Neural Comput & Applic
DOI 10.1007/s00521-009-0277-8

ORIGINAL ARTICLE

Nonlinear Poisson regression using neural networks: a simulation study

Nader Fallah · Hong Gu · Kazem Mohammad · Seyyed Ali Seyyedsalehi · Keramat Nourijelyani · Mohammad Reza Eshraghian

Received: 16 February 2008 / Accepted: 20 April 2009

Abstract We describe a novel extension of the Poisson regression model to be based on a multi-layer perceptron, a type of neural network. This relaxes the assumptions of the traditional Poisson regression model, while including it as a special case. In this paper, we describe neural network regression models with six different schemes and compare their performances in three simulated data sets, namely one linear and two nonlinear cases. From the simulation study it is found that the Poisson regression models work well when the linearity assumption is correct, but the neural network models can largely improve the prediction in nonlinear situations.

N. Fallah · K. Mohammad (✉) · K. Nourijelyani · M. R. Eshraghian
Epidemiology and Biostatistics Department,
University of Tehran/Medical Sciences, Tehran, Iran
e-mail: kmohamad@tums.ac.ir; mohamadk@tums.ac.ir

N. Fallah
e-mail: nader.fallah@dal.ca

K. Nourijelyani
e-mail: nouri4@yahoo.com

M. R. Eshraghian
e-mail: eshraghian@yahoo.com

H. Gu
Department of Mathematics and Statistics,
Dalhousie University, Halifax, Canada
e-mail: hgu@mathstat.dal.ca

S. A. Seyyedsalehi
Biomedical Engineering Faculty,
AmirKabir University of Technology, Tehran, Iran
e-mail: ssalehi@cic.aut.ac.ir

Keywords Generalized linear model · Poisson regression · Nonlinear regression · Multilayer perceptron · Simulation

1 Introduction

After the first paper about neural networks published by McCulloch and Pitts [1], statisticians and artificial intelligence scientists have worked independently on this subject for several decades. Recently, efforts had been made to combine techniques such as regression and classification models, which are common in both disciplines. In real applications, the most commonly used regression and classification models are linear. Currently various nonlinear methods, such as generalized additive models (GAM), classification and regression tree (CART), multivariate adaptive regression splines (MARS) and neural network models have become popular. Among these popular nonlinear methods, neural network models are attractive in their flexibility, and achieve comparable performance in prediction.

There have been some developments on the combination of neural network and statistical models. These include nonlinear multiple regression, nonlinear logistic regression and nonlinear multinomial logistic regression using neural networks described by Ripley [2] and Bishop [3]. Neural networks have also been used in modeling survival data in a variety of ways [4]. Faraggi and Simon [5] suggested an extension of the Cox proportional hazard model, and commented that similar extensions can be made to logistic regression and multinomial logistic regression. Nonlinear extension of ordinal logistic regression was proposed by Mathieson [6].

Published online: 28 July 2009

In addition, reports in some research papers suggest that the extension of generalized linear model (GLM) by neural network models are applicable in real data set such as speech recognition, waveform analysis of electrocardiography, electroencephalography, and signal processing. This kind of neural network has also been successfully applied in clinical outcome prediction of myocardial infarction, mortality, surgical decision making on traumatic brain injury patients, recovery from surgery, pediatric, genecology, head trauma, and transplantation [7–16].

In this paper, we extend the linear Poisson regression to neural network Poisson regression, and examine its performance in comparison to the linear Poisson regression for simulated data. Based on existing literature this model has not been introduced before.

2 Methods

2.1 Neural networks

The most commonly used form of neural network is the multi-layer perceptron (MLP). A MLP consists of one input layer of units, one output layer of units and possibly one or more layers of 'hidden' units. The input units pass their inputs to the units in the first hidden layer or directly to the output units. Each of the hidden layer units adds a constant (termed as 'bias') to a weighted sum of its inputs and calculates an activation function ϕ_h of the result. This is then passed to hidden units in the next layer or to the output unit(s).

