July 2017

Bayesian Artificial Neural Networks in Health and Cybersecurity

Hansapani Sarasepa Rodrigo
University of South Florida, sarasepa@mail.usf.edu

Bayesian Artificial Neural Networks in Health and Cybersecurity

by

Hansapani Sarasepa Rodrigo

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Mathematics & Statistics
College of Arts and Sciences
University of South Florida

Major Professor: Chris P. Tsokos, Ph.D.
Kandethody Ramachandran, Ph.D.
Lu Lu, Ph.D.
Dan Shen, Ph.D.

Date of Approval:
June 22, 2017

Keywords: Artificial Neural Networks, Bayesian Analysis, Diagnostic Models in Health,
Survival Predictions, Vulnerabilty prediction models

DEDICATION

To my loving parents Ranjith Rodrigo and K. D. Kamala,

whom sacrificed their valuable time to enlight my future.

and

To my loving husband Manoj Peiris

and

To my adorable son Neysuka Peiris

for making my life so beautiful.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Chris P. Tsokos for providing me continous support and motivations for my Ph.D. research. His patience, immense knowledge and his guidance helped me in all the time in my research. I could not have imagined having a better advisor than him for my studies.

My sincere thanks also go to Dr. Kandethody Ramachandran, Dr. Dan Shen, Dr. Lu Lu and Dr. Chad Dube for serving in my Ph.D. committee and for their continuous support, and constructive advices during the preparation of this dissertation and my study at USF.

Finally, I would like to thank my loving husband, and my parents. Without their support, I could not have achieved this target. Thank you for everything and all your support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Being in the era of Big data, the applicability and importance of data-driven models like artificial neural network (ANN) in the modern statistics have increased substantially. In this dissertation, our main goal is to contribute to the development and the expansion of these ANN models by incorporating Bayesian learning techniques. We have demonstrated the applicability of these Bayesian ANN models in interdisciplinary research including health and cybersecurity.

Breast cancer is one of the leading causes of deaths among females. Early and accurate diagnosis is a critical component which decides the survival of the patients. Including the well known "Gail Model", numerous efforts are being made to quantify the risk of diagnosing malignant breast cancer. However, these models impose some limitations on their use of risk prediction. In this dissertation, we have developed a diagnosis model using ANN to identify the potential breast cancer patients with their demographic factors and the previous mammogram results. While developing the model, we applied the Bayesian regularization techniques (evidence procedure), along with the automatic relevance determination (ARD) prior, to minimize the network over-fitting. The optimal Bayesian network has 81% overall accuracy in correctly classifying the actual status of breast cancer patients, 59% sensitivity in accurately detecting the malignancy and 83% specificity in correctly detecting non-malignancy. The area under the receiver operating characteristic curve (0.7940) shows that this is a moderate classification model.

We then present a new Bayesian ANN model for developing a nonlinear Poisson regression model which can be used for count data modeling. Here, we have summarized all the important steps involved in developing the ANN model, including the forward-propagation, backward-propagation and the error gradient calculations of the newly developed network.

As a part of this, we have introduced a new activation function into the output layer of the ANN and error minimizing criterion, using count data. Moreover, we have expanded our model to incorporate the Bayesian learning techniques. The performance our model is tested using simulation data.

In addition to that, a piecewise constant hazard model is developed by extending the above nonlinear Poisson regression model under the Bayesian setting. This model can be utilized over the other conventional methods for accurate survival time prediction. With this, we were able to significantly improve the prediction accuracies. We captured the uncertainties of our predictions by incorporating the error bars which could not achieve with a linear Poisson model due to the overdispersion in the data. We also have proposed a new hybrid learning technique, and we evaluated the performance of those techniques with a varying number of hidden nodes and data size.

Finally, we demonstrate the suitability of Bayesian ANN models for time series forecasting by using an online training algorithm. We have developed a vulnerability forecast model for the Linux operating system by using this approach.

CHAPTER 1

ARTIFICIAL NEURAL NETWORK MODELS AND FUNDAMENTAL CONCEPTS

1.1 Introduction

The idea of learning from data has been around for long time [1]. Unlike in the past, the world in thirst of finding advanced techniques which can convert just a bunch of data into valuable information.

The modern artificial intelligence (AI) is originated as a result of that. AI attempts to transfer the human thinking into a machine learning process; a way to induce new knowledge through the experience. Most of these procedures are directly derived from or inspired by the classical statistics. In fact, both AI and statistics are concerned in learning with evidence and making decisions [2]. In general, probabilistic models in classical statistics have proven to be the most effective way of formally structuring the knowledge for machine learning. Artificial neural networks combined with those probabilistic models provide a promising approach to solve different types of regression and classification problems.

1.2 General Objectives

In this dissertation, our main objective is to broaden the horizons of advanced neural networks models with the classical Bayesian learning. Bayesian learning provides several advantages over regular maximum likelihood learning in neural networks. It captures the uncertainties associated with the weight parameters and hence overcomes the problem of network overfitting. This allows us to obtain accurate predictions. Next, we summerize about the other objectives of our research.

The main goal in Chapter 2 is to develop an ANN based diagnostic model to classify the malignant breast cancer patients from non-malignant. Despite the fact that the death rates from breast cancer from all causes were the same in women who got mammograms and those who did not [3], the American Cancer Society recommends for all women ages over 40 to have annual mammograms. Our proposed model can be used as an alternative way of identifying the potential risk in women by obtaining the posterior probabilities associated with being malignant in the Bayesian setting.

In Chapter 3, we have developed a new nonlinear Poisson regression model using Bayesian ANN. In this chapter, we also evaluated the performance of three different Bayesian learning techniques called evidence procedure, Hybrid Monte Carlo sampling procedure and a newly developed Hybrid Bayesian learning method based on the complexity of the sample size and the weight space. We then tried to extend the above nonlinear Poisson regression model to develop a piecewise constant hazard model. We demonstrate that this model is useful in predicting the hazard and the survival times of the patients. Unlike the existing models, this ANN model can efficiently handle a large amount of data without suffering from data redundancy.

In Chapter 5, we have developed a nonlinear time series prediction model with a recurrent neural network using an online training algorithm. This is developed based on Hybrid Monte Carlo sampling method. With the online training, we can minimize the effect from autocorrelated data to the neural network system. This process is demonstrated by building a vulnerability prediction model for the Linux operating system. The identification of the vulnerabilities in ahead of time provides many advantages for system administrators. With that, they can allocate the necessary time and resources to avoid any major attacks from different hackers. In the final Chapter, we discuss the ways to extend our current research projects and our future research work.

1.3 Artificial Neural Networks

An Artificial Neural Network (ANN) is an information processing archetype that is inspired by the biological neural networks systems, such as the brain. They have been successfully applied in almost every field including engineering, computer science, and medicine [4–7]. The popularity of these models have increased mainly due to some inherent features of ANN. They have the strength of making predictions based on both individual attributable variables and possible complex interactions among them. Also, they serve as a powerful tool for modeling nonlinear functions and non-additive effects.

However, they also bring their challenges. The main concern is that their final results are less interpretable. Sometimes, this limitation can be overcome by building hybrid models using both neural networks and other statistical models like multiple regression, logistic regression, and multinomial logistic regression.

There are different types of neural networks depending on the network structure and the learning process, such as feed-forward, radial basis, recurrent, and Kohonen self-organizing maps. In this chapter, we discuss the implementation of feed-forward and recurrent neural networks. Under the feed-forward ANN, we specifically discuss about the multi-layer perceptron neural network.

1.3.1 Multi-Layer Perceptron

Multi-layer perceptrons (MLP) are a popular class of feed-forward networks which represent a multivariate nonlinear function mapping between a set of input vectors $\boldsymbol{x} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^N\}$ and target vectors $D = \{\boldsymbol{t}^1, \boldsymbol{t}^2, ..., \boldsymbol{t}^N\}$ [8]. These networks are organized as several interconnected layers. Each layer is a collection of artificial neurons (nodes) where connections are made among the layers without any feedback loops. MLP follows a supervised learning technique where both inputs and outputs are fed into the network for the training process. Figure 1.1 represents the architecture of a MLP with 3 layers, namely the input, hidden and output. Here we have assumed that it has d inputs, M hidden and

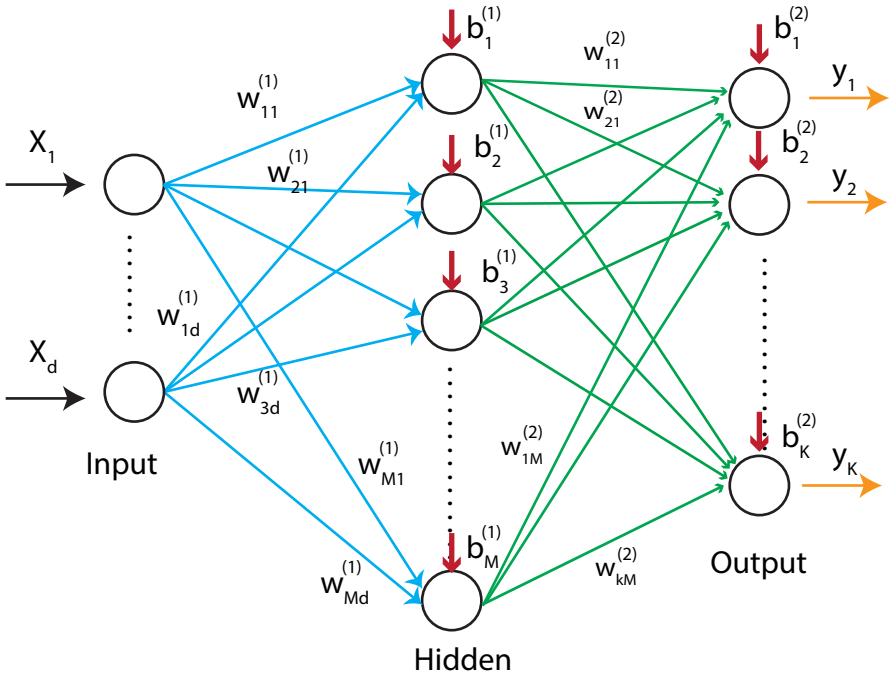

Figure 1.1: A multi-layer perceptron neural network

K output nodes. The process of obtaining the analytical function of an ANN model can be described as follows.

A weighted linear combination of d input values and the corresponding bias of the hidden unit form the input for the j^{th} hidden unit $a_j^{(1)}$, as in the Eq. (1.1).

$$a_j^{(1)} = \sum_{i=1}^{d} w_{ji}^{(1)} x_i + b_j^{(1)}, \tag{1.1}$$

here $w_{ji}^{(1)}$ is the weight associated with the input i and the hidden unit j where as $b_j^{(1)}$ is the bias associated with hidden unit j. By applying a nonlinear differentiable activation function $h(.)$ on Eq. (1.1), we get the activation of hidden unit j.

$$z_j = h(a_j^{(1)}), j = 1, 2, ..., M. \tag{1.2}$$

The most commonly used activation functions are either logistic sigmoid or hyperbolic tangent activations. In our analysis, we have used the hyperbolic tangent activation function of the form Eq. (1.3) since it has a faster convergence than the logistic function.

$$h(a_j^{(1)}) = \frac{e^{a_j^{(1)}} - e^{-a_j^{(1)}}}{e^{a_j^{(1)}} + e^{-a_j^{(1)}}} \tag{1.3}$$

A weighted combination of z_j and the corresponding bias associated with each output node $b_j^{(2)}$ form the input $a_k^{(2)}$ to each output node

$$a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + b_k^{(2)}, \tag{1.4}$$

where $k = 1, 2, ..., K$. The final outcome is obtained by applying a nonlinear transformation $g(.)$ on Eq. (1.4)

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = y_k = g(a_k^{(2)}). \tag{1.5}$$

The choice of the activation function $g(.)$ depends on the nature of the data and the distribution of target variables. For a linear regression model we assume $g(.)$ to be the identity function where as for a classification problem it is the logistic sigmoid function of the form

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{1 + exp\big(-a_k^{(2)}\big)}. \tag{1.6}$$

In the Bayesian context, the $y_k(\boldsymbol{x}, \boldsymbol{w})$ can be interpreted as the probability of membership in class C_1 given the input vector $\boldsymbol{x}$ and the probability of membership of class C_2 is given by $(1 - y_k(\boldsymbol{x}, \boldsymbol{w}))$.

In Chapter 3, we will discuss the *exponential* activation function of the form in Eq. (1.7) in detail.

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = exp\big(a_k^{(2)}\big). \tag{1.7}$$

For any of the activation functions discussed above, the final analytical form of the MLP for the k^{th} output node can be written as,

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = g(a_k^{(2)}) = g\bigg(\sum_{j=1}^{M} w_{kj}^{(2)} h\bigg(\sum_{i=1}^{d} w_{ji}{}^{(1)} x_i + b_j^{(1)} \bigg) + b_k^{(2)} \bigg). \tag{1.8}$$

This model is simply a nonlinear function from a set of input variables $\{x_i\}$ to a set of output variables $\{y_k\}$ linked with adjustable weight parameters [9],

$$\boldsymbol{w} = \{w_{11}^{(1)}, w_{12}^{(1)}, ..., w_{21}^{(2)}, w_{22}^{(2)}, .., w_{KM}^{(2)}\}.$$

It is important to note that the complexity of a neural network is directly proportional to the number of hidden nodes. It has been shown that a network with one hidden layer accompanied by sufficient number of hidden nodes is capable of approximating any continuous function [10].

1.3.2 Network Training and Error Function

Network training plays a major role when using a neural network to find solutions to a given problem. By training, we refer to finding the optimal set of weight parameters $\boldsymbol{w}$ using the training data. For that, we need to have an idea about the network error function.

Let's consider a set of independent training data $\{\boldsymbol{x}^n, \boldsymbol{t}^n\}$ with a joint probability density function $p(\boldsymbol{x}^n, \boldsymbol{t}^n)$, we can write the likelihood function

$$p(D|\boldsymbol{w}, \boldsymbol{x}) = \prod_n p(\boldsymbol{x}^n, \boldsymbol{t}^n) = \prod_n p(\boldsymbol{t}^n|\boldsymbol{x}^n) p(\boldsymbol{x}^n), \tag{1.9}$$

where $p(\boldsymbol{t}^n|\boldsymbol{x}^n)$ is the conditional density $\boldsymbol{t}$ given $\boldsymbol{x}$ and $p(\boldsymbol{x}^n)$ is the unconditional density of $\boldsymbol{x}$. We introduce an error function in the form Eq. (1.10) by taking the negative log-likelihood of the data (it is more convenient to minimize the negative log likelihood function

than maximizing the likelihood).

$$E(\boldsymbol{w}) = -ln\,(p(D|\boldsymbol{w}, \boldsymbol{x})) = -\sum_n ln\, p(\boldsymbol{t}^n|\boldsymbol{x}^n) - \sum_n ln\, p(\boldsymbol{x}^n). \tag{1.10}$$

Note that $p(\boldsymbol{x}^n)$ in Eq. (1.10) does not depend on the network parameter. Therefore, we can modify the error function,

$$E(\boldsymbol{w}) = -ln\,(p(D|\boldsymbol{w}, \boldsymbol{x})) = -\sum_n ln\, p(\boldsymbol{t}^n|\boldsymbol{x}^n). \tag{1.11}$$

This indicates that the choice of the error function depends entirely on the conditional distribution. In the next section we discuss about these conditional distributions and error functions related to both classification and linear regression. The conditional distribution related to the newly developed nonlinear Poisson regression will be discussed in Chapter 3.

