Using Neural Networks to Predict Claim Duration in the Presence of Right Censoring and Covariates

David B. Speights, Ph.D.,
Joel B. Brodsky, Ph.D., and
Darya L. Chudova, M.S.

Using Neural Networks to Predict Claim Duration in the Presence of Right Censoring and Covariates

David B. Speights, Ph.D., Joel B. Brodsky, Ph.D., and Darya I. Chudova, M.S.

HNC Insurance Solutions

formerly Risk Data Corporation

David B. Speights is a research statistician at HNC Insurance Solutions. He holds a doctorate in Biostatistics from the University of California, Los Angeles and is an expert in censored regression techniques. His current research is in the area of individual claims reserving and neural networks for right-censored data.

Joel B. Brodsky is Vice President, Scoring and Segmentation at First USA Bank, N.A. At the time this paper was prepared, he was Vice President, Research at HNC Insurance Solutions. He holds a doctorate degree in Statistics from the University of California, Berkeley and is an expert in the application of advanced statistical methods. He has been a statistical advisor to a variety of industries, including banking and finance, biomedical, education, insurance, and electric and gas utility.

Darya I. Chudova is a statistical analyst at HNC Insurance Solutions. She is a doctorial candidate in applied mathematics. She specializes in the development and application of neural network and statistical models for spectral and financial data.

HNC Insurance Solutions is a business unit of HNC Software Inc. and was formed in the 1998 merger of CompReview Inc. and Risk Data Corporation. HNC Insurance Solutions is an insurance information service and statistical research company which creates and markets solutions for the insurance industry.

Abstract

We present a general methodology for fitting feed-forward neural networks when both right censoring and covariate information (claim attributes) exist. Right censoring occurs when only intermediate, but not final values of a time-dependent variable (such as claim duration) are known for some data points, and final values of the variable are known for all other observations. This situation frequently arises in casualty insurance when there are active claims in an analysis data set. The techniques we develop are applicable for estimating the distribution of claim lifetimes when awards are disbursed over the unknown claim life. The neural-network framework allows us to handle complex relationships between the claim attributes and claim duration.

We will derive a generalization for right-censored data of the back-propagation method used for fitting feed-forward neural networks. A connection between least squares estimation and maximum likelihood estimation will be used to establish the generalization. A typical cross-validation approach to modeling will be described to reduce over-fitting. An application of our methods is demonstrated for predicting the duration of a claim in worker's compensation insurance in the presence of covariates.

1 Introduction

In casualty insurance, it is common for the payments on a claim to be disbursed over time. For example, in workers' compensation insurance, a claim is filed some time after injury to the worker and payments are made on the claim over a period of several years. In this setting, most data samples contain claims that are still active and do not have complete information. Therefore, when building models to estimate claim duration, we need to use techniques designed to handle incomplete observations.

When a claim is open at the time of sampling, the claim duration is said to be right censored. The claim is right censored because all we know is the final claim duration exceeds the current duration. From a graphical perspective, the right end of the claim's timeline has been hidden from view. For example, if the claim is open for 16 months prior to sampling, we know that at closing the claim's duration will exceed 16 months.

When estimating the duration of a claim, it is important to consider the point in the claim's life at which we are making the estimate. For example, if we make a prediction on the day that a claim is reported, we will be limited to available information. Alternately, if our prediction is made after three months of claim activity, we will have more information. Models should reflect the point in time at which data are available. For example, we may want to use the total medical paid at six months as a predictor of duration. However, this information will not be known at the beginning of a claim's life. Therefore, this model is applicable only for predictions at 6 months duration for claims that exceed 6 months duration.

Estimating claim duration and the distribution of durations can be useful for a number of reasons. For example, there may be a need to make an early assessment of the claim's severity based on all available claim information. This type of procedure may be useful in providing an index of the claim's severity relative to claim duration. Methods such as these provide a systematic way of evaluating a large amount of claim information in an efficient and logical manner. Using a neural network to predicts duration provides a comprehensive method that uses complete historical data to develop the predictions of duration.

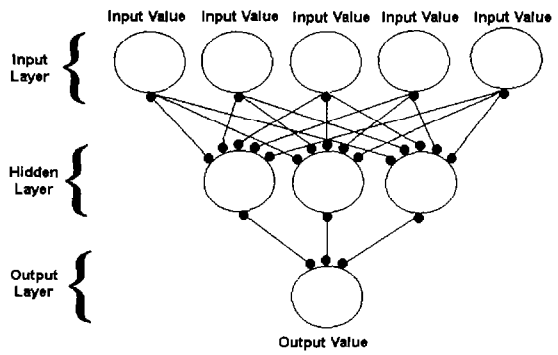
In this paper, we develop methods to model the relationship between claim characteristics and the duration of a claim. These methods use a generalization of the back-propagation algorithm to right-censored data for feed-forward neural networks. Back-propagation is a numerical optimization technique that is commonly used to estimate a neural network's parameters (often referred to as weights in the neural network literature). Feed-forward refers to the specific order in which each subject's information is processed. The techniques developed here build on ideas presented in (Faraggi & Simon 1995, Liestol, Andersen & Andersen 1994). We will generalize the neural network, back-propagation algorithm to right-censored data using a likelihood-based approach.

