Deep Learning for Quantile Regression: DeepQuantreg

A PREPRINT

Yichen Jia
Department of Biostatistics
Graduate School of Public Health
University of Pittsburgh, Pittsrbugh, USA

Jong-Hyeon Jeong *
Department of Biostatistics
Graduate School of Public Health
University of Pittsburgh, Pittsburgh, USA

July 15, 2020

ABSTRACT

The computational prediction algorithm of neural network, or deep learning, has drawn much attention recently in statistics as well as in image recognition and natural language processing. Particularly in statistical application for censored survival data, the loss function used for optimization has been mainly based on the partial likelihood from Cox's model and its variations to utilize existing neural network library such as Keras, which was built upon the open source library of TensorFlow. This paper presents a novel application of the neural network to the quantile regression for survival data with right censoring, which is adjusted by the inverse of the estimated censoring distribution in the check function. The main purpose of this work is to show that the deep learning method could be flexible enough to predict nonlinear patterns more accurately compared to the traditional method even in low-dimensional data, emphasizing on practicality of the method for censored survival data. Simulation studies were performed to generate nonlinear censored survival data and compare the deep learning method with the traditional quantile regression method in terms of prediction accuracy. The proposed method is illustrated with a publicly available breast cancer data set with gene signatures. The source code is freely available at `https://github.com/yicjia/DeepQuantreg`.

Keywords: Huber Check Function; Inverse Probability Censoring Weights (IPCW); Neural Network; Right Censoring; Survival Analysis; Time to Event

1 Introduction

Deep learning algorithms have been developed to perform classification and prediction in many application areas such as image recognition and natural language processing. Recently deep learning has gained much popularity in survival analysis because of its flexible model design and ability of capturing nonlinear relationships. The main challenge of applying deep neural network models to time-to-event data is the presence of censoring. To overcome this challenge, many deep learning models propose to use the Cox model-based, or the partial likelihood-based (Cox, 1975), loss function for predicting patient survival (Faraggi and Simon, 1995; Katzman et al., 2018; Ching et al., 2018). Faraggi and Simon (1995) replaced the linear function in the partial likelihood with a nonlinear functional form of the output in the neural network algorithm, extending the Cox's model to a nonproportional hazards modeling.

To analyze survival data, the quantile could be a preferred summary measure due to commonly encountered skewness and outliers in the observed data. Without any covariates, or predictors, for a positive random variable T the quantile is generally defined as

$$Q_T(\tau) = \inf\{t : \Pr(T \leq t) \geq \tau\}, \tau \in (0, 1).$$

In practice, however, investigators would be more interested in associations between the quantiles of time-to-event distributions and potential predictors in a regression setting. Quantile regression, originally proposed by Koenker and Bassett Jr (1978), is a popular alternative to the least-square approach in the linear regression. It also has been an attractive alternative to the Cox proportional model (Cox, 1972) for time-to-event data, and many methods have been

*Corresponding Author: jjeong@pitt.edu

established to deal with the censoring problem under the quantile regression (Ying et al., 1995; McKeague et al., 2001; Peng and Huang, 2008). Cannon (2011) has developed an `R` package `QRNN`, which implements the quantile regression neural network for continuous response variable, i.e. precipitation amounts truncated at zero, but it cannot be used to incorporate the random censorship commonly encountered in survival data. To our best knowledge, however, there is no literature on deep learning method for the quantile regression on right-censored survival data. In this paper, we present a novel deep censored quantile regression that is flexible to fit both log-linear and log-nonlinear time-to-event data for the purpose of more accurate prediction compared to the traditional quantile regression. The proposed method using the Huber check function with inverse probability weights has been implemented via Python library `Keras`, which is the high-level Application Programming Interface (API) of `TensorFlow 2.0`, and it is available on GitHub at `https://github.com/yicjia/DeepQuantreg`.

In Section 2, we review the existing work on censored quantile regression. In Section 3, we present the explanation and implementation of our deep censored quantile regression. The proposed algorithm is assessed via simulation studies in Section 4 and illustrated with a breast cancer dataset in Section 5. Finally, we conclude our paper with a brief discussion in Section 6.

2 Censored Quantile Regression

In this section, we review the existing method of quantile regression for survival data.

First without censoring, suppose data consist of a positive continuous time to an event of interest T and a covariate vector $\boldsymbol{x}^{'} = (1, x_1, ..., x_p)$ associated with a regression coefficient parameter vector $\beta^{'} = (\beta_0, \beta_1, ..., \beta_p)$. By definition, the τ^{th} conditional quantile function of the dependent variable T given covariates $\boldsymbol{x}$ is defined as

$$Q_{T|\boldsymbol{x}}(\tau) = \text{inf}\{t : \Pr(T \leq t|\boldsymbol{x}) \geq \tau\}, \tau \in (0, 1).$$

Suppose $(T_i, \boldsymbol{x_i})$ is a realization of the random variable T without censoring and a covariate vector for the i^{th} subject. Then the popular log-linear quantile regression model can be specified as

$$Q_{T_i|\boldsymbol{x_i}}(\tau) = \exp(\beta_{\tau}^{'}\boldsymbol{x_i}). \tag{1}$$

We can show that the model (1) stems from the the log-linear failure time model

$$\log(T_i) = \beta_{\tau}^{'}\boldsymbol{x_i} + \epsilon_i, \tag{2}$$

where ϵ_i would be the residual from the fitted systemic part of the model, i.e. $\beta_{\tau}^{'}\boldsymbol{x_i}$, by minimizing the sum of the absolute deviations (LAD, least absolute deviation)

$$\sum_{i=1}^{n} |\log(T_i) - \beta_{\tau}^{'}\boldsymbol{x_i}|. \tag{3}$$

Equivalently the LAD estimators can be obtained by minimizing the check function (Koenker and Bassett Jr, 1978)

$$\rho_{\tau}(u) = u[\tau - I(u \leq 0)], \tag{4}$$

where $u = \log(T_i) - \beta_{\tau}^{'}\boldsymbol{x_i}$. Denoting $S_{T0}(t) = \Pr(T > t|x = 0)$ for the baseline survival function, the log-linear failure time model is well known to be equivalent to the accelerated failure time (AFT) model with only the signs of the regression coefficients being negative, i.e.