In this paper, we fix the activation function (ϕ_h) as tangent hyperbolic function in hidden layer and exponential function in output layer (ϕ_0). Denote the inputs as x_i's and the outputs t_k's, for MLP with one hidden layer

$$t_k = \phi_0\left(\alpha_k + \sum_{j\to k}\omega_{jk}\phi_h\left(\alpha_j + \sum_{i\to j}\omega_{ij}x_i\right)\right) \tag{1}$$

If we have only one output node, k will be equal to one. The weights can be determined by optimizing some proper criterion function such as minimizing the sum of squared errors of the predicted variable or maximizing the log-likelihood of the data in cases where a distribution of the response variable can be assumed.

The structure of MLP made it possible to fit very general nonlinear functional relationships between inputs and outputs. Research results have shown that neural networks with enough hidden units can approximate any arbitrary functional relationships [17, 18]. However, over-fit can be a serious problem in such a framework. This problem is usually overcome either by stopping the optimization early or more often by using *regularization* techniques to penalize the optimization criterion. By adding a penalty term to the optimization criterion, the estimates of the weights will be shrunk which is also termed as shrinkage method. The following smoothness penalty is often used in shrinkage method:

$$L = -\log\text{likelihood} + \lambda\sum_{\text{weights}}\omega_{ij}^2 \tag{2}$$

This process is also known as *weight decay* in neural network literatures. The tuning parameter λ can be chosen by cross-validation. For fixed number of hidden units, we minimize this penalized log-likelihood in Eq. 2 to get the weights estimated [4].

2.2 Optimization criteria

Given a training set comprising a set of input vectors $\{x_n\}$, where $n = 1,\ldots, N$, together with the corresponding target vector $\{y_n\}$, if we assume that data points y_n ($n = 1,\ldots, N$) are independent conditional on x_n, the likelihood function can be written as:

$$P(y|x) = \prod_{n=1}^{N} p(y_n|x_n) \tag{3}$$

The error function can be defined as the negative log-likelihood:

$$E = -\log P(y_1,\ldots,y_N|x_1,\ldots,x_N) = -\sum_{n=1}^{N}\log p(y_n|x_n) \tag{4}$$

For regression problems with normality assumption, this can be reduced to the most commonly used squared error criterion:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y_n - t_n(x_n;w)\}^2 \tag{5}$$

For classification problems, it is often advantageous to associate the network outputs to the posterior probabilities of each class. For a problem with two classes, the target variable $\{y_n\}$ is binary and can be assumed to follow binomial distribution with its probability as $t_n(x_n; w)$. The error function in Eq. 4 then yields the cross-entropy error function:

$$E = -\sum\{y_n\ln t_n + (1-y_n)\ln(1-t_n)\} \tag{6}$$

This definition can be extended to other family of GLM such as multinomial logistic regression and ordinal logistic regression or Cox regression for survival models [2–6]. We will consider the Poisson regression in the following.

2.3 Poisson regression

Suppose we have a single target variable with count response, we consider the nonlinear Poisson regression for neural networks as an extension of generalized linear models.

The Poisson probability distribution for count data is given by:

$$P[Y_n = y_n] = \frac{e^{-\lambda_n}\lambda_n^{y_n}}{y_n!}, \quad y_n = 0, 1, 2, \ldots \tag{7}$$

In linear Poisson regression, the most commonly used formulation is the log-linear link function: $\ln \lambda_n = x_n'\beta$. Thus, the expected value for y_n is given by $E[y_n|x_n] = \lambda_n = e^{x_n'\beta}$.

Here, we model λ_n as a function of x_n by an MLP neural network:

$$t_n = \hat{\lambda}_n = \phi_0\left(\alpha + \sum_j \omega_j \phi_h\left(\alpha_j + \sum_{i\to j} \omega_{ij} x_n\right)\right) \tag{8}$$

Substituting Poisson probability function in Eq. 4 and using Eq. 8 as Poisson means, the negative log-likelihood criterion can be obtained as:

$$E = -\sum_{n=1}^{N}[-t_n + y_n \log t_n - \ln y_n!] \tag{9}$$

Eliminating the last term which is not related to the model fitting, we have:

$$E = -\sum_{n=1}^{N}[-t_n + y_n \log t_n] \tag{10}$$

2.4 Model fitting and gradient calculation

We compare the performances of different models using simulations. A penalized version of likelihood error criterion functions given in Eq. 11 is used to fit models with fixed number of units in hidden layer, to guard against over-fitting.