1.3.2.1 Linear Regression

If we have a single target variable t_k, which follows a Gaussian distribution with a mean $y(\boldsymbol{x}^n, \boldsymbol{w})$ and precision (inverse variance) β then the conditional distribution of targets is given by

$$p(\boldsymbol{t}^n|\boldsymbol{x}^n) = \prod_{k=1}^{K} (\frac{\beta}{2\pi})^{1/2} exp\bigg(- \frac{\beta\big(y_k(\boldsymbol{x}^n, \boldsymbol{w}) - t_k^n\big)^2}{2}\bigg). \tag{1.12}$$

We get the corresponding error function by discarding the multiplcative and additive constants,

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\{y_k(\boldsymbol{x}^n, \boldsymbol{w}) - t_k^n)^2\}. \tag{1.13}$$

1.3.2.2 Classification

Let's consider a case with K separate binary classification with logistic activation function and $t_k \in \{0, 1\}$ for $k = 1, 2, ..., K$ where $t = 1$ denotes class C_1 and $t = 0$ denotes class

C_2. Then the conditional distribution of targets is given as

$$p(\boldsymbol{t}^n|\boldsymbol{x}^n) = \prod_{k=1}^{K} y_k(\boldsymbol{x}^n, \boldsymbol{w})^{t_k^n} \big[1 - y_k(\boldsymbol{x}^n, \boldsymbol{w})^{1-t_k^n}\big]. \tag{1.14}$$

The corresponding cross entropy error [11] can be obtained from Eq. (1.15)

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K}\{t_k^n \, ln \, y_k^n + (1 - t_k^n) \, ln \, (1 - y_k^n)\} \tag{1.15}$$

In each case, we then need to find the optimal weight vector (maximum likelihood solution) $\boldsymbol{w}_{ML}$ which gives the minimum $E(\boldsymbol{w})$. Due to the nonlinearity of the network function, $E(\boldsymbol{w})$ will be nonconvex and hence we only will be able to find the local minima of the error function.

1.3.3 Parameter Optimization

Finding the optimal weight vector is equivalent to finding the stationary points (in this case, local minima) in the weight space with $\nabla E(\boldsymbol{w}) = 0$. However, there exist multiple points in the weight space at which the gradient vanishes due to the nonlinear dependence on the weights and bias parameters of the error function. Therefore, we might need to compare at least several local minima in order to find a sufficiently good solution for the global minima.

Moreover, it is impossible to find an analytical solution for $\nabla E(\boldsymbol{w}) = 0$, and therefore we need to rely on iterative numerical procedures. Most of these techniques start with choosing an initial value $\boldsymbol{w}^{(0)}$, for the weight vector and then move through the weight space in successive steps like in Eq. (1.16).

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} + \Delta\boldsymbol{w}^{(\tau)}, \tag{1.16}$$

where τ is the iteration step and $\Delta\boldsymbol{w}^{(\tau)}$ is the weight vector update. The simplest approach is to use gradient descent optimization [12]. In our analysis, we have used much robust

and faster algorithms like conjugate gradient and quasi-Newton [13] for this optimization process. Many of them use the gradient information, and therefore need to evaluate $\nabla E(\boldsymbol{w})$ after each update. This will be discussed in the next section.

1.3.4 Error Back-propagation and Evaluation of Error Gradients

The back-propagation procedure allows the derivatives of an error function on the network weights and biases to be evaluated efficiently. That is to find,

$$\frac{\partial E^n}{\partial {w_{ji}}^{(1)}}, \frac{\partial E^n}{\partial {w_{kj}}^{(2)}}, \frac{\partial E^n}{\partial {b_j}^{(1)}} \text{ and } \frac{\partial E^n}{\partial {b_k}^{(2)}}.$$

This uses the chain rule of partial derivatives and leads to an algorithm in which error derivatives are propagated backward through the network starting from the output units. We discuss this process in detailed here.

Consider a three layer ANN system as given in Section 1.3.1 where the inputs of the j^{th} hidden node $a_j^{(1)}$ is given as in Eq. (1.1) and the outputs of that hidden node z_j is given as in Eq. (1.2). The corresponding input to the k^{th} output node is given in Eq. (1.4) and the final output of the ANN for a regression problem is given in Eq. (1.5) where as the final output of the binary classification problem is given in Eq. (1.6).

In either case, the first step in evaluating the error derivatives is to perform a forward propagation through the network for the complete data set in order to evaluate the activations ${z_j}^n$ of the hidden units and the activations ${y_k}^n$ of the output units for each data point n in the data set. We assume our training data are independently and identically distributed and hence the total error function can be written as a sum of n individual error functions as in Eq. (1.17).

$$E(\boldsymbol{w}) = \sum_{n=1}^{N} E^n(\boldsymbol{w}) \tag{1.17}$$

Therefore, we now discuss the backpropagation technique for each individual error function $E^n(\boldsymbol{w})$. Note that this individual error function depends on the weight $w_{ji}^{(1)}$ only via $a_j^{(1)}$.

Hence,

$$\frac{\partial E^n}{\partial {w_{ji}}^{(1)}} = \frac{\partial E^n}{\partial {a_j}^{(1)}} \frac{\partial a_j^{(1)}}{\partial {w_{ji}}^{(1)}}. \tag{1.18}$$

From Eq. (1.1), we know that $\frac{\partial a_j^{(1)}}{\partial {w_{ji}}^{(1)}} = x_i^n$ and $\frac{\partial a_j^{(1)}}{\partial {b_j}^{(1)}} = 1$. Further, using the notation, $\delta_j^{(1)n} = \frac{\partial E^n}{\partial a_j^{(1)}}$, we get

$$\frac{\partial E^n}{\partial {w_{ji}}^{(1)}} = \delta_j^{(1)n} x_i^n, \tag{1.19}$$

and

$$\frac{\partial E^n}{\partial {b_j}^{(1)}} = \delta_j^{(1)n}. \tag{1.20}$$

Note that,

$$\delta_j^{(1)n} = \sum_k \frac{\partial E^n}{\partial a_k^{(2)}} \frac{\partial a_k^{(2)}}{\partial a_j^{(1)}} = h'(a_k^{(2)}) \sum_{k=1}^{K} {w_{kj}}^{(2)} \delta_k^{(2)n}, \tag{1.21}$$

where $\frac{\partial E^n}{\partial {a_k}^{(2)n}} = \delta_k^{(2)n}$. This $\delta_k^{(2)n}$ can be calculated depending on the error function used in the problem. For example, we can use the sum of square error function for a linear regression problem where as the cross entropy error function is used for a classification problem. As we have used the corresponding canonical error functions in either case we have,

$$\delta_k^{(2)n} = {y_k}^n - {t_k}^n. \tag{1.22}$$

The derivative of the total error E can then be obtained by repeating the above steps for each data point in the training set and then summing over all data points.

$$\frac{\partial E}{\partial {w_{ji}}^{(1)}} = \sum_{n=1}^{N} \frac{\partial E^n}{\partial {w_{ji}}^{(1)}} \tag{1.23}$$

A well trained ANN is capable of making reasonable predictions to unseen data, which is known as generalization. This is achieved by incorporating the regularization term, α,

to the error function as in Eq (1.24) which is known as a weight decay [8].

$$\tilde{E}(\boldsymbol{w}) = E(\boldsymbol{w}) + \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} \tag{1.24}$$

The value of α is usually determined by using cross validation techniques which involves reserving a validation dataset to evaluate the error $\tilde{E}(\boldsymbol{w})$ of models trained using a range of different values of α and selecting the value of α with the smallest $\tilde{E}(\boldsymbol{w})$.

1.4 Bayesian Neural Networks

Bayesian neural networks provide a more intuitive approach for network training. A significant amount of research in this area was conducted by David Mackay in 1992 [14, 15]. In the maximum likelihood method, we find a single set of weight parameters by minimizing the error function. In contrast to that, in the Bayesian approach, a probability distribution is used to capture the uncertainties associated with the weight parameters [8].

Use of Bayesian learning in ANN provides several advantages. In fact, the use of regularization parameters can be given a natural interpretation. Moreover, it allows of using a relatively large number of regularization parameters while they can be optimized during the training process. The automatic relevance determination [16–18] prior helps to identify the relative importance of the input variables. Additionally, we can create error bars to the regression problems when making the new predictions. We also can improve the prediction accuracies by creating network committees after combining different networks.

In this method, we first introduce a prior distribution $p(\boldsymbol{w})$ for the weights where it represents our knowledge of the weight parameters before observing the data. Once we observe the data, the Bayes' theorem is used to update our beliefs and the posterior probability density $p(\boldsymbol{w}|\mathcal{D}, \boldsymbol{x})$ of the weight parameters is obtained.

$$p(\boldsymbol{w}|\mathcal{D}, \boldsymbol{x}) = \frac{p(\mathcal{D}|\boldsymbol{w}, \boldsymbol{x})p(\boldsymbol{w})}{p(\mathcal{D}|\boldsymbol{x})} \tag{1.25}$$

Here, $p(\mathcal{D}|\boldsymbol{w}, \boldsymbol{x})$ is the likelihood function, and $p(\mathcal{D}|\boldsymbol{x})$ is the normalization factor which is given by,

$$p(D|\boldsymbol{x}) = \int p(D|\boldsymbol{w}, \boldsymbol{x}) p(\boldsymbol{w}|\boldsymbol{x}) d\boldsymbol{w}. \tag{1.26}$$

We then use this posterior distribution to make inferences. That is to make new predictions based on

$$p(t|\boldsymbol{x}^*, \mathcal{D}) = \int p(t|\boldsymbol{x}^*, \boldsymbol{w}) p(\boldsymbol{w}|D, \boldsymbol{x})\, d\boldsymbol{w} \tag{1.27}$$

1.4.1 Prior and Posterior Distribution of the Weight Parameters

In this section, we discuss the process of obtaining the posterior distribution for the weight parameters. From Eq. (1.25), it is clear that we need to have a prior distribution for the weights and the likelihood of the data in order to obtain the posterior distribution. In our analysis, we consider a zero mean Gaussian prior of the form Eq. (1.28),

$$p(\boldsymbol{w}|\boldsymbol{x}) = \frac{1}{Z_w(\alpha)} exp(-\alpha E_p(\boldsymbol{w})) = \frac{1}{Z_w(\alpha)} exp(-\frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w}), \tag{1.28}$$

where $Z_w = (\frac{2\Pi}{\alpha})^{w/2}$ and α is the inverse variance of the distribution, also known as the hyperparameter of the prior distribution. As a part of Bayesian learning, we can optimize this hyperparameter (This will be discussed in Section 1.4.2). The error term $E_p(\boldsymbol{w})$ is chosen to be $\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$, as it penalizes the weights of large magnitudes for a better generalization. This is same as having a weight decay regularizer as in Eq. (1.24).

The posterior probability distribution for weights can be determined according to the Bayes' theorem by incorporating the prior of Eq. (1.28) and the corresponding log likelihood $p(\mathcal{D}|\boldsymbol{w}, \boldsymbol{x})$ of the data.

$$P(\boldsymbol{w}|\mathcal{D}, \boldsymbol{x}) = \frac{1}{Z_s} exp[-(ln\ P(\mathcal{D}|\boldsymbol{w}, \boldsymbol{x}) + \alpha E_p(\boldsymbol{w}))] = \frac{1}{Z_s} exp[-S(\boldsymbol{w})], \tag{1.29}$$

where Z_s is the normalization constant of the posterior distribution and $S(\boldsymbol{w})$ is the regularized cost function. Due to the analytical difficulties in evaluating the above posterior,

we introduce a Gaussian approximation to this posterior distribution. For that, we first need to find the most probable weight vector $\boldsymbol{w}_{MAP}$, by minimizing the regularized cost function $S(\boldsymbol{w})$ using the standard nonlinear optimization algorithms such as conjugate gradients (Here we have assumed that α is known).

Having found a mode $\boldsymbol{w}_{MAP}$, we then build a local Gaussian approximation by evaluating the matrix of second derivatives of the negative log posterior distribution.

$$\boldsymbol{A} = -\nabla\nabla \, ln \, p(\boldsymbol{w}|\mathcal{D}, \boldsymbol{w}). \tag{1.30}$$

The corresponding Gaussian approximation to the posterior is then given by,

$$q(\boldsymbol{w}|\mathcal{D}, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_{MAP}, \boldsymbol{A}^{-1}). \tag{1.31}$$

To make prediction at a new input vector $\boldsymbol{x}^*$ for a regression problem, we need to calculate the predictive distribution,

$$p(t|\boldsymbol{x}^*, \mathcal{D}) = \int p(t|\boldsymbol{x}^*, \boldsymbol{w}) q(\boldsymbol{w}|\mathcal{D}, \boldsymbol{x}) \, d\boldsymbol{w}. \tag{1.32}$$

For a classification problem, the probability that a new input vector belongs to class C_1 as in Eq. (1.33). Though this prediction is not directly achievable, we can use the marginalized predictions to obtain the results as suggested by MacKay [14].

$$p(C_1|\boldsymbol{x}^*, \mathcal{D}) = \int (C_1|\boldsymbol{x}^*, \boldsymbol{w}) q(\boldsymbol{w}|\mathcal{D}, \boldsymbol{x}) dw = \int y(\boldsymbol{x}, \boldsymbol{w}) q(\boldsymbol{w}|\mathcal{D}, \boldsymbol{x}) dw. \tag{1.33}$$

1.4.2 The Evidence Procedure and the Automatic Relevance Determination Prior

The evidence procedure is an iterative algorithm for determining the optimal weights and hyperparameters. Instead of integrating over all unknown hyperparameters, this searches for optimal hyperparameters. Evidence procedure has given good results on many

applications [19] and less computationally costly compared to other Bayesian approaches. This process is summarized here.

