

2.1.1.2 Backpropagation Algorithm

Training of a neural network studies the existing information from the data and modifies weight parameters iteratively to minimize the least squared error

$$E = \sum_{i=1}^{n} \{y_i - \hat{m}(x_i)\}^2.$$

There have been many different optimization algorithms to train neural network. Among those choices, an algorithm called Backpropagation (Rumelhart et al., 1986) is the most popular one for MLP neural network.

In backpropagation, the errors (the difference between desired and estimated values of output) are propagated backward from output to input. It calculates the gradient of the errors with respect to the weight parameters and moves forward according to the gradient descent direction. Therefore, it is closely related to the gradient steepest method, Newton-Raphson (N-R) algorithm, but with its own feature. A learning rate (a step size) is adopted instead of using the inverse of the hessian matrix in each iterative weights update. In consequence, the results are

somehow sensitive to the learning rate. Smaller learning rates may cause slower convergence while larger steps converge faster but may lead to the possibility of oscillation (Rumelhart et al., 1986). One way to improve the original backpropagation method is to include a dynamic learning rate (LeCun, Simard, & Pearlmutter, 1993; Yu, Chen, & Cheng, 1995). However, there are no certain formulae or systems for universal optimal learning rates. The improvement is sometimes limited, and choosing the step size largely depends on experimentation.

The procedure of backpropagation (Warner & Misra, 1996) is as follows:

1. Initialize the weights $(\boldsymbol{\omega}_0^0, \boldsymbol{\omega}^0, \boldsymbol{\beta}^0, \boldsymbol{\beta}_0^0)$ to small random values.

2. At t^{th} step, choose an observation (x_i, y_i) along with the initial weights to get $\hat{y}_i$ via (2.23) and calculate $E_i = (y_i - \hat{y}_i)^2/2$.

3. Compute $\triangle\boldsymbol{\beta}$ and $\triangle\boldsymbol{\omega}$ for $j = 1, \ldots, M$ by

$$\triangle\beta_0 = -\eta\frac{\partial E_i}{\partial\beta_0}, \quad \triangle\beta_j = -\eta\frac{\partial E_i}{\partial\beta_j}, \quad \triangle\omega_{0j} = -\eta\frac{\partial E_i}{\partial\omega_{0j}}, \quad \triangle\omega_j = -\eta\frac{\partial E_i}{\partial\omega_j}$$

   where η is the learning rate decided in advance.

4. Update weights at $(t+1)^{\text{th}}$ iteration according to

$$\beta_0^{t+1} = \beta_0^t + \triangle\beta_0,\ \beta_j^{t+1} = \beta_j^t + \triangle\beta_j,\ \omega_{0j}^{t+1} = \omega_{0j}^t + \triangle\omega_{0j},\ \omega_j^{t+1} = \omega_j^t + \triangle\omega_j$$

5. Repeat step 2-4 for each observation until convergent criterion meets.

Similar to other optimization algorithms for MLP, backpropagation also suffers from convergence to local minimums. No algorithm so far guarantees the finding of global minimum. Provided the estimator does not diverge, under some regularity conditions mentioned in White (1989b), the estimator is ensured to converge almost surely to a parameter value that locally minimizes expected squared error loss.

2.1.2 Other Nonparametric Smoothing Methods

There are various nonparametric techniques of smoothing if $m(x)$ in (2.2) is believed to be smooth. Basically, these methods could be classified as global or local approximations. Besides neural network approach, wavelet thresholding (Donoho & Johnstone, 1994, 1995a, 1995b) and spline smoothing (Eubank, 1988; Wahba, 1990) are known as global smoothing methods. For local approximation methods, there are locally weighted scatter plot smoothing (LOWESS, (Cleveland, 1979)) and local polynomial including kernel (Nadaraya, 1964; Watson, 1964; Gasser & Müller, 1979) and local linear estimations (Fan, 1992; Fan & Gijbels, 1992; Fan, 1993; Ruppert & Wand, 1994). Fan and Gijbels (1996) provided a detailed summary of nonparametric regression including the smoothing methods mentioned above. In this thesis, we present three commonly used approaches: local polynomial regression, polynomial regression splines and smoothing splines.

2.1.2.1 Local Polynomial Regression

Back to equation (2.2), more specifically, we assume that

$$Y_i = m(X_i) + \sigma(X_i)\epsilon_i, \qquad i = 1, \ldots, n, \tag{2.7}$$

where $E(\epsilon) = 0$, $\mathrm{Var}(\epsilon) = 1$, and X_i's and ϵ_i's are independent. Then the conditional mean and variance of Y given $X = x_0$ are denoted by $m(x_0)$ and $\sigma^2(x_0)$ respectively. Suppose the $(p+1)^{\text{th}}$ derivative of $m(x)$ exists at the point x_0, we can approximate the unknown function $m(x)$ locally by a polynomial of order p. For x in a neighborhood of x_0, we use Taylor expansion and have:

$$m(x) \approx m(x_0) + m'(x_0)(x - x_0) + \frac{m''(x_0)}{2\,!}(x - x_0)^2$$

$$+\cdots+\frac{m^{(p)}(x_0)}{p\,!}(x-x_0)^p$$
$$\triangleq \sum_{j=0}^{p}\beta_j(x-x_0)^j. \tag{2.8}$$

We can model $m(x)$ locally by

$$\hat{m}(x)=\sum_{j=0}^{p}\beta_j(x-x_0)^j, \quad \text{for } x \text{ near } x_0. \tag{2.9}$$

At point x_0, we can estimate $\{\beta_j\}_{j=0}^p$ through a weighted least squares regression by minimizing

$$\sum_{i=1}^{n}\left\{Y_i-\sum_{j=0}^{p}\beta_j(X_i-x_0)^j\right\}^2 K_h(X_i-x_0), \tag{2.10}$$

where h is called bandwidth that controls the size of neighborhood, and $K_h(\cdot)=K(\cdot/h)/h$ with $K(\cdot)$ is called a kernel function assigning weights to data points depending on how far away from the center of the neighborhood. $K(\cdot)$ is usually assumed to be a symmetric probability density function.

Denote $\{\hat{\beta}_j\}_{j=0}^p$ to be the minimizer to (2.10). For $\nu=0,1,\ldots,p$, $m^\nu(x_0)$ can be estimated by $\nu!\hat{\beta}_\nu$ according to (2.8). To obtain the estimate of $m^\nu(\cdot)$ over x, we need to solve the weighted least squared problem over a fine grid points of x.

For a more convenient purpose, matrix notation is used and minimizing (2.10) can be written as

$$\min_{\boldsymbol{\beta}}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T\mathbf{W}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}), \tag{2.11}$$

where

$$\mathbf{W}=\text{diag}\{K_h(X_i-x_0)\},$$

and

$$\mathbf{X} = \begin{pmatrix} 1 & (X_1 - x_0) & \dots & (X_1 - x_0)^p \\ \vdots & \vdots & & \vdots \\ 1 & (X_n - x_0) & \dots & (X_1 - x_0)^p \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_p \end{pmatrix}.$$

The estimation of $\boldsymbol{\beta}$ in (2.11) is provided by weighted least squared theory and is given by

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T W \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \tag{2.12}$$

It follows directly from the least squares theory that

$$\begin{aligned} E(\hat{\boldsymbol{\beta}}|\mathcal{X}) &= (\mathbf{X}^T W \mathbf{X})^{-1} \mathbf{X}^T W \mathbf{m} \\ &= \boldsymbol{\beta} + (\mathbf{X}^T W \mathbf{X})^{-1} \mathbf{X}^T W \mathbf{r}, \\ \mathrm{Var}(\hat{\boldsymbol{\beta}}|\mathcal{X}) &= (\mathbf{X}^T W \mathbf{X})^{-1} (\mathbf{X}^T W \mathrm{Cov}(\mathbf{y}) \mathbf{X}) (\mathbf{X}^T W \mathbf{X})^{-1} \\ &\approx \sigma^2(x) (\mathbf{X}^T W \mathbf{X})^{-1} (\mathbf{X}^T W^2 \mathbf{X}) (\mathbf{X}^T W \mathbf{X})^{-1}, \end{aligned}$$

where $\mathbf{m} = (m(X_1), \dots, m(X_n))^T$, and $\mathbf{r} = \mathbf{m} - \mathbf{X}\boldsymbol{\beta}$ is the approximation error. When $\sigma^2(x) = \sigma^2$, " $\approx$ " becomes " $=$ " and σ^2 can be estimated using difference method. However, the exact expressions of the conditional mean and variance are not applicable because they depend on some unknown quantities.

Remark 2.1.1 (Asymptotic Bias and Variance). It is necessary to approximate the bias (i.e., $E(\hat{\boldsymbol{\beta}}|\mathcal{X}) - \boldsymbol{\beta}$) and the variance. Let $S_{n,j} = \sum_{i=1}^{n} (X_i - x_0)^j K_h(X_i - x_0)$. Then

$$S_n = \mathbf{X}^T W \mathbf{X} = (S_{n,i+j}).$$

Also, denote the moments of K and K^2 respectively by

$$\mu_j = \int t^j K(t)dt \quad \text{and} \quad \nu_j = \int t^j K^2(t)dt.$$

Furthermore, some matrices and vectors of moments are denoted by

$$\begin{aligned} S &= (\mu_{j+l})_{0\le j,\ l\le p} & c_p &= (\mu_{p+1}, \ldots, \mu_{2p+1})^T \\ S^* &= (\mu_{j+l+1})_{0\le j,\ l\le p} & \tilde{c}_p &= (\mu_{p+2}, \ldots, \mu_{2p+2})^T \end{aligned}$$

Then under some regularity conditions (Theorem 3.1 of Fan & Gijbels, 1996), the asymptotic conditional bias and variance are given by:
if $p-\nu$ is odd,

$$\begin{aligned} \text{Bias}\{\hat{m}_\nu(x_0)|\mathcal{X}\} &= e_{\nu+1}^T S^{-1} c_p \frac{\nu!}{(p+1)!} m^{(p+1)}(x_0) h^{p+1-\nu} \\ &\quad + o_p(h^{p+1-\nu}), \end{aligned} \tag{2.13}$$

and if $p-\nu$ is even,

$$\begin{aligned} \text{Bias}\{\hat{m}_\nu(x_0)|\mathcal{X}\} &= e_{\nu+1}^T S^{-1} \tilde{c}_p \frac{\nu!}{(p+2)!} \{m^{(p+2)}(x_0) \\ &\quad + (p+2) m^{(p+1)}(x_0) \frac{f'(x_0)}{f(x_0)}\} h^{p+2-\nu} \\ &\quad + o_p(h^{p+2-\nu}), \end{aligned} \tag{2.14}$$

and

$$\begin{aligned} \text{Var}\{\hat{m}_\nu(x_0)|\mathcal{X}\} &= e_{\nu+1}^T S^{-1} S^* S^{-1} e_{\nu+1} \frac{\nu!^2 \sigma^2(x_0)}{f(x_0) n h^{1+2\nu}} \\ &\quad + o_p\left(\frac{1}{nh^{1+2\nu}}\right), \end{aligned} \tag{2.15}$$

where $e_{\nu+1} = (0, \ldots, 0, 1, 0, \ldots, 0)^T$ with 1 on the $(\nu+1)^{\text{th}}$ position.