$$S_{T_i}(t|x_i) = S_{T0}(t\exp(-\beta^{'}\boldsymbol{x_i})),$$

which induces a proportional quantile regression model

$$Q_{T_i|\boldsymbol{x_i}}(\tau) = Q_{T0}(\tau)\exp(\beta_{\tau}^{'}\boldsymbol{x_i}), \tag{5}$$

because the baseline quantile function can be expressed as $Q_{T0}(\tau) = t\exp(-\beta_{\tau}^{'}\boldsymbol{x_i})$ and the conditional quantile given $\boldsymbol{x_i}$ as $Q_{T_i|\boldsymbol{x_i}}(\tau) = t$. For the simplest case with a single binary covariate as a group indicator $x_i = 0$ (control) or 1 (intervention), the model (5) can be presented as $Q_{T_i|x_{i,1}=0}(\tau) = Q_{T0}(\tau)\exp(\beta_{\tau,0})$ and $Q_{T_i|x_{i,1}=1}(\tau) = Q_{T0}(\tau)\exp(\beta_{\tau,0} + \beta_{\tau,1})$, so that $\beta_{\tau,1}$ can be interpreted as the log-ratio or the log-difference of the two quantile functions regardless of the baseline quantile function $Q_0(\tau)$ or the intercept parameter $\beta_{\tau,0}$. In addition, the baseline quantile function $Q_{T0}(\tau)$ is the quantile of a time-to-event distribution when the covariate values are 0, neither depending on time progress, unlike the baseline hazard function in the proportional hazards model (Cox, 1972), nor being subject-specific as in the typical regression model, so it can be considered as a common additive constant that

can be absorbed into the intercept parameter. This implies that the regression coefficient parameters in the quantile regression model (2) can be inferred through the log-linear failure time model (3) by minimizing the checking function (4).

We now consider right-censored survival data. Let T_i and C_i denote potential failure time and potential censoring time, respectively, and they are independent conditional on the covariate vector $\boldsymbol{x_i}$. In many clinical trials and biomedical studies, we only observe (Y_i, δ_i, $\boldsymbol{x_i}$), where $Y_i = \min(T_i, C_i)$ is the observed survival time, and $\delta_i = I(T_i \leq C_i)$ is the event indicator. To incorporate the right-censoring, we include a weight function

$$w_i = \frac{\delta_i}{\hat{G}(Y_i)},$$

in (4) where $\hat{G}(\cdot)$ is the Kaplan-Meier estimator of the censoring distribution based on the observed data $\{Y_i, I(\delta_i = 0)\}$. This weight function can be shown to be equivalent to jumps of the Kaplan-Meier estimator of the distribution function of the logarithm of the failure time (Huang et al., 2007). Therefore, the estimator $\hat{\beta}_\tau$ is the minimizer of

$$\sum_{i=1}^{n} w_i \cdot \rho_\tau(\log(Y_i) - \beta_\tau^{'} \boldsymbol{x_i}). \tag{6}$$

The solutions to equation (6) can be obtained by the function `rq.wfit()` with the option of specifying the weights w_i from the `quantreg` package in R. The consistency and asymptotic normality of the estimator $\hat{\beta}_\tau$ has been established in Huang et al. (2007).

3 Deep Censored Qunatile Regression: DeepQuantreg

3.1 Model Architecture

We propose a deep feed-forward neural network to predict the conditional quantile. Figure 1 shows the basic model architecture. The input to the network is the covariate vector x_j ($j = 1, 2, ..., J$). The hidden layers of the network are dense, i.e. fully connected by the nodes. The output of the hidden layer is given by applying the activation function to the inner product between the input and the hidden-layer weights plus the hidden-layer bias. For example, suppose there are J input variables and two hidden layers. Then, the output of the k^{th} hidden node for the first hidden layer would be

$$g_k = f_1\left(\sum_{j=1}^{J} x_j w_{jk}^{(h)} + b_k^{(h)}\right), \quad k = 1, 2, ..., K,$$

and the output of the l^{th} hidden node in the second hidden layer would be

$$h_l = f_2\left(\sum_{k=1}^{K} g_k w_{kl}^{(h)} + b_l^{(h)}\right), \quad l = 1, 2, ..., L,$$

where $f_1(\cdot)$ and $f_2(\cdot)$ denotes the activation functions for hidden layers, and $w^{(h)}$ and $b^{(h)}$ represent the hidden-layer weights and bias, respectively, both of which get updated at each training iteration. The bias terms allow for shifting the activation function outputs left or right (Gallant and Gallant, 1993; Bishop et al., 1995; Reed and Marks II, 1999). Lastly, the output layer of the network is a single node with a linear activation function which gives the estimate of the conditional τ^{th} quantile for the i^{th} subject as

$$\hat{Q}_i^{(\tau)} = \sum_{l=1}^{L} h_l w_l^{(o)} + b^{(o)},$$

where $w^{(o)}$ and $b^{(o)}$ denote the output-layer weights and bias, respectively. Popular activation functions are the logistic function (sigmoid), rectified linear unit (RELU) defined as $\max(0, y)$, where y is a linear response function, and scaled exponential linear unit (SELU), among other things. It is known that the sigmoid activation function can cause vanishing or exploding gradient problem, so the RELU or SELU is generally preferred.

3.2 Cost, or Loss, Function

Similar to the ordinary censored quantile regression, a modified form of the check function needs to be used in the cost function for the deep quantile regression. Following Faraggi and Simon (1995), we replace the linear functional $\beta_\tau^{'} x_i$ in

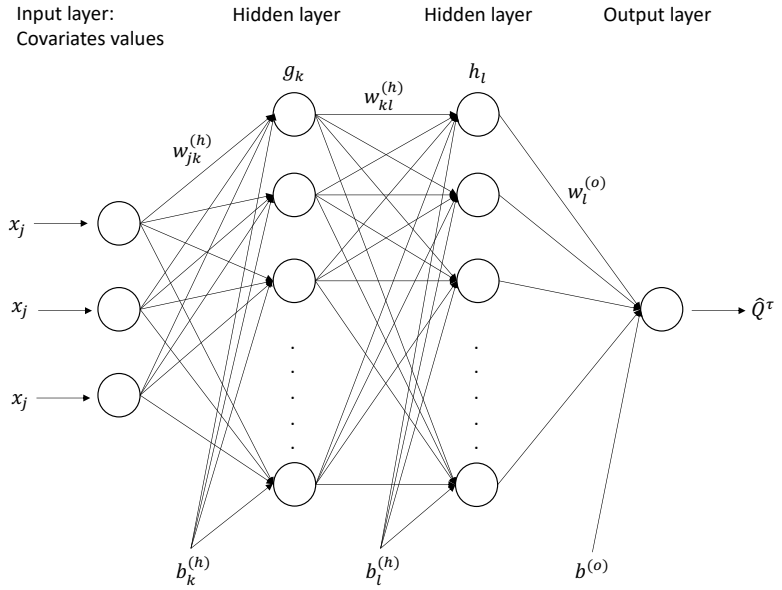


Figure 1: An overview of the deep censored quantile regression architecture used in this study.

equation (6) by the output from the neural network $\hat{Q}_i^{(\tau)}$, i.e.

$$\sum_{i=1}^{n} w_i \cdot \rho_\tau(\log(Y_i) - \hat{Q}_i^{(\tau)}). \tag{7}$$

Note that, a single layer neural network with linear activation function could approximate the ordinary quantile regression.