The posterior distribution of the weight parameters can be rewritten by highlighting dependency of that on its hyperparameters,

$$p(\boldsymbol{w}|\mathcal{D},\boldsymbol{x}) = \int\int p(\boldsymbol{w},\alpha,\beta|\mathcal{D},\boldsymbol{x})\, d\alpha\, d\beta = \int\int p(\boldsymbol{w},\alpha,\beta,\mathcal{D},\boldsymbol{x}) p(\alpha,\beta|\mathcal{D},\boldsymbol{x})\, d\alpha\, d\beta. \quad (1.34)$$

Under the evidence procedure, we assume that the posterior density of the hyperparameters $p(\alpha,\beta|\mathcal{D})$ is sharply peaked around the most probable values of those hyperparameters α_{MAP} and β_{MAP}. That is we use the Laplace approximation. With that we have,

$$p(\boldsymbol{w}|\mathcal{D},\boldsymbol{x}) \approx p(\boldsymbol{w}|\alpha_{MAP},\beta_{MAP},\mathcal{D},\boldsymbol{x}) \int\int p(\boldsymbol{w},\alpha,\beta|\mathcal{D},\boldsymbol{x})\, d\alpha\, d\beta. \quad (1.35)$$

Therefore, the first step in the evidence procedure is to evaluate the posterior distribution of hyperparameters by approximating with the most probable values of hyperparameters. Using Bayesian inference, the posterior distribution of the hyperparameters can be obtained by

$$p(\alpha,\beta|\mathcal{D},\boldsymbol{x}) = \frac{p(\mathcal{D}|\alpha,\beta,\boldsymbol{x}) p(\alpha,\beta|\boldsymbol{x})}{p(\mathcal{D}|\boldsymbol{x})}. \quad (1.36)$$

For the rest of the analysis, we assume that $p(\alpha,\beta|\boldsymbol{x})$ to be uniform and $p(\mathcal{D}|\boldsymbol{x})$ is ignored as we are only interested in the peaks of this density. Therefore, we only need to maximize $p(\mathcal{D}|\alpha,\beta,\boldsymbol{x})$,that is the evidence of the hyperparameters

$$p(\mathcal{D}|\alpha,\beta,\boldsymbol{x}) = \int p(\mathcal{D}|\boldsymbol{w},\alpha,\beta,\boldsymbol{x}) p(\boldsymbol{w}|\alpha,\beta,\boldsymbol{x})\, d\boldsymbol{w} = \frac{1}{Z_D(\beta)} \frac{1}{Z_W(\alpha)} \int exp(-S(\boldsymbol{w}))\, d\boldsymbol{w}. \quad (1.37)$$

We then take the log of the evidence and optimize it with respect to α and β. This procedure can be repeated for each local minimum, and α and β can be re-estimated using the,

$$\alpha^{new} = \frac{\gamma}{2E_p(\boldsymbol{w})}$$
$$\text{and} \tag{1.38}$$
$$\beta^{new} = \frac{N-\gamma}{2E(\boldsymbol{w})}.$$

where $\gamma = \sum_{i=1}^{W} \frac{\lambda_i}{\lambda_i+\alpha}$ and $\lambda_1, ..., \lambda_W$ are the eigen values of the Hessain matrix of the regularized cost function $S(\boldsymbol{w})$.

Having found the α_{MAP} and β_{MAP} using the evidence procedure, we can approximate the regularized cost function using the second-order Taylor series expansion around the most probable weight vector $\boldsymbol{w}_{MAP}$,

$$S(\boldsymbol{w}) \approx S(\boldsymbol{w}_{MAP}) + \frac{1}{2}(\boldsymbol{w}-\boldsymbol{w}_{MAP})^T \boldsymbol{A}(\boldsymbol{w}-\boldsymbol{w}_{MAP}). \tag{1.39}$$

This is used for the Gaussian approximation to the posterior distribution

$$q(\boldsymbol{w}|\mathcal{D}, \boldsymbol{x}) = \frac{1}{Z_s^*} exp(-S(\boldsymbol{w}_{MAP}) - \frac{1}{2}\Delta\boldsymbol{w}^T \boldsymbol{A}\Delta\boldsymbol{w}). \tag{1.40}$$

In the Bayesian setting, we can associate a separate hyperparameter to each input variable representing the inverse variance of the prior distribution of the weights fanning out from that input [20]. Optimal values for these hyperparameters can then be obtained using the evidence procedure. So the weights connected to the irrelevant inputs are automatically set to small values. This is known as the automatic relevance determination (ARD) prior.

1.4.3 Hybrid Monte Carlo Method

In the evidence procedure, we have used several approximations to get the posterior weight distribution and to optimize the hyperparameters. R. M. Neal in 1994 [18] proposed a Bayesian learning method based on Hybrid Monte Carlo sampling. Here, he has suggested

approximating the predictive distribution given in Eq. (1.32) by a finite sum of the form,

$$\langle p(t) \rangle = p(\boldsymbol{t}|\boldsymbol{x}, \mathcal{D}) \simeq \frac{1}{M} \sum_{m=1}^{M} p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}_m), \tag{1.41}$$

where $\{\boldsymbol{w}_m\}$ represents a sample of weight vectors generated from the posterior distribution. We also can obtain a statistical error estimate for our predictions by considering the variance of this statistic,

$$SE = \sqrt{\frac{\langle p(t)^2 \rangle - (\langle p(t) \rangle)^2}{N}}, \tag{1.42}$$

With this one can construct the confidence interval which represents one standard deviation around the expected value of the statistic.

Hybrid Monte Carlo method of sampling uses the information of gradients which makes it ideal for neural networks. Furthermore, the accuracy of the above estimator does not depend on the dimensionality of $\boldsymbol{w}$ and therefore high accuracy can be achievable with a relatively small number of samples.

However, in reality, samples $\{\boldsymbol{w}_m\}$, might not be independent and therefore we might need relatively large samples. Most importantly, the posterior distribution depends on the prior distribution of the weights as well as the selection of hyperparameter values. If we do not choose an informative prior based on our data and different values of hyperparameters, this method might require a huge number of samples to achieve a sufficient accuracy.

1.4.4 Network Committees

It is recommended in literature to train several networks with different random initial weight configurations. This will avoid the cost function being stuck in a local minimum. We can simply form a committee of networks by combining these networks. Suppose we have a set of L trained network models where $i = 1, 2, ..., L$. These networks may have different number of hidden units, or networks with the same architecture but trained to different local minima of the cost function.

The simplest form of a committee, which involves taking the average predictions of the outputs of the L networks, is given by Eq. (1.43). This will improve the accuracy of the predictions over an individual network output [20]. That is,

$$y_{com}(\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{L} \sum_{i=1}^{L} y_i(\boldsymbol{x}, \boldsymbol{w}). \tag{1.43}$$

1.5 Sparse Kernel Methods

The kernel concept was first introduced by the Aizerman *et* al. [21] into the field of pattern recognition in 1964. A kernel function for a model with fixed feature space mapping $\phi(\boldsymbol{x})$ is defined to be, $k(\boldsymbol{x}, \boldsymbol{x}') = \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{x}')$, and it is a symmetric function of its arguments.

Usually, kernel methods use set of all training data to obtain either a point estimate of the parameter vector or to determine a posterior distribution over the above vector during the training phase. Therefore, it takes a significant amount of computational time to make the predictions for unseen data, based on the learned parameter vector $\boldsymbol{w}$.

In contrast to the above, sparse kernel methods use only a subset of the training data [9] to make the predictions. Here, we discuss about two of such models support vector machine and relevance vector machine.

1.5.1 Support Vector Machine

Support vector machine (SVM) [22, 23] have been widely used to solve classification problems by constructing an optimal separating hyperplane in a feature space. They are important mainly because of several reasons. One reason is being robust to very large number of variables and small samples and another reason is they can build both simple and highly complex classification models. However, unlike ANN and RVM, SVM does not provide any posterior probabilities.

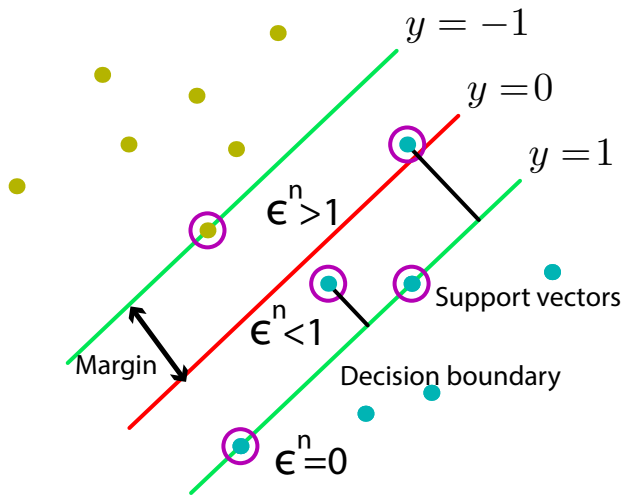

Figure 1.2: Illustration of the margin, decision boundary and support vectors

For a two-class classification problem with a linear model of the form,

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \Phi(\boldsymbol{x}) + \boldsymbol{b}, \tag{1.44}$$

where $\Phi(\boldsymbol{x})$ is the feature space mapping introduced in the previous section. If the training data set is linearly separable in $\Phi(\boldsymbol{x})$, then there exist at least one possible solution for $\boldsymbol{w}$ and $\boldsymbol{b}$ such that $y(\boldsymbol{x}^n) > 0$ for the class with $t^n = +1$ and $y(\boldsymbol{x}^n) < 0$ for the other class with $t^n = -1$, so that $t^n y(\boldsymbol{x}^n) > 0$ for all training data $n = 1, 2, .., N$.

If there are more than one solution, then we find a solution with the smallest generalization error. SVM handles this through a concept called "margin". The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown in the Fig. 1.2. Maximizing the margin leads to a particular choice of decision boundary. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles. With this concept, we can minimize an error function where it gives an infinite error when a data point is misclassified and a zero error for correctly classified data.

When the training data are not linearly separable, slack variables $\epsilon^n \geq 0$ are introduced. For the data points which are on or inside the correct margin boundary, we have $\epsilon^n = 0$ and for the other data points we have, $\epsilon^n = |t^n - y(\boldsymbol{x}^n)|$. Thus, a data point that is on the decision boundary $y(\boldsymbol{x}^n) = 0$ has $\epsilon^n = 1$. We can now define a new classification as $t^n y(\boldsymbol{x}^n) \geq 1 - \epsilon^n$. Data points for which $\epsilon^n = 0$ are correctly classified. Points for which $0 < \epsilon^n \leq 1$ lie inside the margin, but on the correct side of the decision boundary, and those data points for which $\epsilon^n > 1$ lie on the wrong side of the decision boundary and are misclassified as illustrated in Fig. 1.2. Hence, our goal is to maximize the margin while softly penalizing the points that lie on the wrong side of the margin boundary by minimizing,

$$C \sum_{n=1}^{N} \epsilon^n + \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w}, \tag{1.45}$$

where the parameter C controls the trade-off between the slack variable penalty and the margin.

1.5.2 Relevance Vector Machine

Relevance vector machine is based on the Bayesian formulation [24] and provides posterior probabilistic outputs by applying Bayesian treatment to Eq. (1.43). RVM provides equivalent generalization performance like SVM, but they utilize dramatically fewer kernel functions and hence provide much sparser solutions than SVM.

In RVM, a prior over the model weights, one associated with each weight governed by a set of hyperparameters, is introduced and their most probable values are iteratively estimated from the data. In practice when that the posterior distributions of many of the weights are sharply peaked around zero, sparsity is achieved. Those training vectors associated with the remaining non-zero weights are known as "relevance vectors", in reference to the principle of automatic relevance determination prior.

CHAPTER 2

AN EFFECTIVE DIAGNOSTIC ARTIFICIAL NEURAL NETWORK MODEL FOR BREAST CANCER

2.1 Introduction

Breast cancer is the second most fatal disease in women worldwide. Self-awareness and evaluation of breast cancer risk play important roles in detecting cancer in its early stages. Including the well-known "Gail model" [25], some other statistical models have been proposed to assess the risk of being diagnosed with breast cancer [26–28]. However, these models imposed some limitations on their use of risk prediction [29, 30]. Usually, physicians advice women who are 40 years or older to do an annual mammogram screening test for their benefits. However, in addition to their cost in thousand dollars, there is a significant controversy about the usefulness of mammograms [3]. Typically, the sensitivity of a mammogram (probability of correctly identifying a malignant lump) varies between 68% and 79% [31].

The primary objective of our study is to develop a better statistical model to correctly classify the malignant breast cancer patients with their demographic factors and previous mammogram results using a Bayesian artificial neural network (ANN) model. This allows us to successfully evaluate the probability of diagnosing with malignant breast cancer [32]. Moreover, we can find out the relevance importance of risk factors for the network predictions. Hence, women can use our proposed model as a decision supportive system, before proceeding to their next annual mammogram or at least along with that.

In addition to the proposed neural network model, we have built several other diagnosis models with support vector machine (SVM), relevance vector machine (RVM), and some hybrid models, combining several of them above. We present their discrimination capacities

based on the accuracies, specificities, sensitivities and the area under the receiver operating characteristic curve (AUC).

2.2 Literature Review

The statistical models which have been developed in the past on the breast cancer can be classified into two main groups: diagnosis and prognostic models. Each type of model serves for different purposes [33], a diagnosis model is used to identify the malignant and nonmalignant cancerous patterns where a prognosis model is used for prediction of future development of cancerous cell [31].

Detecting malignancy in breast cancer using a neural network with mammogram data has experimented in several studies [4–7, 34]. The popularity of ANN in health related problems has been increased rapidly over the past decades due to its capability of identifying complicated patterns inherited in the patients' data.

Janghel *et al.* [35] have used neural networks to develop a diagnosis model using a back-propagation algorithm, with a 52% overall accuracy. Ayer *et al.* [4] have quantified the breast cancer risk using an ANN model, and have obtained a significant improvement in their model accuracy. Utomo *et al.* [34] have used extreme learning machine neural network to develop a better generalization classifier model than the commonly used gradient based ANN. Most of other studies have compared the accuracies between ANN and the classical methods like logistic regression [36–38]. The majority of them have reported similar performance between those two, while some have reported that one or the other model performed better depending on their data.

Singh et al. [39] have used Bayesian regularization techniques in developing a breast cancer diagnosis model using ANN. However, none of the existing studies have utilized the evidence approach or the automatic relevance determination prior which lead to a minimum network overfitting. Those approaches provide efficient solutions for the problems that has with some medical statisticians including misuses of neural networks with significant network overfitting [40–42] and identifying the importance of risk factors [43].

Several other studies [44–46] have found that the support vector machines provide higher prediction accuracies than any other data mining techniques including ANN and Bayesian network. However, SVM does not provide class probabilities in classifications. Though there exist a heuristic approach to map non-probabilistic SVM outputs to probabilities via a logit function [47], it fails to provide much insight. Though RVM provides posterior class probabilities and has better sparse property, generalization ability, and decision speed [48, 49], it also has almost an equal training efficiency and a classification accuracy as SVM.