1.1 Introduction to Neural Networks

Neural network models are closely related in form to many commonly used statistical techniques. (Wasserman 1989) provides a technical introduction to neural networks. (Sarle 1994) describes connections between several statistical procedures and neural networks. Among the procedures he discusses are linear regression, logistic regression, discriminant analysis, multivariate linear regression, and principal component analysis. Most of these techniques are shown to be special cases of neural networks. The flexibility of neural networks to apply to a wide variety of modeling situations makes them valuable as a general framework for statistical analysis. This paper will draw one more connection between neural networks and statistical procedures. We will show how the neural network framework can be used to model continuous outcome data with right censoring.

Figure 1 is a graphic representation of a typical neural network architecture. Such a diagram is commonly used in literature on neural networks. In the figure, the flow of information, or data processing sequence, is downward. Because the flow is only one-way and begins with the input variables, the network is said to be a feed-forward network. Each circle in the figure is called a node, or "processing unit." In actuality, each node represents the evaluation of a function. Estimation of the functional parameters is called "fitting." Thus, each node can be thought of as a separate regression. Also, each row of circles in Figure 1

Figure 1: Diagram of a Feed-Forward Neural Network



is referred to as a "layer."

Consider the nonlinear regression

$$y = 4\cos(7 + 3x).$$

For a given value of x, the function $7 + 3x$ is first evaluated and then the cosine of the intermediate value is calculated. In neural network problems, x corresponds to the input level, $7 + 3x$ refers to the input to the node in the hidden layer, and cos(.) is the activation function of the hidden layer's node, and the result of $\cos(7 + 3x)$ is the output of the hidden layer's node. The layer of nodes is said to be "hidden" because it is unavailable to the network's user. The output of the hidden layer is then multiplied by 4 and passed to the output layer. The information flow is said to be one-way because a given x value determines the value for $7 + 3x$ which in turn determines the output of the hidden layer and the output layer through the model weights. In this setup, there is only one hidden node and the model weights are 7, 3, and 4.

In the general representation of Figure 1, the top layer of nodes represents the input data or predictor variables, where each circle signifies one continuous variable, or one level of a categorical variable. The middle layer represents the hidden layer of the network. There are different projections of the input layer into each circle in the hidden layer. A projection is simply a linear combination of the input variables. The output from each node of the first hidden layer is typically scaled to the unit interval by an activation function. The final bottom layer represents a single linear combination of the hidden layer and is called the prediction or output layer. This diagram depicts a one hidden-layer model, but more hidden layers can be added.

A neural network can model complex relationships between the input and output variables. Such relationships include interactions between multiple input variables and nonlinear transformations of input variables. With more traditional analysis methods, discovering subtle interactions and transformations may be time-consuming and difficult, if not impossible. With a neural network, the network architecture is easily adapted to include subtle interactions and transformations.

Neural networks can be powerful tools for modeling claim duration and costs. To intuitively understand this assertion, assume that the mean of the output variable can be accurately approximated by a (possibly very complex) continuous function. Consider Figure 1 with only one hidden layer and assume the output of each hidden node is a simple continuous function. With linear combinations of the certain simple continuous functions, the result can be made arbitrarily complex by utilizing a sufficiently large number of hidden nodes. This allows the neural network to approximate a wide class of functions.

Parameters of a feed-forward neural network are often estimated using a technique known as the back-propagation algorithm. The algorithm is an optimization technique and is related to the gradient descent algorithm. Some details of the algorithm are presented in section 2.2. Interested readers are referred to (Wasserman 1989) for more details.

Example 1.1: Representing Nonlinear Deterministic Functions To demonstrate the ability of neural networks to capture nonlinear relationships, we generated data randomly from the polynomial equation

$$z = \frac{1}{3}x^3 - x + 1. \tag{1.1}$$

We generated values of x from a uniform distribution on the interval $[-3, 3]$ and determined z values using equation 1.1.

Figure 2 shows the fit to these data of a feed-forward neural network with one hidden layer and three nodes in the hidden layer. Methods for specifying the form, or architecture, of a neural network and for estimating its parameters will be described in the next section. This example is intended solely to demonstrate that neural networks can accurately approximate nonlinear relationships.

The solid line in Figure 2 represents the neural network equation and the superimposed scatter plot represents the true values that were generated. Figure 2 demonstrates the ability of the neural network to adapt to nonlinear relationships with relatively few nodes in the hidden layer. The general mean structure of the neural network allows us to represent a polynomial relationship without specifying quadratic or nonlinear terms in our model.