3.3 Optimization

The optimization of the neural network is achieved by the gradient descent process in the backpropagation phase (Rumelhart et al., 1986) to minimize the cost, or loss, function. The minimum can be obtained as the derivative of the loss function with respect to the weights through the chain rule involving the inputs and outputs of the activation functions reaches 0. Since it is often impossible to obtain a closed form solution for the weights including the biases in the deep learning setting, however, the gradient descent method is often used to iteratively search for the minimum of the loss function by updating old weight values by the amount of the slope of the loss function at those values multiplied by a learning parameter (usually small) that controls the size of convergence steps. The backpropagation algorithm through the gradient descent method was shown to be simple and computationally efficient (Goodfellow et al., 2016).

Optimizers implemented in TensorFlow typically include Stochastic Gradient Descent (SGD), SGD with Momentum, Adaptive gradient optimizer (AdaGrad) (Duchi et al., 2011), AdaDelta (Zeiler, 2012), Adam (adaptive moment estimation) (Kingma and Ba, 2014), and Adamax (a variant of Adam based on the infinity norm), among other things. For example, the ordinary Gradient Descent (GD) algorithm directly converges but requires heavy computational burden because of optimizing the loss function defined on all data points. Stochastic GD (SGD) is computationally more efficient because it optimizes the loss function defined on a single point at a time. The SGD with momentum algorithm smooths out the stochastic path of the weight function convergence in SGD by using exponential moving average like ARIMA, which makes the convergence more direct and faster. The AdaGrad algorithm changes the learning rate for each iteration of the propagation, each different layer, neuron, and weight, so that the learning rate is getting smaller with more iterations by dividing by sum of squares (up to the current iteration) of the derivative of the loss function with respect to the weights, leading to smoother convergence. AdaDelta optimization algorithm improves AdaGrad weakness of the learning rate converging to zero with iterations. Adam is an efficient stochastic optimization algorithm that only requires first-order gradients with little memory requirement, computing individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. The initial values of the weights for these algorithms could be generated from some uniform or normal distributions that depend on the numbers of input and/or output variables, but more research seems to be needed.

The check function in equation (4) is undefined at the origin, and thus not differentiable everywhere. Therefore, we adopted the Huber function (Huber et al., 1973) to smooth the check function as in Chen (2007) and Cannon (2011), which is defined as

$$\rho_\tau(u) = \begin{cases} \tau h(u) & \text{if} \quad u \geq 0 \\ (\tau - 1)h(u) & \text{if} \quad u < 0 \end{cases} \tag{8}$$

where

$$h(u) = \begin{cases} \frac{u^2}{\xi} & \text{if} \quad 0 \leq |u| \leq \xi \\ |u| - \frac{\xi}{2} & \text{if} \quad |u| > \xi \end{cases}$$

is the Huber function.

3.4 Hyperparameter tuning

Hyperparameter tuning is essential in machine learning methods such as neural network and random forests since each different training run of the algorithm could provide a different output even with the same set of hyperparameters. As will be shown later, in addition, a model with more layers and more nodes per layer tends to capture the nonlinear patterns in the data better, so hyperparameter tuning is also crucial to prevent overfitting. In this paper, hyperparameters involved in neural network, including number of hidden layers, number of nodes in each layer, activation function, optimizer, number of epochs and batch size, are tuned using 5-fold cross validation.

3.5 Model evaluation

To evaluate the prediction performance of our model, we use both concordance-index (*C*-index) (Harrell Jr et al., 1984) and modified mean error sum of squares (MMSE), which is newly proposed here for the censored quantile regression.

The C-index is one of the most common metrics used to assess the prediction accuracy of a model in survival analysis and it is considered concordant if a case that fails at a earlier time point is predicted with a worse outcome. In our case, we use the τ^{th} quantile estimated from the model as the predicted outcome while the failure, or event, time is the observed counterpart. The value of C-index is between 0 and 1, where 0.5 indicates a random prediction and 1 is a perfect association of predicted and observed outcomes.

Similar to the the mean error sum of squares (MSE) from the ordinary least squares method, the MMSE is defined as the mean of residual sum of squares between observed true event times and predicted quantile estimates under censoring. Since the prediction procedure already adjusts for censored observations via the inverse probability weighting to predict the true event times, the MMSEs are only calculated over event times. Mathematically, for the traditional quantile regression, the MMSE can be defined as

$$\sum_{i=1}^{n} \left[\delta_i \{ \log(Y_i) - \hat{\beta}_\tau' x_i \} \right]^2,$$

and for the neural network algorithm,

$$\sum_{i=1}^{n} \left[\delta_i \{ \log(Y_i) - \hat{Q}_i^{(\tau)} \} \right]^2.$$

4 Simulation Studies

In this section, we compare our deep censored quantile regression (DeepQuantreg) with the traditional censored quantile regression (Quantreg) under different simulation scenarios. We also compare the predicted results with and without Huber function in our deep neural network model. Thus, three models we compare are (i) traditional censored quantile regression with ordinary check function, (ii) deep censored quantile regression with ordinary check function, and (iii) deep censored quantile regression with Huber check function. The hyperparameters we used in this simulation study are summarized in Table 1.

4.1 Data Generation

One binary predictor and one continuous predictor were involved in simulating nonlinear patterns of time to event data. Specifically, failure times were first generated from 15 sequentially combined exponential distributions, with event rates changing over the main continuous predictor without a group effect, which will be referred to as “no group effect data" Second, an additional binary predictor was included to induce a multiplicative group effect, referred to as

Table 1: Hyperparameters used in simulation studies

Data type	Censoring	# of layers	# of nodes/layer	Activation function	Optimizer	# of epochs	Batch size
No effect data	10%	2	300	Sigmoid	Nadam	500	64
	30%	2	300	Hard-sigmoid	Adadelta	500	64
	50%	2	300	Sigmoid	Adadelta	500	64
	70%	2	300	Hard-sigmoid	Nadam	500	64
Group effect data	10%	2	300	Hard-sigmoid	Adam	500	64
	30%	2	300	Hard-sigmoid	Adam	500	64
	50%	2	300	Hard-sigmoid	Nadam	500	64
	70%	2	300	Sigmoid	Nadam	500	64
Subgroup effect data	10%	2	300	SELU	Nadam	100	64
	30%	2	300	Hard-sigmoid	Adadelta	100	64
	50%	2	300	SELU	AdaMax	100	64
	70%	2	300	Sigmoid	Adam	100	64

Acronyms: SELU - Scaled Exponential Linear Unit;
Adam - Adaptive Moment Estimation;
Nadam - Nesterov-accelerated Adaptive Moment Estimation

"group effect data". Finally, censoring times were drawn from a uniform distribution between 0 and c, where c is chosen to give desired censoring proportions of 10%, 30%, 50% and 70%, and the minimum of failure times and censoring times were created as observed survival times, together with associated event indicators. For example, Figure 2 shows simulated no group effect data and group effect data under 10% (Figure 2a and 2b) and 50% censoring (Figure 2c and 2d), respectively, where $\triangle$ indicates control group and $*$ does intervention group. Note that under 50% censoring the larger true failure times tend to be censored in the middle. We generated training and test data sets with sample size of $n = 1500$ for both no group effect data and group effect data. We performed 1000 simulations and compared the C-index and MMSE metrics on the test sets.