2.3 Methodology

We have used a multi-layer perceptron neural network to develop the proposed breast cancer diagnosis model. The model is trained based on demographic risk factors and previous mammogram results from the white women. The corresponding posterior class probabilities of malignancy for each woman is obtained as the outcome. The developed ANN model with one hidden layer is represented in Fig. 2.1. The final analytical form of the output is given by Eq. (2.1),

$$y(\boldsymbol{x}, \boldsymbol{w}) = g(a) = g\Bigg(\sum_{j=0}^{M} w_{1j}^{(2)} h\Bigg(\sum_{i=0}^{d} {w_{ji}}^{(1)} x_i + b_j^{(1)} \Bigg) + b_1^{(2)} \Bigg). \tag{2.1}$$

Here, $x_1, x_2, ..., x_d$ are the risk factors and y is the posterior class probability of falling into malignant breast cancer class. The hyperbolic tangent and the logistic sigmoid activation functions are selected as the h and g respectively. The network training is performed by minimizing the cross entropy error [Eq. (1.15)] or the evidence function (Eq. [1.37)] which we have discussed in detail in Chapter 1.

2.3.1 Study Population

The data for this study are taken from the breast cancer surveillance consortium [50] for the period 1996 to 2002. For each white woman, information on her menopausal type, age,

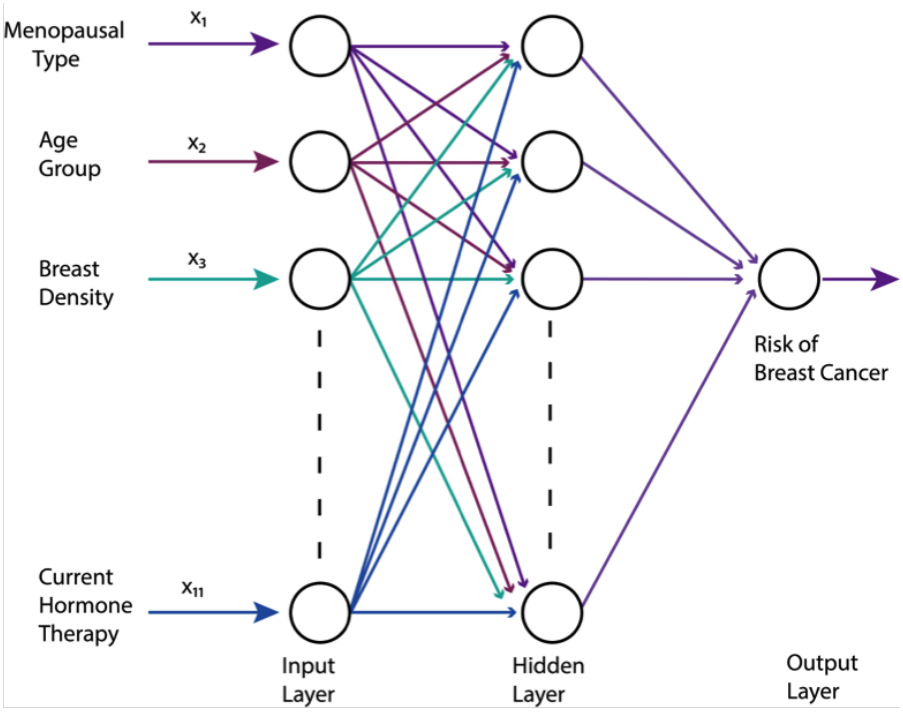

Figure 2.1: The proposed ANN model for breast cancer diagnosis

breast density, ethnicity (Hispanic), body mass index (BMI), age at first birth, personal or family history of breast cancer, prior breast procedures, results of the last mammogram, type of menopause and current hormone therapy are taken into consideration. Ages of the women in this study vary from 35 to 84 years, and specific details are given in Table 2.1.

2.3.2 Implementation of the ANN Model

Training and testing data sets were created by partitioning the whole data set into two parts each with 75% and 25% of the data. A random sample out of the non-malignant group from the training set is selected and merged with the malignant group in order to obtain a balanced training set as given in Table 2.2.

We trained different ANN models using both standard (maximum likelihood method) and Bayesian approaches with a varying number of hidden nodes from 1 to 25. The scaled

Table 2.1: Details of the study population

			Malignant	Malignant(%)	Non Malignant	Non Malignant (%)	Total	Total(%)
	Total		1053	6.47	15218	93.53	16271	100
1	Menopausal Type (X1)							
		Premenopausal	227	21.56	2882	18.94	3109	19.11
		Postmenopausal	826	78.44	12336	81.06	13162	80.89
2	Age Group (X2)							
		35-39	6	0.57	496	3.26	502	3.09
		40-44	72	6.84	788	5.18	860	5.29
		45-49	137	13.01	2355	15.48	2492	15.32
		50-54	168	15.95	2695	17.71	2863	17.6
		55-59	150	14.25	1872	12.3	2022	12.43
		60-64	141	13.39	1663	10.93	1804	11.09
		65-69	131	12.44	1533	10.07	1664	10.23
		70-74	96	9.12	1477	9.71	1573	9.67
		75-79	93	8.83	1343	8.83	1436	8.83
		80-84	59	5.6	996	6.54	1055	6.48
3	Breast Density (X3)							
		Almost entirely fat	31	2.94	2575	16.92	2606	16.02
		Scattered fibroglandular densities	405	38.46	5319	34.95	5724	35.18
		Heterogeneously dense	506	48.05	4993	32.81	5499	33.8
		Extremely dense	111	10.54	2331	15.32	2442	15.01
4	Hispanic (X4)							
		No	1026	97.44	12476	81.98	13502	82.98
		Yes	27	2.56	2742	18.02	2769	17.02
5	Body Mass Index (X5)							
		10-24.99	432	41.03	4969	32.65	5401	33.19
		25-29.99	326	30.96	4404	28.94	4730	29.07
		30-34.99	181	17.19	3304	21.71	3485	21.42
		35 or more	114	10.83	2541	16.7	2655	16.32
6	Age at First Birth (X6)							
		Age<30	692	65.72	7654	50.3	8346	51.29
		Age 30 or greater	154	14.62	3412	22.42	3566	21.92
		Nulliparous	207	19.66	4152	27.28	4359	26.79
7	Number of first degree relatives with breast cancer (X7)							
		Zero	763	72.46	8515	55.95	9278	57.02
		One	252	23.93	5077	33.36	5329	32.75
		Two or more	38	3.61	1626	10.68	1664	10.23
8	Previous breast procedure (X8)							
		No	716	68	8925	58.65	9641	59.25
		Yes	337	32	6293	41.35	6630	40.75
9	Result of last mammogram before the index mammogram (X9)							
		Negative	1032	98.01	13244	87.03	14276	87.74
		False positive	21	1.99	1974	12.97	1995	12.26
10	Surgical menopause (X10)							
		Natural	576	54.7	7000	46	7576	46.56
		Surgical	250	23.74	5336	35.06	5586	34.33
		Unknown	227	21.56	2882	18.94	3109	19.11
11	Current hormone therapy(X11)							
		No	400	37.99	6382	41.94	6782	41.68
		Yes	426	40.46	5954	39.12	6380	39.21
		Unknown or not menopausal	227	21.56	2882	18.94	3109	19.11

Table 2.2: Summary of the training and testing data

Data	**Malignant**	**Non-Malignant**	**Total**
Train	829	1658	2487
Test	224	3030	3254
Total	1053	4688	5741

conjugate gradient algorithm is used for network training as it automatically adjusts the learning rate with a faster learning [51].

Neural networks in the standard setting are trained using a 10-fold cross validation method, both with and without a weight regularization. In the 10-fold cross validation, the training set is divided into ten distinct segments where 9 of them are used to train the network, and the remaining segment is used for validation. This process is repeated for each of the ten possible choices of the segments which are omitted from the training process, and the validation errors (cross entropy error) are averaged over all ten segments. The best network in this approach is the one with the smallest average cross entropy in the validation data set [52].

Under the Bayesian approach, we trained another three types of networks with different weight regularization techniques. The first two networks were trained using Bayesian evidence procedure [8], one without and the other with automatic relevance determination prior. For both of the above types, ten different networks were trained with ten different random initializations to examine the effect of local minima on solutions, and these were taken to construct the network committees. The optimal ANN model with the highest average log evidence from each committee is then selected to predict the posterior probability of malignancy. In addition to above, another simple neural network with one hidden node was built, and this is functionally equivalent to a logistic regression model.

The final Bayesian ANN model, along with the evidence process and ARD prior is developed using a 10-fold cross validation method. Here also, we built several ANN models by varying the number of hidden nodes and the best ANN is selected based on the minimum regularized cost function.

2.3.3 Model Evaluation

The optimal ANN models in each case are evaluated based on their accuracy, sensitivity, specificity values and AUC values for the testing data [53, 54]. The proportions of correctly identified malignant and non-malignant women from the ANN models are known as the model "accuracies". The proportions of actual malignant patients who are correctly identified from the models are known as the "sensitivities, " and the proportions of non-malignant women who are correctly identified from the models are known as the "specificities." A perfect desirable predictor would be described as 100% sensitive (that is predicting all people from the malignant group as malignant) and 100% specific (that is predicting all non-malignant people as nonmalignant). However, for any test, there is usually a trade-off between these two measures and this can be represented graphically by the receiver operating characteristic curve.

A summary of our six optimal neural networks is given in Table 2.3. According to that, the overall accuracy of the logistic neural network (6^{th} MLP) is lower than all the other models except for the ANN trained without the ARD prior. Moreover, it has the second lowest sensitivity and specificity values with the highest error. However, these models are not directly comparable with respect to their errors, as they have different settings and different training samples.

Table 2.3: Classification summary of the ANN models

No	ANN Model	Error(Cross Entropy/Cost)	Accuracy	Sensitivity	Specificity
MLP 1	Standard ANN without a weight regularization	641.96(valid error 16.50)	78.43%	55.36%	80.13%
MLP 2	Standard ANN with a weight regularization	434.77(valid error 8.28)	74.09%	53.57%	75.61%
MLP 3	ANN with evidence, but without ARD prior	548.63	72.99%	60.71%	73.89%
MLP 4	ANN with both evidence and ARD prior	582.28	74.15%	59.82%	75.21%
MLP 5	ANN with evidence and ARD prior along with cross validation	908.78	81.35%	59.38%	82.97%
MLP 6	ANN with one hidden node (logistic)	1123.1	73.11%	55.35%	74.42%

Out of these ANN models, the best network with respect to the highest accuracy and specificity is found to be the ANN trained using the evidence procedure and ARD

prior along with a cross validation (5^{th} MLP). As can be seen, use of evidence procedure and the ARD prior have always resulted in better sensitivities. However, use of weight regularization without any optimization (evidence process) does not provide any significant improvement over the standard network training process.

After a careful investigation of our results, we can conclude that the use of weight regularization techniques along with the evidence and the cross-validation processes provide better results in Bayesian classification, for most of the time. Apart from that, use of the ARD prior helps to identify the most contributing variables in the model. Also, by forming committees, we were able to reduce the network training error. Therefore, we prefer Bayesian learning methods over the standard method of training the neural networks. We can see that the minimum and maximum prediction accuracies from these ANN models are 73% and 81%, respectively. Sensitivity values are varying from a minimum of 54% up to a maximum of 61% while specificity values are varying from 74% to 83%. The AUC

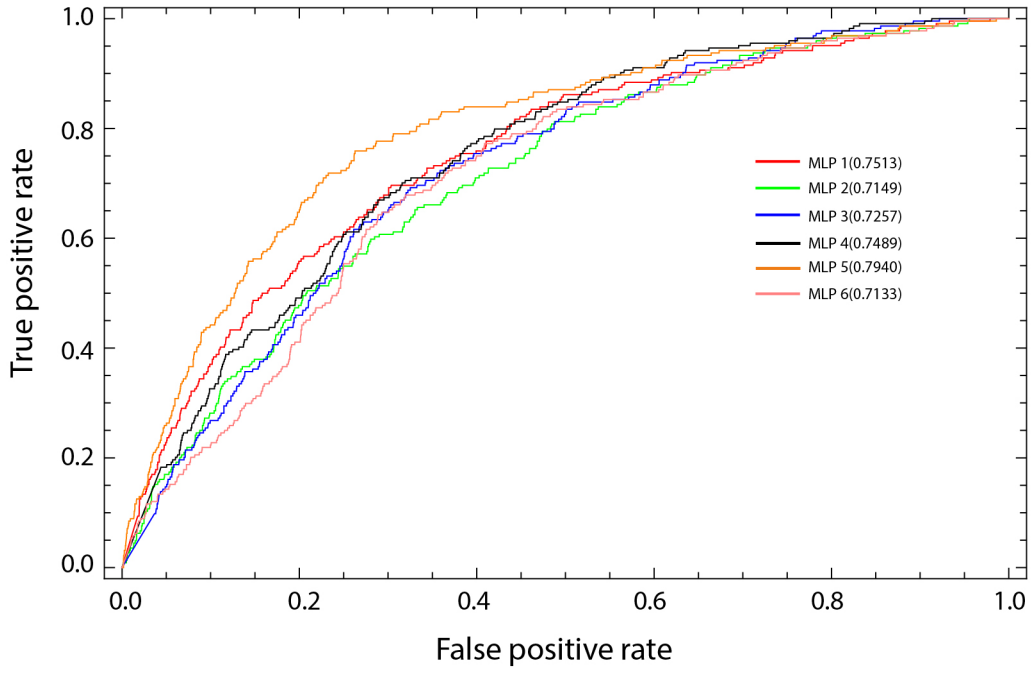

Figure 2.2: The receiver operating characteristic curves

values of all the above ANN models are greater than the AUC values of all the above ANN models are greater than 70%, which implies a moderate classification model. Figure 2.2 represents the receiver operating characteristic curves with the corresponding AUC values. In the end, we obtained the posterior probabilities of malignancy from the best Bayesian ANN model.

The relevance importance of the inputs identified by the ARD prior, depending on their eventual hyperparameter values is presented in Table 2.4. Risk factors with smaller hyperparameters correspond to a large variance prior and hence allow weights of large magnitude. Such variables are highly contributing to the model outcome. As can be seen, being in the age group 75-79 is the most critical factor in diagnosing with the malignant breast cancer. Having a prior false-positive mammogram can be an indication of malignant breast cancer. In accordance with the cancer literature [50], we found that the risk factors like, having heterogeneously or extremely dense breast densities, and having a BMI of "35 or more" are significantly contributing to the model.

2.4 SVM, RVM and Ensemble Modeling

To evaluate our ANN model performance over the other classification techniques, we implemented classification models with SVM and RVM methods. Further, some ensemble classification models were constructed by combining the best Bayesian network with SVM and RVM models.

