Statistics and Its Interface Volume 3 (2010) 437–444

# Robust neural network with applications to credit portfolio data analysis

Yijia Feng*, Runze Li†, Agus Sudjianto and Yiyun Zhang‡

In this article, we study nonparametric conditional quantile estimation via neural network structure. We proposed an estimation method that combines quantile regression and neural network (robust neural network, RNN). It provides good smoothing performance in the presence of outliers and can be used to construct prediction bands. A Majorization-Minimization (MM) algorithm was developed for optimization. Monte Carlo simulation study is conducted to assess the performance of RNN. Comparison with other nonparametric regression methods (e.g., local linear regression and regression splines) in real data application demonstrate the advantage of the newly proposed procedure.

AMS 2000 subject classifications: Primary 62G35, 62P99.
Keywords and phrases: Conditional quantile, Nonparametric regression.

## 1. INTRODUCTION

Nonparametric regression models are useful in exploring the fine features in the data. Many methods were proposed in the literature, including local polynomial (Fan, 1992, 1993), splines smoothing (Fan and Gijbels, 1996; Stone et al., 1997), neural network (Ripley, 1993; Haykin, 1994; Bishop, 1995) and etc. Most of these existing works focus on the estimation of the mean regression function originally. While the mean function characterizes the important central information of the response, in many situations, conditional quantiles may reveal a more comprehensive view of the distribution, especially when it is non-normal and/or in the presence of outliers. Quantile regression (Koenker and Bassett, 1978) may be viewed as an extension from estimation of conditional mean under least squared loss to conditional quantiles under general quantile loss. Later, Koenker et al. (1994) proposed an estimation for nonparametric quantile regression function using splines, and Yu and Jones (1998) developed an estimation for nonparametric quantile regression function using local linear techniques.

In this article, we propose to use a neural network method to estimate the nonparametric quantile regression function. The newly proposed method is motivated by an empirical analysis of credit card portfolio data. Figure 1 depicts the scatter plots of credit card portfolio data for two typical segments. The vertical axis stands for the charge-off (i.e., customers defaulted their loan) rate, and horizontal axis is the age of the loan since the time of origination (in months), which is often called *months on book*. Since the regression curve shows how the charge-off rate changes over the age of the loan, it is also called *maturation curve*.

Both local linear regression with a selected optimal bandwidth and cubic regression splines are employed to estimate maturation curve. From Figure 1 it can be seen that they are wiggling over the month on book. This is an undesired feature because it is believed from our business experience that the maturation curve is stable after a ceratin number of months on book. This motivates us to seek other nonparametric estimation procedure for estimation of the maturation curve which naturally provides desirable smoothness characteristics. Thus, we further use a neural network to estimate the maturation curve. The estimated curves are also displayed in Figure 1. Unlike the estimates of local linear and cubic regression splines, the neural network estimate is smooth and does not wiggle around.

In addition, the scatter plots clearly show that there are some outliers in both data sets, and the second set seems to contain more outliers than the first set. This leads us to further develop nonparametric quantile regression with a neural network, which we call robust neural network (RNN) which will be discussed in detail in later sections. With the aid of RNN, we can also obtain prediction intervals based on the estimated quantile functions.

Computation is the major challenge in the implementation of neural network quantile regression because the objective function is a nonlinear and nonconvex function of high dimensional parameters. We develop an algorithm to estimate the nonparametric quantile regression function using the Majorization-Minimization (MM) algorithm (Hunter and Lange, 2000).

*Feng's research was supported by a National Institute on Drug Abuse (NIDA) grant R21 DA024260 as a graduate research assistant. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIDA or the NIH.
†Runze Li is the corresponding author. Li's research was supported by a National Science Foundation grant DMS 0348869.
‡Zhang's research was supported by a National Institute on Drug Abuse grant P50 DA10075 when he was a graduate research assistant in Penn State University. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIDA or the NIH.