Remark 2.1.2 (Choice of Order p). When $p = 0$ and $\nu = 0$, local linear regression reduce to the famous Nadaraya-Watson (N-W) estimation (Nadaraya, 1964; Watson, 1964). In (2.13) and (2.14), it is clear that the asymptotic bias for $p - \nu$ odd has a simpler structure without the term $f'(x_0)$ involved. This directly results in the fact that the N-W estimator ($p = 0$ and $\nu = 0$) has an additional term in the asymptotic bias compared to local linear estimator ($p = 1$ and $\nu = 0$). In general, the polynomial fits with $p-\nu$ odd outperform those with $p-\nu$ even in that they have smaller bias but same variance. In order to balance the conditional bias and variance, we favor local linear or local cubic estimation.

Remark 2.1.3 (Selection of Kernel Function K). For the selection of kernel function K, Fan and Gijbels (1996) showed that the optimal weight function is the Epanechnikov kernel, $K(z) = 0.75(1 - z^2)_+$, which minimizes the asymptotic MSE of the resulting local linear estimators for all choices of p and ν. Other kernels like the Gaussian kernel ($K(z) = \frac{1}{\sqrt{2\pi}} \exp(-z^2/2)$) and the uniform kernel ($K(z) = 1_{[-0.5,\ 0.5]}(z)$) also have comparative performances.

Remark 2.1.4 (Choice of Bandwidth h). There are usually two types of bandwidths: constant or local (variable) bandwidth. Variable bandwidth selection is desirable when the estimating curve has complicated structure (Fan & Gijbels, 1995). More often, we focus on constant bandwidth. For $m^{(\nu)}(x)$, we intuitively obtain the ideal optimal bandwidth by minimizing MISE$= \int \text{MSE}(x)\omega(x)dx$ and typically set $\omega(x) = \omega_0(x)f(x)$:

$$h_{opt} = C_{\nu,p}(K) \left[\frac{\int \sigma^2(x)\omega_0(x)dx}{\int \{m^{(p+1)}(x)\}^2 \omega_0(x) f(x) dx} \right] n^{-1/(2p+3)}, \tag{2.16}$$

where

$$C_{\nu,p}(K) = \left[\frac{(p+1)!^2(2\nu+1)\int K_\nu^{*2}(t)dt}{2(p+1-\nu)\{\int t^{p+1}K_\nu^*(t)dt\}^2} \right]^{1/(2p+3)}$$

and

$$K_\nu^*(t) = (\sum_{l=0}^{p} S^\nu t^l)K(t).$$

GCV and CV criteria can be used to select the suitable bandwidth with respect to the data. However, the computation is very expensive. A quicker and more preferable way is to obtain a suitable bandwidth by using "plug-in" method, which substitutes the unknown quantities in (2.16) by their rough estimates.

2.1.2.2 Regression and Penalized Splines

Consider the model

$$Y_i = m(X_i) + \epsilon_i,$$

where $E(\epsilon_i|X_i) = 0$ and $Var(\epsilon_i|X_i) = \sigma^2$. Let $t_1 < \cdots < t_J$ be J usually equally spaced fixed knots in the range of $x-$coordinate. We approximate $m(x)$ with a polynomial regression spline of order p, which is defined by

$$s(x) = \sum_{j=1}^{J+p} \beta_j B_j(x).$$

Here $B_j(x)$'s are bases, of which two most popular kinds are:

(1) Power basis: $1, x, ..., x^p,\ (x-t_j)_+^p\ (j = 1, \ldots, J)$, where $u_+ = u$ if $u > 0$, and $u_+ = 0$ otherwise;

(2) B-spline basis (de Boor, 1978): defined iteratively, no expressive form.

After some algebraic simplifications, the polynomial regression spline is piecewise p-th order polynomial, and the locations of discontinuity occur at each knots. The

most commonly used regression spline is the *cubic spline*, i.e. $p = 3$.

To fit a polynomial regression spline, the objective is to find $\{\hat{\beta}_j\}_{j=1}^{J+4}$ that minimizes

$$\sum_{i=1}^{n} \left\{ Y_i - \sum_{j=1}^{J+4} \beta_j B_j(x) \right\}^2, \tag{2.17}$$

which can be easily reformatted to an OLS problem.

The choice of knots is a state of art. Obviously, zero knots lead to a global p-th order polynomial fit, and too many knots can result in serious overfitting. To account for the overfitting problem, Ruppert (2002) introduced the penalized spline as an improvement over the polynomial regression splines. Penalized spline adds an extra penalty term to (2.17) for a general p-th order polynomial spline and minimizes

$$\sum_{i=1}^{n} \left\{ Y_i - \sum_{j=1}^{J+p} \beta_j B_j(x) \right\}^2 + \lambda \sum_{j=1}^{J} \beta_{j+p}^2, \tag{2.18}$$

where λ is a smoothing parameter. It is clear that the larger the value of λ, the more the spline fit is shrunk towards a global polynomial fit where $\beta_{j+p} = 0$ for $j = 1, \ldots, J$.

Let us use power basis as an example, let $\mathbf{Y} = (y_1, \ldots, y_n)^T$, and $\mathbf{X}$ be the design matrix whose ith row is

$$\mathbf{X}_i = [1, x_i, \ldots, x_i^p, (x - t_1)_+^p, \ldots, (x - t_J)_+^p]^T.$$

Then the solution $\hat{\beta}(\lambda)$ to (2.18) is given by

$$\hat{\beta}(\lambda) = (\mathbf{X}^T\mathbf{X} + \lambda D)^{-1}\mathbf{X}^T\mathbf{y}, \tag{2.19}$$

where D is a diagonal matrix whose first $(p + 1)$ diagonal elements are 0.

To choose the smoothing parameter λ, Ruppert (2002) provided the corresponding GCV criterion. Define the average squared residuals as

$$ASR(\lambda) = n^{-1} \sum_{i=1}^{n} \left\{ y_i - m(X_i; \hat{\beta}(\lambda)) \right\}^2,$$

and $S(\lambda) = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda D)^{-1}\mathbf{X}^T$ to be the "hat" matrix. Then the generalized cross-validation statistics for choosing λ is given by

$$GCV(\lambda) = \frac{ASR(\lambda)}{[1 - \lambda n^{-1} tr\{S(\lambda)\}]^2}.$$

2.1.2.3 Smoothing Splines

Consider finding function m to minimize the least squared problem

$$\sum_{i=1}^{n} \{Y_i - m(X_i)\}^2. \tag{2.20}$$

The naive solution will be any curve m that interpolates the data and it can reduce the summation of errors to 0. However, such solution induces a huge amount of variability of the parameter estimates. In consequence, a penalty for over-parametrization is imposed to (2.20)

$$\sum_{i=1}^{n} \{Y_i - m(X_i)\}^2 + \lambda \int (m''(x))^2 dx, \tag{2.21}$$

where λ is a nonnegative smoothing parameter. It is obvious that $\lambda = 0$ results in an interpolation through every x and $\lambda = +\infty$ gives a linear regression $m(x) = \beta_0 + \beta_1 x$. We are able to control the rate of exchange between error residual and roughness of the fitting curve through adjusting the value λ . The solution to minimization of (2.21) is denoted as $\hat{m}_\lambda$ and is called the *smoothing spline*

estimator.

In the class of all twice differentiable functions on the interval $[X_{(1)}, X_{(n)}]$, (2.21) has an unique minimizer

$$\hat{m}_\lambda(x) = \sum_{j=1}^{n+2} \beta_j B_j(x), \tag{2.22}$$

where $\{B_j(x)\}_{j=1}^{n+2}$ are the spline basis mentioned in Section 2.1.2.2. $\hat{m}_\lambda$ has a form of cubic splines with knots $X_{(2)}, \ldots, X_{(n-1)}$.

Denote $Q = (q_{ij})_{(n+2)\times(n+2)}$ by

$$q_{ij} = \int_{X_{(1)}}^{X_{(n)}} B_i^{''}(x) B_j^{''}(x) dx$$

and $B = (B_j(X_i))_{n\times(n+2)}$. Then (2.21) can be written as

$$\|Y - B\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}^T Q \boldsymbol{\beta}$$

Thus,

$$\hat{\boldsymbol{\beta}} = (B^T B + \lambda Q)^{-1} B^T \mathbf{y},$$

where $\mathbf{y} = (Y_1, \ldots, Y_n)$.

$$\hat{\mathbf{y}} = B\hat{\boldsymbol{\beta}} = B(B^T B + \lambda Q)^{-1} B^T \mathbf{y} \tag{2.23}$$

Therefore, define

$$S(\lambda) = B\hat{\boldsymbol{\beta}} = B(B^T B + \lambda Q)^{-1} B^T.$$

Then $\hat{\mathbf{y}} = S(\lambda)\mathbf{y}$ is a linear estimator in term of y's.

Remark 2.1.5 (Smoothing splines and kernel regression). Following Härdle

(1990), we write

$$\hat{m}_\lambda(x) = n^{-1} \sum_{i=1}^{n} W_i(x, \lambda; X_1, \ldots, X_n) Y_i,$$

where W_i is called effective weight, which is independent of the response $\{Y_i\}$. Though the form of W_i is complicated, the relation with kernel regression has been studied by Silverman (1984), where it was pointed out that the smoothing spline is a basically a local kernel average with a variable bandwidth. To see this, for X_i away from boundary, and for large n and small λ,

$$W_i(x, \lambda; X_1, \ldots, X_n) \approx f(X_i)^{-1} h(X_i)^{-1} K_s\{(X_i - x)/h(X_i)\}, \tag{2.24}$$

where the *local bandwidth* $h(X_i)$ satisfies

$$h(X_i) = [\lambda/\{nf(X_i)\}]^{1/4}$$

and the *kernel function* K_s is given by

$$K_s(t) = 1/2 \exp(-|t|/\sqrt{2}) \sin(|t|/\sqrt{2} + \pi/4)$$

Equation (2.24) actually gives an intuition of how the spline allocates the weights to the local neighbor of a point x.

Remark 2.1.6 (Choosing regularization parameter). Choice of parameter λ is essential in the smoothing procedure. There are usually *cross-validation* (CV) and *generalized cross-validation* (GCV) criterions:

$$\mathrm{CV}(\lambda) = n^{-1} \sum_{i=1}^{n} \{Y_i - \hat{m}_{\lambda,-i}(X_i)\}^2 \tag{2.25}$$

We select λ through minimizing cross-validation criterion in (2.25), where $\hat{m}_{\lambda,-i}$ is the estimator of optimizing (2.21) without the i^{th} observation (Allen, 1974; Stone, 1974). However, using CV approach is computational expensive. Therefore, generalized cross-validation (Craven & Wahba, 1979) was proposed

$$\text{GCV}(\lambda) = \frac{\|\mathbf{y} - \hat{\mathbf{y}}(\lambda)\|^2}{n\{1 - e(\lambda)/n\}^2},$$

where $e(\lambda) = tr\{S(\lambda)\}$, $\mathbf{y}$ and $\hat{\mathbf{y}}$ are given in (2.23).

2.2 Quantile Regression

The simple linear regression model (2.1) suffers from outliers. More specifically, if the error is heavy-tailed, the estimator will be substantially influenced by potential large errors. Introduced by Koenker and Bassett (1978), quantile regression has been a popular method that extends the classical method of estimating conditional mean to estimating quantiles of the distribution conditional on the values of covariates. Regression quantiles are robust against the influence of outliers and capable of providing a complete picture of the conditional distribution rather than a single point estimate of the mean.