2 Neural Networks for Right-Censored Data

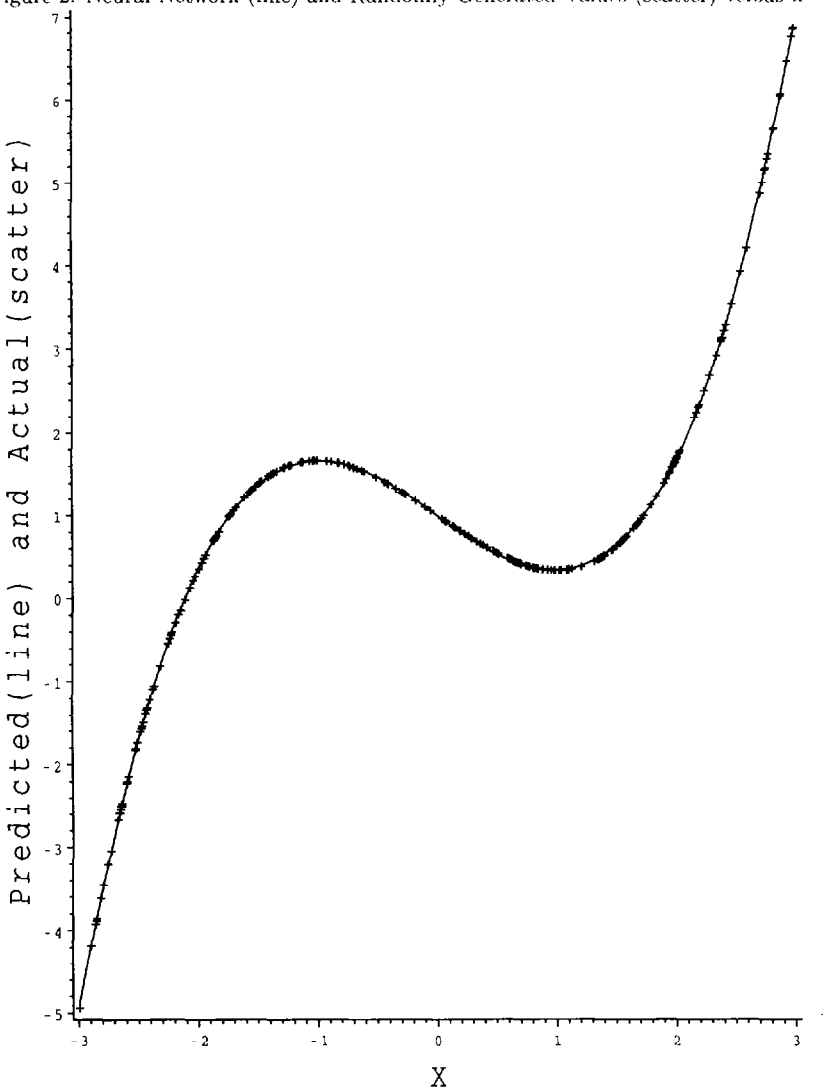
The feed-forward neural network is analogous to a regression model because there is a set of input values, typically called predictors in statistical models, and an output variable, usually known as the response variable. In regression analysis, the model is

$$Z_i = \beta' X_i + \sigma\epsilon_i, \tag{2.1}$$

where Z_i is the response, β is a p x 1 vector of parameters, X_i is a p x 1 [1] vector of predictor variables, σ is a scale parameter, and ϵ_i is a random error term with distribution function

[1] For simplicity of notation, the first element of X_i is assumed to be identically 1. With this formulation, the right hand side of 2.1 includes an additive term that is analogous to the intercept term in traditional regression

Figure 2: Neural Network (line) and Randomly Generated Values (scatter) versus x



F and density function f. The covariate vector, X_i, is hypothesized to have an additive relationship to the outcome Z_i [2].

While properties of such a regression model are well known and parameter estimates are straightforward to obtain, the model in equation 2.1 is often inappropriate due to model misspecification. The primary misspecification issue is the additivity in the mean structure. An alternative to the linear model is a mean structure with a more general formulation.

In order to employ a neural network model, replace the linear mean structure, $\beta' X_i$, in equation 2.1 with a more general mean function, $h(\theta, X_i)$, as

$$Z_i = h(\theta, X_i) + \sigma\epsilon_i. \tag{2.2}$$

Here, h is an arbitrary function with a univariate response and θ is a parameter vector corresponding to the mean structure being fit. By choosing h properly, we can represent many feed-forward network architectures with equation 2.2. We will restrict our attention to feed-forward neural networks with a single hidden layer. Our methods generalize to multiple hidden layers without much difficulty.