Table 2.5 summarizes the classification details for each method. From that, we can see that the SVM has outperformed our best ANN method in all three evaluation aspects. Also, the ensemble method with ANN and SVM tends to give the second best accuracy values where as ANN alone slightly inferior to both of them. We have used penalized logistic regression to make the ensemble models. This matches with the general findings that the SVM performs better for classification problems. However, the disadvantage of using SVM is that it does not provide the actual posterior class probabilities like an ANN model. The RVM model, which is the Bayesian version of SVM, did not outperform the

Table 2.4: Relative importance of the risk factors based on the ARD prior

Rank	Alpha (hyperparameter)	Variable	Risk Group
1	0.3841	agegrp9	Age group 75-79
2	0.555	lastmamm	Result of last mammogram before the index mammogram - False positive
3	0.6489	density3	Density - Heterogeneously dense
4	0.6846	density4	Density - Extremely dense
5	0.8251	bmi4	35 or more
6	1.3072	agegrp2	Age group 40-44
7	1.3872	agegrp7	Age group 65-69
8	1.6989	hispanic	Hispanic or not - Yes
9	1.7403	nrelbc2	Number of first degree relatives with breast cancer - 2 or more
10	1.951	hrtYes	Current hormone therapy ? Yes
11	2.0528	agegrp10	Age group 80-84
12	2.0826	bmi2	25-29.99
13	2.198	agegrp8	Age group 70-74
14	2.2112	hrtNo	Current hormone therapy - No
15	2.8161	agegrp6	Age group 65-69
16	2.9341	bmi3	30-34.99
17	3.2299	agegrp5	Age group 55-59
18	3.652	nrelbc1	Number of first degree relatives with breast cancer - One
19	3.7138	surgnatural	Surgical menopause - Natural
20	4.2249	agegrp4	Age group 50-54
21	5.0616	surgsurgical	Surgical menopause - Surgical
22	5.1547	brstproc	Previous breast procedure - Yes
23	5.7224	density2	Density - Scattered fibroglandular densities
24	7.2989	menopaus	Postmenopausal or age>=55
25	10.1388	agenulli	Age at first birth - Nulliparous
26	10.5538	agegrp3	Age group - 45-49
27	11.4664	agegreater30	Age at first birth - Age 30 or greater

other models. Therefore, we stick to the best ANN model to preserve our original goal of finding posterior class probabilities of malignancy.

Table 2.5: Summary of the classification methods

Method	**Accuracy**	**Sensitivity**	**Specificity**
ANN	81.35%	59.38%	82.97%
SVM	85.46%	60.27%	87.33%
RVM	78.86%	54.02%	80.69%
ANN+SVM	82.67%	58.48%	84.46%
ANN+RVM	81.90%	52.68%	84.06%
ANN+SVM+RVM	81.32%	51.79%	83.50%

2.5 Conclusions and Contributions

In the present research, we have created different ANN models using both standard and Bayesian analytics. The Bayesian ANN models outperformed the standard ANN models. Additionally, it provides the posterior probabilities for a classification, and that can be used as a priori risk of diagnosing breast cancer. Moreover, we can use the evidence procedure for network regularization along with the ARD prior. By applying those two analytics along with cross-validation procedure, we were able to achieve a significant difference in the accuracy of our neural network models. By using network committees, we were able to significantly improve our prediction accuracies due to the lower variances.

The highest accuracy was obtained from one of the Bayesian ANN, and it is about 81%. This is a significant improvement over the other methods which used for the same set of real data with respect to the discriminative accuracy. ROC curve provides information about a model's classification efficiency. We were able to get a good classification model with the third and the fifth ANN model where it gives an AUC, more than 75%. Relevance importance of the risk factors were obtained with the aid of the ARD prior, which is very useful information for any women with certain risk factors.

We have introduced a breast cancer diagnosis model using artificial neural network analytics. This can be used as a decision supportive system in evaluating the potential risk of diagnosing a woman with malignant breast cancer.

In this study, we were able to accomplish the following goals.

1. We have developed an effective diagnosis model for potential breast cancer patients using ANN.

2. Unlike the existing diagnosis models, we were able to increase the validity of the proposed ANN model by incorporating the Bayesian regularization analytics.

3. White women can use our model assess their preliminary risk of diagnosis with malignant breast cancer.

4. The proposed analytic ANN model can identify the risk of breast cancer and proceed for medical treatment if necessary.

5. The proposed model can also be used to determine if an individual should proceed to have a mammogram.

6. The information obtained from the proposed model would improve the financial aspects of health by avoiding unnecessary treatments.

Finally, the present research confirms the fact that ANN have an important role in improving the accuracy and consistency of medical diagnosis. We believe that we can improve this model further by considering more relevant risk factors and more recent data. Additionally, we would like to re-implement this model for different races since the race is one of the significant risk factors which contributes to the malignancy of breast cancer [55].

CHAPTER 3

BAYESIAN MODELING OF NONLINEAR POISSON REGRESSION WITH ARTIFICIAL NEURAL NETWORKS

3.1 Introduction

Poisson regression is a form of regression analysis which is used to model count data [56]. This plays an important role in interdisciplinary research including health, finance, social, etc. For example, Poisson regression can be used to model the number of mutations on a strand of DNA per unit length, model number of claims occurring in a given period, or to model the students drop out rates from schools.

When developing a Poisson regression model, we assume that its mean is related to a function of covariates. More specifically, it assumes that log-transformed outcomes are linearly related to the covariates. However, in reality, this assumption may or may not be true. Another strong assumption which involves in Poisson regression is that its mean is same as its variance. Any violation of this assumption might lead to significantly underestimated standard errors. Eventually, this may incorrectly assess the significance of individual regression parameters.

In this Chapter, we discuss about a method of developing a nonlinear Poisson regression model using artificial neural networks (ANN). In fact, we introduce a new Bayesian artificial neural network for developing a nonlinear Poisson regression. With our approach, we can overcome the most serious issues of a conventional Poisson regression model of overdispersion, by introducing error bars over the standard errors and using relevance determination prior to assess the importance of each covariate into the model. Additionally, we introduce a new Hybrid Bayesian learning method for neural networks based on the

evidence procedure and Hybrid Monte Carlo (HMC) sampling. A real world application of this new ANN model will be presented in the next Chapter.

3.2 Literature Review

Researchers have an ever increasing interest on modeling with nonlinear regression models, mainly due to the availability of Big data. These include nonlinear analysis methods like as generalized additive models, classification and regression trees, multivariate adaptive regression splines and neural network models which exhibit their unique strengths and weaknesses.

Among those, nonlinear modeling with artificial neural networks has gained an immense attraction due to their flexibility and high predictive performances. Consequently, a significant amount of researchers has contributed in developing nonlinear versions of generalized linear models with neural networks. Development of a nonlinear logistic regression model with an ANN is one of the pioneering study [43] in this field. C. M. B. Bishop in 2006 [9], then introduced a nonlinear multinomial logistic regression model using ANN and both of these models have been extensively applied for solving various interdisciplinary research problems [4, 57]. A nonlinear extension of ordinal logistic regression using ANN has been introduced in financial engineering by Mathieson *et al.* [58].

A nonlinear Poisson regression model has first been developed using the maximum likelihood (ML) method by Fallah *et al.* [59] in 2009. An application of that in predicting the cause-specific hazard of the breast cancer patients can be found in [60]. As per our knowledge, only these two studies have contributed in this research regime. In this study, our goal is to significantly improve these existing models by introducing the Bayesian learning techniques. Although the Bayesian learning approaches have been used for regular regression [61], none of the existing studies have utilized it for count modeling.

3.3 Methodology

3.3.1 Neural Network and Poisson Regression

Let $D = (\boldsymbol{t}^1, ..., \boldsymbol{t}^n)$ be a vector of count responses with input vectors $\boldsymbol{x^n}$ for $n = 1, 2, ...N$. Note that $\boldsymbol{t}^1 = \{t_1^1, t_2^1, ..., t_K^1\}$. Then, we can obtain the likelihood function according to Eq. (1.9) in Chapter 1. The probability mass function of the Poisson distribution, $p(\boldsymbol{t}^n|\boldsymbol{x}^n)$, is given in Eq. (3.1).

$$p(\boldsymbol{t}^n|\boldsymbol{x}^n) = \prod_{k=1}^{K} \frac{e^{-\lambda_k^n}(\lambda_k^n)^{t_k^n}}{t_k^n!}, t_k^n = 0, 1, 2, ... \tag{3.1}$$

The average value, λ_k^n can be modeled with a multi-layer perceptron artificial neural network model of the form given in Eq. (3.2) with d inputs, M hidden nodes and K outputs,

$$y_k(\boldsymbol{x}^n, \boldsymbol{w}) = \hat{\lambda_k^n} = g\bigg(\sum_{j=1}^{M} w_{kj}^{(2)} h\bigg(\sum_{i=1}^{d} w_{ji}^{(1)} x_i^n + b_j^{(1)}\bigg) + b_k^{(2)}\bigg), \tag{3.2}$$

where h and g are the hyperbolic tangent and the exponential activation functions in the hidden and output layers.

3.3.2 Training the ANN

The training phase of a neural network plays an important role to gain better predictions. This can be achieved either using the maximum likelihood (ML) or the Bayesian methods. With the ML approach, we can find an optimal set of weights by minimizing the network error function. However, training with ML tends to provide poor predictions due to its inherent problem of network overfitting which had lead to bias parameter estimations. This can be minimized by introducing a regularization parameter.

Unlike the ML method, the Bayesian approach provides a more intuitive learning of the weight parameters. Bayesian learning of ANN involves introducing a prior distribution for the weights. In developing our proposed model, we used a zero mean Gaussian prior of the form Eq. (1.28) given in Chapter 1.

In either methods, ML or Bayesian, we need to minimize the canonical error function as a part of the learning process. When developing a nonlinear Poisson regression model, this error function is obtained by taking the negative log likelihood of the Poisson distribution of the form Eq. (3.3),

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K}\Big[-y_k(\boldsymbol{x}^n,\boldsymbol{w}) + t_k^n\, log(y_k(\boldsymbol{x}^n,\boldsymbol{w})) - ln(t_k^n)!\Big]. \tag{3.3}$$

This error function can be modified by eliminating the terms which are not involved in model fitting and introducing a weight decay parameter α,

$$\begin{aligned}\tilde{E}(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K}\Big[&-y_k(\boldsymbol{x}^n,\boldsymbol{w}) + t_k^n\, log(y_k(\boldsymbol{x}^n,\boldsymbol{w})) \\ &+\frac{\alpha}{2}\sum_{weights}\big({w_{ij}^{(1)}}^2 + {w_{kj}^{(2)}}^2 + (b_j^{(2)})^2 + (b_k^{(2)})^2\big)\Big].\end{aligned} \tag{3.4}$$

In the ML approach, we used a cross-validation method which involves reserving a validation dataset to evaluate the error $\tilde{E}(\boldsymbol{w})$ of models trained using a range of different values of α from $\{0.01.0.025, 0.05, 0.075, 0.1\}$ and selecting the value of α that gives the smallest $\tilde{E}(\boldsymbol{w})$. This is based on the fact that the weight decay values between 0.01 and 0.1 are sufficient for model regularization [43].

Under the Bayesian approach, we used the evidence procedure, Hybrid Monte Carlo (HMC) method and a new Hybrid Bayesian learning method. As we have introduced the first and second methods in Chapter 1, here, we discuss about the new Hybrid Bayesian method.

3.4 New Bayesian Learning for Neural Networks

In the HMC method, we need to generate several samples out of the posterior distribution of the weights in order to approximate the integral Eq. (1.32) to make the predictions. The generation of these samples highly depends on the initial hyperparameter value in the weight posterior.

Prior to HMC sampling, we can use the evidence procedure to optimize these hyperparameter value in the nonlinear Poisson regression model. This optimized hyperparameter value along with the weight parameters can then be used to generate the samples from the posterior distribution. This new approach, called the Hybrid Bayesian, provides relatively high prediction accuracies, specially compared to HMC method. Additionally, we can identify the relative importance of the inputs to the final ANN model. Moreover, we can capture the uncertainties associated with our network predictions by constructing the confidence intervals as discussed in Section 1.4.3. We have summarized the steps of this new approach with respect to the nonlinear Poisson regression model in Fig. 3.1.

3.4.1 Convergence Diagnostic Statistics

When using the methods which utilized the Monte Carlo sampling, i.e., HMC and Hybrid Bayesian, our goal is to generate samples out of the stationary distribution of the Markov chain. Therefore, we need to check whether the chain has converged or not. In order to assess this, we used a convergence diagnostic test statistic called, "estimated potential scale reduction" (EPSR) which was introduced by Gelman and Rubin [62].

Here, they have assumed that if a chain has converged, then it has forgotten its starting point. So, several sequences drawn from different starting points should be indistinguishable. EPSR assess this quantitatively by calculating the variance between different sequences of a similar size and the variance within each of those sequence. Conventionally, a group of sequence of samples can be accepted if their EPSR statistic falls below 1.10 for all statistic of interest including the regularized error function. Further details can be found in [20].

3.5 Development of the Nonlinear Poisson ANN Model

3.5.1 Parameter Optimization

For an ANN model with fixed number of hidden units, we can minimize the regularized error function $\tilde{E}(\boldsymbol{w})$ to obtain the optimal weight vector $\boldsymbol{w}$. That is to find a weight vector

Step 1
Choose an initial value for the hyperparameter α. Initialize the weights and the bias parameters in the network.

Step 2
Train the network with a suitable optimisation algorithm to minimize the reguralized cost function $S(\boldsymbol{w})$ given in the Eq. (3.4).

Step 3
When the network training has achieved a local minimum, use the Gaussian apprx. to compute the evidence for the hyperparameter. α can be re-estimated using,

$$\alpha^{new} = \frac{\gamma}{2E_p(\boldsymbol{w})}$$

and obtain the optimal value α_{MAP}

Step 4
Having found the α_{MAP}, and the weights and bias parameters, use HMC to sample from the posterior distribution of the weights to approximate the predictive distribution

$$p(t|\boldsymbol{x}^*, D) = \int p(t|\boldsymbol{x}^*, \boldsymbol{w}) p(\boldsymbol{w}|D) d\boldsymbol{w}$$

by the finite sum

$$p(t|\boldsymbol{x}^*, D) \simeq \frac{1}{M} \sum_{m=1}^{M} p(t|\boldsymbol{x}^*, \boldsymbol{w}_m)$$

Use Eq. (1.42) to calculate the standard errors.

Step 5
Repeat the steps 1 to 4 for random initial choices for the network weights in order to generate network commitees.

Step 6
Make the predictions based on the network committees.

Figure 3.1: Steps of the new bayesian learning procedure

$\boldsymbol{w}$ which gives the smallest $\tilde{E}(\boldsymbol{w})$. However, There exist multiple points in the weight space at which the gradient vanishes due to the nonlinear dependence on the weights and bias parameters of the error function. Hence it may be necessary to compare several local minima in order to find a sufficiently good solution. In order to find these points with $\tilde{E}(\boldsymbol{w}) = 0$, we need to rely on iterative numerical procedures.