$$E_r = E + \lambda \sum_{\text{weights}} \omega_{ij}^2 \tag{11}$$

For each data set, to identify the number of units in hidden layer, both criteria Akaike Information Criterion (AIC) and Schwarz Bayesian Information Criterion (BIC) are calculated:

$$\text{AIC} = -2 \times \text{Loglikelihood} + 2 \times m \tag{12}$$

$$\text{BIC} = -2 \times \text{Loglikelihood} + m \times \log(N) \tag{13}$$

where m is the number of the estimated parameters (proportion to units) and N is the number of the observations. The model with the smallest value of the information criterion is considered to be the best. However, it should be noticed that in our neural network model fittings, for each setting of fixed number of hidden units, the negative log-likelihood score we get is suboptimal since the weights are optimized on a penalized version of Eq. 11, We thus can only get approximations of the AIC and BIC values.

We also calculated MSE for testing set as a reference measure for accuracy, where MSE is defined as

$$\frac{1}{N}\sum_{n=1}^{N}(\lambda_n - t_n)^2. \tag{14}$$

The predictions by different models are ranked by MSE.

The models considered include 2, 3, 4, 5, 10, 20 hidden units. To save the computation time, the weight decay parameter is pre-fixed at 0.012 in our simulations. This value is chosen based on some empirical study for different choices of weight decay parameter.

Back propagation is a general computing technique to fit parameters in MLP. The computation involves the numerical evaluation of derivatives of the error function with respect to the weights and biases. The general form of back propagation is described elsewhere [2, 3]. Here, we use a special algorithm based on the article by Pearlmutter [19] for computation of Hessian matrix, similar to Nabney [20] approach. The scaled conjugate gradient algorithm is used for optimization. The code is written in R 2.6.2 and Matlab 7.5.

3 Simulation

The purpose of the simulation study is to check the model selection and prediction accuracy based on neural network Poisson regression by comparison to the Poisson linear regression. Three different simulation schemes are used. For each data set we generate 1,000 independent observations. The first 500 observations will constitute a training set and the last 500 observations a testing set.

Simulation 1 (linear): a simple linear Poisson regression model is used with a single covariate X following uniform [0,1] distribution. The response variable is generated as

$$Y_i \sim \text{Pois}(\exp(X))$$

Simulation 2 (nonlinear): two covariates X_1 and X_2 are simulated as independently from U (0,1) and U (0,2). The response variable is generated as

$$Y_i \sim \text{Pois}(\exp(1 + 1.2\sqrt{X_1} + 0.25(\sqrt[4]{X_2})))$$

Simulation 3 (nonlinear): three covariates X_1, X_2, and X_3 are simulated as independently from U (0,1), U (1,2), and U (0,0.1), respectively. The response variable is generated as

$$Y_i \sim \text{Pois}(\exp(0.5 + 1.2X_1^3 + 2X_2^2 - 0.01X_3))$$

Table 1 Results of neural network models for 3 simulated data (training data)

Model	Simulation 1			Simulation 2			Simulation 3		
	P^*	AIC	BIC	P^*	AIC	BIC	P^*	AIC	BIC
MLP ($H^* = 2$)	7	758.3	787.8	9	1,787.6	1,791.5	11	1,603.5	1,649.3
MLP ($H = 3$)	10	762.5	804.6	13	1,789.1	1,794.6	16	1,615.2	1,686.5
MLP ($H = 4$)	13	768.2	823.1	17	1,790.5	1,797.6	21	1,630.3	1,718.0
MLP ($H = 5$)	16	774.2	841.6	21	1,791.6	1,800.5	26	1,649.7	1,751.1
MLP ($H = 10$)	31	804.2	934.8	41	1,796.8	1,814.1	51	1,707.5	1,924.4
MLP ($H = 20$)	61	864.2	1,123.1	81	1,805.7	1,839.8	101	1,820.3	2,240.3

P^* number of parameters, H^*number of hidden neuron

Table 2 Measure of accuracy MSE on testing data

Model	Simulation 1	Simulation 2	Simulation 3
Poisson regression	0.001	0.603	0.369
MLP ($H^* = 2$)	0.005	0.081	0.054
MLP ($H = 3$)	0.013	0.121	0.110

H^* number of hidden neuron

In simulations 2 and 3 we try to produce nonlinear and more complicated schemes in comparison to the linear model used in simulation 1.