For a feed-forward neural network with one hidden layer, specify

$$h(\theta, x) = f(\alpha_0 + \sum_{j=1}^{H} \alpha_j s_j(\beta_j' x)). \tag{2.3}$$

In this equation, $\alpha_0, ..., \alpha_H$ are scalars, $\beta_1, ..., \beta_H$ are p x 1 vectors, H is the number of nodes in the hidden layer, f is known as the activation function of the ouput layer, $s_j(.)$ are known as the activation functions for the hidden layer, and $\theta = \{\alpha_0, ..., \alpha_H, \beta_1', ..., \beta_H'\}'$ is the vector of all parameters in the neural network. For the work presented in this paper, $f(x) = x$ is assumed to be the identity function and $s_1(x) = ... = s_H(x)$ are all assumed to be equal. Using the same activation functions for $s_1, ..., s_H$ is common in most neural network literature, but this is not necessary. Some commonly chosen activation functions are linear ($s(x) = ax + b$) and logistic ($s(x) = [1 + \exp(-x)]^{-1}$). The reader should note that

[2]With the formulation of equation 2.1, interactions between and transformations of the input variables are represented as additional covariates.

this model is a special case of projection pursuit regression which is described in (Huber 1985).

If all of the activation functions in the hidden layer $s_1, \ldots s_H$ are set to the identity function, this procedure is equivalent to traditional regression analysis. In this setting, many of the parameters in the neural network will not be identifiable, but the equation can be reduced to identifiable elements that are equivalent to regression parameters.

The neural network's ability to represent complex relationships between the input values and the output value is derived through the activation functions. By taking linear combinations of simple nonlinear functions, it is possible to represent complex relationships. By coupling this ability with multiple projections (linear combinations) of the input variables onto the hidden layer, the nonlinear relationships and interactions can be represented by the network structure.

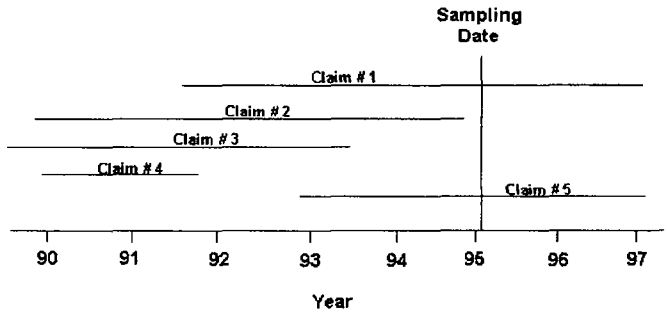
Using Equation 2.2, we can develop a likelihood equation for the data when a form is specified for the error distribution, F. In the next section, we will use this formulation to generalize the back-propagation algorithm to accommodate right-censored data.

2.1 Parametric Estimation

Let $T_1, \ldots, T_n$ represent a random sample of claim durations and let $O_1, \ldots O_n$ represent the associated injury dates for the claims. Define the sampling date as S_0. The associated fixed censoring times for each claim are $C_i = S_0 - O_i$. We observe $Y_i = \min(T_i, C_i)$. If a claim is open, $Y_i = C_i$, otherwise, $Y_i = T_i$. Censoring is represented by an indicator variable $\delta_i = I(Y_i = T_i)$. If $\delta_i = 1$ the claim is uncensored and if $\delta_i = 0$, the claim is censored. Let $X_i = (X_{1i}, \ldots, X_{pi})'$ represent the p x 1 vector of covariates, or claim attributes, for the i^{th} individual.

Censored regression techniques are developed under the assumption that T_i is independent of C_i conditional on X_i. We consider C_i to be a fixed censoring time since our samples are collected at a fixed point in time. When the censoring variable is considered fixed, but each individual's censoring time can be different, then the censoring is often referred to as

Figure 3: Diagram of Sample Worker's Compensation Claims



generalized Type I censoring. The independence assumption is satisfied when T_i is independent of O_i conditional on X_i. This assumption implies that any association the duration of claim has with injury date is explained by the covariates.

Consider the following situation to illustrate the notation. Suppose we sample on a particular day, say January 31, 1995. In our notation, January 31, 1995 minus the injury date, is the censoring time. Since each claim has a different injury date, they have different censoring times. The situation is depicted for five sample claims in Figure 3. In Figure 3, claims 2, 3, and 4 are uncensored, while claims 1 and 5 are censored. We have partial information on the censored claims and would have technical difficulties accurately calculating the mean duration of a claim without incorporating censored data analysis techniques.

Let $\Theta = (\theta', \sigma)$ be the complete vector of model parameters. In equation 2.2, let $Z_i = \log(T_i)$ and $e_i = (Z_i - h(\theta, X_i))/\sigma$. If ϵ has a standard normal distribution, then the likelihood of the data is

$$L(\Theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \left[\exp\left(-\frac{1}{2}e_i^2\right)\right]^{\delta_i} \left[\int_{e_i}^{\infty} e^{-\frac{1}{2}u^2} du\right]^{1-\delta_i}.$$

With maximum likelihood estimation, estimates of members of the parameter vector, Θ, will

be those values which maximize L.