Most techniques involve choosing some initial value $\boldsymbol{w}^{(0)}$ for the weight vector and then moving through weight space in a successive steps of the form,

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} + \Delta\boldsymbol{w}^{(\tau)}. \tag{3.5}$$

where τ is the iteration step. Different algorithm involve different choices for the weight vector update $\Delta\boldsymbol{w}^{(\tau)}$. Many of them use the gradient information, and therefore need to evaluate $\tilde{E}(\boldsymbol{w})$ after each update. This will be discussed in the next section.

3.5.2 Error Back-Propagation and Evaluation of Error Gradients

The back-propagation procedure allows the derivatives of an error function with respect to the network weights and biases to be evaluated efficiently. This uses the chain rule of partial derivatives and leads to an algorithm in which error derivatives are propagated backward through the network starting from the output units.

For the explanation purpose of the error back-propagation technique involved with nonlinear Poisson regression, we introduce the following notations for an ANN. Let $a_j^{(1)}$ and z_j be the inputs and outputs of the hidden layer as given in Eqns. (3.6) and (3.7).

$$a_j^{(1)} = \sum_{i=1}^{d} w_{ji}^{(1)} x_i + b_j^{(1)}, \tag{3.6}$$

and

$$z_j = h(a_j^{(1)}) = tanh(a_j^{(1)}), \;\; j = 1, 2, ..., M. \tag{3.7}$$

Eq. (3.7) has the property

$$\frac{dz_j}{da_j^{(1)}} = (1 - z_j^2). \tag{3.8}$$

The corresponding input to the output layer is given in Eq. (3.9),

$$a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + b_k^{(2)}, k = 1, 2, ..., K, \tag{3.9}$$

where K is the total number of outputs. So the final output of the ANN for the Poisson regression model is given by,

$$y_k = y_k(\boldsymbol{x}, \boldsymbol{w}) = exp(a_k^{(2)}). \tag{3.10}$$

The first step in evaluating the error derivatives is to perform a forward propagation for the complete data set in order to evaluate the activations, ${z_j}^n$ of the hidden units and the activations, $y_k(\boldsymbol{x}^n, \boldsymbol{w})$ [or y_k^n] of the output units for each pattern n in the data set. We assume our training data are independently and identically distributed and hence the total error function can be written as a sum of n individual error functions,

$$E(\boldsymbol{w}) = \sum_{n=1}^{N} E^n(\boldsymbol{w}). \tag{3.11}$$

Because of the canonical choice of the error function and its corresponding activation function, it can be seen that the partial derivative of the error with respect to ${a_k}^{(2)n}$, is given by,

$$\frac{\partial E^n(\boldsymbol{w})}{\partial {a_k}^{(2)n}} = \delta_k^{(2)n} = {y_k}^n - {t_k}^n. \tag{3.12}$$

This represents the 'error' of the output unit by the data pattern n. The derivatives of $E(\boldsymbol{w})$ with respect to the second layer weights and bias are given by,

$$\frac{\partial E^n(\boldsymbol{w})}{\partial {w_{kj}}^{(2)}} = \sum_{n=1}^{N} \delta_k^{(2)n} z_j^n, \tag{3.13}$$

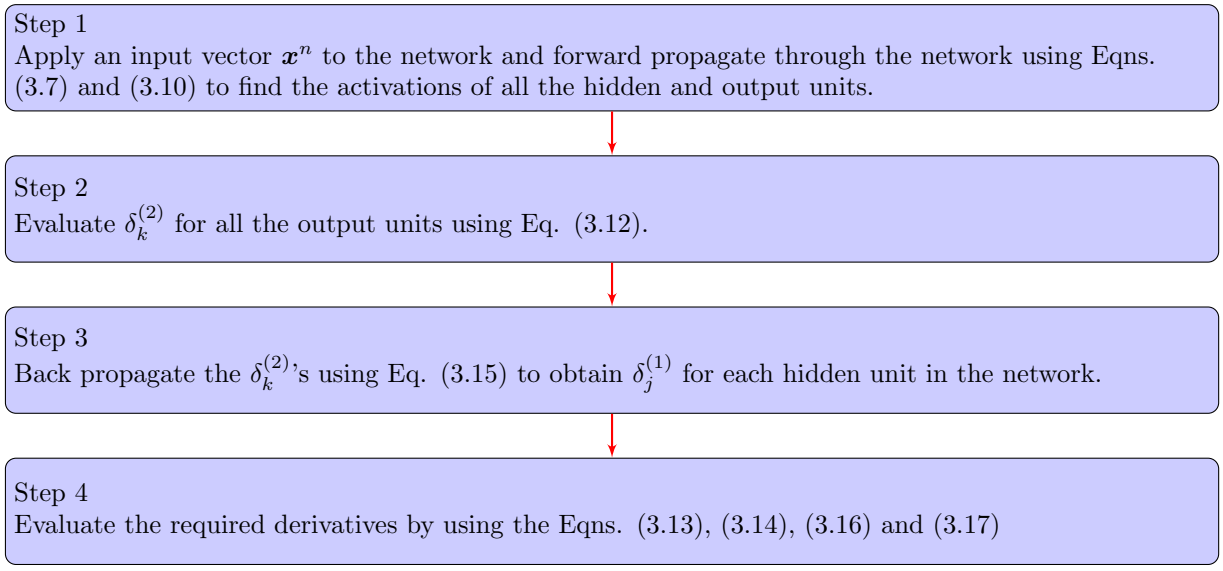

Figure 3.2: Error back-propagation procedure

and

$$\frac{\partial E^n(\boldsymbol{w})}{\partial {b_k}^{(2)}} = \sum_{n=1}^{N} \delta_k^{(2)n}. \tag{3.14}$$

The corresponding derivatives of the hidden layer parameters can then be obtained by back propagating the 'errors', $\delta_k^{(2)n}$ through the output layer weights. Hence the 'error' signals for the hidden nodes,

$$\delta_j^{(1)n} = h'(a_k^{(2)}) \sum_{k=1}^{K} {w_{kj}}^{(2)} \delta_k^{(2)n} = (1 - (z_j^n)^2) \sum_{k=1}^{K} {w_{kj}}^{(2)} \delta_k^{(2)n}. \tag{3.15}$$

Hence,

$$\frac{\partial E^n(\boldsymbol{w})}{\partial {w_{ji}}^{(1)}} = \sum_{n=1}^{N} \delta_j^{(1)n} x_i^n, \tag{3.16}$$

and

$$\frac{\partial E^n(\boldsymbol{w})}{\partial {b_j}^{(1)}} = \sum_{n=1}^{N} \delta_j^{(1)n}. \tag{3.17}$$

Therfore, we can summarize the above discussed process as given in Fig. 3.2.

The derivative of the total error $E(\boldsymbol{w})$ can then be obtained by repeating the above steps for each data pattern in the training set and then summing over all patterns. i.e.,

$$\frac{\partial E(\boldsymbol{w})}{\partial {w_{ji}}^{(1)}} = \sum_{n=1}^{N} \frac{\partial E^n(\boldsymbol{w})}{\partial {w_{ji}}^{(1)}}. \tag{3.18}$$

This error function is then used for batch optimizations.

3.5.3 The Fast Multiplication by the Hessian

The second derivatives of the error function which consist of $\frac{\partial E(\boldsymbol{w})}{\partial w_{ji} w_{kj}}$ form the elements of the Hessian matrix $\boldsymbol{H}$. This plays an important role in many aspects of neural computing. This has been used in several nonlinear optimization algorithms, as the network training is based on the second-order properties of the error surface. Moreover, this provides a fast procedure for re-training the network even after a small change in the training data.

However, for many applications of the Hessian, the quantity of interest is not the Hessian matrix $\boldsymbol{H}$, itself, but the product of $\boldsymbol{H}$, with some vector $\boldsymbol{\nu}$. This product, $\boldsymbol{\nu}^T \boldsymbol{H}$, shows interesting properties with fewer calculations and also nearly same efficiency. When developing our nonlinear Poisson ANN model we utilized this procedure using the R-propagation algorithm proposed by Pearlmutter [81]. Further details can be found in [9, 20].

We first need to note that,

$$\boldsymbol{\nu}^T \boldsymbol{H} = \boldsymbol{\nu}^T \nabla (\nabla E(\boldsymbol{w})), \tag{3.19}$$

where ∇ is the gradient operator in the weight space. $\nabla E(\boldsymbol{w})$ is evaluated using standard forward-propagation and back-propagation equations given in the previous section. We then apply Eq. (3.19) to these equations to obtain a set of forward-propagation and back-propagation equations in order to evaluate $\boldsymbol{\nu}^T \boldsymbol{H}$. This is similar to applying the differential operator $\boldsymbol{\nu}^T \nabla$ on the original forward-propagation and back-propagation equations, and

we denoted it by $R\{.\}$ using Pearlmutter's notation. Then we have,

$$R\{\boldsymbol{w}\} = \boldsymbol{\nu}. \tag{3.20}$$

Applying $R\{.\}$ operator on forward-propagation equations given in Eqns. (3.6), (3.7), (3.9) and (3.10) we get,

$$R\{a_j^{(1)}\} = \sum_i \nu_{ji} x_i, \tag{3.21}$$

$$R\{z_j\} = h'(a_j) R\{a_j\}, \tag{3.22}$$

$$R\{a_k^{(2)}\} = \sum_j w_{kj}^{(2)} R\{z_j\} + \sum_j \nu_{kj} z_j, \tag{3.23}$$

$$R\{y_k\} = y_k R\{a_k^{(2)}\}, \tag{3.24}$$

where ν_{ji} is the element of the vector $\boldsymbol{\nu}$ that corresponds to the weight w_{ji}. Applying $R\{.\}$ operator on back propagation Eqns. (3.12) and (3.15) we get R-backward propagation equations:

$$R\{\delta_k^{(2)}\} = R\{y_k\}, \tag{3.25}$$

and

$$\begin{aligned} R\{\delta_j^{(1)}\} = h''(a_j^{(1)}) R\{a_j^{(1)}\} \sum_k {w_{kj}}^{(2)} \delta_k^{(2)} + \\ h'(a_j^{(1)}) \sum_k \nu_{kj} \delta_k^{(2)} + h'(a_j^{(1)}) \sum_k w_{kj} R\{\delta_k^{(2)}\} \end{aligned} \tag{3.26}$$

We also know that the gradients with respect to the output and the hidden layer weights are given by Eqns. (3.13) and (3.16). Finally, applying R operator on those two equations, we get the elements of $\boldsymbol{\nu}^T \boldsymbol{H}$ as follows.

$$\frac{\partial E(\boldsymbol{w})}{\partial w_{kj}} = R\{\delta_k^{(2)}\} z_j + \delta_k^{(2)} R\{z_j\}, \tag{3.27}$$

and

$$\frac{\partial E(\boldsymbol{w})}{\partial w_{ji}} = x_i R\{\delta_j^{(1)}\}. \tag{3.28}$$

3.5.4 Evaluations Measures

For the evaluation purposes, we have used several error measurement criteria. These include the root mean square error (RMSE), mean absolute error (MAE), mean percentage error (MPE) and relative squared error (RSE) as given in equations from Eqns. (3.29) to (3.32).

$$RMSE = \left[\frac{1}{N}\sum_{n=1}^{N}\left(y(\boldsymbol{x}^n, \boldsymbol{w}) - t^n\right)^2\right]^{(1/2)}, \tag{3.29}$$

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|y(\boldsymbol{x}^n, \boldsymbol{w}) - t^n|, \tag{3.30}$$

$$MPE = \frac{1}{N}\sum_{n=1}^{N}|\frac{y(\boldsymbol{x}^n, \boldsymbol{w}) - t^n}{t^n}|, \tag{3.31}$$

and

$$RSE = \frac{\sum_{n=1}^{N}\left(y(\boldsymbol{x}^n, \boldsymbol{w}) - t^n\right)^2}{\sum_{n=1}^{N}(t^n - \bar{t})^2}. \tag{3.32}$$

These error measurements help to provide an overall assessment of the predictions in different aspects. RMSE and MAE can be used to assess the prediction accuracies of the models whereas MPE acts as a good measure of bias in the predictions. RSE gives the relative error to what it would have been if a simple predictor (the average of the actual values) had been used.

3.6 Simulation Study

After constructing the nonlinear Poisson regression model, we evaluated its performance using six simulation studies. We have chosen these simulation schemes in a way that the Poisson expected value depends both linearly and nonlinearly on the covariates.

- Simulation 1

  The response variable is generated with a single covariate $x \sim Uni(0,1)$,

$$Y_i \sim Poi\,(exp(x)). \tag{3.33}$$

- Simulation 2

  The response variable is generated with a single covariate $x \sim Uni(0,1)$,

$$Y_i \sim Poi\,(exp(1 + 1.5exp(x + 0.2))). \tag{3.34}$$

- Simulation 3

  The response variable is generated with two covariates, $x_1 \sim Uni(0,1)$ and $x_2 \sim Uni(0,2)$,

$$Y_i \sim Poi\,(exp(1 + 1.2x_1^{1/2} + 0.25x_2^{1/4})). \tag{3.35}$$

- Simulation 4

  The response variable is generated with two covariates, $x_1, x_2 \sim Uni(0,1)$,

$$Y_i \sim Poi\left(exp\big(\frac{0.5exp(1 + 2x_1)}{1 + exp(x_2 + 1)}\big)\right). \tag{3.36}$$

- Simulation 5

  The response variable is generated with three covariates, $x_1 \sim Uni(0,1), x_2 \sim Uni(1,2), x_3 \sim Uni(0,1)$,

$$Y_i \sim Poi\left(exp\big(\frac{(0.5x_1^2 + x_2^2)}{1 + 0.2exp(x_3 + 0.2)}\big)\right). \tag{3.37}$$

- Simulation 6

  The response variable is generated with three covariates, $x_1 \sim Uni(1,4), x_2 \sim Uni(0,1), x_3 \sim Uni(0,0.2)$,

$$Y_i \sim Poi\left(exp\big(1 + 1.25\,log(x_1) + 0.5x_2 + 0.25x_3^2\big)\right) \tag{3.38}$$

With each of the above simulation schemes, we generated samples of different sizes, 500, 5,000 and 50,000 and each of them are partitioned into a training set (80%) and a testing set (20%). Nonlinear Poisson regression models are then created with ANN following both ML and Bayesian approaches (evidence, HMC and Hybrid Bayesian). A 5-fold cross validation technique is used with ML approach to minimize the network overfitting. We repeat the same process for different weight decay values $\alpha = \{0.01, 0.025, 0.05, 0.075, 0.1\}$ and for different hidden nodes from 3 to 13. The final predictions are based on network committees created with ten different random initializations.

In the Bayesian approach, we utilized a zero mean Gaussian prior to initialize the weight parameters. Moreover, an automatic relevance determination prior was used in the evidence setting where it assumes different distributions for each of the input variables. When using the HMC method, we discard some initial samples to avoid the susceptibility of sampling from any non-stationary distribution. The same procedure is followed with the proposed Hybrid Bayesian method. We also constructed the error bars within one standard deviation of our predictions. When evaluating the models, we have used the actual λ instead of $y(\boldsymbol{x}^n, \boldsymbol{w})$ for the previously discussed error measurements.