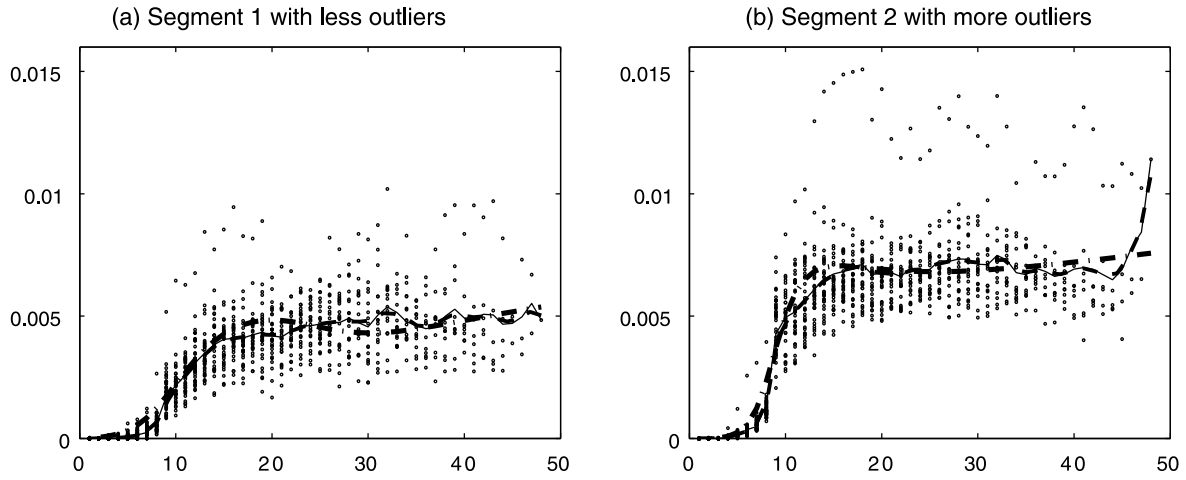


*Figure 1. Comparison of three nonparametric methods under least squared loss. Dots are the scatter plot of data. Solid, dashed and dotted-dashed curves are the estimated mean functions by cubic splines, local linear regression and neural network, respectively. Note that solid curve and dashed curve are almost identical.*

This paper is organized as follows. Section 2 outlines the RNN methodology. The MM algorithm for RNN is proposed along with technical details of its implementation. In Section 3, Monte Carlo simulation studies are conducted to assess the performance of the proposed estimation procedure and algorithm. Thereafter, we apply RNN to analyze credit card portfolio data. Conclusion remarks and some discussions are given in Section 4.

## 2. ROBUST NEURAL NETWORKS

### 2.1 Some preliminary

Suppose that $(x_1, y_1), \ldots, (x_n, y_n)$ is a random sample from

$$y = m(x) + \epsilon, \tag{1}$$

where $m(\cdot)$ is an unknown smoothing function to be estimated, and $\epsilon$ is a random error with mean zero. We will employ a neural network model known as multi-layer perceptron (MLP) network to estimate the $m(x)$. In particular, we will use a 3-layer perceptron neural network where it consists of three components as depicted in Figure 2:

- *An input layer* – used to input information from the explanatory variable $\{x_i\}_{i=1}^n$;
- *A hidden layer* – used to process the summation from an input layer via some nonlinear transformation and each layer contains certain amount of neurons;
- *An output layer* – used to output the estimation of response variable $\{y_i\}_{i=1}^n$.

Let $\nu_{ij}$ be the weighted summation of each input $x_i$

$$\nu_{ij} = x_i\omega_j + \omega_{0j}, \quad j = 1, \ldots, M. \tag{2}$$

where $\omega_j$ is the connection weight between input and $j^{\text{th}}$ neuron while $\omega_{0j}$ is a bias adjusted at the $j^{\text{th}}$ neuron. Furthermore, $g(\cdot)$ is an activation function, which is typically chosen to be a logistic-type function:

$$g(\nu_{ij}) = \frac{1}{1+\exp(-\nu_{ij})} \qquad \text{or} \qquad g(\nu_{ij}) = \tanh(\nu_{ij}).$$

The 3-layer perceptron neural network with $M$ neurons approximate $m(x_i)$ by

$$m(x_i) \approx \hat{m}(x_i; \boldsymbol{\beta}, \boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_M) = \beta_0 + \sum_{j=1}^{M} \beta_j g(\nu_{ij}), \tag{3}$$

where $\beta_0$ is a bias adjusted for the output, and $\beta_j$ is the weight connecting $j^{\text{th}}$ neuron and the output.

From (2) and (3), it can be seen in the 3-layer network that the role of a hidden layer is to expand the input space into sigmoidal basis functions where the output layer calculates the superposition of the constructed basis functions. The parameters of the network (i.e. *weights*) are estimated by optimizing an objective function (e.g., via minimizing least square error).