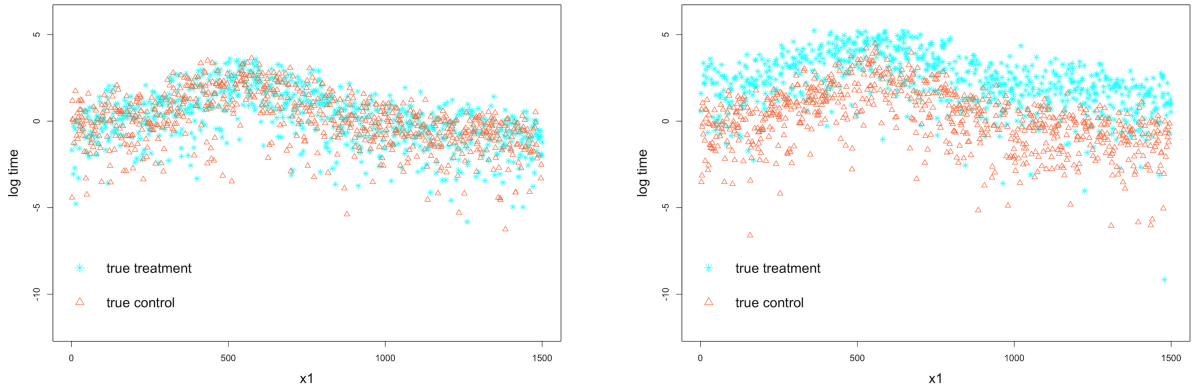
4.2 Main Comparison

First we compared performance among the models (i)-(iii) described at the beginning of this section for different τ values and censoring proportions. Figure 3 shows the box plots of the C-indices. One can observe that DeepQuntreg outperforms the traditional quantile regression in all scenarios, as expected, even though the Huber check function hardly affect the performance of the deep learning algorithm in these simulated data. The MMSE results (Table 2) are consistent with the C-index results, where our deep learning algorithms have much smaller MMSEs than the traditional quantile regression method. Interestingly note that the MMSEs become smaller with higher censoring proportions, which will be explained later in this section. Again, using the Huber function in the loss function doesn't affect the MMSE results, but it will be adopted in the subsequent analyses in this paper due to its differentiability at the origin.

We also present the median ($\tau = 0.5$) prediction results on the test sets with censoring proportions of 10% and 50% for the no group effect data and group effect data. Figure 4 shows the fitting results when the censoring proportion was 10%, where $\circ$ indicates the fitted values for the control group and $+$ does for the intervention group, implying that DeepQuantreg captures the nonlinear patterns in the data reasonably well in both scenarios, unlike the traditional quantile regression model with the restricted linear assumption, even though the same check function was used in the loss function. Figure 5 shows a similar plot for the case of 50% censoring. In Figure 5, one can observe that DeepQuantreg also captures the nonlinear patterns appropriately under heavy censoring, demonstrating ability of capturing different patterns for different groups in this example, i.e. unimodal shape for control group and a flatter shape for intervention group due to the censoring effect of the larger true event times in the middle, which also explains reduction of the MMSEs for the heavier censoring case in our simulation studies.

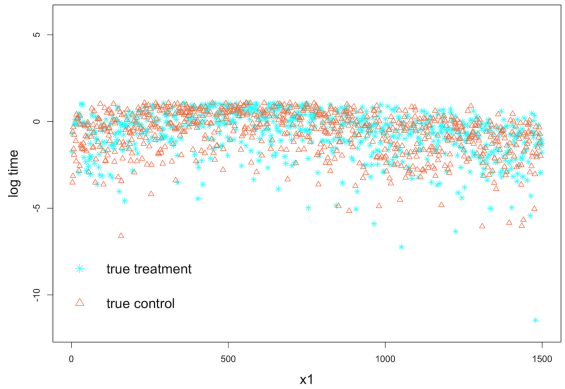
4.3 Effects of Number of Nodes

We evaluated the effect of the number of hidden layers and the number of nodes per layer on the performance of the proposed deep censored median regression model, i.e. when $\tau = 0.5$, for the group effect data set. Figure 6a shows that the model using one hidden layer with 4 nodes can only predict a linear fit. However, increasing the number of layers seems to help capturing the non-linearity. Figure 6b shows that with 300 nodes per layer, even one hidden layer can approximate the non-linearity reasonably well, and increasing the number of layers will further sharpen the nonlinear fit. In many studies using deep neural network, the number of nodes in hidden layers tends to decrease toward the output, being less than or equal to the number of input variables. Therefore, our finding here interestingly indicates that given the same number of layers the neural network predicts the nonlinear patterns better when the number of nodes in the hidden layer is larger than the number of input variables.

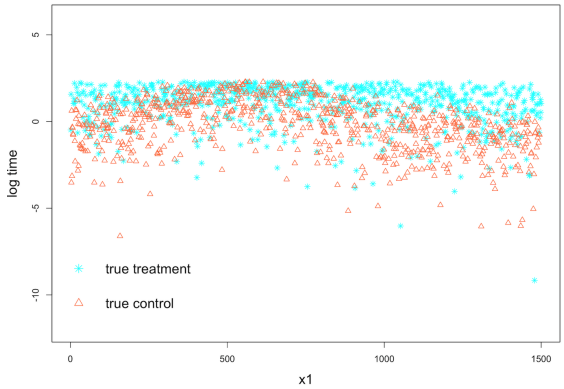


(a) No group effect data with 10% censoring

(b) Group effect data with 10% censoring



(c) No group effect data with 50% censoring



(d) Group effect data with 50% censoring

Figure 2: Simulated data for no group effect data (left) and group effect data (right) under different censoring proportions

5 Real Data Application

In this section, we compared DeepQuantreg with traditional quantile regression with the Netherlands Cancer Institute 70 gene signature data set (Van De Vijver et al., 2002), which is publicly available from the R package `penalized`. The data set contains 144 lymph node positive breast cancer patients' information on metastasis-free survival, 5 clinical risk factors, and gene expression measurements of 70 genes found to be prognostic for metastasis-free survival in an earlier study. The censoring proportion is around 67%. The covariates we included in the model are age at diagnosis, ER status, and 6 genes that show nonlinear relationship with time to metastasis or last follow-up (Figure 7). The data set was randomly split into $\frac{2}{3}$ training and $\frac{1}{3}$ test sets and repeated for 50 times to calculate the average performance based on the C -index and MMSE.

Table 3 shows that the predication performance of the traditional quantile regression decreases as the quantile increases. However, our deep learning algorithm was stable for all three quantiles, and it outperforms the traditional quantile regression in both the C-index and MMSE. From this application to a real data set, one can also observe that the proposed deep learning algorithm performs reasonably well even the sample size is small/moderate and and the number of covariates is small.