Both linear Poisson and neural network Poisson models are fitted using the training set. Models are selected based on AIC and BIC. The mean squared error (MSE) is calculated for the testing set as a confirmation.

Based on the AIC and BIC values in Table 1, the models with fewer parameters correspond to smaller AIC and BIC values. Therefore, our expectation is that the model with fewer parameters gives us better prediction. MSEs are calculated on the testing sets for MLP with two units in the hidden layer in Table 2. We also include result for MLP with three units in hidden layer as a reference.

Result in Table 2 shows that the nonlinear neural network predictions are slightly less accurate than the Poisson linear regression when the truth is linear (first column in Table 2), which can be anticipated. However, the gains in accuracy are significant when the truth is nonlinear (second and third columns in Table 2). It is also noticed that the model selected based on AIC and BIC values in Table 1 was the model with two nodes in hidden layer. This model indeed performs better than the model with three nodes in the hidden layer as confirmed by MSE in Table 2. This shows such a model selection procedure is basically valid. And even the suboptimal nonlinear model performs much better than the linear model if the truth is nonlinear.

Figure 1 shows comparisons of real lambda values with their predictions by linear Poisson regression and neural network Poisson regression for the testing set in simulation 2. It is obvious that the neural network predictions are much closer to the actual values in comparison to the linear Poisson regression predictions.

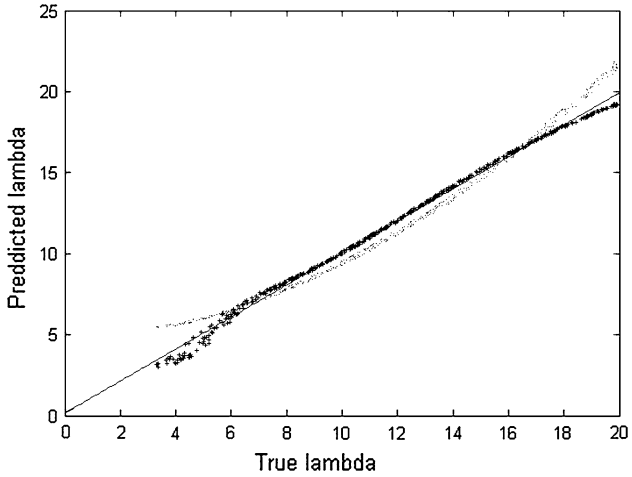

Fig. 1 Plots of real lambda and its prediction based on second simulation data. Linear Poisson prediction (“.”), Poisson neural networks (“+”)

4 Conclusion

Any nonlinear models in statistical methodology need some sort of assumptions about either distributions and/or prior knowledge for suitable functions to be used in modeling the nonlinearity, but in real applications (particularly in modeling the phenomena in medical settings) we actually never know which suitable functions to use. One major benefit of neural networks is their flexibility. As a consequence, in many applications, neural networks have shown better prediction ability compared to classical statistical methods. MLP as a landmark of neural networks is a flexible nonlinear regression model, and can always be used to approximate a continuous function. It applies well when response variable is real valued. If the response variable is integer valued, in particular measured as count data, the MLP is not suitable to be applied directly. The

hybridization of neural network with generalized linear models provides a natural solution to such situations. In this paper, we proposed a new nonlinear extension of Poisson regression based on neural networks. We compared the performance of two types of models for some simulated data. Poisson regression, as a conventional statistical method, has the ability to model some linear relationship, between the independent and dependent variables, while as an artificial neural network structure can increase this flexibility in modeling the nonlinearity in independent variables. While the neural network and linear Poisson regression have similar performances and accuracy rates, according to prediction when the truth is in linear form, the neural networks achieve much higher accuracy in predictions then when the truth is in nonlinear form. Both linear and nonlinear Poisson regression can be implemented by existing software. However, neural networks require a more elaborate setup of parameters, and programming. Other researchers have examined the extension of ordinary linear regression, logistic regression, multinomial logistic regression, and ordinal logistics regression by neural networks. This research, by extending Poisson regression to neural networks, completed another part of the generalized linear models.