Rather than maximizing L, it is generally easier to maximize the log likelihood, $l(\Theta)$.

$$l(\Theta) = log(L(\Theta)) = -\sum_{i=1}^{n}\left[\frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2}\delta_i c_i^2 - (1-\delta_i)\log\left(\int_{c_i}^{\infty} e^{-\frac{1}{2}u^2}du\right)\right].$$

For the back-propagation algorithm, typically a cost function is used for estimation that represents the amount of prediction error in our model. This cost function is minimized to obtain estimates of the model parameters. To accommodate right-censored data, we propose using the negative of the log likelihood as the cost function for parameter estimation.

$$C(\Theta) = \sum_{i=1}^{n}\left[\frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2}\delta_i c_i^2 - (1-\delta_i)\log\left(\int_{c_i}^{\infty} e^{-\frac{1}{2}u^2}du\right)\right]. \tag{2.4}$$

If all of the data in equation 2.4 are uncensored and f represents the normal density, equation 2.4 can be written as

$$C(\Theta) = \sum_{i=1}^{n}\frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2}\left(\frac{Z_i - h(X_i,\theta)}{\sigma}\right)^2 \tag{2.5}$$

$$= \frac{n}{2}\log(2\pi) + n\log(\sigma) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(Z_i - h(X_i,\theta))^2. \tag{2.6}$$

The first two terms on the right-hand side of equation 2.6 do not depend on θ and the value of θ which minimizes the third term will be the same for all values $\sigma > 0$. Therefore, the value of θ which minimizes $C(\Theta)$ is the same θ which minimizes

$$C_1(\Theta) = \sum_{i=1}^{n}(Z_i - h(X_i,\theta))^2. \tag{2.7}$$

This θ is known as the least squares estimate and $C_1(\Theta)$ is the cost function typically used in fitting feed-forward neural networks without censored data. Thus, the proposed cost function given by equation 2.4 provides a generalization to the standard back-propagation algorithm for fitting feed-forward neural networks.

2.2 Numerical Estimation Procedures

Minimization of equation 2.4 can be performed with a variety of algorithms. We propose the back-propagation algorithm because it has proven successful for fitting neural network mean

structures. Unfortunately, since $C(\Theta)$ is not differentiable with respect to σ at $\sigma = 0$, the algorithm does not perform adequately for estimating σ. Therefore, we employ a two-step estimation approach with θ being estimated using back-propagation and σ being estimated using maximum likelihood.

The back-propagation algorithm is related to the gradient-descent algorithm and can be found in its general form in many neural network textbooks (see for example (Hecht-Nielsen 1990)). The algorithm minimizes $C(\Theta)$ with respect to the parameter θ while considering σ to be fixed. Unlike traditional optimization routines, estimates are typically updated one observation at a time. The model parameters are updated for the i^{th} observation and the p^{th} iteration by the following updating mechanism:

$$\theta_{i+n(p-1)} = \theta_{i-1+n(p-1)} + \Delta\theta_{i-1+n(p-1)}, \tag{2.8}$$

where

$$\Delta\theta_{i-1+n(p-1)} = \lambda \nabla_\theta C_i(\theta_{i-1+n(p-1)}, \sigma),$$

λ is known as the learning rate, and $C_i()$ represents the i^{th} term in the summation of equation 2.4, and $\nabla_\theta C_i()$ is the partial derivative of $C_i()$ with respect to θ. The reader should note that the parameter estimates (network weights) are updated at each observation. Typical values for λ range from 0.0001 to 0.1 and are typically chosen by trial and error methods.

This defines the basic version of the back-propagation algorithm. Many modifications for adjusting the learning rate, λ, for estimating the parameters have been proposed. The learning rate is typically decreased if there is an increase in the cost function through one pass of the data. For more details on this algorithm see (Wasserman 1989).

We assume that σ is fixed through each pass of the data. After each pass through the data, σ is re-estimated using maximum-likelihood techniques treating θ as fixed. Considering θ to be fixed, we estimate σ by using the Newton-Raphson algorithm

$$\sigma_{j+1} = \sigma_j - [\nabla^2_\sigma C(\theta, \sigma_j)]^{-1} \nabla_\sigma C(\theta, \sigma_j). \tag{2.9}$$

This procedure can be initialized by choosing σ_0 to be the previous value of σ or by using

$$\sigma_0 = \sqrt{\frac{\sum_{i=1}^{n}(Z_i - h(\theta, X_i))^2}{n}},$$

where θ represents the most recent value for the Θ parameters. The reader should note that choosing a good initial value for σ is crucial for the stability of our algorithm [3]

3 Example of a Neural Network With Simulated Data

In this section an example with simulated data is used to demonstrate the prediction potential of neural networks. In this example, simulated data were used so we could reconstruct the true values that would be censored in a real data set. This example will provide some indication of the accuracy of our proposed methods for prediction.

For this example, we randomly generated data from a model with true values distributed as

$$T_i = \exp(x_i^2 + 0.5 * \epsilon_{1i}),$$

and censoring values distributed as

$$C_i = \exp(0.25 + x_i^2 + 0.5 * \epsilon_{2i}),$$

where ϵ_{1i} and ϵ_{2i} are deviates from a standard normal distribution and X_i is a uniform random deviate on the interval $(-3, 3)$. We consider the minimum of these two quantities, $Y_i = \min(T_i, C_i)$, to be the observation when censoring is present.