### 2.2 Robust neural network via MM algorithm

Define the quantile loss function for a given quantile $q \in (0, 1)$ as

$$\rho_q(r) = |r| + (2q-1)r. \tag{4}$$

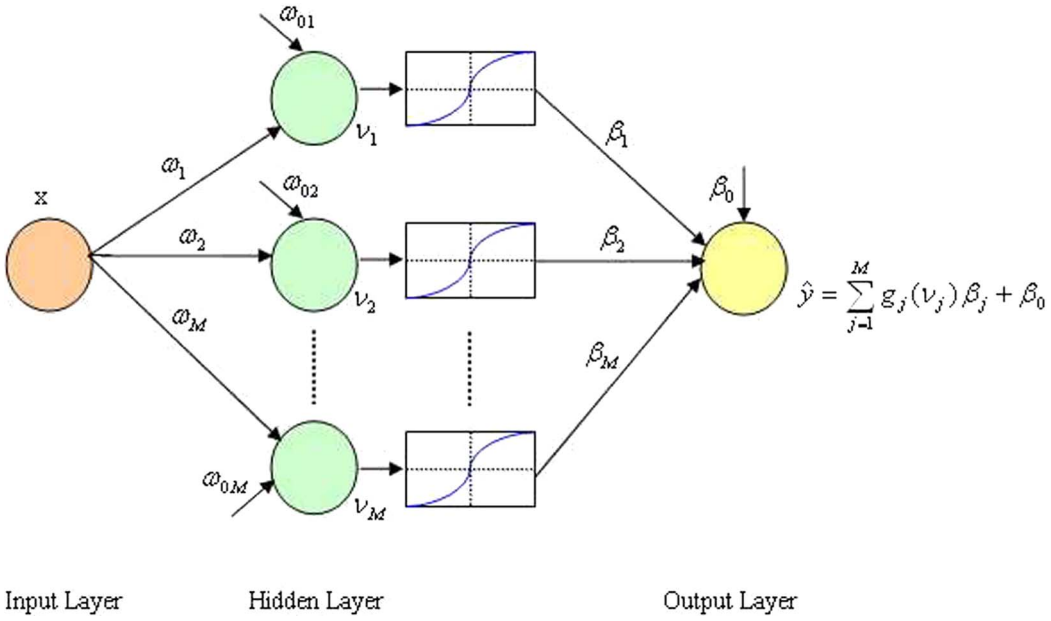


*Figure 2. Three-layer perceptron.*

The RNN is obtained via minimizing the quantile loss

$$(5)\qquad E = E(\boldsymbol{\beta}, \boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_M) = \sum_{i=1}^{n} \rho_q \left[ y_i - \left\{ \sum_{j=1}^{M} \beta_j g(x_i \omega_j + \omega_{0j}) + \beta_0 \right\} \right],$$

with respect to weights $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_M)^T$ and $\boldsymbol{\omega}_j = (\omega_{0j}, \omega_j)$, $j = 1, \ldots, M$. Let $(\hat{\boldsymbol{\beta}}_q, \hat{\boldsymbol{\omega}}_{q1}, \ldots, \hat{\boldsymbol{\omega}}_{qM})$ be the minimizer of equation (5). The resulting $\hat{m}_q(x) = \hat{m}(x_i; \hat{\boldsymbol{\beta}}_q, \hat{\boldsymbol{\omega}}_{q1}, \ldots, \hat{\boldsymbol{\omega}}_{qM})$ estimates the $q^{\text{th}}$ conditional quantile at point $x$. In particular, when $q = 1/2$, the RNN becomes the least absolute deviation regression for the neural network, a robust estimate for the mean and median function when the error $\epsilon$ has a symmetric distribution about the origin.

The major challenge in the implementation of RNN is computation because the objective function $E$ is a high-dimensional, nonconvex, and nonlinear function of the parameters $\boldsymbol{\beta}$ and $\boldsymbol{\omega}_j$. Here we propose an algorithm for the neural network quantile regression adopted from a Majorization-Minimization (MM) algorithm (Hunter and Lange, 2000) for optimization. It creates a surrogate function that majorizes the objective function. Then, optimizing the objective function is driven by optimizing the surrogate function instead. Denote the residual of $y_i$ at the $k^{\text{th}}$ step iteration by

$$r_{ki} = y_i - \left\{ \beta_0^{(k)} + \sum_{j=1}^{M} \beta_j^{(k)} g\big(x_i \omega_j^{(k)} + \omega_{0j}^{(k)}\big) \right\}.$$

In the "majorization" step, we use a quadratic function, namely $\rho_{q,m}(r)$, to approximate the quantile loss function $\rho_q(r)$,

$$(6)\qquad \rho_{q,m}(r) = \frac{r^2}{2|r_{ki}|} + \frac{|r_{ki}|}{2} + (2q-1)r.$$

By the Cauchy-Schwarz inequality, it follows that $\rho_{q,m}(r) \geq \rho_q(r)$ and $\rho_{q,m}(r_{ki}) = \rho_q(r_{ki})$. As a result, $\rho_{q,m}(r)$ majorizes $\rho_q(r)$, and therefore,

$$E_{k,m} = \sum_{i=1}^{n} \rho_{q,m} \left[ y_i - \left\{ \sum_{j=1}^{M} \beta_j g(x_i \omega_j + \omega_{0j}) + \beta_0 \right\} \right]$$

majorizes $E$ in (5) at the $k^{\text{th}}$ step. In practice, when $r_{ki}$ is very close to zero, then the $i^{\text{th}}$ data point will receive too much weight. As advocated by Fan and Li (2001), we consider a perturbed-version of majorization in which we replace $\rho_{q,m}(r)$ in (6) by $\rho_{q,m}(r, a_n)$ as

$$(7)\qquad \rho_{q,m}(r, a_n) = \frac{r^2}{2|r_{ki}| + a_n} + \frac{|r_{ki}| + 0.5a_n}{2} + (2q-1)r,$$

with $a_n$ being the $(2/\sqrt{n})$-th sample percentile of $|r_{k1}|, \ldots, |r_{kn}|$.