Under the l_2 loss, we obtain conditional mean estimate of $E(Y|X)$ through minimizing $\sum(y_i - \hat{y}_i)^2$. In the presence of outlier, the residual $y_i - \hat{y}_i$ is amplified by taking squares. This motivates us to consider the l_1 distance $|y_i - \hat{y}_i|$, to alleviate the impacts of outliers. It turns out that, if we minimize $\sum |y_i - \hat{y}_i|$ instead of $\sum(y_i - \hat{y}_i)^2$, the resulting estimator is the conditional median of $Y|X$. More generally, the conditional quantile are estimable based on some further extension

of the loss form. For any $0 < \tau < 1$, we define a quantile loss function by

$$\rho_\tau(u) = u\{\tau - I(u < 0)\} = \frac{1}{2}\{|u| + (2\tau - 1)u\}, \tag{2.26}$$

which is illustrated in Figure 2.3. It assigns asymmetric weights to absolute errors to achie [illegible] quantile

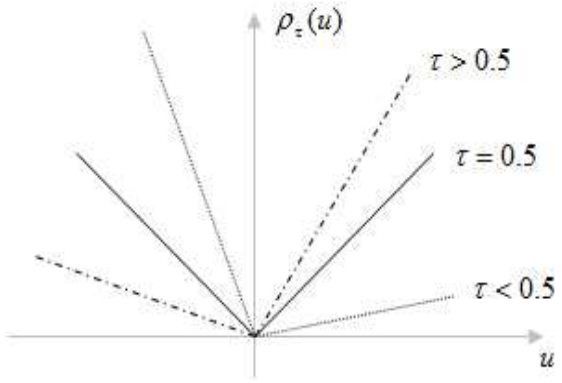

Figure 2.3. Quantile Regression ρ Function

function in a linear form as $Q_y(\tau|x) = x^T\beta(\tau)$, then the estimation of $\beta(\tau)$ will be obtained through

$$\min_{\beta \in \Re^p} \sum_{i=1}^{n} \rho_\tau(y_i - x_i^T\beta) \tag{2.27}$$

Denote $\hat{Q}_y(\tau|x) = x^T\hat{\beta}(\tau)$ the estimated τth conditional quantile of $Y|X = x$. It is obvious that letting $\tau = 0.5$ in (2.27) leads to the special case of conditional median estimates.

Ordinary gradient decent methods do not directly apply in finding the solution of (2.27), because the target function is not twice differentiable. Fortunately, the minimization problem here can be transformed into minimizing a series of expressions subject to certain constraints, which can be solved by linear programming,

see Wagner (1959) for example. To solve linear programs in a parametric sense, there are simplex approach, interior point method, etc. Interior point method is more preferable of its capability of large quantile regression applications. For non-linear model, interior point method is highly relevant too (Koenker & Park, 1996). Other than using linear programming to solve (2.27), Hunter and Lange (2000) proposed a MM algorithm, which can be viewed as an extension of the popular EM algorithm.

Asymptotic Theory. Assume the conditional τth quantile of $Y|X$ follows a linear relationship $Q_y(\tau|x) = x^T\beta(\tau)$. We denote $\hat{\beta}_n(\tau)$ as the solution to (2.27), and use it to estimate $\beta(\tau)$. We wish to establish the asymptotic properties for this quantile estimate. The results are summarized below. See Koenker (2005) for more details.

The following assumptions are the asymptotic behavior of the sequence of the design matrices $X^{(n)}$:

Condition 2.2.1. *There exists $d > 0$ such that*

$$\liminf_{n\to\infty} \inf_{\|u\|=1} n^{-1} \sum I\left(|x_i^T u| < d\right) = 0$$

Condition 2.2.2. *There exists $D > 0$ such that*

$$\limsup_{n\to\infty} \sup_{\|u\|=1} n^{-1} \sum \left(x_i^T u\right)^2 \leq D.$$

Condition 2.2.1 is set for identifiability and it ensures that the $\{x_i\}_{i=1}^n$ are not concentrated on a proper linear subspace of $\mathbb{R}^p$. Condition 2.2.2 bounds the rate of growth of the $\{x_i\}_{i=1}^n$ and holds for the classical condition that $n^{-1}\sum_{i=1}^n x_i x_i^T$ converges to a positive definite matrix.

Theorem 1 (Consistency). *Suppose Condition 2.2.1 and Condition 2.2.2 hold, then the quantile estimator is consistent in probability, i.e.,*

$$\hat{\beta}_n(\tau) - \beta(\tau) = o_p(1)$$

if and only if for any $\epsilon > 0$,

$$\sqrt{n}(a_n(\epsilon) - \tau) \to \infty, \qquad \text{and} \qquad \sqrt{n}(\tau - b_n(\epsilon)) \to \infty,$$

where

$$a_n(\epsilon) = n^{-1} \sum F_i(x_i^T \beta(\tau) - \epsilon),$$

$$b_n(\epsilon) = n^{-1} \sum F_i(x_i^T \beta(\tau) + \epsilon),$$

and F_i *is the conditional distribution function of* Y_i *for* $i = 1, 2, \ldots$.

Besides the consistency of $\hat{\beta}_n(\tau)$, the asymptotic distribution can also be derived. Rewrite the conditional distribution function of Y_i as $P(Y_i < y|x_i) = F_{Y_i}(y|x_i) = F_i(y)$. Then

$$Q_{Y_i}(\tau|x_i) = F_{Y_i}^{-1}(\tau|x_i) \equiv \xi_i(\tau)$$

Theorem 2 (Asymptotic normality, Thm 4.1 in Koenker, 2005). *Assume the following condition hold:*

Condition 2.2.3. *The distribution functions* $\{F_i\}$ *are absolutely continuous, with continuous densities* $f_i(\xi)$ *uniformly bounded away from 0 and* ∞ *at the points* $\xi_i(\tau)$, $i = 1, 2, \ldots$

Condition 2.2.4. *There exist positive definite matrices* D_0 *and* $D_1(\tau)$ *such that*

(i) $\lim_{n\to\infty} n^{-1}\sum x_i x_i^T = D_0$,

(ii) $\lim_{n\to\infty} n^{-1}\sum f_i(\xi_i(\tau))x_i x_i^T = D_1(\tau)$,

(iii) $\max_{i=1,\ldots,n} \|x_i\|/\sqrt{n} \to 0$.

We have

$$\sqrt{n}(\hat{\beta}_n(\tau) - \beta(\tau)) \sim \mathcal{N}(0, \tau(1-\tau)D_1^{-1}D_0D_1^{-1})$$

In the iid error model, $\sqrt{n}(\hat{\beta}_n(\tau) - \beta(\tau)) \sim \mathcal{N}(0, \omega^2 D_0^{-1})$, *where* $\omega^2 = \tau(1-\tau)f_i^2(\xi_i(\tau))$.

Conditions 2.2.3 and 2.2.4 sets some very mild restrictions on the conditional distribution and the design respectively. The above theorem gives us an idea of how quickly the converge occurs along with the magnitude of convergence rate. Furthermore, the asymptotic covariance matrix is in an explicit form. Inferences and tests can be carried out accordingly, and are omitted here.

2.3 Nonparametric Quantile Regression

Though parametric model plays an important role in the realm of scientific data, there are may occasions that we have to turn to a more flexible way in nonparametric models. Due to many approaches of nonparametric regression, there are numerous literatures on the incorporations with quantile regression, see, for example, Koenker, Ng, and Portnoy (1994) and Yu and Jones (1998). Following the discussion of nonparametric smoothing methods in Section 2.1.2, only quantile smoothing splines and quantile local linear will be considered in this section.

2.3.1 Quantile Smoothing Splines

Similar to smoothing splines under least squared loss in (2.21), Koenker et al. (1994) extended the loss to a general quantile form and aim to minimize

$$R_{\tau,\lambda}(g) = \sum_{i=1}^{n} \rho_\tau\{y_i - m(x_i)\} + \lambda \left(\int_0^1 |m''(x)|^p dx \right)^{1/p},$$

where $\rho_\tau(u)$ is defined in (2.26), and $m(x)$ is the unknown function in (2.2) without assuming any parametric form . The regularization parameter $\lambda \in \mathbb{R}_+$ controls the smoothness of the smoothing splines. When $p = 1$, the L_1 penalty term yields a solution of linear splines, and is also easy computation. Therefore, we only consider the case when $p = 1$ for convenience. Define the solution space of function g as

$$U^2 = \left\{ m : m(x) = a_0 + a_1 x + \int_0^1 (x - y)_+ \, d\mu(y),\ V(\mu) < \infty,\ a_i \in \mathbb{R},\ i = 0, 1 \right\},$$

where $V(\cdot)$ is the total variation. The total variation of an absolutely continuous function is the integral of the absolute value of its derivative. Then we will have Theorem 3 presented as follows.

Theorem 3 (Thm 1 in Koenker et al., 1994). *The function $m \in U^2$ minimizing $\sum \rho_\tau\{y_i - m(x_i)\} + \lambda V(m')$ is a linear spline with knots at the points x_i $(i = 1, \ldots, n)$.*

Theorem 3 clarifies the existence of a linear spline as the solution. With its support, the details of solving the linear smoothing spline has been derived. Given the spline solved, choice of parameter λ is critical. It is suggested that we can

minimize criterion

$$\mathrm{SIC}(p_\lambda) = \log\left[n^{-1}\sum_{i=1}^{n}\rho_\tau\{y_i - \hat{m}(x_i)\}\right] + \frac{1}{2}n^{-1}p_\lambda \log n$$

to select parameter λ. Here, p_λ is the number of active knots for different λ's.

2.3.2 Local Linear quantile regression

Several local polynomial approaches have been developed (Chaudhuri, 1991; Fan, Hu, & Truong, 1994; Fan, Yao, & Tong, 1996; Yu & Jones, 1998) embed with quantile regression. We will focus on the quantile local linear approach ($p = 1$). Denote the τth conditional quantile $m_\tau(x)$. The idea of the local linear fit is to approximate the unknown $m_\tau(x)$ by a linear function

$$m_\tau(x) = m_\tau(x) + m'_\tau(x)(x - x_0) \equiv \beta_0 + \beta_1(x - x_0)$$

for x in a neighborhood of x_0. With such intuition, we aim to minimize

$$\sum_{i=1}^{n}\rho_\tau\{Y_i - \beta_0 - \beta_1(X_i - x_0)\}K\left(\frac{X_i - x_0}{h}\right)$$

It is obvious that $\hat{\beta}_0$ estimates $m_\tau(x)$ and $\hat{\beta}_1$ estimates $m'_\tau(x)$. To obtain $\hat{\beta}_0$ and $\hat{\beta}_1$, an iteratively reweighted least squares algorithm can be adopted, see, for example, Yu and Jones (1997).

Remark 2.3.1 (MSE). To evaluate the performance of $\hat{m}_\tau(x)$, it is necessary to derive its mean squared error (MSE). Under certain conditions given by Fan et al. (1994), as $n \to \infty$, $h = h(n) \to 0$ and $nh \to \infty$, for x is not too near a boundary

of the support and a smoothness condition,

$$\mathrm{MSE}(\hat{m}_\tau(x)) \simeq \frac{1}{4}h^4\mu_2(K)^2 m''_\tau(x)^2 + \frac{R(K)\tau(1-\tau)}{nhm(x)f(m_\tau(x)|x)^2},$$

where $\mu_2(K) = \int t^2K(t)dt$, $R(K) = \int K^2(t)dt$. Fan et al. also presented the boundary situation for MSE. We take K to have support $[-1, 1]$ and m to have support $[0, 1]$. It is shown that if x is a boundary point ($x = ch, 0 < c < 1$), then

$$\mathrm{MSE}(\hat{m}_\tau(x)) \simeq \frac{1}{4}h^4\alpha_c(K)^2 m''_\tau(0+)^2 + \frac{\beta_c(K)\tau(1-\tau)}{nhm(0+)f(m_\tau(0+)|0+)^2},$$

where

$$\alpha_c(K) = \frac{a_2^2(c;K) - a_1(c;K)a_3(c;K)}{a_0(c;K)a_2(c;K) - a_1^2(c;K)},$$

$$\beta_c(K) = \frac{\int_{-1}^{c}\{a_2(c;K) - a_1(c;K)t\}^2K(t)dt}{\{a_0(c;K)a_2(c;K) - a_1^2(c;K)\}^2},$$

and

$$a_l(c;K) = \int_{-1}^{c} t^lK(t)dt, \qquad l = 0, 1, 2.$$

These MSE expressions reflect two advantages: One is that the asymptotic bias is independent of the design density $m(x)$; The other is the good behaviors have been kept at boundaries with regard to orders of magnitude.