Both T_i and C_i follow log-normal distributions conditional on X_i. To see this, note that

$$\begin{aligned} \log(T_i) &= X_i^2 + \epsilon_{1i} \quad \text{and} \\ \log(C_i) &= X_i^2 + \epsilon_{2i}. \end{aligned}$$

where

$$\begin{aligned} \epsilon_{1i} &\sim N(0, 0.25) \quad \text{and} \\ \epsilon_{2i} &\sim N(0.25, 0.25). \end{aligned}$$

[3]The Newton-Raphson procedure still contains derivatives of $C(\Theta)$ with respect to σ. Therefore, it will experience similar problems near $\sigma = 0$. We have found that with a good starting value, this problem is minimized and the above algorithm is reasonably stable.

Thus, conditional on X_i,

$$\begin{aligned} \log(T_i) &\sim N(X_i^2, 0.25) \quad \text{and} \\ \log(C_i) &\sim N(X_i^2 + 0.25, 0.25). \end{aligned}$$

We generated 1000 observations and achieved approximately 35% censoring. This level of censoring is moderately heavy. We fit this data with the algorithms described in section 2.2. The architecture employed was a five-node, feed-forward neural network with a single hidden layer and normally distributed error terms. This network can be described with the following model equation,

$$\log(T) = \alpha_0 + \sum_{j=1}^{5} \alpha_j s(\beta_{0j} + \beta_{1j} X) + \sigma\epsilon.$$

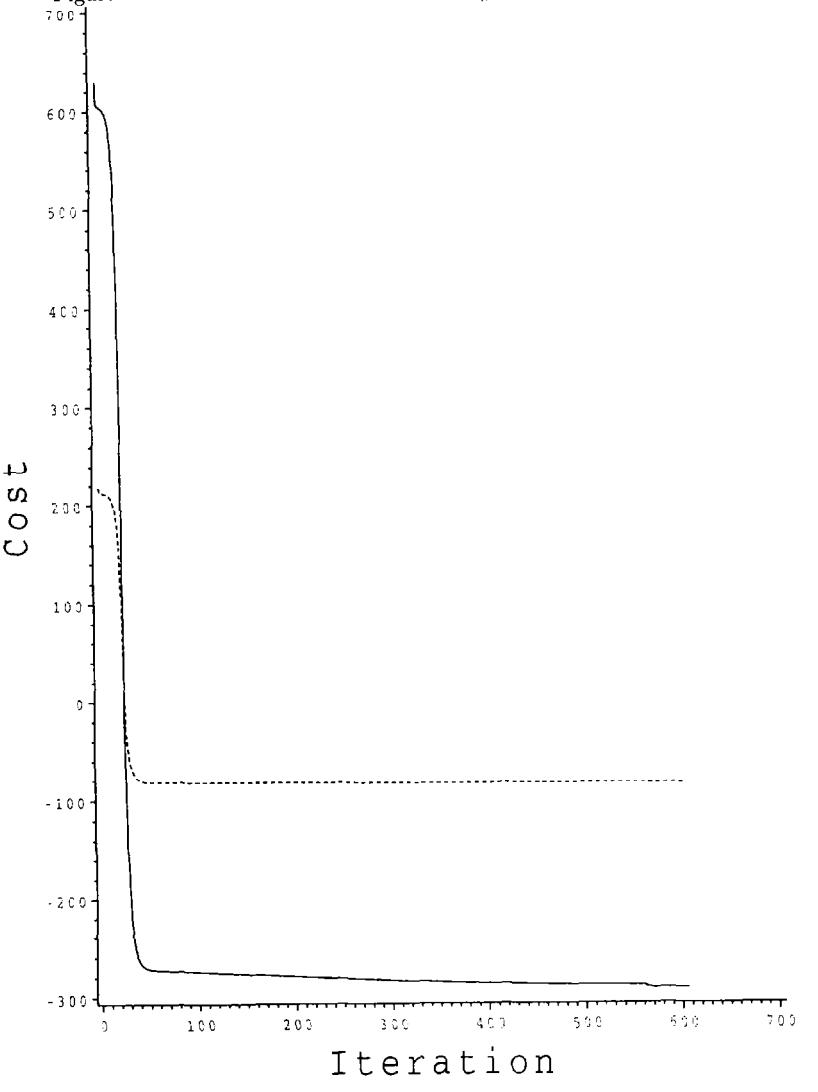
In this equation $s(u) = [1 + exp(-u)]^{-1}$ is the logistic function, and epsilon has a standard normal distribution.

Our data set of 1000 observations was split randomly into two parts with approximately 75% in the training set and 25% in the testing set. The data in the training set were used to fit or "train" the network. The data in the test set were used to assess or "test" the network's predictive abilities. Historically, the 75/25 split has been found to be adequate in most circumstances and is the common choice for training networks, but this choice is somewhat arbitrary.