In the "minimization" step, back propagation algorithm (Rumelhart et al., 1986) is applied for the perturbed majorization function

$$E_{k,m} = \sum_{i=1}^{n} \rho_{q,m}(r_{ki}, a_n),$$

at the $k^{\text{th}}$ step. Back propagation algorithm propagates information forward once it studies predictor $x$ given some initial weights. After calculating the errors between observed and predicted $y$'s, it goes back to adjust the weights and starts the next round of study. In this way, 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 inversion of a large dimensional matrix. The procedure of back propagation in neural network quantile regression can be summarized as an algorithm below.

Step 1. Set initial value of $\boldsymbol{\beta}$ and $\boldsymbol{\omega}_j$ to be $\boldsymbol{\beta}_j^{(1)}$ and $\boldsymbol{\omega}_j^{(1)}$, and set a learning rate. In our numerical study, constant learning rate $\eta = 0.01$ is used.

Step 2. Let $k = 1$, and set $\boldsymbol{\beta}^{(k,1)} = \boldsymbol{\beta}^{(k)}$ and $\boldsymbol{\omega}_j^{(k,1)} = \boldsymbol{\omega}_j^{(k)}$.

Step 2a. For $i^{\text{th}}$ observation $(x_i, y_i)$, define $E_i = \rho_{q,m}(r_{ki}, a_n)$. Calculate the gradient $\triangle_i\boldsymbol{\beta}$ and $\triangle_i\boldsymbol{\omega}$: for $j = 1, \ldots, M$,

$$\triangle_i\beta_0 = -\eta\frac{\partial E_i}{\partial\beta_0}, \quad \triangle_i\beta_j = -\eta\frac{\partial E_i}{\partial\beta_j}, \tag{8}$$

and

$$\triangle_i\omega_{0j} = -\eta\frac{\partial E_i}{\partial\omega_{0j}}, \quad \triangle_i\omega_j = -\eta\frac{\partial E_i}{\partial\omega_j}. \tag{9}$$

Step 2b. Update weights

$$\begin{aligned}
\beta_0^{(k,i+1)} &= \beta_0^{(k,i)} + \triangle_i\beta_0,\\
\beta_j^{(k,i+1)} &= \beta_j^{(k,i)} + \triangle_i\beta_j,\\
\omega_{0j}^{(k,i+1)} &= \omega_{0j}^{(k,i)} + \triangle_i\omega_{0j},\\
\omega_j^{(k,i+1)} &= \omega_j^{(k,i)} + \triangle_i\omega_j,
\end{aligned} \tag{10}$$

where $\triangle_i\beta_0$, $\triangle_i\beta_j$, $\triangle_i\omega_{0j}$ and $\triangle_i\omega_j$ $(j = 1, \ldots, M)$ are from (8) and (9).

Step 2c. Repeat Steps 2a and 2b for $i = 1, \ldots, n$, and update the weights

$$\begin{aligned}
\beta_0^{(k+1)} &= \beta_0^{(k,n+1)}, \quad \beta_j^{(k+1)} = \beta_j^{(k,n+1)},\\
\omega_{0j}^{(k+1)} &= \omega_{0j}^{(k,n+1)}, \quad \omega_j^{(k+1)} = \omega_j^{(k,n+1)},
\end{aligned} \tag{11}$$

Step 3. Let $k = 2, 3, \ldots$, and repeat Step 2 until convergent criterion meets.

**Remark.** The $\rho_{q,m}(r, a_n)$ does not majorize $\rho_q(r)$. Thus, the descent property of MM algorithm may not be valid here. It can be shown that

$$\begin{aligned}
&\frac{1}{n}E(\boldsymbol{\beta}^{(k+1)}, \boldsymbol{\omega}_1^{(k+1)}, \ldots, \boldsymbol{\omega}_M^{(k+1)})\\
&\quad \le \frac{1}{n}E(\boldsymbol{\beta}^{(k)}, \boldsymbol{\omega}_1^{(k)}, \ldots, \boldsymbol{\omega}_M^{(k)}) + \frac{a_n}{4}.
\end{aligned}$$

if $\{\boldsymbol{\beta}^{(k+1)}, \boldsymbol{\omega}_1^{(k+1)}, \ldots, \boldsymbol{\omega}_M^{(k+1)}\}$ is the minimizer of $E_{k,m}$, and $\{\boldsymbol{\beta}^{(k)}, \boldsymbol{\omega}_1^{(k)}, \ldots, \boldsymbol{\omega}_M^{(k)}\}$is the minimizer of $E_{k-1,m}$. Note that $a_n \to 0$, the descent property of MM algorithm may be approximately valid when $n$ is large enough.