Tables 3.1 and 3.2 summarize these measurements for ANN models with 5 and 10 hidden nodes, which are initiated with a hyperparameter value of 0.075. As can be seen, for Simulation 1, linear Poisson regression model has outperformed ANN models regardless of the amount of data. This confirms the fact that, a linear model is superior when there exist a simple linear relationship between the response and the covariates.

In contrast to that, when there exist nonlinear dependencies on the covariates, ANN models have outperformed the linear Poisson regression model. More specifically, Bayesian ANN models have given the smallest prediction errors compared to the ANN models constructed with the ML method. For relatively smaller sample sizes, both evidence and Hybrid Bayesian perform well interchangebly. However, for extremly large sample sizes (N=50,000), the prediction accuracies of the proposed Hybrid Bayesian method shows a significant improvement over the other two Bayesian methods. Moreover, except for few

Table 3.1: Model evaluation using ANN with 5 hidden nodes with testing data

ANN with 5 Hidden Nodes and Alpha 0.075												
	N = 500				N = 5,000				N = 50,000			
	RMSE	**MAE**	**MPE**	**RSE**	**RMSE**	**MAE**	**MPE**	**RSE**	**RMSE**	**MAE**	**MPE**	**RSE**
Simulation 1												
Linear Poisson Reg	0.09200	0.07020	0.03670	0.04670	0.00300	0.00270	0.00190	0.00003	0.01250	0.01030	0.00610	0.00065
ML	0.20540	0.15410	0.09560	0.23280	0.18310	0.16040	0.10600	0.13420	0.12470	0.06560	0.03210	0.06500
HMC	0.09360	0.06880	0.03660	0.04830	0.05620	0.04620	0.02690	0.01280	0.04260	0.03510	0.02560	0.00740
Evidence	0.09460	0.06180	0.02950	0.04940	0.03430	0.02730	0.01520	0.00480	0.01640	0.01080	0.00540	0.00110
Hybrid Bayesian	0.09240	0.05980	0.02840	0.04710	0.03280	0.02660	0.01500	0.00440	0.01690	0.01110	0.00560	0.00120
Simulation 2												
Linear Poisson Reg	9.86790	7.63730	0.11460	0.01220	10.67720	7.55080	0.11300	0.11380	10.70620	7.65760	0.11150	0.01392
ML	2.36640	2.07470	0.04000	0.00060	1.06750	0.80440	0.01290	0.00010	0.94980	0.59080	0.00900	0.00011
HMC	0.99040	0.68660	0.00790	0.00012	0.70100	0.43470	0.00480	0.00006	0.42230	0.25150	0.00460	0.00002
Evidence	1.13740	0.81470	0.00970	0.00016	0.50140	0.30150	0.00300	0.00003	0.26580	0.21220	0.00320	0.00000
Hybrid Bayesian	1.01510	0.72400	0.00950	0.00013	0.66290	0.41950	0.00480	0.00005	0.20910	0.16130	0.00310	0.00000
Simulation 3												
Linear Poisson Reg	0.32240	0.22260	0.04370	0.03650	0.27680	0.22830	0.03890	0.02500	0.29680	0.24610	0.04280	0.02800
ML	1.01190	0.76610	0.14030	0.26580	0.24160	0.19570	0.03190	0.02120	0.09880	0.08230	0.01250	0.00310
HMC	0.23120	0.18680	0.03310	0.01870	0.12670	0.09710	0.01670	0.00520	0.07040	0.04660	0.00900	0.00160
Evidence	0.19100	0.14580	0.02800	0.01280	0.11210	0.08740	0.01460	0.00410	0.04470	0.02810	0.00530	0.00064
Hybrid Bayesian	0.18980	0.15510	0.02750	0.01270	0.09970	0.07670	0.01240	0.00330	0.03880	0.02330	0.00450	0.00048
Simulation 4												
Linear Poisson Reg	7.02420	5.40880	0.13180	0.01360	10.42930	5.94110	0.12410	0.02350	13.91590	7.27130	0.12570	0.02850
ML	5.32640	3.90830	0.08170	0.00790	1.45810	1.12320	0.02460	0.00045	1.39260	1.00040	0.02020	0.00028
HMC	1.28920	0.93750	0.01770	0.00046	1.33320	0.57800	0.00960	0.00038	0.83480	0.49440	0.00820	0.00010
Evidence	1.69010	1.20570	0.02080	0.00079	0.54110	0.40540	0.00850	0.00004	0.53470	0.40010	0.00840	0.00004
Hybrid Bayesian	1.48750	1.07050	0.02030	0.00041	0.91030	0.45580	0.00960	0.00018	0.44120	0.31900	0.00630	0.00006
Simulation 5												
Linear Poisson Reg	0.73730	0.49270	0.08760	0.01990	0.71300	0.43370	0.06460	0.01900	0.58500	0.42360	0.07200	0.01450
ML	0.68970	0.41050	0.06740	0.01810	0.78440	0.48810	0.08080	0.02550	0.30900	0.20530	0.02890	0.00400
HMC	0.58940	0.35400	0.05650	0.01270	0.42720	0.23370	0.03070	0.00680	0.24040	0.16050	0.02380	0.00250
Evidence	0.60430	0.44130	0.08440	0.01340	0.40880	0.22680	0.03000	0.00620	0.24770	0.17230	0.02380	0.00260
Hybrid Bayesian	0.58730	0.43340	0.08320	0.01260	0.32640	0.19000	0.02530	0.00400	0.19650	0.13450	0.02010	0.00170
Simulation 6												
Linear Poisson Reg	0.69150	0.54970	0.06170	0.02140	0.74220	0.64700	0.07370	0.02200	0.74150	0.62940	0.06930	0.02160
ML	1.08560	0.92540	0.09680	0.04480	0.48050	0.39280	0.04240	0.00910	0.10380	0.07560	0.00680	0.00042
HMC	0.39770	0.33120	0.03780	0.00710	0.15660	0.10260	0.00930	0.00098	0.05080	0.03150	0.00320	0.00010
Evidence	0.37690	0.32250	0.03550	0.00640	0.18280	0.14050	0.01430	0.00130	0.04730	0.03280	0.00320	0.00009
Hybrid Bayesian	0.36120	0.28760	0.03480	0.00580	0.18280	0.14060	0.01440	0.00130	0.04540	0.03140	0.00320	0.00008

cases, we were able to obtain better results with the new Hybrid Bayesian model compared with the HMC sampling method. We observe the same pattern for other α values as well. Our findings are consistent with the study [61] which was done for ordinary regression.

We also interested in analyzing the impact associated with the expansion of the weight parameter space. For that, we analyze the model performances with a varying number of hidden nodes from 3 to 13. However, we did not notice any significant improvement in the prediction accuracies when increasing the number of hidden nodes. In fact, we found that the optimal number of hidden nodes depend on the amount of the data and the complexity of the problem. Therefore, prior to finding an ANN model, it is important to consider several ANN models with different hidden nodes.

Table 3.2: Model evaluation using ANN with 10 hidden nodes with testing data

ANN with 10 Hidden Nodes and Alpha 0.075												
	N = 500				**N = 5,000**				**N = 50,000**			
	RMSE	**MAE**	**MPE**	**RSE**	**RMSE**	**MAE**	**MPE**	**RSE**	**RMSE**	**MAE**	**MPE**	**RSE**
Simulation 1												
Linear Poisson Reg	0.08960	0.06050	0.03130	0.04420	0.00300	0.00270	0.00190	0.00003	0.01250	0.01030	0.00610	0.00065
ML	0.20550	0.15410	0.09560	0.23280	0.20540	0.15410	0.09560	0.23270	0.10310	0.09400	0.06280	0.04320
HMC	0.09200	0.07020	0.03670	0.04670	0.02610	0.01970	0.01070	0.00280	0.03860	0.03390	0.02450	0.02160
Evidence	0.09950	0.07380	0.03700	0.05460	0.02600	0.02010	0.01110	0.00270	0.01660	0.01080	0.00530	0.00120
Hybrid Bayesian	0.09990	0.07450	0.03750	0.05510	0.02530	0.01960	0.01080	0.00260	0.01680	0.01080	0.00530	0.00120
Simulation 2												
Linear Poisson Reg	9.86790	7.63730	0.11460	0.01220	10.67720	7.55080	0.11300	0.11380	10.70620	7.65760	0.11150	0.01392
ML	2.25310	1.95060	0.03700	0.00062	0.60900	0.34150	0.00450	0.00003	0.49100	0.39910	0.00590	0.00003
HMC	1.01910	0.69160	0.00760	0.00013	0.82840	0.43890	0.00360	0.00008	0.26350	0.18880	0.00300	0.00001
Evidence	1.06220	0.77390	0.00990	0.00014	0.54380	0.32230	0.00320	0.00004	0.25730	0.20540	0.00310	0.00001
Hybrid Bayesian	1.14410	0.80680	0.00980	0.00016	0.57020	0.33170	0.00310	0.00004	0.24020	0.18850	31.00000	0.00001
Simulation 3												
Linear Poisson Reg	0.32240	0.22260	0.04370	0.03650	0.27680	0.22830	0.03890	0.02500	0.29680	0.24610	0.04280	0.02800
ML	0.90550	0.70130	0.13740	0.21350	0.32400	0.25800	0.04190	0.03780	0.11180	0.08740	0.01410	0.00400
HMC	0.21200	0.54400	0.02880	0.01580	0.13460	0.09800	0.01680	0.00590	0.05180	0.33900	0.00650	0.00085
Evidence	0.21610	0.15990	0.03080	0.01640	0.11220	0.08710	0.01460	0.00410	0.07040	0.05240	0.00890	0.00160
Hybrid Bayesian	0.20810	0.15310	0.03000	0.01520	0.11960	0.09010	0.01490	0.00470	0.04510	0.02880	0.00550	0.00065
Simulation 4												
Linear Poisson Reg	7.02420	5.40880	0.13180	0.01360	10.42930	5.94110	0.12410	0.02350	13.91590	7.27130	0.12570	0.02850
ML	4.96230	3.68190	0.07670	0.00690	2.05150	1.46700	0.03430	0.00090	1.74740	1.08240	0.02200	0.00045
HMC	1.60200	1.15380	0.02470	0.00071	0.90310	0.38980	0.00640	0.00018	0.65950	0.42660	0.00792	0.00013
Evidence	1.55850	1.08310	0.01820	0.00067	0.83010	0.43270	0.00930	0.00015	0.51690	0.39800	0.02020	0.00004
Hybrid Bayesian	1.41490	0.98090	0.01830	0.00055	0.55400	0.39900	0.00780	0.00005	0.46690	0.33868	0.00665	0.00003
Simulation 5												
Linear Poisson Reg	0.73730	0.49270	0.08760	0.01990	0.71300	0.43370	0.06460	0.01900	0.58500	0.42360	0.07200	0.01450
ML	1.74850	1.10990	0.20880	0.08540	1.19630	0.66130	0.10120	0.05900	0.82000	0.46150	0.07220	0.02920
HMC	0.63740	0.43010	0.08430	0.01490	0.43800	0.20020	0.02550	0.00720	0.19650	0.13446	0.02010	0.00168
Evidence	0.59420	0.42900	0.08190	0.01290	0.43780	0.16820	0.02020	0.00710	0.19210	0.13760	0.02200	0.00160
Hybrid Bayesian	0.68430	0.50320	0.09700	0.01710	0.38320	0.15190	0.01830	0.00550	0.18150	0.11490	0.01780	0.00140
Simulation 6												
Linear Poisson Reg	0.69150	0.54970	0.06170	0.02140	0.74220	0.64700	0.07370	0.02200	0.74150	0.62940	0.06930	0.02160
ML	1.26540	1.04940	0.10470	0.06050	0.48210	0.39150	0.44000	0.00920	0.11380	0.08510	0.00810	0.00051
HMC	0.43610	0.37250	0.03960	0.00850	0.18260	0.11470	0.01020	0.00130	0.05770	0.03630	0.00350	0.00013
Evidence	0.37020	0.31420	0.03530	0.00610	0.13430	0.08810	0.00830	0.00072	0.04570	0.03160	0.00310	0.00008
Hybrid Bayesian	0.36580	0.31080	0.03490	0.00600	0.13660	0.08680	0.00780	0.00074	0.04670	0.03320	0.00330	0.00009

As we mentioned in Section 3.5, when using hybrid Monte Carlo methods, we need to be careful about sampling from the stationary distribution. In order to check that, we first used a visualization technique where we overlaid a 5 sequence samples.

Figure 3.3, depicits the cost function values for the HMC and the proposed Hybrid Bayesian methods after a burn-in period of 5,000 samples for the Simulation 6. As can be seen, the 5 different sequences drawn from different starting points of the two chains (two methods) are indistinguishable, which confirms the fact that samples are drawn from a stationary distribution of the Markov chain. Nevertheless, the Hybrid Bayesian shows both a lesser variation and an error (cost function) than in the HMC method.