The graph in Figure 4 shows values for the cost equation 2.4 plotted against p from equation 2.8 for the training set and the testing set. The algorithm described by equations 2.8 and 2.9 was applied to the training set only. In this graph the dashed lines (- - - -) represent the loss function calculated on the testing set and the solid line (———) represents the cost function calculated on the training set. Convergence was considered obtained when the testing set's cost function failed to decrease for 40 consecutive iterations. The point at which the testing set's cost function stopped decreasing was considered the convergence point. This approach guards against the dangers of over fitting that can occur in over-parameterized models.

After the neural network model was fit, we reconstructed the log predictions and plotted them against the log of the true observations $\log(T_i)$ for the test set. Figure 5 shows a plot of

Figure 4: Loss function values for the testing and training data sets



the estimated relationship between x and $E(\log(T)|x)$ (shown by the solid line) superimposed on a scatter plot of the log of the true values, T_i. With this comparison, we demonstrate the ability of neural networks to produce accurate predictions of true values even with censoring.

4 Application of Neural Networks to Workers' Compensation Data

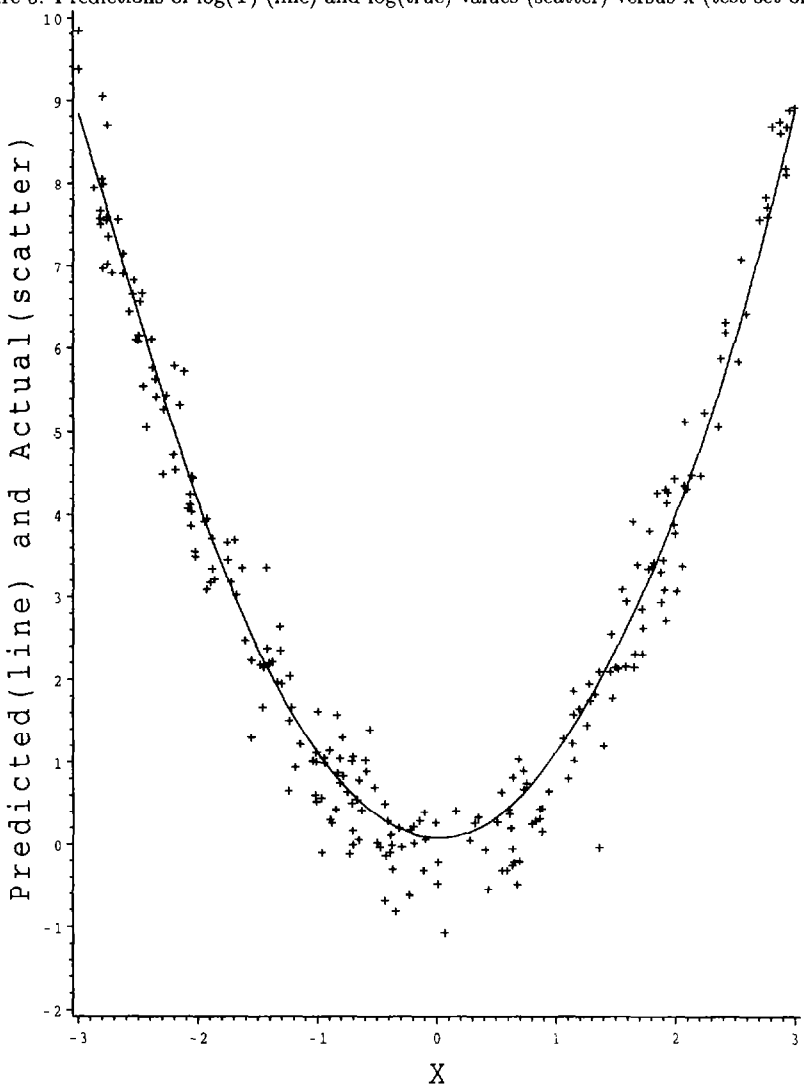
In this section we apply the methods outlined in this paper to a single state insurance carrier. Our data set consisted of all claims that opened after December 31, 1987. The data were sampled in June of 1997 and all claims that were open at that time are considered to be right censored. We construct a prediction model for estimating the duration on an individual claim with data containing right-censored observations.

Our prediction model uses several covariates that are typically available early on in a claim's life so that our models will be valid from the beginning of a claim. The characteristics used for the model are accident code, gender, weekly wage, zip code, injury type, class code, body part, nature of injury, and age at the time of injury. Accident code, injury type, class code, body part, and nature of injury variables are encoded using the National Council on Compensation Insurance (NCCI) standards.

The duration of a claim is considered to be the duration since the claim was reported to the insurance carrier. Only claims with indemnity payments were used in modeling and claims with permanent total disabilities were excluded since they typically last until a claimant is deceased. The assumptions on the distribution of the error term and censoring mechanism are defined in section 2.1.