### 2.3 Some practical issues

Some practical issues of network training such as the selections of initial value, learning rate, network structure and training algorithm can be found in Haykin (1994); Hassoun (1995); Bishop (1995). Here we provide a brief summary for these issues.

Our experience indicates that a small learning rate is more successful in searching for the optimum parameters. Using a small learning rate prevents taking large steps in the gradient direction although it means longer computational time. In our numerical study, constant learning rate $\eta = 0.01$ is used. There are proposals for dynamically adjusting the learning rate to improve convergence rate; see, for example, (Darken et al., 1992; LeCun et al., 1993; Roy, 1993).

Since the objective function in the network learning is nonlinear and nonconvex, we have to deal with issues of local minima and stationary points. Several trials of different initial values are beneficial to avoid local minima. We set our initial values randomly between 0 and 1.

Numerical stability may benefit from standardization of variables. In our numerical study, we standardize both independent and dependent variables. The number of hidden neurons is important because too many or too few hidden neurons may result in over-fitted or under-fitted models respectively. Therefore, it is necessary to choose a proper number of neurons in every case study. This can be evaluated by monitoring prediction errors using cross validation, for example. In our credit card case study, the data and business experience suggest that there is a period of increasing, decreasing, and relatively constant rate of charge-off. Thus, we select two or three hidden units to sufficiently capture the dynamics of consumer default rate.

## 3. NUMERICAL STUDY AND APPLICATIONS

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

### 3.1 Simulation study

It is well known that a neural network is a good smoother in general, however, its performance can be sensitive to the presence of extreme observations in the training data set, similar to other regression methods that utilize squared loss

as objective functions. RNN alleviates the aforementioned problem of sensitivity to outliers.

We now conduct a Monte Carlo simulation study to assess the performance of the proposed procedure. In our simulation, we generate 1000 data sets, each having the sample size $n = 200$, from

$$y_i = m(x_i) + \epsilon_i, \quad i = 1, \ldots, n, \tag{12}$$

where $x_i$ is generated from uniform distribution over [0,1], and

$$m(x) = e^{-x} \left\{ \sin(2\pi x + 1) + 2 \right\}. \tag{13}$$

In our simulation, we consider three scenarios for random error $\epsilon$'s:

- Normal error, $\epsilon \sim N(0, 0.2^2)$,
- Mixtures of normals error, $0.9N(0, 0.2^2) + 0.1N(0, 1)$,
- Laplace Distribution with $\mu = 0$ and $\sigma = 0.2$.

In our simulation, we use 3-Layer MLP network with 5 neurons in the hidden layer and $g(x) = \tanh(x)$ as the activation function. A typical fitted curve selected from 1,000 simulations is depicted in Figure 3, from which it is shown that the fitted curve is very close to the true curve. The proposed algorithm and estimation procedure performed very well in all three scenarios for error distribution.

The true quantiles and their estimates are summarized in Table 1. The standard deviations of conditional quantile estimations are provided in parentheses. The true $q^{\text{th}}$ conditional quantile evaluated at $x$ is calculated by

$$F^{-1}_{Y|X=x}(q) = F_e^{-1}(q) + m(x) \tag{14}$$

where $F^{-1}_{Y|X=x}$, and $F_e^{-1}$ is the quantile functions of corresponding conditional distribution $Y|X = x$ and the error distributions, respectively. Due to the difficulty of obtaining the quantile function of a mixture of normal distributions, we calculate the estimate of true quantile values via Monte Carlo method instead.

The column labeled "Est(SE)" in Table 1 is the sample average and the standard error of 1,000 estimates over 1,000 simulations. To gauge the performance of the proposed estimation procedures in a specific point of $x$, we select three representative $x$-values: $x = 0.3, 0.6, 0.9$. Note that $x = 0.3$ lies in the interval where $m(x)$ sharply decreases; $x = 0.6$ is in the neighborhood of a local minimum of $m(x)$ ; $x = 0.9$ is located in the interval where the curve is relatively flat. At each grid point, we compare the estimate with the true value at 5 different quantiles with $q = 0.05, 0.25, 0.50, 0.75, 0.95$.