6 Discussion

In this paper, we developed a deep learning algorithm for the quantile regression under right censoring. We adopted the Huber check function in the loss function with inverse probability weights to adjust for censoring. A common knowledge seems to be that the utility of the deep learning algorithm is only reserved for high-dimensional data.

Table 2: MMSE (mean(SD)) of the two simulation scenarios under different censoring proportion and quantiles τ

Censoring	Method	Quantiles τ		
		0.25	0.5	0.75
No group effect data				
10%	Quantreg	19.71 (1.99)	16.61 (1.82)	18.88 (1.75)
	DeepQuantreg w/o Huber	11.98 (1.41)	12.00 (1.4)	12.01 (1.46)
	DeepQuantreg w/ Huber	12.00 (1.41)	12.01 (1.41)	12.02 (1.47)
30%	Quantreg	1.99 (0.16)	1.48 (0.13)	3.37 (1.29)
	DeepQuantreg w/o Huber	1.44 (0.14)	1.43 (0.13)	1.43 (0.13)
	DeepQuantreg w/ Huber	1.43 (0.13)	1.44 (0.14)	1.44 (0.14)
50%	Quantreg	0.4 (0.03)	0.32 (0.05)	0.89 (0.24)
	DeepQuantreg w/o Huber	0.34 (0.03)	0.34 (0.03)	0.34 (0.03)
	DeepQuantreg w/ Huber	0.34 (0.03)	0.34 (0.03)	0.34 (0.03)
70%	Quantreg	0.08 (0.01)	0.08 (0.02)	0.22 (0.05)
	DeepQuantreg w/o Huber	0.07 (0.01)	0.07 (0.01)	0.07 (0.01)
	DeepQuantreg w/ Huber	0.07 (0.01)	0.07 (0.01)	0.07 (0.01)
Group effect data				
10%	Quantreg	376.24 (44.05)	306.76 (38.83)	334.04 (40.48)
	DeepQuantreg w/o Huber	215.2 (26.5)	215.48 (26.68)	215.63 (27.17)
	DeepQuantreg w/ Huber	215.17 (26.58)	215.23 (26.88)	215.52 (28.24)
30%	Quantreg	27.02 (2.45)	20.41 (1.79)	44.21 (10.84)
	DeepQuantreg w/o Huber	18.5 (1.6)	18.5 (1.63)	18.51 (1.6)
	DeepQuantreg w/ Huber	18.49 (1.61)	18.5 (1.59)	18.5 (1.6)
50%	Quantreg	3.47 (0.33)	2.98 (0.45)	7.47 (2.48)
	DeepQuantreg w/o Huber	2.6 (0.21)	2.59 (0.21)	2.6 (0.21)
	DeepQuantreg w/ Huber	2.6 (0.21)	2.6 (0.21)	2.59 (0.21)
70%	Quantreg	0.42 (0.04)	0.38 (0.08)	1.04 (0.31)
	DeepQuantreg w/o Huber	0.35 (0.03)	0.35 (0.03)	0.35 (0.03)
	DeepQuantreg w/ Huber	0.35 (0.03)	0.35 (0.03)	0.35 (0.03)

Table 3: C-index and MMSE (mean(SD)) of the NCI data under different quantiles τ

τ	0.25		0.5		0.75	
Methods	Quantreg	DeepQuantreg	Quantreg	DeepQuantreg	Quantreg	DeepQuantreg
C-index	0.710 (0.07)	0.710 (0.06)	0.64 (0.09)	0.712 (0.07)	0.62 (0.06)	0.708 (0.06)
MMSE	16.11 (10.12)	17.12 (2.70)	34.41 (26.17)	16.93 (3.55)	59.75 (30.45)	17.24 (2.78)

However, in this paper, we demonstrated that our proposed deep learning algorithm performs very well even with a small number of covariates, or input variables. In addition, we illustrated how the number of hidden layers and the number of hidden nodes per layer affect the predication ability of the proposed algorithm. Our finding is that to capture the nonlinear patterns properly, one may need more hidden layers and/or more nodes per layer than the number of input variables. It is highly recommended to do the hyperparameter tuning before running the model to get the best performance while avoiding overfitting.

Typical methods for regularization to prevent overfitting in machine learning would be to use a penalty function, to resample both subjects and features (predictors) as in random forest, or to drop out layers as in neural network, but we haven't applied such methods in this paper simply because our simulated data and real data set were not high-dimensional.

Acknowledgments

Dr. Jeong's research was supported in part by National Institute of Health (NIH) grant 5-U10-CA69651-11.

References

Bishop, C. M. et al. (1995). *Neural Networks for Pattern Recognition*. Oxford university press.

Cannon, A. J. (2011). Quantile regression neural networks: Implementation in r and application to precipitation downscaling. *Computers & Geosciences* **37,** 1277–1284.

Chen, C. (2007). A finite smoothing algorithm for quantile regression. *Journal of Computational and Graphical Statistics* **16,** 136–164.

Ching, T., Zhu, X., and Garmire, L. X. (2018). Cox-nnet: an artificial neural network method for prognosis prediction of high-throughput omics data. *PLoS Computational Biology* **14,** e1006076.

Cox, D. R. (1972). Regression models and life-tables. *Journal of Royal Statistical Society-Series B* **34,** 187–220.

Cox, D. R. (1975). Partial likelihood. *Biometrika* **62,** 269–276.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* **12,** 2121–2159.

Faraggi, D. and Simon, R. (1995). A neural network model for survival data. *Statistics in Medicine* **14,** 73–82.

Gallant, S. I. and Gallant, S. I. (1993). *Neural Network Learning and Expert Systems*. MIT Press.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Harrell Jr, F. E., Lee, K. L., Califf, R. M., Pryor, D. B., and Rosati, R. A. (1984). Regression modelling strategies for improved prognostic prediction. *Statistics in Medicine* **3,** 143–152.

Huang, J., Ma, S., and Xie, H. (2007). Least absolute deviations estimation for the accelerated failure time model. *Statistica Sinica* pages 1533–1548.

Huber, P. J. et al. (1973). Robust regression: asymptotics, conjectures and monte carlo. *The Annals of Statistics* **1,** 799–821.

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., and Kluger, Y. (2018). Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Medical Research Methodology* **18,** 24.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Koenker, R. and Bassett Jr, G. (1978). Regression quantiles. *Econometrica: Journal of the Econometric Society* pages 33–50.

McKeague, I. W., Subramanian, S., and Sun, Y. (2001). Median regression and the missing information principle. *Journal of Nonparametric Statistics* **13,** 709–727.

Peng, L. and Huang, Y. (2008). Survival analysis with quantile regression models. *Journal of the American Statistical Association* **103,** 637–649.