Remark 2.3.2 (Choice of Bandwidth h). Under the guidance of MSE expressions, the strategies of bandwidth selection are provided (Ruppert, Sheather, & Wand, 1995). The sophisticated methods to select h_{mean} is, for $\tau > 0$, to utilize

$$h_p = h_{\mathrm{mean}}\{\tau(1-\tau)\phi(\Phi^{-1}(\tau))^2\}^{1/5}.$$

Chapter 3

Robust Neural Network

Suppose that $\{(x_i, y_i)\}_{i=1}^n$ is a random sample from distribution (X, Y). Often, linearity is not enough to model the relation between X and Y. This motivates us to adopt an nonparametric function $m(\cdot)$ to replace the simple linear model. Furthermore, while there are outliers in the data, the conditional mean function of $Y|X$ is not reliable as an estimator for the central information. Conditional median will be a good candidate under such situation. To obtain an complete picture of the conditional distribution of $Y|X$ at some specific $X = x$, we denote the conditional τth quantile of $Y|X$ as

$$Q_y(\tau|x) = m_\tau(x).$$

$\tau = 0.5$ gives the special case of conditional median.

We use neural network as the nonparametric smoothing method and incorporate it with the quantile regression. We will demonstrate in next chapter that the neural network possesses some desired property. Equipped with quantile regression, the robust neural network (RNN) is a powerful tool for estimating the conditional quantile curves.

3.1 Robust Neural Network

Suppose $\hat{y}_i$ is the estimator of y_i. Under least squared loss, we aim

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{3.1}$$

to be small. When mean is not a good representative of the center, we consider absolute loss as a robust alternative candidate:

$$\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{3.2}$$

which leads to the median for the conditional distribution of $Y|X$. More generally, with the quantile loss,

$$\sum_{i=1}^{n}\rho_\tau(y_i - \hat{y}_i), \tag{3.3}$$

where $\rho_\tau(\cdot)$ is defined in (2.26), the resulting estimate is $\hat{Q}_y(\tau|x)$. Least absolute deviation (LAD) is a special case of quantile loss with $\tau = 0.5$. Furthermore, with quantile loss criterion, we are capable of giving the prediction intervals for new observation y^* given $X = x^*$.

For $i = 1, \ldots, n$, let $r_i = y_i - \hat{y}_i$. Applying the concept of robust estimation to neural network structure, we aim to minimize the error

$$E = \sum_{i=1}^{n}\rho_\tau(r_i) = \sum_{i=1}^{n}\rho_\tau\left[y_i - \left\{\sum_{j=1}^{M}\beta_j g(x_i\omega_j + \omega_{0j}) + \beta_0\right\}\right], \tag{3.4}$$

with respect to weights $\boldsymbol{\beta}$ and $\boldsymbol{\omega}$, where $(\boldsymbol{\beta}, \boldsymbol{\omega})$ are in parameter space (B, Ω) and

have form

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_M \end{pmatrix}, \quad \boldsymbol{\omega} = \begin{pmatrix} \omega_{01} & \omega_1 \\ \omega_{02} & \omega_2 \\ \vdots & \vdots \\ \omega_{0M} & \omega_M \end{pmatrix}, \tag{3.5}$$

In (3.5), β_0 and ω_{0j} $(j = 1, \ldots, M)$ are bias weights that are added to output and hidden neurons respectively as shown in Figure 2.2. β_j, ω_j $(j = 1, \ldots, M)$ are weights connecting between neurons and input/output.

3.2 MM Algorithm in Robust Neural Network

Gradient descent is a common optimization algorithm for finding a minimum of a function. It involves the first derivative and sometimes the second derivative of the objective function. However, in our setting, $\rho_\tau(\cdot)$ is not even first order differentiable. To minimize (3.4), MM algorithm (Hunter & Lange, 2000) is adopted in our proposal. In our context, MM algorithm stands for Majorization-Minimization. Basically, it creates a surrogate function that majorize the objective function. Then, optimizing objective function is driven by optimizing surrogate function instead.

In "Majorization" step, we use quadratic surrogate function $\rho_\tau^m(r)$

$$\rho_\tau^m(r) = \frac{r^2}{2|r_k|} + \frac{|r_k|}{2} + (2\tau - 1)r \geq |r| + (2\tau - 1)r = \rho_\tau(r), \tag{3.6}$$

to majorize $\rho_\tau(r)$ at $r = r_k$. If r is close to 0, a small perturbation a_n will be added

to the denominator of the first term in (3.6) as

$$\rho_\tau^m(r, a_n) = \frac{r^2}{2|r_k| + a_n} + \frac{|r_k|}{2} + (2\tau - 1)r, \tag{3.7}$$

where $a_n \to 0$. For $i = 1, \ldots, n$, define R_i to be the absolute value of residual r_i. Then in our case, a_n in (3.7) is chosen as

$$a_n = \min\{a_{r,n}, 0.01\},$$

where $a_{r,n}$ is the $(2/\sqrt{n})$ 100th-percentile of the order statistics $R_{(1)}, \ldots, R_{(n)}$, i.e., $a_{r,n} = R_{(\lfloor 2\sqrt{n} \rfloor)}$.

In the "Minimization" step, an iterative algorithm, Backpropagation (Rumelhart et al., 1986) is applied. It studies a single observation (x_t, y_t) in each iteration, may revisit the datum after a loop of going through all data and stops the iteration when the estimators converge. In the study of (x_t, y_t), it propagates information from predictor x_t forward through neurons by current input/output weights. After calculating the error between observed and predicted y_t, it goes back to adjust the weights and starts the next round of study with another datum (x_{t^*}, y_{t^*}). Unlike the Newton-Raphson algorithm, back propagation updates the connecting weights one at a time. Since the number of parameters (weights) grows rapidly as hidden neurons increase, it actually avoids the action of inverse of a huge matrix though it seems not so efficient as N-R method, which updates all parameters at the same time. The procedure of back propagation in robust neural network is as follows:

Step 1: Initialize the weights (β, ω) to small random values.

Step 2: Choose an observation (x_i, y_i) along with the initial weights to get $\hat{y}_i$ via (2.23) and calculate $E_i = \rho_\tau^m(r_i)$, where $r_i = y_i - \hat{y}_i$.

Step 3: Compute $\triangle\boldsymbol{\beta}$ by

$$\triangle\beta_0 = -\eta\frac{\partial E_i}{\partial\beta_0}, \quad \triangle\beta_j = -\eta\frac{\partial E_i}{\partial\beta_j}, \quad j = 1, \ldots, M \tag{3.8}$$

where η is the learning rate decided in advance,

$$\frac{\partial E_i}{\partial\beta_0} = \frac{\partial E_i}{\partial r_i}\cdot\frac{\partial r_i}{\partial\beta_0} = \left.\frac{\mathrm{d}\rho_\tau^m(r)}{\mathrm{d}r}\right|_{r=r_i}\cdot(-1)$$

and

$$\frac{\partial E_i}{\partial\beta_j} = \frac{\partial E_i}{\partial r_i}\cdot\frac{\partial r_i}{\partial\beta_j} = \left.\frac{\mathrm{d}\rho_\tau^m(r)}{\mathrm{d}r}\right|_{r=r_i}\cdot\left\{-g(\omega_j x_i + \omega_{0j})\right\}.$$

Step 4: Compute $\triangle\boldsymbol{\omega}$ by

$$\triangle\omega_{0j} = -\eta\frac{\partial E_i}{\partial\omega_{0j}}, \quad \triangle\omega_j = -\eta\frac{\partial E_i}{\partial\omega_j}, \quad j = 1, \ldots, M \tag{3.9}$$

where η is the same as the last step,

$$\frac{\partial E_i}{\partial\omega_{0j}} = \frac{\partial E_i}{\partial r_i}\cdot\frac{\partial r_i}{\partial g}\cdot\frac{\partial g}{\partial\omega_{0j}} = \left.\frac{\mathrm{d}\rho_\tau^m(r)}{\mathrm{d}r}\right|_{r=r_i}\cdot\left\{-\beta_j g'(\omega_j x_i + \omega_{0j})\right\}\cdot 1$$

and

$$\frac{\partial E_i}{\partial\omega_j} = \frac{\partial E_i}{\partial r_i}\cdot\frac{\partial r_i}{\partial g}\cdot\frac{\partial g}{\partial\omega_j} = \left.\frac{\mathrm{d}\rho_\tau^m(r)}{\mathrm{d}r}\right|_{r=r_i}\cdot\left\{-\beta_j g'(\omega_j x_i + \omega_{0j})\right\}\cdot x_i.$$

Step 5: Update weights at $(t+1)^{\text{th}}$ iteration according to

$$\begin{aligned} \beta_0^{t+1} &= \beta_0^t + \triangle\beta_0, \qquad \beta_j^{t+1} = \beta_j^t + \triangle\beta_j, \\ \omega_{0j}^{t+1} &= \omega_{0j}^t + \triangle\omega_{0j}, \quad \omega_j^{t+1} = \omega_j^t + \triangle\omega_j, \end{aligned} \tag{3.10}$$

where $\triangle\beta_0,\ \triangle\beta_j,\ \triangle\omega_{0j}$ and $\triangle\omega_j$ $(j = 1, \ldots, M)$ are from (3.8) and (3.9).

Step 6: Repeat all the steps above for each observation until estimators converge.

Remark 3.2.1 (Neural network structure). Too many hidden neurons or too few may cause over- or under-fitting issues respectively. Therefore, it is necessary to choose a proper number of neurons according to different data. More details about selecting a proper number of neurons will be discussed in next section.

Remark 3.2.2 (Activation function). Selection of activation function g is also essential. The most popular ones are *Sigmoid* and *tanh*. Sometimes, not both of them works well in real data. Having different trials are helpful.

Remark 3.2.3 (Initial values). Due to the high dimension of the parameter space (B, Ω), several trials of different initial values is beneficial to avoiding being trapped in local extremes. In practice, it is better to set initial values between 0 & 1.

Remark 3.2.4 (Learning rate). It is rather to give small learning rate to prevent taking large steps in the gradient direction. Small learning rate provides careful search in parameter space though it is time-consuming. In our numerical study, constant learning rate $\eta = 0.01$ is used. Moreover, there are researches on dynamic or optimal learning rate to improve convergence rate. Interested readers are referred to the considerable literature on neural networks (Darken, Chang, & Moody, 1992; LeCun et al., 1993; Roy, 1993).

Remark 3.2.5 (Normalization). Normalization is essential. Due to the character of sigmoid-type function, it will not have good result if x's are too wild. Usually, we use $\dfrac{x - \bar{x}}{\mathrm{std}(x)}$ before data are sent to train neural network. Some normalization or transformation should also be applied to y's if necessary.

Remark 3.2.6 (Stopping rules). Stopping rules should be selected carefully to avoid early stop or non-stop. Some authors also suggest to split the data into two sets, one of which is training set and the other of which is testing set. This is doable if data have enough observation for splitting.