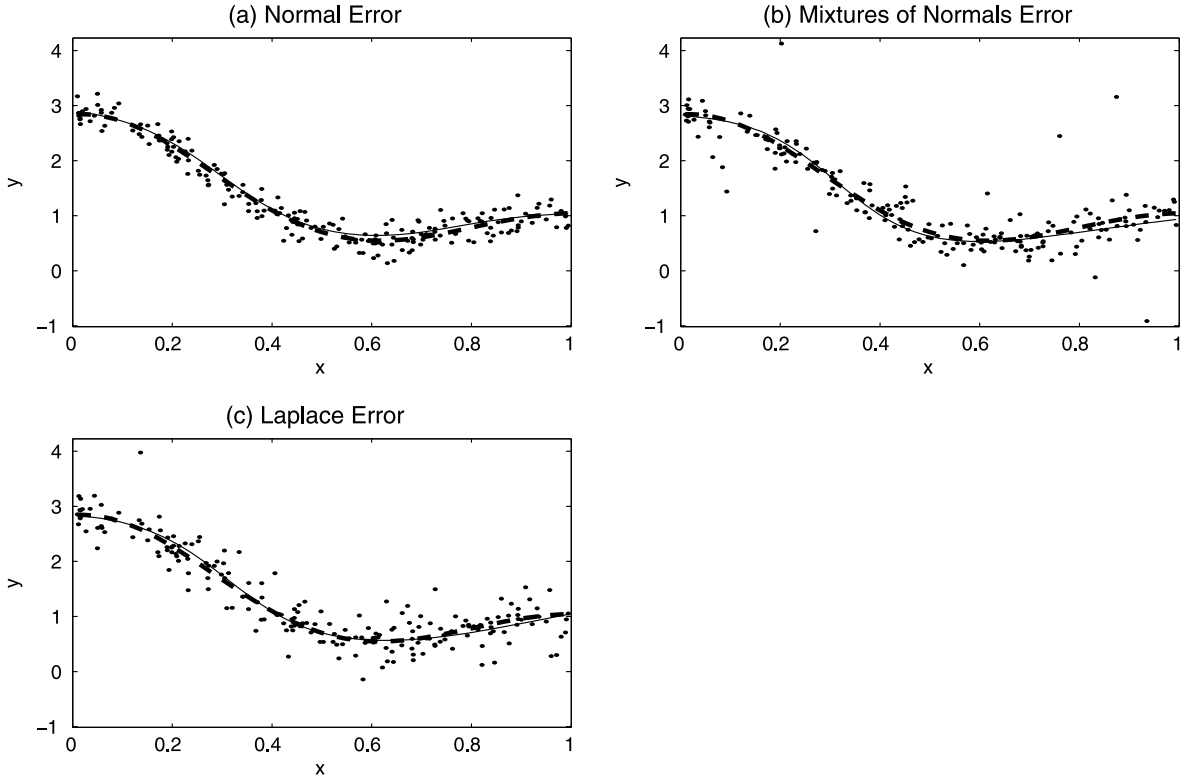


*Figure 3. A typical median fit under different error distributions. Dots are scatter plot of simulated data, thick dashed line is the true median function, and solid line is the estimated median using five neurons.*

*Table 1. Simulation results*

| | $x = 0.3$ | | $x = 0.6$ | | $x = 0.9$ | |
|---|---|---|---|---|---|---|
| quantile | True | Est(SE) | True | Est. (SE) | True | Est.(SE) |
| Normal error - $N(0, 0.2^2)$ | | | | | | |
| 5% | 1.2512 | 1.2261 (0.0809) | 0.2207 | 0.3516 (0.0695) | 0.6318 | 0.4290 (0.0908) |
| 25% | 1.5348 | 1.4668 (0.0741) | 0.4148 | 0.3733 (0.0544) | 0.8259 | 0.7322 (0.0766) |
| 50% | 1.6697 | 1.7126 (0.0885) | 0.5497 | 0.5682 (0.0630) | 0.9608 | 0.9420 (0.0819) |
| 75% | 1.8046 | 1.9052 (0.742) | 0.6846 | 0.7464 (0.0625) | 1.0957 | 1.1728 (0.0858) |
| 95% | 1.9986 | 2.0581 (0.962) | 0.6846 | 0.9186 (0.0715) | 1.2898 | 1.2963 (0.0761) |
| Mixture of normals error - $0.9N(0, 0.2^2) + 0.1N(0, 1^2)$ | | | | | | |
| 5% | 1.2520 | 1.1124 (0.1452) | 0.1324 | 0.1318 (0.0932) | 0.5459 | 0.3750 (0.1310) |
| 25% | 1.5201 | 1.5546 (0.0754) | 0.4014 | 0.4054 (0.0516) | 0.8146 | 0.7891 (0.0670) |
| 50% | 1.6707 | 1.7432 (0.0786) | 0.5503 | 0.5928 (0.0431) | 0.9643 | 0.9406 (0.0564) |
| 75% | 1.8174 | 1.9111 (0.0784) | 0.6977 | 0.7554 (0.0604) | 1.1105 | 1.1241 (0.0714) |
| 95% | 2.0902 | 2.2263 (0.1590) | 0.9709 | 1.2162 (0.1190) | 1.3843 | 1.4386 (0.1257) |
| Laplace error - Laplace Distribution with $\mu = 0$ and $\sigma = 0.2$ | | | | | | |
| 5% | 1.2092 | 1.2482 (0.1239) | 0.0892 | 0.0316 (0.1508) | 0.5003 | 0.4747 (0.1266) |
| 25% | 1.5310 | 1.4021 (0.1007) | 0.4111 | 0.5315 (0.0687) | 0.8222 | 0.8158 (0.0891) |
| 50% | 1.6697 | 1.7222 (0.0536) | 0.5497 | 0.5670 (0.0420) | 0.9608 | 0.9590 (0.0619) |
| 75% | 1.8083 | 1.8575 (0.0627) | 0.6883 | 0.7282 (0.0534) | 1.0994 | 1.1131 (0.0697) |
| 95% | 2.1302 | 2.1958 (0.1583) | 1.0102 | 1.0145 (0.1190) | 1.4213 | 1.4578 (0.1378) |