Figure 6 demonstrates the ratio of the neural network model prediction to the actual duration against the actual duration in days. The axes are displayed in log-base 10 increments. For open cases, the duration to date was used in the plot. If all predictions are perfect, the cloud of points would lie directly on the line "1/1." Typically, the model under-predicts long duration claims and over-predicts short duration claims. The plot demonstrates that most

Figure 5: Predictions of log(T) (line) and log(true) values (scatter) versus x (test set only)



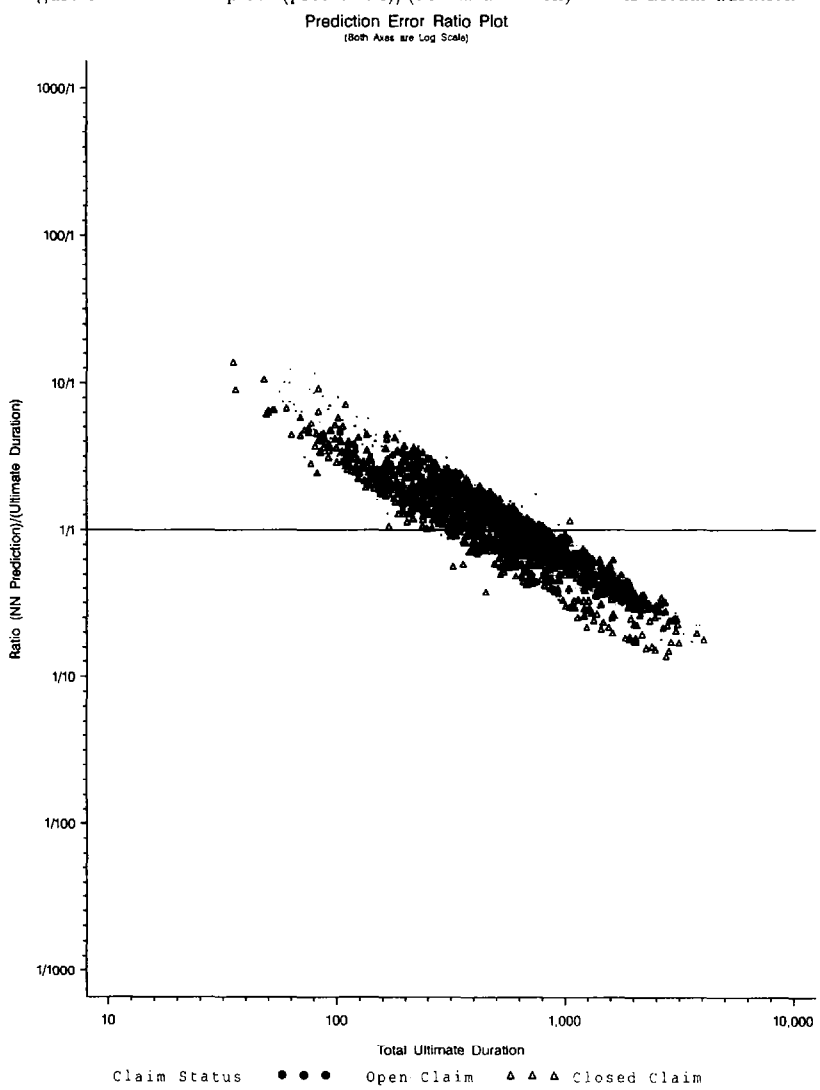
predicted durations are reasonably close to the actual duration.

5 Conclusions

This paper presented a generalization of a commonly used algorithm for neural networks using a likelihood-based approach. A connection between this algorithm and the typical least squares approach to estimation was demonstrated. We showed that our algorithm could make accurate predictions in the presence of right-censored data. The example with the simulated data demonstrated the ability of neural networks to identify nonlinear relationships even in the presence of right censoring. The example from workers' compensation insurance showed how this method can be applied to estimating duration in the presence of many covariates.

The ideas presented in this paper are general in nature and there are many other applications that could benefit from these techniques. We merely scratched the surface of possible applications. Neural networks have proven very useful in modeling complex situations. By adding a generalization to handle the problem of right censoring, this powerful technique can be applied to a new range of actuarial problems.

Figure 6: Error ratio plot. (prediction)/(actual duration) versus actual duration



References

Faraggi, D. & Simon, R. (1995), 'A Neural Network Model for Survival Data', *Statistics in Medicine* **14**, 73–82.

Hecht-Nielsen, R. (1990), *Neurocomputing*, Addison-Wesley.

Huber, P. J. (1985), 'Projection Pursuit', *Annals of Statistics* **13**, 435–475.

Liestol, K., Andersen, P. K. & Andersen, U. (1994), 'Survival Analysis and Neural Nets', *Statistics in Medicine* **13**, 1189–1200.

Sarle, W. S. (1994), 'Neural Networks and Statistical Models', *Proceeding: SAS Users Group International* pp. 1538–1550.

Wasserman, P. D. (1989), *Neural Computing: Theory and Practice*, Van Nostrand Reinhold.