More discussions regarding issues worth concerning such as the selections of initial value, learning rate, network structure and training algorithm can be found in the literature, for example, in (Haykin, 1994; Hassoun, 1995; Bishop, 1995; Fang, Li, & Sudjianto, 2005).

3.3 Selecting Number of Neurons

As mentioned in the previous section, the number of hidden neurons is important to function approximation via neural network. Too many neurons will lead to over-fitting and computational time-consuming. Suppose we have M neurons, then there are $3M + 1$ parameters, i.e., M each for input weight, input bias, output weight and 1 for output bias. Whenever one extra neuron is added, three additional parameters are on board immediately, i.e., input weight ω_{M+1}, input bias $\omega_{0,M+1}$and output weight β_{M+1}. Since the parameter space (B, Ω) is already complex and high dimensional, some variable selection method should be adopted as to reduce the dimension as low as possible.

We use traditional forward stepwise selection for the choice of M. The checking criterions are AIC and BIC as follows:

$$AIC(M) = n \log \frac{RSS}{n} + 2(3M + 1)$$

and

$$BIC(M) = n \log \frac{RSS}{n} + \log(n) \cdot (3M + 1)$$

where $RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$. The value of M which minimizes AIC or BIC will be selected as the number of neurons in the hidden layer.

Chapter 4

Numerical Results

In this section, we explore the behavior of robust neural network via simulation studies and a real data example. In our simulations, three types of random errors are used to illustrate the robustness of our method. For real data study, we compare neural network with other nonparametric function approximation methods.

4.1 Simulation Study

4.1.1 Basic Settings

We propose four functions with different features:

$$\begin{aligned}
&\text{Model 1}: \quad m(x) = e^{-x}\left\{\sin(2\pi x + 1) + 2\right\} \\
&\text{Model 2}: \quad m(x) = \cos(3\pi x) \\
&\text{Model 3}: \quad m(x) = 1.6x + 0.2 \\
&\text{Model 4}: \quad m(x) = x^2 \sin(4x^2 - 4)
\end{aligned}$$

For each of these functions, data of sample size $n = 500$ are generated from

$$y_i = m(x_i) + \epsilon_i, \quad i = 1, ..., n \tag{4.1}$$

where x_i is generated from $Uniform(0, 1)$ and random error ϵ_i's are drawn from the following distributions respectively:

(a) Gaussian Distribution $\mathcal{N}(0, \sigma^2)$,

(b) Mixed Gaussian Distribution, 90% from $\mathcal{N}(0, \sigma^2)$ and 10% from $\mathcal{N}(0, (3\sigma)^2)$,

(c) Laplace Distribution with $E(\epsilon) = 0$ and $Var(\epsilon) = 2\sigma^2$. Therefore, the scalar parameter $b = \sigma$ in our setting,

In these four cases, σ takes different values: (1) $\sigma = 0.2$; (2) $\sigma = 0.1$; (3) $\sigma = 0.15$ and (4) $\sigma = 0.4$. The numerical results will be checked from two aspects: graphical fitting of median (50% quantile) and table summaries of various quantile estimations with their standard error. Meanwhile, the true values of these quantiles are provided for comparison purpose.

4.1.2 Overall Fitting

3-Layer MLP network is used with 5 neurons in the hidden layer. The median fitting results of all four models are shown in Figure 4.1 - Figure 4.4. Hyperbolic tangent function $\tanh(x)$ is chosen for $g(x)$ as the activation function. In all plots, true curve is represented by dashed line and dots are the observed samples. Solid line gives the smoothed $\hat{y}$'s evaluated at each x in one typical simulation. As we can see, robust neural network approximates well under Gaussian errors without surprise. Furthermore, under errors of heavy tail distributions such as mixed Gaussian error and Laplace error, it also provide satisfactory results.