The estimates for a median are the best among all quantiles regardless of the error distributions. This is expected since the estimate for median has the smallest asymptotic variance among the quantile regression with $q \in (0, 1)$. As a quantile goes more extreme toward 0 or 1, the quality of the estimates are degrading though they still include the true value within 2 standard errors in most cases. Comparing the performance at different grid values, the estimate at $x = 0.6$ is not as good as those at $x = 0.3$ and $x = 0.9$. Also it is notable that, for heavy tail distributions (e.g., mixtures of normals and Laplace), when the quantiles are more extreme, the estimated quantiles have larger standard errors than that in the normal error case.

## 3.2 Credit card portfolio data analysis

In this section, we apply the proposed methodology to model the performance of credit card portfolio which consists of hundreds of credit card loans. For an illustrative purpose, we selected data from two segments. Each segment consists of pools of credit card loans from monthly origination. That is, each pool within a segment is a collection of credit cards that are issued in a particular month. We tracked the monthly performance of each pool in terms of default or charge off rates from the time of issuance to date. The charge off rate is defined as the number of charge off accounts given the number of active (i.e., at risk) accounts at any given month. The oldest vintage in this analysis has 48 months on book. Subsequently, newer monthly pools will have shorter monthly performance data. The purpose of the analysis is to model the *maturation* characteristic of each segment. That is, to model the charge off rate as the credit card pool aged in terms of number of *months on book* (MoB). Let $x$ be the MoB and $y$ be the charge-off rate, respectively, and consider the following nonparametric model

$$y = m(x) + \epsilon.$$

The regression function, $m(x)$, is called the *maturation* curve because it represents the maturation characteristic of the pool of loans as it aged. Typically, one would like to represent the maturation effect $m(x)$ as a smooth function over $x$ so that a reasonable *extrapolation* can be made to project future performance.

There are several practical nonparametric methods for function approximation that can be used to fit maturation curve, such as kernel regression, local polynomial, spline regression, etc. For this particular application, the neural network (i.e., sigmoidal basis functions) are desirable because it naturally provides a well regularized smooth representation of a maturation curve while other nonparametric methods are lacking the desired smooth representation over month on book. For an illustrative purpose, we choose a local linear and cubic spline as competitors to the neural network approach. A 3-Layer MLP with 3 hidden neurons is employed in this study. Figure 1 gives the graphical results of data fitting for all three methods under least squared loss. Figure 1(a) has less potential outliers in the data while the other set (i.e., segment) in Figure 1(b) contains more outliers. As opposed to smooth behavior of a neural network, local linear (in dashed line) and cubic spline (in solid line) both have undesirable wiggly curve fit. In the presence of outliers, however, all these three methods have a drawback because of their lack of robustness of least squares estimate. Note the fit at large month on book where the data is sparser, the estimations are highly influenced by outliers.

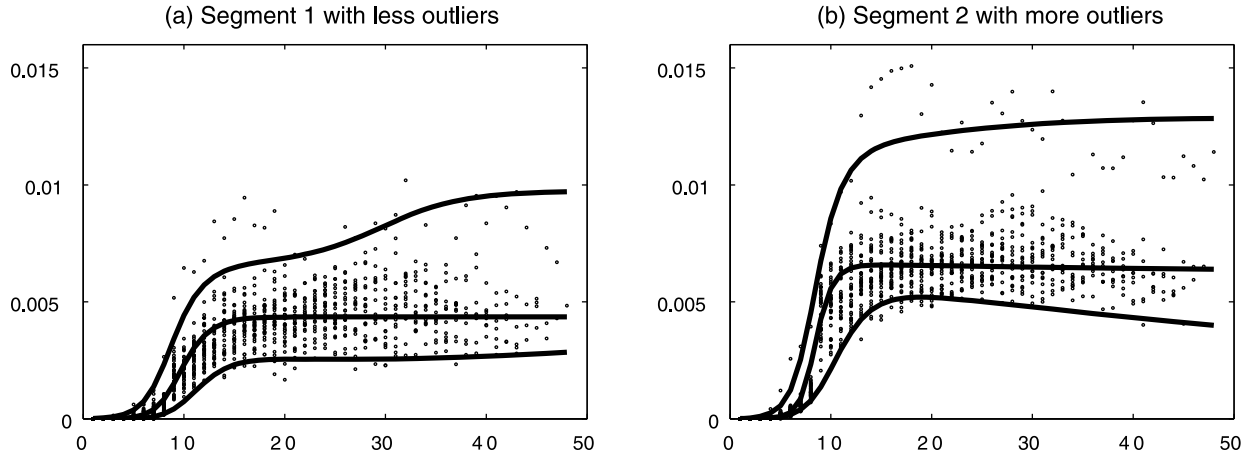


*Figure 4. Robust neural network estimation under general Quantile Loss. Dots are the scatter plot of data. The curves from bottom to top are* $5\%$, $50\%$ *and* $95\%$ *quantile estimates respectively.*

In particular, the case for more outliers shown in Figure 1(b), local linear and cubic spline estimation are significantly more impacted by the outliers. Hence, we go further and alter the setting from the least squared loss to the quantile loss.