Acknowledgments This study was sponsored by Tehran University of Medical Sciences. Most of this work was carried out during a sabbatical year (student visitor period) of Nader Fallah at the Department of Mathematics and Statistics in Dalhousie University. Authors wish to thank R. Ripley and I. Nabney for their help on R and Matlab Codes. The authors thank Catherine Pretty and Janet Brush for editing this manuscript.

References

1. McCulloch W, Pitts W (1943) A logical calculus of ideas immanent in nervous activity. Bull Math Biophys 5:115–133. doi:10.1007/BF02478259
2. Ripley BD (1996) Pattern recognition and neural networks. Cambridge University Press, London
3. Bishop CM (2006) Pattern recognition and machine learning. Springer, New York
4. Ripley RM, Harris AL, Tarassenko L (2004) Non-linear survival analysis using neural networks. Stat Med 23:825–842. doi:10.1002/sim.1655
5. Faraggi D, Simon R (1995) The maximum likelihood neural network as a statistical classification model. J Statist Plann Inference 46:93–104. doi:10.1016/0378-3758(95)99068-2
6. Mathieson MJ (1996) Ordinal models for neural networks. Neural networks in financial engineering. In: Refences A-PN, Abu-Mostafa Y, Moody J, Weigend A (eds), Proceedings of the third international conference on neural networks in the capital markets, pp 23–536
7. Nejadgholi I, Seyyedsalehi SA (2009) Nonlinear normalization of input patterns to speaker variability in speech recognition neural networks. Neural Comput Appl 18:45–55. doi:10.1007/s00521-007-0151-5
8. Baxt WG (1990) Use of an artificial neural network for data analysis in clinical decision-making: the diagnosis of acute coronary occlusion. Neural Comput 2:480–489. doi:10.1162/neco.1990.2.4.480
9. Huang YL, Wang KL, Chen DR (2006) Diagnosis of breast tumors with ultrasonic texture analysis using support vector machines. Neural Comput Appl 15:164–169. doi:10.1007/s00521-005-0019-5
10. Akin M, Kurt MB, Sezgin N, Bayram M (2008) Estimating vigilance level by using EEG and EMG signals. Neural Comput Appl 17:227–236. doi:10.1007/s00521-007-0117-7
11. Bailey TC, Everson RM, Fieldsend JF, Krzanowski WJ, Partridge D, Schetinin V (2007) Representing classifier confidence in the safety critical domain: an illustration from mortality prediction in trauma cases. Neural Comput Appl 16:1–10. doi:10.1007/s00521-006-0053-y
12. Eftekhar B, Mohammad K, Eftekhar H, Ghodsi M, Ketabchi E (2005) Comparison of artificial neural network and logistic regression models for prediction of mortality in head trauma based on initial clinical data. BMC Med Inform Decis Mak 5:3. doi:10.1186/1472-6947-5-3
13. Sadat-Hashemi SM, Kazemnejad A, Lucas C, Badie K (2005) Predicting the type of pregnancy using artificial neural networks and multinomial logistic regression: a comparison study. Neural Comput Appl 14:198–202. doi:10.1007/s00521-004-0454-8
14. Leondes CT (1998) Neural network systems techniques and applications. Academic Press, San Diego
15. Shafi I, Ahmad J, Shah SI, Kashif FM (2008) Computing deblurred time frequency distributions using artificial neural networks. Circuits Syst Signal Process 27:277–294. doi:10.1007/s00034-008-9027-x
16. Shafi I, Ahmad J, Shah SI, Kashif FM (2007) Evolutionary time-frequency distributions using Bayesian regularised neural network model. IET Signal Process 1:97–106. doi:10.1049/iet-spr:20060311
17. Funahashi K (1989) On the approximate realization of continuous mapping by neural networks. Neural Netw 2:183–192. doi:10.1016/0893-6080(89)90003-8
18. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2:359–366. doi:10.1016/0893-6080(89)90020-8
19. Pearlmutter BA (1994) Fast exact multiplication by the Hessian. Neural Comput 6:147–160. doi:10.1162/neco.1994.6.1.147
20. Nabney I (2001) Netlab algorithms for pattern recognition. Springer London Ltd, UK