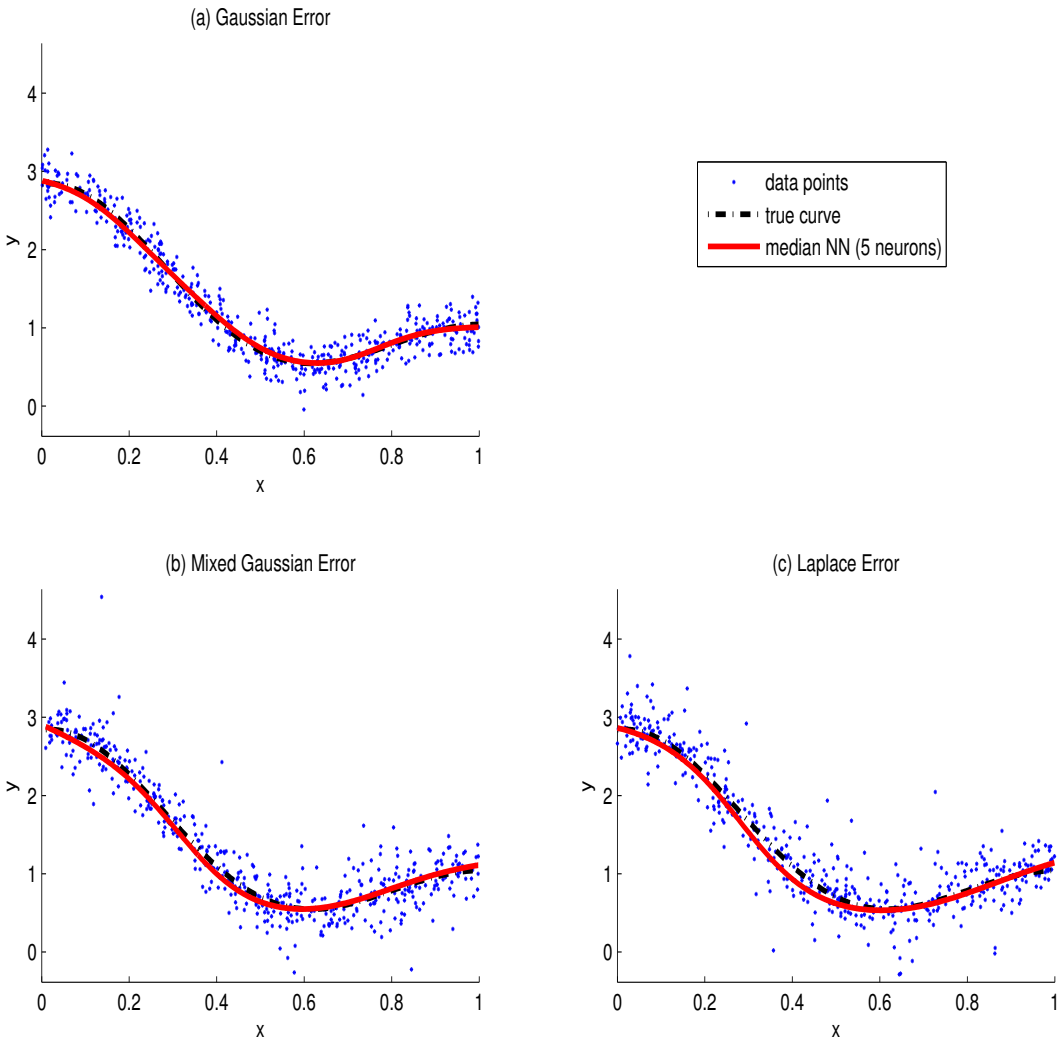

Figure 4.1. Model 1 - Median fit under three error distributions

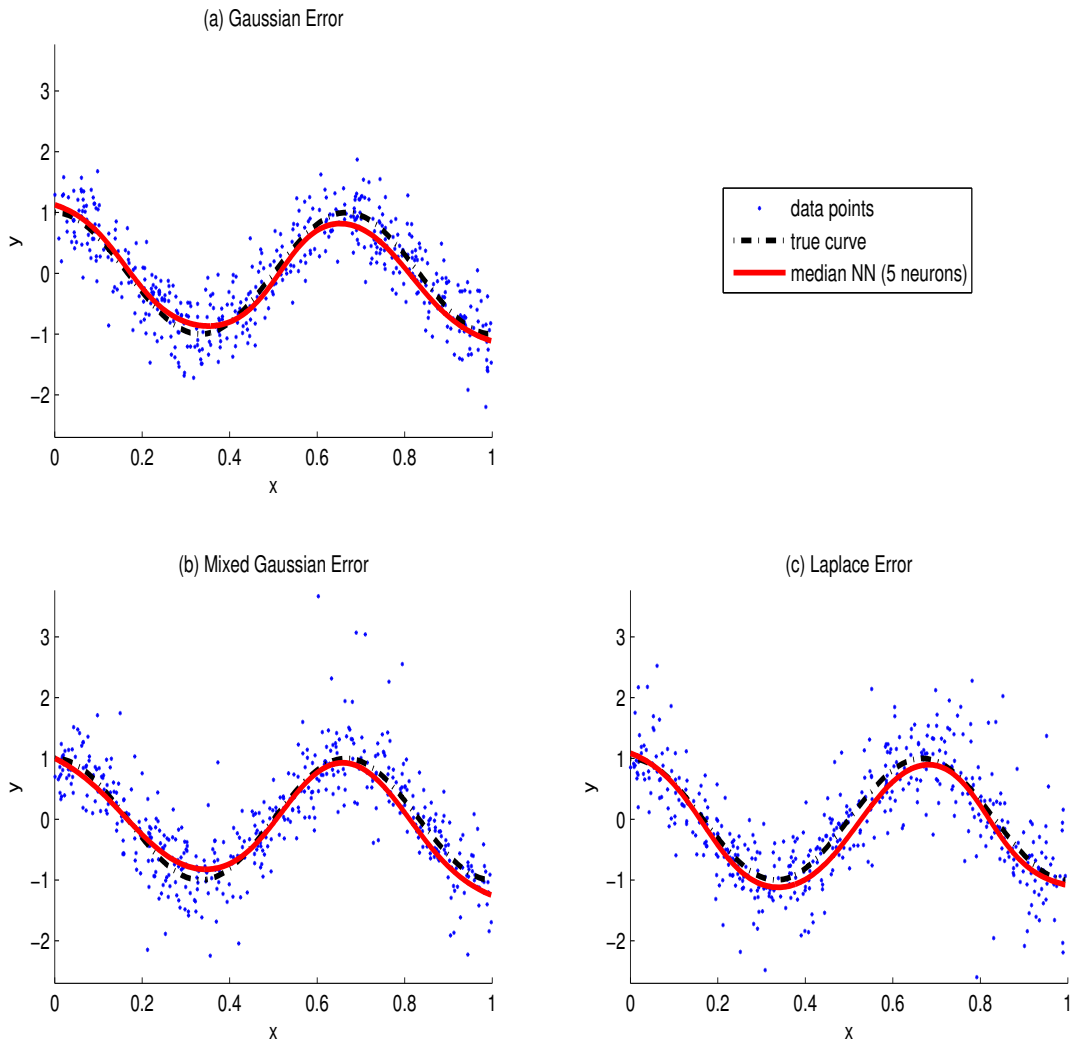

Figure 4.2. Model 2 - Median fit under three error distributions

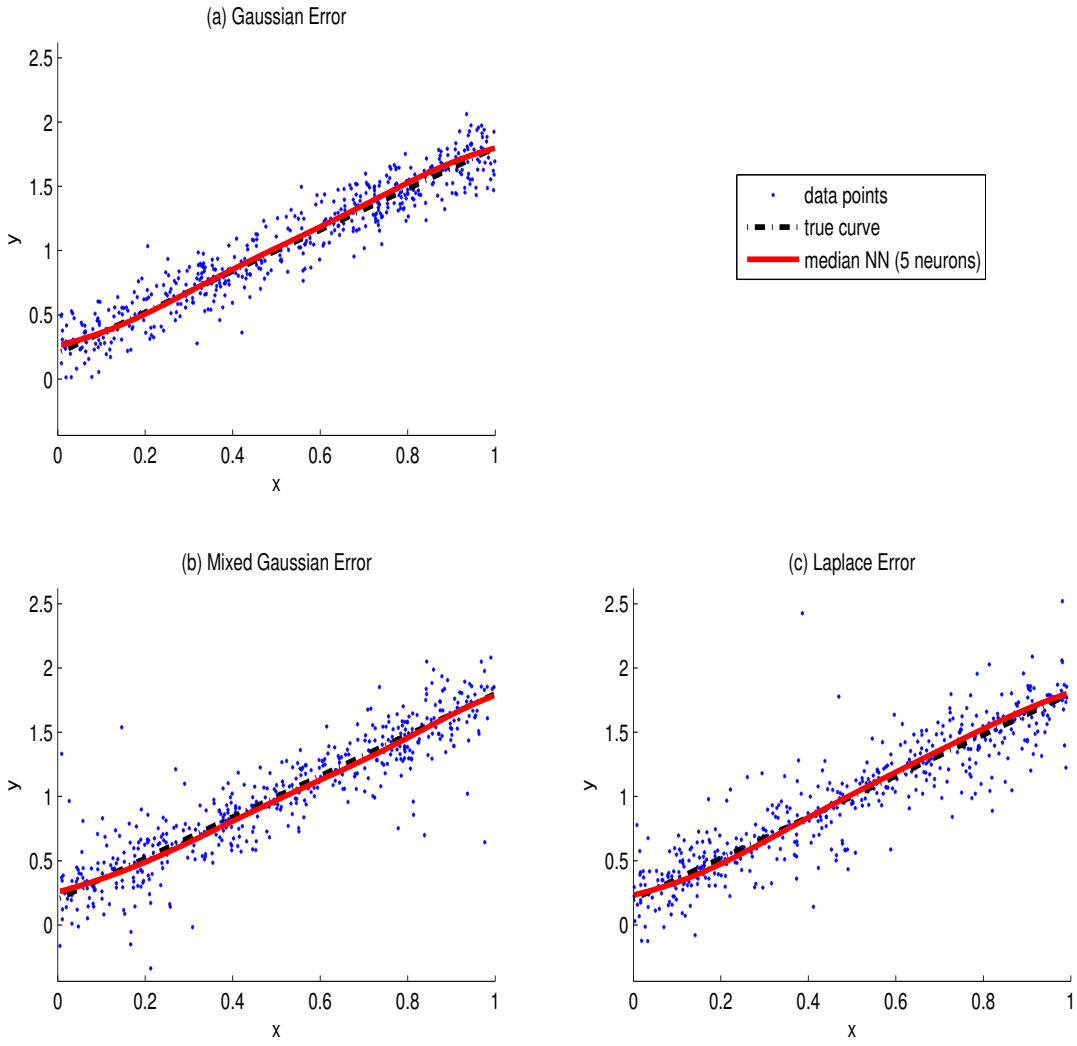

Figure 4.3. Model 3 - Median fit under three error distributions

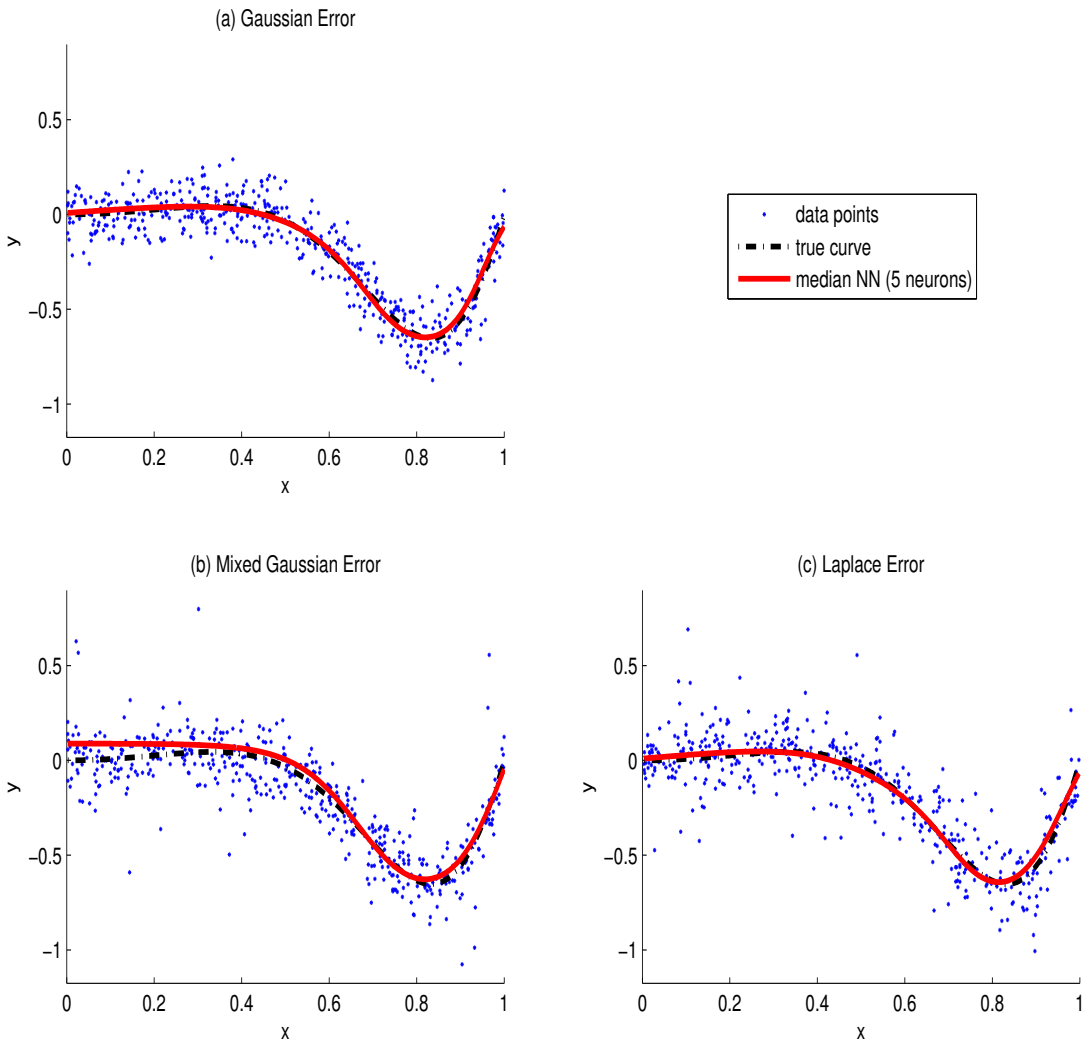

Figure 4.4. Model 4 - Median fit under three error distributions

4.1.3 Quantile Estimation

The true quantiles and their estimates of model 3 are listed in Table 4.1, 4.2 and 4.3 based on 1000 simulations. The standard deviations of 1000 conditional quantile estimations are provided in parentheses.

The true τth conditional quantile evaluated at x is calculated by

$$\Phi^{-1}_{Y|X=x}(\tau) = \Phi^{-1}_{\epsilon}(\tau) + m(x),$$

where $\Phi^{-1}_{Y|X=x}$ is the inverse CDF of corresponding conditional distribution $Y|X = x$. It is composed of the inverse CDF of its error distribution Φ^{-1}_{ϵ} and true function m in (4.1) evaluated at x. Due to the difficulty of obtaining inverse CDF of a Mixed Gaussian distribution, we calculate estimate of true quantile values via Monte Carlo method instead.

Table 4.1. Model 1 - Gaussian error

	$x = 0.3$		$x = 0.6$		$x = 0.9$	
Quantile	True	Predicted	True	Predicted	True	Predicted
5%	1.3407	1.3156	0.2207	0.2511	0.6318	0.5979
		(0.0792)		(0.0392)		(0.0594)
25%	1.5348	1.5359	0.4148	0.4234	0.8259	0.8206
		(0.0707)		(0.0396)		(0.0647)
50%	1.6697	1.6801	0.5497	0.5623	0.9608	0.9608
		(0.0755)		(0.0447)		(0.0686)
75%	1.8046	1.8293	0.6846	0.7077	1.0957	1.0966
		(0.0887)		(0.0524)		(0.0719)
95%	1.9986	2.0571	0.8787	0.9531	1.2898	1.2963
		(0.1272)		(0.0639)		(0.0698)

We provide simulation results for the above three types of error distributions at three grids $x = 0.3, 0.6, 0.9$. These grid points represent three typical situations in

Table 4.2. Model 1 - Mixed Gaussian error

	$x = 0.3$		$x = 0.6$		$x = 0.9$	
Quantile	True	Predicted	True	Predicted	True	Predicted
5%	1.2812	1.2575	0.1632	0.1633	0.5790	0.5030
		(0.0991)		(0.0739)		(0.0944)
25%	1.5233	1.5131	0.4046	0.3969	0.8142	0.7925
		(0.0827)		(0.0593)		(0.0762)
50%	1.6718	1.6799	0.5494	0.5536	0.9599	0.9514
		(0.0843)		(0.0614)		(0.0751)
75%	1.8176	1.8496	0.6952	0.7183	1.1064	1.1135
		(0.1018)		(0.0690)		(0.0813)
95%	2.0556	2.1660	0.9370	0.9996	1.3525	1.3866
		(0.1269)		(0.0931)		(0.1042)

Table 4.3. Model 3 - Laplace error

	$x = 0.3$		$x = 0.6$		$x = 0.9$	
Quantile	True	Predicted	True	Predicted	True	Predicted
5%	1.2092	1.1475	0.0892	0.1556	0.5003	0.3508
		(0.1094)		(0.0722)		(0.0998)
25%	1.5310	1.5122	0.4111	0.4374	0.8222	0.7663
		(0.1123)		(0.0592)		(0.0780)
50%	1.6697	1.6807	0.5497	0.5819	0.9608	0.9353
		(0.0816)		(0.0532)		(0.0666)
75%	1.8083	1.8514	0.6883	0.7383	1.0994	1.0943
		(0.0957)		(0.0646)		(0.0855)
95%	2.1302	2.2130	1.0102	1.0979	1.4213	1.4198
		(0.1298)		(0.0990)		(0.1112)

interval $[0, 1]$: $x = 0.3$ is in the area that curve sharply decreases; $x = 0.6$ is in the neighborhood of a local minimum; $x = 0.9$ is located where the curve tends flat. At each grid point, we compare predicted value with the true value at 5 different quantiles $= \{0.05, 0.25, 0.50, 0.75, 0.95\}$.

The estimates for median are the best among all quantiles regardless of the

error distribution. As quantile goes more extreme to 0 or 1, the estimates are not as good as the ones of median but still include the true value within 2 standard errors most the cases. For comparison among different grids, estimates at $x = 0.6$ seem to behave not as good as those at $x = 0.3$ and $x = 0.9$. It is common in function approximation for fitting the local extremes. From all three tables above, standard errors of median estimations are similar for all distributions. When the quantiles are more extreme, the estimated quantiles have larger standard errors for mixed Gaussian and Laplace error distributions than for Gaussian due to their heavy tails.