To alleviate the robustness issue of least square objective function, we applied the quantile learning objective function. Figure 4 shows the estimation results for both data sets under a general quantile loss. The bottom, middle, and top curves represents 5%, 50% and 95% quantile estimates, respectively. Notice that the fit at a larger MoB are no longer affected by the outliers. Thus, the robust neural network estimates are rather robust to the outliers. Additionally, the fitted curve is desirable of parametric-like smoothness of maturation characteristics. In particular, the upper quantile (e.g., 90% quantile) estimate at given MoBs can be used as the *downside* risk estimate of a charge-off rate of a segment.

## 4. DISCUSSION

In this paper we proposed the robust neural network for conditional quantile function estimation. We develop an algorithm for training RNN based on the MM algorithm. We empirically test the proposed algorithm by simulation studies, and illustrate the proposed procedure by an analysis of credit portfolio data. In our real data analysis, we compare the performance of local linear regression, regression splines and the neural network estimates. The comparison indicates that the robust neural network approach provides desirable characteristics of both smooth function approximation as well as its robustness to outliers. Additionally, by applying quantile loss estimate, the approach produces interval estimates of the approximation curve. Its ability to provide the estimation of a potential downside risk of charge-off rate upper quantile curve is very useful for risk management purpose.

*Received 18 September 2009*

## REFERENCES


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

Darken, C., Chang, J. and Moody, J. Learning Rate Schedules For Faster Stochastic Gradient Search. In *Neural Networks for Signal Processing 2 — Proceedings of the 1992 IEEE Workshop*, Piscataway, NJ, USA, 1992. IEEE Press.

Fan, J. Design-adaptive nonparametric regression. *Journal of the American Statistical Association*, 87(420):998–1004, 1992.

Fan, J. Local linear regression smoothers and their minimax efficiencies. *The Annals of Statistics*, 21(1):196–216, 1993.

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

Fan, J. and Li, R. Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001.

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

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

Hunter, D. and Lange, K. Quantile Regression via MM Algorithm. *Journal of Computational and Graphical Statistics*, 9:60–70, 2000.

Koenker, R. and Bassett, G. J. Regression Quantiles. *Econometrica*, 46(1):33–50, 1978.

Koenker, R., Pin, N. and Portnoy, S. Quantile smoothing splines. *Biometrika*, 81(4):673–680, 1994.

LeCun, Y., Simard, P. Y. and Pearlmutter, B. A. Automatic Learning Rate Maximization by On-line Estimation of the Hessian's Eigenvectors. In *Advances in Neural Information Processing Systems 5*, pages 156–163. Morgan Kaufmann, 1993.

Ripley, B. D. Statistical Aspects of Neural Networks. *Networks and Chaos - Statistical and Probabilistic Aspects*, pages eds. O. Barndorff–Nielsen, J. Jensen, and W. Kendall, London: Chapman & Hall, pp. 40–123, 1993.

Roy, S. Near-optimal Dynamic Learning Rate for Training Backpropagation Neural Networks. In *Proc. SPIE Vol. 1966, p. 277–283, Science of Artificial Neural Networks II*, 1993.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. Learning Representations by Back-propagating Errors. *Nature*, 323(6088): 533–536, 1986.

Stone, C. J., Hansen, M. H., Kooperberg, C. and Truong, Y. K. Polynomial Splines and Their Tensor Products in Extended Linear Modeling. *The Annals of Statistics*, 25(4):1371–1470, 1997.

Yu, K. and Jones, M. C. Local Linear Quantile Regression. *Journal of the American Statistical Association*, 93(441):228–237, 1998.

Yijia Feng
Department of Statistics
The Pennsylvania State University
University Park, PA 16802, USA
E-mail address: yijia@psu.edu

Runze Li
Department of Statistics and the Methodology Center
The Pennsylvania State University
University Park, PA 16802, USA
E-mail address: rli@stat.psu.edu

Agus Sudjianto
Bank of America
Charlotte, NC 28255, USA
E-mail address: Agus.Sudjianto@bankofamerica.com

Yiyun Zhang
Novartis Oncology
OGD BDM Gbl Bios Full Dev
Florham Park, NJ 07932, USA
E-mail address: yiyun.zhang@novartis.com