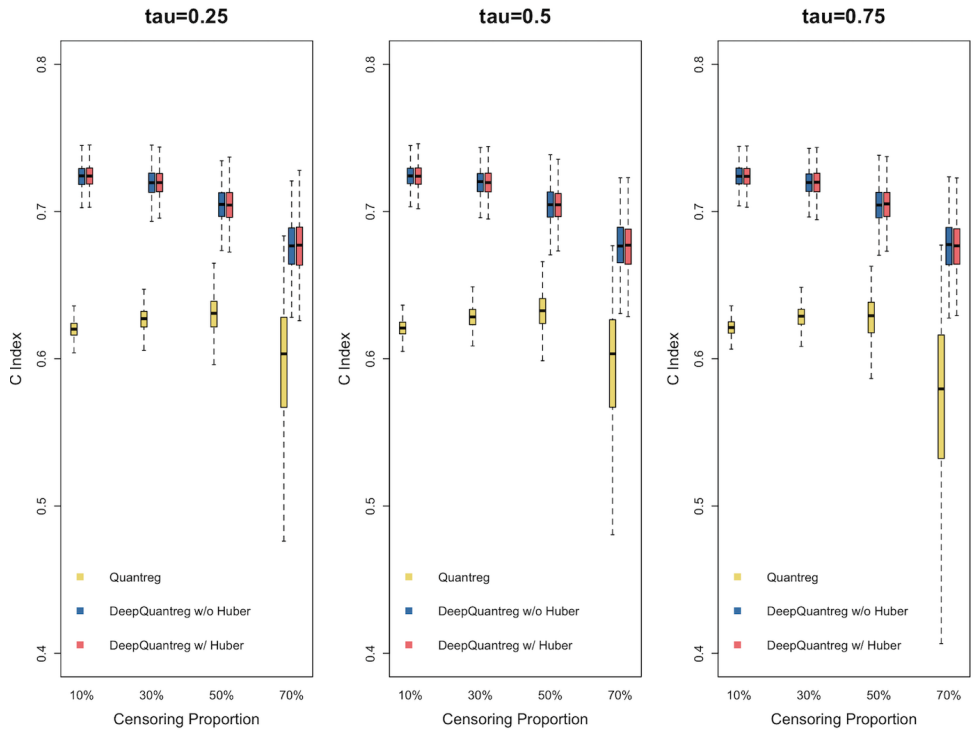
Reed, R. and Marks II, R. J. (1999). *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature* **323,** 533–536.

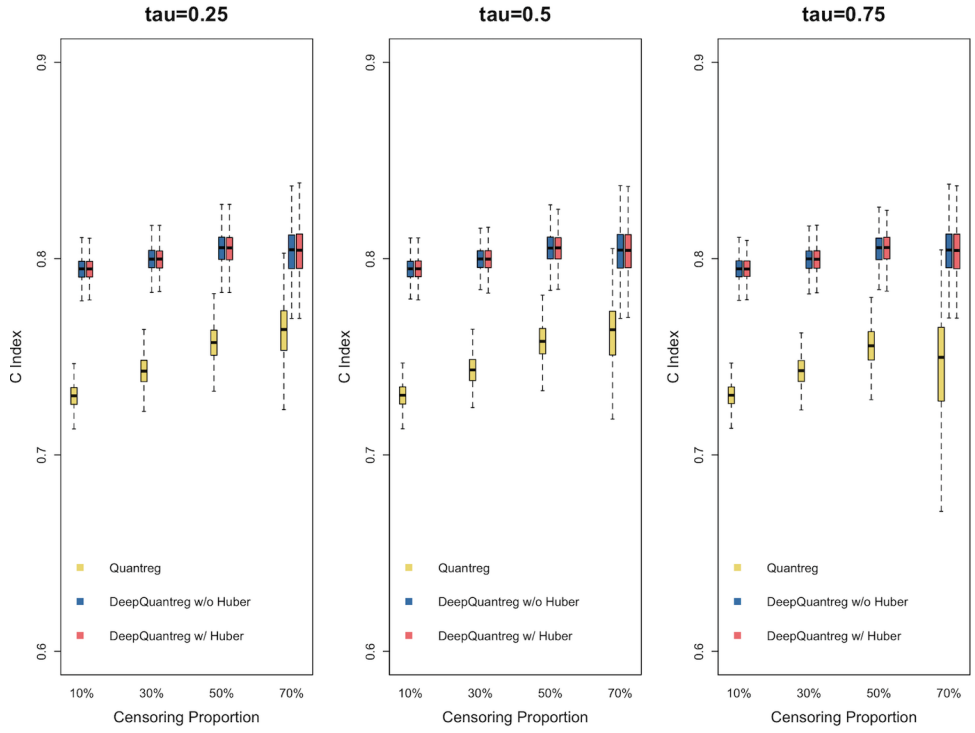
Van De Vijver, M. J., He, Y. D., Van't Veer, L. J., Dai, H., Hart, A. A., Voskuil, D. W., Schreiber, G. J., Peterse, J. L., Roberts, C., Marton, M. J., et al. (2002). A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine* **347,** 1999–2009.

Ying, Z., Jung, S.-H., and Wei, L.-J. (1995). Survival analysis with median regression models. *Journal of the American Statistical Association* **90,** 178–184.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .



(a) Boxplot of C-Index for no group effect data



(b) Boxplot of C-Index for group effect data

Figure 3: Boxplot of C-Index for different simulated data scenario

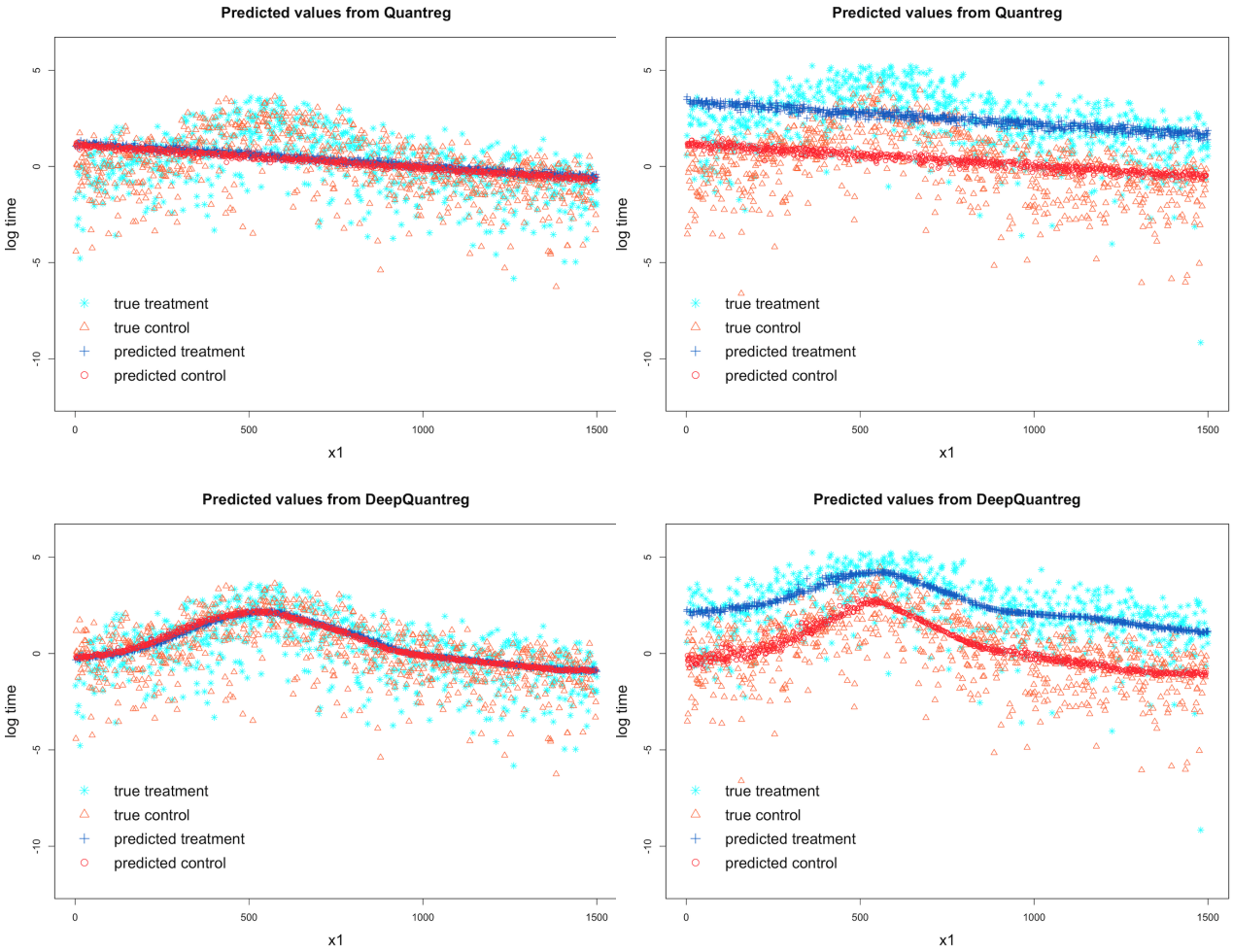


Figure 4: Median ($\tau = 0.5$) prediction plots for no group effect data (left) and group effect data (right) under 10% censoring proportion

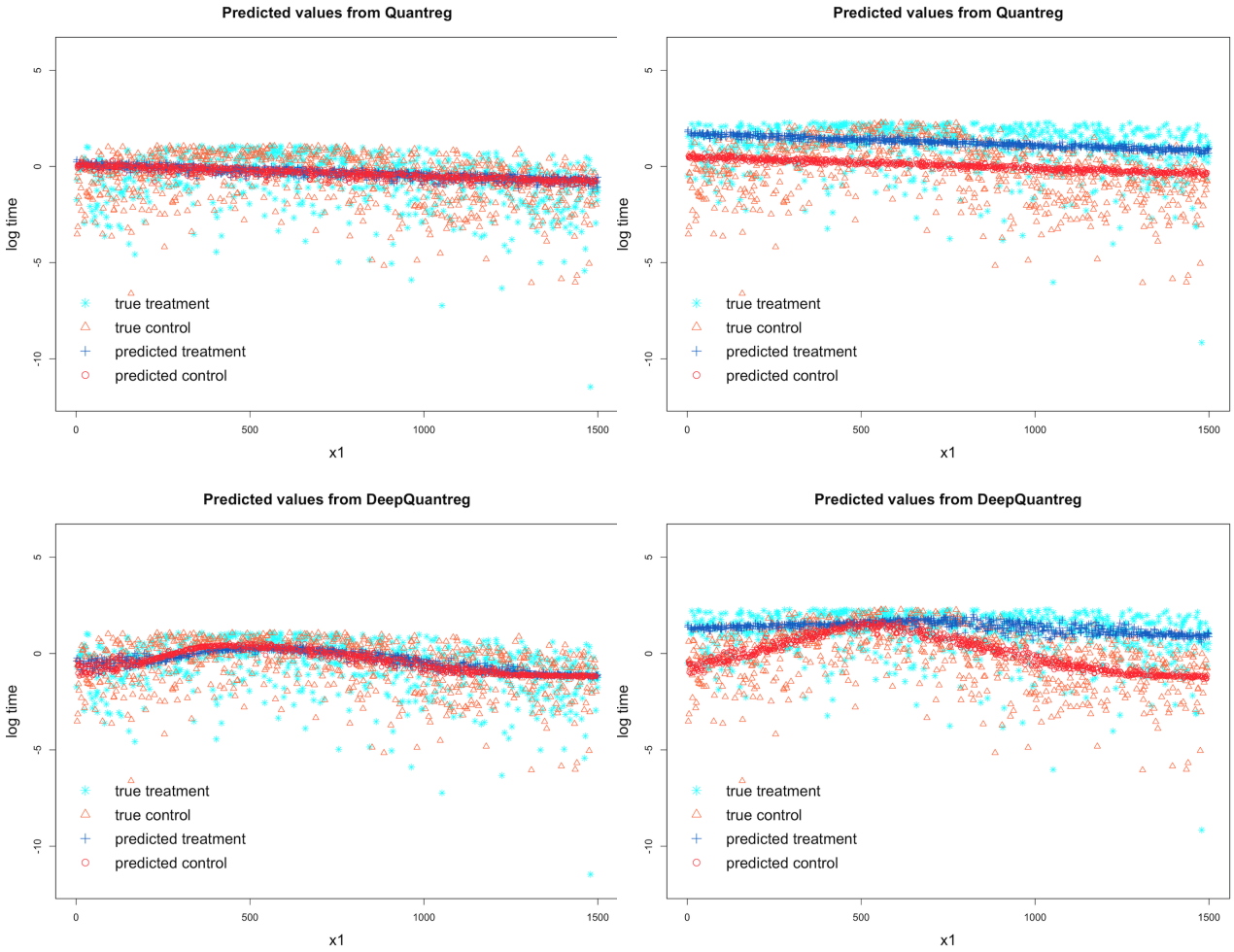
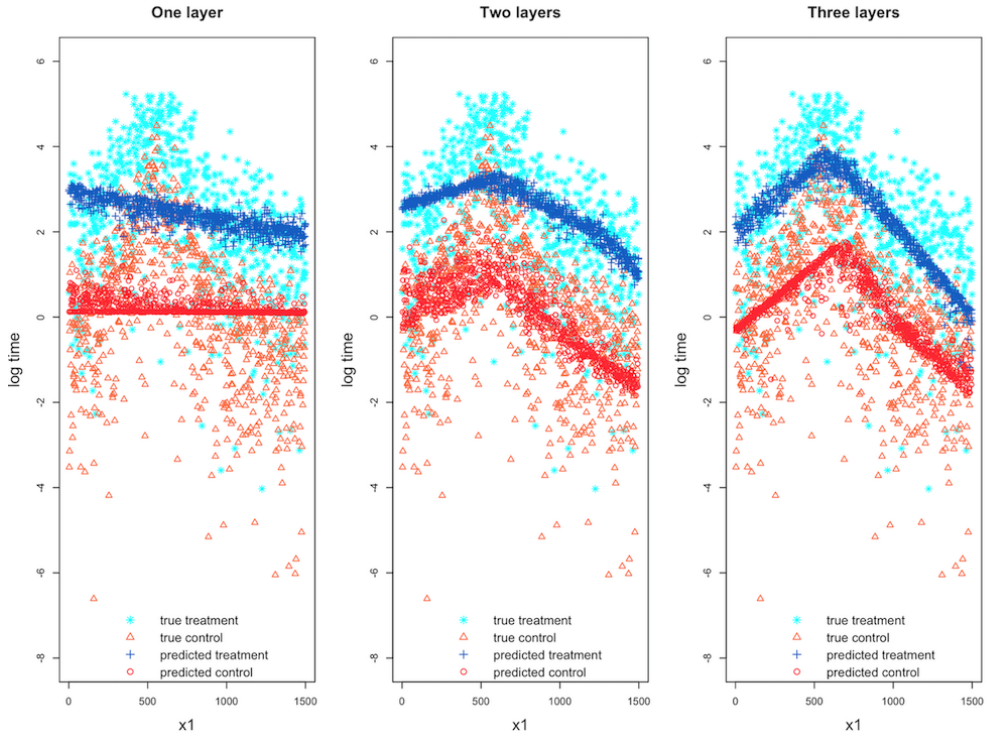
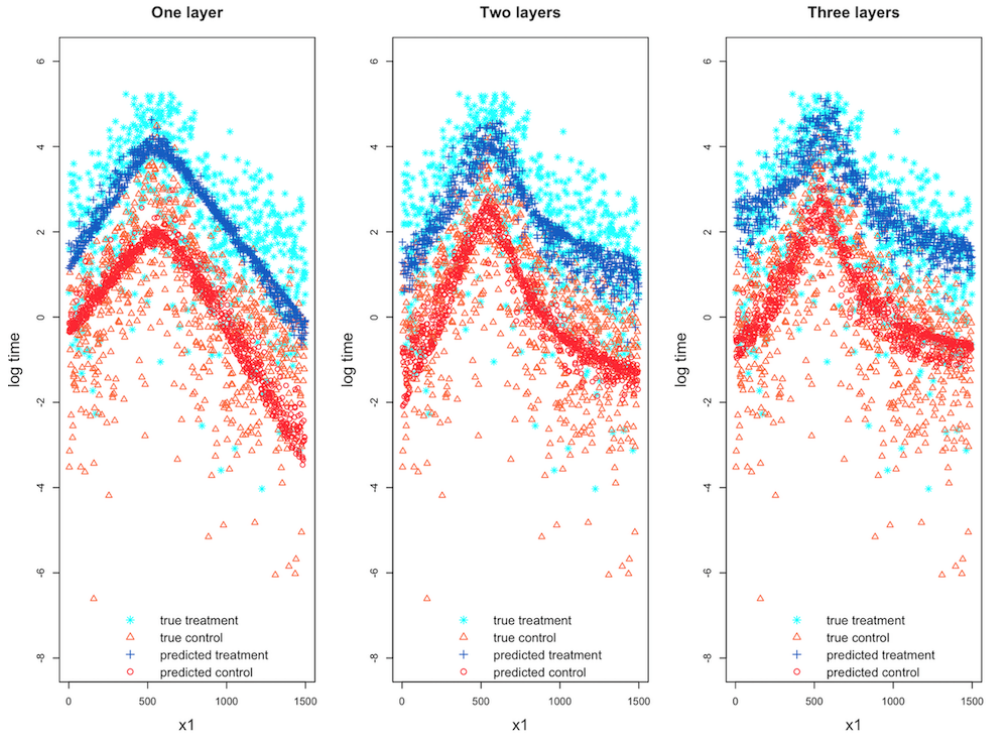