The results for the other models are summarized as below:

Table 4.4. Model 2 - Gaussian error

	$x = 0.3$		$x = 0.6$		$x = 0.9$	
Quantile	True	Predicted	True	Predicted	True	Predicted
5%	-1.6090	-1.5374	0.1511	0.2136	-1.2457	-1.2505
		(0.0954)		(0.1312)		(0.1306)
25%	-1.2209	-1.1756	0.5392	0.5753	-0.8576	-0.8507
		(0.0927)		(0.1101)		(0.1395)
50%	-0.9511	-0.9089	0.8090	0.8435	-0.5878	-0.5754
		(0.0961)		(0.1019)		(0.1492)
75%	-0.6813	-0.6034	1.0788	1.1420	-0.3180	-0.2468
		(0.1348)		(0.1050)		(0.1665)
95%	-0.2931	-0.1967	1.4670	1.5257	0.0702	0.1828
		(0.1488)		(0.0977)		(0.1546)

Table 4.5. Model 2 - Mixed Gaussian error

	$x = 0.3$		$x = 0.6$		$x = 0.9$	
Quantile	True	Predicted	True	Predicted	True	Predicted
5%	-1.7198	-1.6840	0.0404	0.0116	-1.3538	-1.3456
		(0.1625)		(0.2066)		(0.1708)
25%	-1.2413	-1.1801	0.5165	0.5802	-0.8806	-0.8197
		(0.1369)		(0.1621)		(0.1735)
50%	-0.9536	-0.9136	0.8105	0.8526	-0.5890	-0.5409
		(0.1292)		(0.1366)		(0.1715)
75%	-0.6580	-0.5486	1.1007	1.1972	-0.2986	-0.1635
		(0.1744)		(0.1404)		(0.1814)
95%	-0.1845	0.1254	1.5892	1.7124	0.1812	0.3860
		(0.2470)		(0.1781)		(0.2344)

Table 4.6. Model 2 - Laplace error

	$x = 0.3$		$x = 0.6$		$x = 0.9$	
Quantile	True	Predicted	True	Predicted	True	Predicted
5%	-1.8721	-1.8818	-0.1120	-0.0789	-1.5088	-1.5128
		(0.1959)		(0.2021)		(0.1979)
25%	-1.2283	-1.2005	0.5318	0.5602	-0.8650	-0.8501
		(0.1449)		(0.1707)		(0.1734)
50%	-0.9511	-0.9375	0.8090	0.8001	-0.5878	-0.6107
		(0.1218)		(0.1199)		(0.1231)
75%	-0.6738	-0.5903	1.0863	1.1568	-0.3105	-0.2305
		(0.1670)		(0.1379)		(0.1814)
95%	-0.0300	0.2537	1.7301	1.8687	0.3332	0.4577
		(0.2430)		(0.2040)		(0.2450)

Table 4.7. Model 3 - Gaussian error

	$x = 0.3$		$x = 0.6$		$x = 0.9$	
Quantile	True	Predicted	True	Predicted	True	Predicted
5%	0.4333	0.4353	0.9133	0.8839	1.3933	1.3850
		(0.0382)		(0.0425)		(0.0439)
25%	0.5788	0.5743	1.0588	1.0471	1.5388	1.5428
		(0.0401)		(0.0494)		(0.0485)
50%	0.6800	0.6837	1.1600	1.1543	1.6400	1.6495
		(0.0433)		(0.0417)		(0.0417)
75%	0.7812	0.7896	1.2612	1.2635	1.7412	1.7576
		(0.0453)		(0.0446)		(0.0392)
95%	0.9267	0.9400	1.4067	1.4230	1.8867	1.9075
		(0.0461)		(0.0405)		(0.0365)

Table 4.8. Model 3 - Mixed Gaussian error

	$x = 0.3$		$x = 0.6$		$x = 0.9$	
Quantile	True	Predicted	True	Predicted	True	Predicted
5%	0.3899	0.3390	0.8703	0.8521	1.3505	1.3010
		(0.0502)		(0.0617)		(0.0840)
25%	0.5713	0.5562	1.0507	1.0452	1.5310	1.5251
		(0.0447)		(0.0511)		(0.0583)
50%	0.6798	0.6663	1.1608	1.1657	1.6399	1.6399
		(0.0453)		(0.0513)		(0.0556)
75%	0.7899	0.7909	1.2703	1.2778	1.7499	1.7619
		(0.0463)		(0.0484)		(0.0476)
95%	0.9705	1.0213	1.4511	1.4677	1.9295	1.9703
		(0.0660)		(0.0615)		(0.0711)

Table 4.9. Model 3 - Laplace error

	$x = 0.3$		$x = 0.6$		$x = 0.9$	
Quantile	True	Predicted	True	Predicted	True	Predicted
5%	0.3346	0.2445	0.8146	0.7550	1.2946	1.2960
		(0.0596)		(0.0698)		(0.0737)
25%	0.5760	0.5659	1.0560	1.0427	1.5360	1.5231
		(0.0462)		(0.0529)		(0.0562)
50%	0.6800	0.6818	1.1600	1.1556	1.6400	1.6427
		(0.0428)		(0.0456)		(0.0434)
75%	0.7840	0.7975	1.2640	1.2737	1.7440	1.7528
		(0.0570)		(0.0535)		(0.0510)
95%	1.0254	1.0448	1.5054	1.5955	1.9854	1.9686
		(0.0652)		(0.0595)		(0.0614)

Table 4.10. Model 4 - Gaussian error

	$x = 0.3$		$x = 0.6$		$x = 0.9$	
Quantile	True	Predicted	True	Predicted	True	Predicted
5%	-0.1215	-0.1258	-0.3623	-0.4036	-0.7225	-0.7718
		(0.0236)		(0.0332)		(0.0251)
25%	-0.0244	-0.0441	-0.2652	-0.2500	-0.6255	-0.6223
		(0.0212)		(0.0324)		(0.0368)
50%	0.0430	0.0380	-0.1978	-0.1819	-0.5580	-0.5428
		(0.0240)		(0.0319)		(0.0490)
75%	0.1105	0.1104	-0.1303	-0.1167	-0.4906	-0.4591
		(0.0210)		(0.0301)		(0.0569)
95%	0.2075	0.2093	-0.0333	-0.0072	-0.3935	-0.3156
		(0.0187)		(0.0319)		(0.0631)

Table 4.11. Model 4 - Mixed Gaussian error

	$x = 0.3$		$x = 0.6$		$x = 0.9$	
Quantile	True	Predicted	True	Predicted	True	Predicted
5%	-0.1485	-0.1827	-0.3898	-0.3968	-0.7481	-0.7616
		(0.0415)		(0.0456)		(0.0646)
25%	-0.0295	-0.0574	-0.2701	-0.2502	-0.6308	-0.6131
		(0.0329)		(0.0396)		(0.0516)
50%	0.0430	0.0232	-0.1976	-0.1749	-0.5578	-0.5384
		(0.0327)		(0.0388)		(0.0464)
75%	0.1161	0.1234	-0.1247	-0.1189	-0.4846	-0.4459
		(0.0358)		(0.0388)		(0.0618)
95%	0.2368	0.2550	-0.0053	0.0315	-0.3651	-0.2031
		(0.0404)		(0.0582)		(0.0766)

Table 4.12. Model 4 - Laplace error

	$x = 0.3$		$x = 0.6$		$x = 0.9$	
Quantile	True	Predicted	True	Predicted	True	Predicted
5%	-0.1872	-0.2256	-0.4280	-0.4241	-0.7883	-0.7801
		(0.0394)		(0.0529)		(0.0528)
25%	-0.0263	-0.0487	-0.2671	-0.2588	-0.6273	-0.6274
		(0.0305)		(0.0352)		(0.0401)
50%	0.0430	0.0427	-0.1978	-0.1922	-0.5580	-0.5372
		(0.0304)		(0.0333)		(0.0522)
75%	0.1123	0.1044	-0.1285	-0.1001	-0.4887	-0.4501
		(0.0276)		(0.0354)		(0.0489)
95%	0.2733	0.3090	0.0325	0.0120	-0.3278	-0.0990
		(0.0379)		(0.0479)		(0.0545)

To summarize the conclusions, we note that overall neural network is a good approximator for complicated functions. However, it is also sensitive to extreme observations like the other methods that utilize least squared loss. Robust neural network, in our setting, provides good estimation in the presence of outliers.

4.2 Real Data Application

4.2.1 Introduction

In this example we apply the robust neural network methodology in a credit card portfolio dataset. The charge-off rate, i.e. the proportion of accounts that defaults, is an important statistic in credit card risk analysis. To study the charge-off rate, we classify each account into various segments according to their characteristics. Furthermore, the data are grouped by vintages, i.e. by the month in calender time that the account was opened. Therefore each observation in our dataset represents one vintage. The number of months that an individual account exists is called Month-on-Book (MoB). It is believed that the trend of charge-off rate can be decomposed into two additive components: (1) a maturation curve to characterize overall trend under "economic-neutral" condition; and (2) an exogenous curve to characterize the effect of exogenous factors (e.g., economic cycle, management policy changes, financial crisis).

Our data include charge-off rate information of 1176 vintages from June 2002 to May 2006 for two segments of accounts respectively. The maximum MoB is 48 months in both segments. We focus on modelling the maturation curve for every segment. More specifically, within one segment, we model the charge-off rate at some value of MoB as

$$\text{charge-off rate} = \frac{\#\text{ of accounts default}}{\#\text{ of total accounts in this vintage}}.$$

Typically in a credit card data, it is expected that the charge-off rate is a stable and monotone curve over MoB.

4.2.2 Maturation Curve Fitting

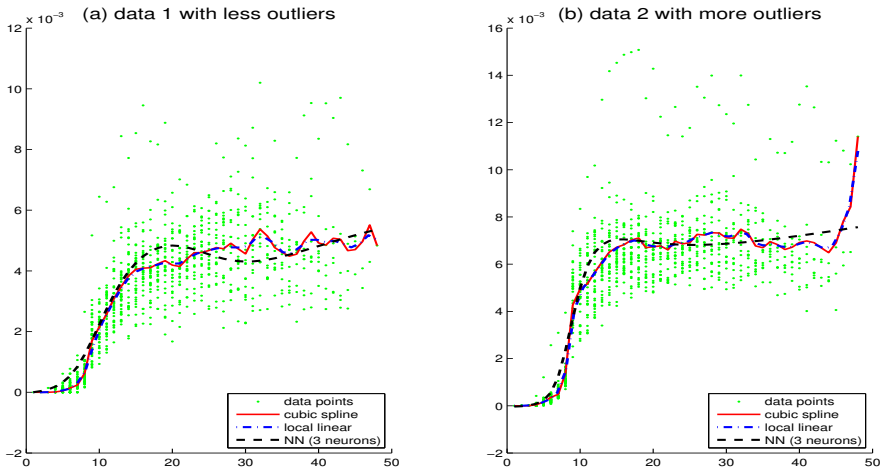

Figure 4.5. Three nonparametric method under Least Squared Loss

We first fit the nonparametric regression models introduced in Chapter 2 for this dataset. There are several practical nonparametric methods for function approximation besides neural network, such as kernel regression, local polynomial, regression splines, smoothing splines, etc. To illustrate the performances, we fit the model with local linear regression, cubic splines and 3-layer MLP neural network with 3 hidden neurons in this study. Figure 4.5 gives the graphical results of data fitting for all three methods under least squared loss for two segments respectively. In our context, neural network method is superior due to its smooth fitting while other methods will oscillate over month-on-book.

As we can see in Figure 4.5, both segments have some potential outliers. The impact is more severe in the second segment. As opposed to smooth behavior of neural network, local linear in solid line and cubic spline in dash-dot line both have wiggling characteristic that is undesirable. However, all these three methods have a fatal disadvantage due to the least squared loss setting. Near the right-hand-side boundary, the estimations are pulled up by outliers. Especially in Figure 4.5 (b),

local linear and cubic spline estimation have tails being pulled up quite seriously. In that case, we go further and alter the setting from least squared loss to general quantile loss, and use robust neural network for both segments.

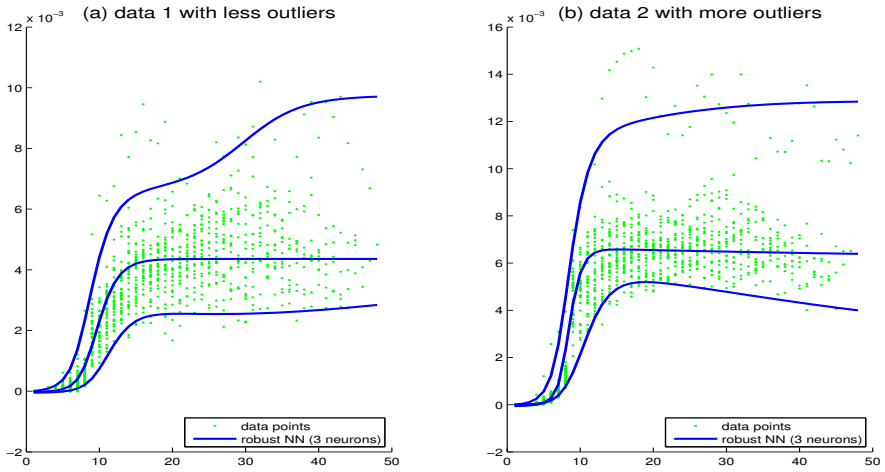

Figure 4.6. Robust Neural Network Estimation under General Quantile Loss

Figure 4.6 shows the quantile estimation results for both segments. In both plots, the lines from bottom are 5%, 50% and 95% quantile estimations respectively. The tails of each curve are not pulled up anymore, thus the estimations are rather robust against outliers. Furthermore, smoothness is no doubt a good match with stability and monotone of the maturation curve. In a more practical point of view, the upper and lower curves evaluated at each x value provide the 90% prediction interval for charge-off rate given the information of a certain vintage.

Chapter 5

Discussion and Future Work

In this thesis, we combine the ideas of neural network with quantile regression, and propose the robust neural network (RNN). RNN is a powerful global nonparametric estimation tool, and is insensitive to outliers. Our numerical results indicates it is comparative to other popular nonparametric regression methods, and its approximation behavior is even better in the case of underlying function is flat. Besides all these features, it gives the prediction interval based on the general quantile loss setting, which makes RNN a desirable tool in many areas which a prospective view of the future observation is desired.

Optimization in neural network has always been very challenging, and so it is in RNN. Future work may include improving the current optimization procedure. Studies of choosing proper initial weights and dynamic learning rate might be helpful to build a model more efficiently. Furthermore, it is also of interest that introducing more sophisticated variable selection methods on the selection of neurons.

References

Allen, D. M. (1974). The Relationship between Variable Selection and Data Agumentation and a Method for Prediction. *Technometrics*, *16*(1), 125-127.

Baxt, W. G. (1990). Use of an Artificial Neural Network for Data Analysis in Clinical Decision-Making: The Diagnosis of Acute Coronary Occlusion. *Neural Comput.*, *2*(4), 480-489.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc.

Chaudhuri, P. (1991). Nonparametric Estimates of Regression Quantiles and Their Local Bahadur Representation. *The Annals of Statistics*, *19*(2), 760-777.

Cheng, W., & Titterington, D. M. (1994). Neural Networks: A Review from a Statistical Perspectivee. *Statist. Sci*, *9*(1), 2-30.

Cleveland, W. S. (1979). Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of the American Statistical Association*, *74*(368), 829-836.

Craven, P., & Wahba, G. (1979). Smoothing Noisy Data with Spline Functions: Estimating the Correct Degree of Smoothing by the Method of Generalized Cross-Validation. *Numer. Math.*, *31*, 377-403.

Cybenko, G. (1989). Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals, and Systems*, *2*, 303-314.

Darken, C., Chang, J., & Moody, J. (1992). Learning rate schedules for faster stochastic gradient search. In *Neural Networks for Signal Processing 2 — Proceedings of the 1992 IEEE Workshop*. Piscataway, NJ, USA: IEEE Press.

de Boor, C. (1978). *A practical guide to splines*. Applied Mathematical Sciences, New York: Springer.

Donoho, D. L., & Johnstone, I. M. (1994). Ideal Spatial Adaptation via Wavelet Shringkage. *Biometrika*, *81*, 425-455.

Donoho, D. L., & Johnstone, I. M. (1995a). Adapting to Unknown Smoothness via Wavelet Shringkage. *Journal of the American Statistical Association*, *90*(432), 1200-1224.

Donoho, D. L., & Johnstone, I. M. (1995b). Minimax Estimation via Wavelet Shringkage. *26*(3), 879-921.

Eubank, R. L. (1988). *Spline Smoothing and Nonparametric Regression*. New York: Marcel Dekker.

Fan, J. (1992). Design-adaptive Nonparametric Regression. *Journal of the American Statistical Association*, *87*(420), 998-1004.

Fan, J. (1993). Local Linear Regression Smoothers and Their Minimax Efficiencies. *The Annals of Statistics*, *21*(1), 196-216.

Fan, J., & Gijbels, I. (1992). Variable Bandwidth and Local Linear Regression Smoothers. *The Annals of Statistics*, *20*(4), 2008-2036.

Fan, J., & Gijbels, I. (1995). Data-driven Bandwidth Selection in Local Polynomial Fitting: Variable Bandwidth and Spatial Adaptation. *Journal of the Royal Statistical Society. Series B (Methodological)*, *57*(2), 371-394.

Fan, J., & Gijbels, I. (1996). *Local Polynomial Modelling and Its Applications*. London: Chapman & Hall.

Fan, J., Hu, T. chung, & Truong, Y. (1994). Robust Nonparametric Function Estimation. *Scandinavian journal of statistics*, *21*(4), 433-446.

Fan, J., Yao, Q., & Tong, H. (1996). Estimation of Conditional Densities and Sensitivity Measures in Nonlinear Dynamical Systems. *Biometrika*, *83*(1), 189-206.

Fang, K.-T., Li, R., & Sudjianto, A. (2005). *Design and Modeling for Computer*

Experiments. Chapman & Hall/CRC.

Gasser, T., & Müller, H.-G. (1979). Kernel estimation of Regression Functions. In *Smoothing Techniques for Curve Estimation.* New York: Springer-Verlag.

Gorman, R. P., & Sejnowski, T. J. (1988). Analysis of Hidden Units in a Layered Network to Classify Sonar Targets. *Neural Networks*, *1*, 75-89.

Härdle, W. (1990). *Applied Nonparametric Regression*. Boston: Cambridge Univerisity Press.

Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks*. Cambridge, MA, USA: MIT Press.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice Hall PTR.

Hopfield, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. In *Proceedings of the National Academy of Sciences of the USA* (Vol. 79, p. 2554-2558).

Hornik, K. (1993). Some New Results on Neural Network Approximation. *Neural Networks*, *6*(9), 1069-1072.

Hunter, D., & Lange, K. (2000). Quantile Regression via MM Algorithm. *Journal of Computational and Graphical Statistics*, *9*, 60-70.

Hutchinson, J. M., Lo, A. W., & Poggio, T. (1994). A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks. *Journal of Finance*, *49*(3), 851-889.

Koenker, R. (2005). *Quantile Regression*. Cambridge University Press.

Koenker, R., & Bassett, G., Jr. (1978). Regression quantiles. *Econometrica*, *46*(1), 33-50.

Koenker, R., Ng, P., & Portnoy, S. (1994). Quantile smoothing splines. *Biometrika*, *81*(4), 673-680.

Koenker, R., & Park, B. J. (1996). An Interior Point Algorithm for Nonlinear Quantile Regression. *Journal of Econometrics*, *71*(1-2), 265-283.

Lapedes, A., & Farber, R. (1988). Nonlinear Signal Processing Using Ueural Networks: Prediction and System Modelling. In D. Z. Anderson (Ed.), *Neural Information Processing Systems, American Institute of Physics* (p. 442-456).

LeCun, Y., Simard, P. Y., & Pearlmutter, B. A. (1993). Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. In *Advances in Neural Information Processing Systems 5* (p. 156-163). Morgan Kaufmann.

Lippmann, R. P. (1988). An Introduction to Computing with Neural Nets. *SIGARCH Comput. Archit. News*, *16*(1), 7-25.

Nadaraya, E. (1964). On Estimating Regression. *Theory of Probability and its Applications*, *9*(1), 141-142.

Reinsch, C. H. (1967). Smoothing by Spline Functions. *J. Nun. Math*, *10*(3), 177-183.

Ripley, B. D. (1993). Statistical Aspects of Neural Networks. *Networks and Chaos - Statistical and Probabilistic Aspects.*, 40-123.

Rosenblatte, F. (1958). The Perceptron: a Probabilistied Model for Information Storage and Organization in the Brain. *Psychological Review*, *65*, 386C408.

Roy, S. (1993). Near-optimal dynamic learning rate for training backpropagation neural networks. In *Proc. SPIE Vol. 1966, p. 277-283, Science of Artificial Neural Networks II.*

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature*, *323*(6088).

Ruppert, D. (2002). Selecting the Number of Knots for Penalized Splines. *Journal of Computational & Graphical Statistics*, *11*(4), 735-757.

Ruppert, D., Sheather, S. J., & Wand, M. P. (1995). An Effective Bandwidth Selector for Local Least Squares Regression. *Journal of the American Statistical Association*, *90*(432), 1257-1270.

Ruppert, D., & Wand, M. P. (1994). Multivariate Locally Weighted Least Squares Regression. *The Annals of Statistics*, *22*(3), 1346-1370.

Silverman, B. W. (1984). Spline Smoothing: The Equivalent Variable Kernel Method. *12*(3), 898-916.

Silverman, B. W. (1985). Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, *47*(1), 1-52.

Stern, H. S. (1996). Neural Networks in Applied Statistics. *Technometrics*, *38*(3), 205-214.

Stone, M. (1974). Cross-Validatory Choice and Assessment of Statistical Predictions. *36*, 111-147.

Wagner, H. M. (1959). Linear Programming Techniques for Regression Analysis. *Journal of the American Statistical Association*.

Wahba, G. (1990). *Spline Models for Observational Data*. Philladelphia: SIAM.

Warner, B., & Misra, M. (1996). Understanding Neural Networks as Statistical Tools. *The American Statistician*, *50*(4), 284-293.

Watson, G. S. (1964). Smooth Regression Analysis. *Sankhya*, *Series A 26*, 359-372.

White, H. (1989a). Learning in Artificial Neural Networks: A Statistical Perspective. *Neural Computation*, *1*(4), 425-464.

White, H. (1989b). Some Asymptotic Results for Learning in Single Hidden-Layer Feedforward Network Models. *Journal of the American Statistical Association*, *84*(408), 1003-1013.

Widrow, B., Rumehart, D. E., & Lehr, M. A. (1994). Neural Networks : Applica-

tions in Industry, Business and Science. *Commun. ACM*, *37*(3), 93-105.

Yu, K., & Jones, M. (1997). A Comparison of Local constant and Local Linear Regression Quantile Estimators. *Computational Statistics & Data Analysis*, *25*(2), 159-166.

Yu, K., & Jones, M. (1998). Local linear quantile regression. *Journal of the American Statistical Association*, *93*(441), 228-237.

Yu, X., Chen, G. A., & Cheng, S. X. (1995). Dynamic Learning Rate Optimization of the Backpropagation Algorithm. *IEEE Transactions on Neural Networks*, *6*(3), 669-677.