Figure 5: Median ($\tau = 0.5$) prediction plots for the no group effect data (left) and group effect data (right) under 50% censoring proportion



(a) Prediction plots for different number of layers with 4 nodes/layer



(b) Prediction plots for different number of layers with 300 nodes/layer

Figure 6: Median ($\tau = 0.5$) prediction plots for deep censored quantile regression model with different number of hidden layers and hidden nodes per layer using group effect data with 10% censoring proportion

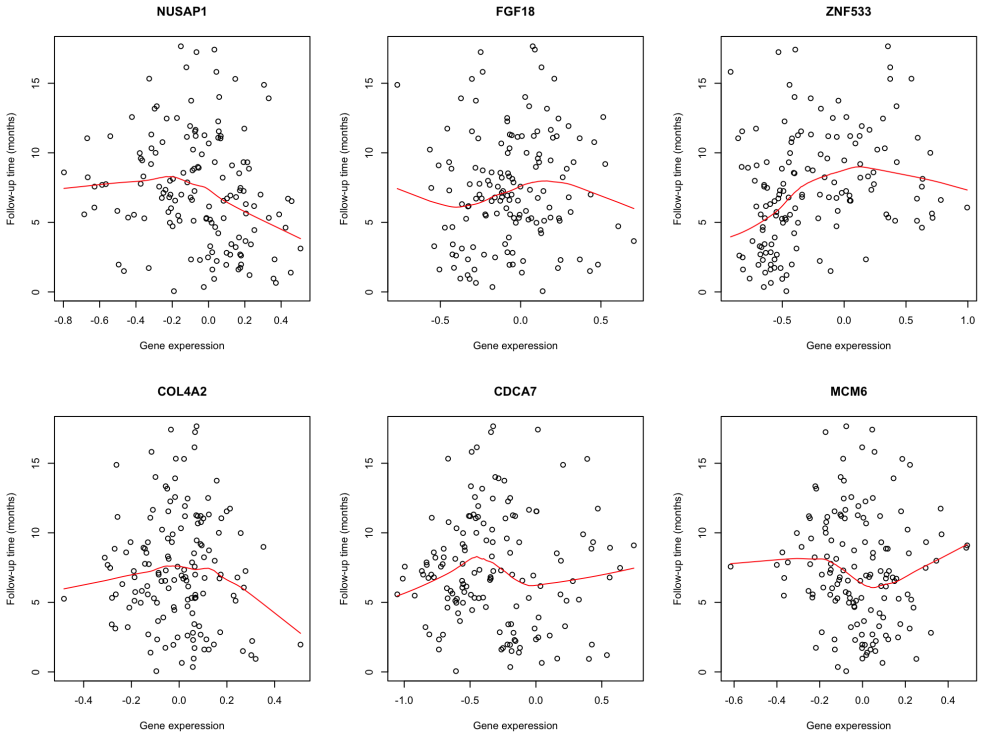


Figure 7: Scatter plots of gene expression with follow up time