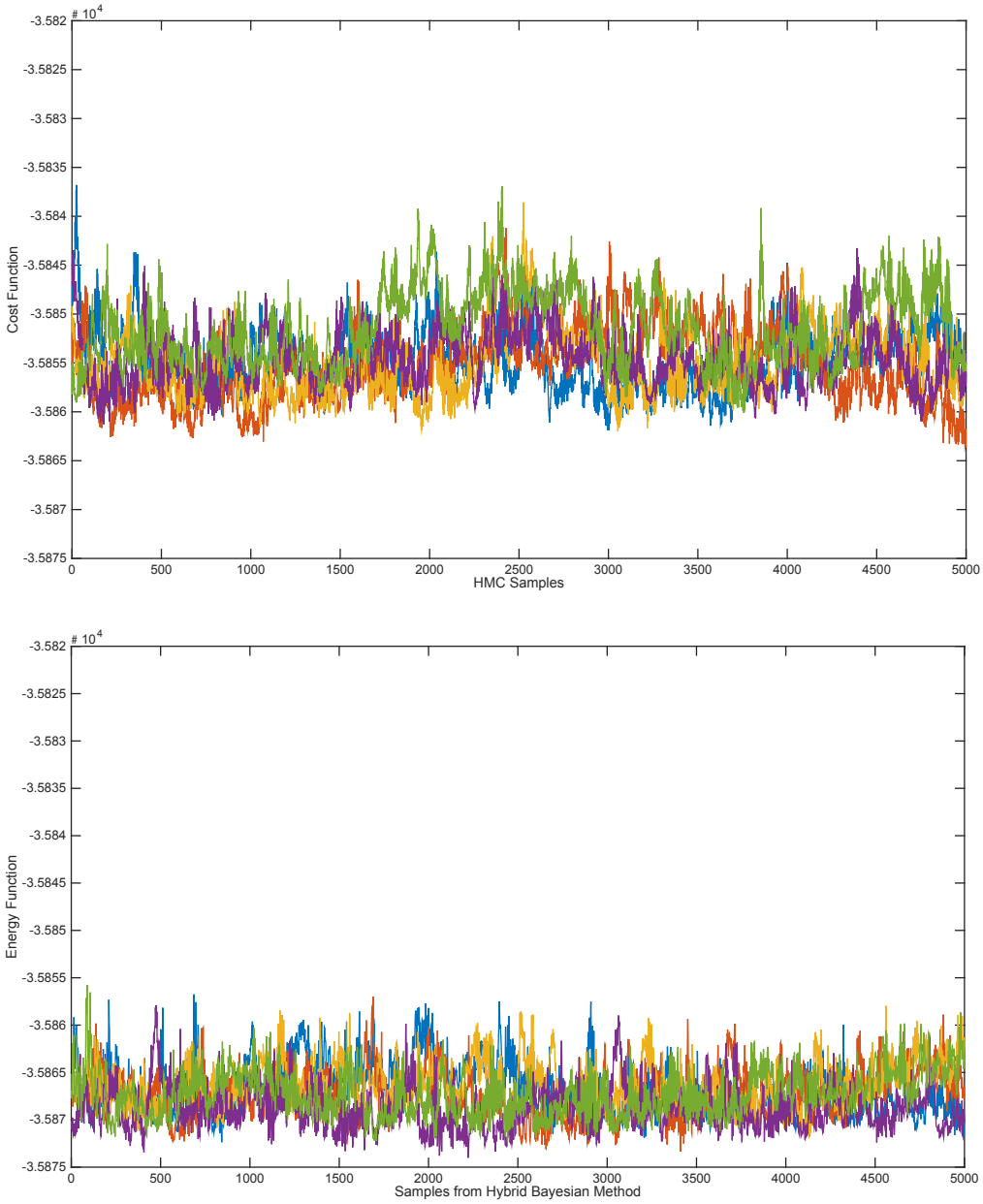

Figure 3.3: Cost functions for 5 sequences drawn from the HMC and Hybrid Bayesian methods after a 5000 burn-in period for Simulation 6 using ANN with 5 hidden nodes

As to quantify the effect of convergence, we calculate the EPSR, conergence diagnostic test statistic values which was introduced in the Section 3.4.1 for each simulation. Table 3.3 summerizes the EPSR values of each weight parameter obtained using the ANN model

Table 3.3: Covergence diagnostics test statistic EPSR values for HMC and Hybrid Bayesian methods: Simulation 6

Weights	**HMC**	**Hybrid Bayesian**
w(1)_11	3.0594	1.0380
w(1)_21	1.7617	1.0893
w(1)_31	1.9997	1.1391
w(1)_41	6.5150	1.0164
w(1)_51	1.9991	1.0407
w(1)_12	2.5444	1.0406
w(1)_22	2.0649	1.0076
w(1)_32	3.2607	1.0679
w(1)_42	2.8220	1.0306
w(1)_52	3.1978	1.0326
w(1)_13	4.2560	1.1904
w(1)_23	2.3057	1.0430
w(1)_33	3.2699	1.0292
w(1)_43	3.9502	1.1431
w(1)_53	3.9652	1.0431
b(1)_1	4.2990	2.0730
b(1)_2	4.7093	1.9725
b(1)_3	2.7830	1.3010
b(1)_4	4.2292	1.0527
b(1)_5	3.1815	1.2040
w(2)_11	3.0967	2.5771
w(2)_21	4.5723	1.9710
w(2)_31	3.6035	1.4738
w(2)_41	2.0472	1.3179
w(2)_51	3.5783	2.1786
b(2)_1	2.3792	1.0397
Cost Function	1.2373	1.0418

with 5 hidden nodes for the Simulation 6. In contrast to the EPSR values associated with HMC weight parameters and the cost function, most of the EPSR values associated with the Hybrid Bayesian method are less than the cut-off 1.10. This indicates that 5,000 samples are not nearly enough for the chain to converge with the HMC method for this data set. However, we can see that our new proposed hybrid Bayesian method converge relatively faster than the HMC method as EPSR for that is less than 1.10. We observe similar results for other simulations.

Figure 3.4 is a visualization of the histogram of the actual mean values and their actual vs predicted values for the Simualtion 6. The green lines represent the error bars within

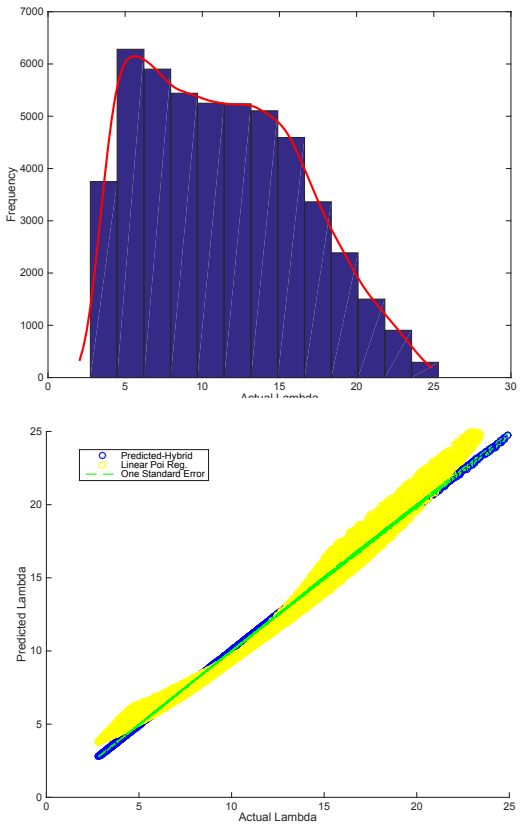

Figure 3.4: Simulation 6: Actual vs predicted lambda

one standard deviation from the predicted mean, which creates a confidence interval for our predictions. This enhances the reliability of our predictions compared to the too precise predictions given by an incorrect model due to the overdispersion.

3.7 Conclusions and Contributions

In this Chapter, we have constructed a new Bayesian nonlinear Poisson regression model. This has a significant potential to be used in interdisciplinary research, in addressing timely important problems related to count modeling. Our contribution to this Chapter can be summarized as follows.

1. We have developed a new nonlinear Poisson regression model using Bayesian artificial neural network.

2. As a part of that, we have developed "exponential" activation function for ANN models.

3. We have introduced a new Hybrid Bayesian learning method which incorporates the core properties of evidence procedure and the Hybrid Monte Carlo Sampling. This new learning method tends to provide accurate results over the other existing methods, specifically for larger samples.

4. With this new model, one can create the error bars for the predictions without having a concern on model overdispersion.

5. ARD prior can be used to identify the relative importance of the covariates, without being misled by false significant results caused by overdispersion.

This proposed nonlinear Poisson regression model can be very useful when handling the Big data. This is because ANN is capable of implicitly detecting all the significant interactions among the predictor variables which we can not achieve with the regular Poisson regression model. Our future goal is to utilize this new model in several real world applications and obtain better predictions. In the next chapter, we discuss a way to extend this nonlinear Poisson model to build a piecewise constant hazard model.

CHAPTER 4

PIECEWISE CONSTANT HAZARD MODEL WITH BAYESIAN ARTIFICIAL NEURAL NETWORK

4.1 Introduction

Accurate prediction of the survival is a challenging, yet substantial task which depends on the underlying the hazard function. These hazard functions can often be complex and might not follow a particular distribution. Moreover, its behavior can significantly be affected by the risk factors which drives the function. Even with decades of research dedicated to survival analysis (and hence in hazard modeling), medical practitioners still search for exclusive predictive models which can handle the modern biomedical data.

In this Chapter, we develop a piecewise constant hazard model using Bayesian neural networks with the intention of introducing a flexible survival prediction model. This was achieved assuming a piecewise constant hazard within smaller time intervals over a certain period. This model provides better survival predictions compared with the conventional methods like linear Poisson regression and generalized estimating equations (GEE). This has been demonstrated with lung cancer patient data taken from Surveillance, Epidemiology and End Results (SEER) program. Just like in the previous Chapter, we have evaluated the model performances using several error measurements criteria.

4.2 Literature Review

Survival or hazard analysis is one of the oldest statistical disciplines which has its roots in demography and actuarial science. A distinguishing feature of survival data is the inevitable presence of incomplete observations, particularly when the terminal event for some individuals is not observed.

Starting from the seventeenth century, a variety of survival analysis models have been developed. The development of renowned Cox proportional hazard model [63] is one of the milestones in survival analysis history. However, this method needs several assumptions to be satisfied, For example, it assumes that the hazard of two or more individuals is proportional to each other, which is an unrealistic assumption. These factors significantly limit its applications.

Parametric modeling is a powerful technique which is used in a wide variety of research for hazard modeling. Nevertheless, they assume specific shapes for the hazard function and hence restrict its flexibility [64]. The generalized linear model with a Poisson error has also been used for survival analysis [65]. An another intuitive approach is to assume that the hazard function is constant within a shorter period, creating a piecewise constant hazard model. This has been advocated as a flexible and parsimonious tool in the literature [66]. This assumption is particularly useful for interpreting cancer survival and to facilitate the treatments and diagnoses [67].

Flexible modeling of survival analysis has become popular during the past decades. We can find studies which have utilized techniques like kernel density [72], artificial neural networks [68–70] and cubic splines [71]. ANN based survival analysis models are widely used mainly due to the capability of handling complex nonlinear relationships among the predictor variables and also due to the fewer assumptions involved with the modeling. Faraggi and Simon have used ANN as a basis for a non-linear proportional hazard model [68]. Another method based on popular multi-layer perceptron: partial logistic regression is developed by [69]. Ravdin and Clark [70] have shown that ANN can be used to predict a patient outcome with censored survival data including time as a covariate.

The development of piecewise exponential model using ANN has been proposed by Fornili *et al.* [60]. This method accommodates a greater flexibility in modeling the complex hazard functions. Here, in this Chapter, we present a new method for developing a piecewise constant hazard model using Bayesian ANN by extending the Fornili's work.

4.3 Methodology

Let T be the survival or the follow-up time for subjects $i = 1, 2, ..., N$ where $T = \min$ {Survival Time, Censoring Time}, and $\boldsymbol{x}$ be the covariates. Let's consider R number of competing risks, which causes the subject to observe the same event of interest [73]. Then Eq. (4.1) defines the hazard function for the r$^{\text{th}}$ risk,

$$\lambda(r, t, \boldsymbol{x}) = \lim_{\Delta t \to 0^+} \frac{P(t < T \leq t + \Delta t, R = r | T \geq t, \boldsymbol{X})}{\Delta t}. \tag{4.1}$$

Then the corresponding survival function and the probability density function can be obtained by Eqns. (4.2) and (4.3) as given below.

$$S(t, \boldsymbol{x}) = exp\bigg(- \int_0^t \lambda(., u, \boldsymbol{x}) du \bigg), \tag{4.2}$$

and

$$f(t|\boldsymbol{x}) = \lambda(., u, \boldsymbol{x}), S(t, \boldsymbol{x}), \tag{4.3}$$

where $\lambda(., u, \boldsymbol{x}_i) = \sum_{r=1}^{R_i} \lambda(r, u, \boldsymbol{x}_i)$ for each individual with R possible competing risks. Thus, for independent observations, assuming non-informative censoring, the likelihood function L can be written as in Eq. (4.4),

$$L = \prod_{i=1}^{N} f(t_i|\boldsymbol{x}_i)^{\delta_i} S(t_i, \boldsymbol{x}_i)^{1-\delta_i} = \prod_{i=1}^{N} \frac{\lambda(., t_i, \boldsymbol{x}_i)}{exp\bigg(\int_0^{t_i} \lambda(., u, \boldsymbol{x}_i) du \bigg)}, \tag{4.4}$$

where δ_i is equal to 0 if the subject i is censored and 1 otherwise.

Under the piecewise constant hazard model, the follow up time T is divided into several disjoint time intervals, $a_0, a_1, ..., a_J$ where $a_0 = 0$ and $a_J = \infty$ and the hazard function for r^{th} risk is assumed to be constant during the j^{th} time period $[a_{j-1}, a_j)$. Hence, we have, $\lambda(., t, \boldsymbol{x}_i) = \lambda(., j, \boldsymbol{x}_i)$ where $\lambda(., j, \boldsymbol{x}_i) = \sum_{r=1}^{R_i} \lambda(r, j, \boldsymbol{x}_i)$ for each subject. Then, the modified likelihood function can be written as in Eq. (4.5),

$$
\begin{aligned}
L &= \prod_{i=1}^{N} \frac{\prod_{j=1}^{J_i} \left(\lambda(.,j,\boldsymbol{x}_i)^{\delta_{ij}} \right)}{exp\left(\sum_{j=1}^{J_i} \left(\lambda(.,j,\boldsymbol{x}_i \right) \tau_{ij} \right)} \\
&= \frac{1}{\prod_{i=1}^{N} \prod_{j=1}^{J_i} \tau_{ij}^{\delta_{ij}}} \prod_{i=1}^{N} \prod_{j=1}^{J_i} \frac{\left(\lambda(.,j,x_i)\tau_{ij} \right)^{\delta_{ij}}}{\delta_{ij}! exp\left(\lambda(.,j,\boldsymbol{x}_i)\tau_{ij} \right)},
\end{aligned}
\tag{4.5}
$$

where

$$
\delta_{ij} = \begin{cases} 1, \text{ if the } i^{th} \text{ subject is deceased during the } j^{th} \text{ interval} \\ 0, \text{ otherwise,} \end{cases}
$$

J_i is the last interval that the subject i is observed and τ_{ij} is the corresponding exposure time which is defined by,

$$
\tau_{ij} = \begin{cases} a_j - a_{j-1}, \text{ if } t_i \geq a_j \\ t_i - a_{j-1}, \text{if } a_{j-1} < t_i \leq a_j \\ 0, \text{if } t_i \leq a_{j-1} \end{cases}
$$

The kernel given in Eq. (4.5) corresponds to the likelihood of a Poisson random variable δ_{ij} with mean $\mu_{ij} = \lambda(.,j,\boldsymbol{x}_i)\tau_{ij}$. By applying the logarithm on both sides of this, we get,

$$
\log(\mu_{ij}) = \log(\lambda(.,j,\boldsymbol{x}_i)) + \log(\tau_{ij}). \tag{4.6}
$$

We can model, $\lambda(.,j,\boldsymbol{x}_i)$ in Eq. (4.6) with a Poisson log-linear model of the form $\log(\lambda(.,j,\boldsymbol{x}_i)) = \alpha_j + \boldsymbol{x_i}'\boldsymbol{\beta}$ as given in [74, 75]. However, modeling with this approach become difficult, sometimes impractical, when there is a large number of δ_{ij} observations, due to a large amount of subject data or longer follow up.

4.4 The Proposed Bayesian ANN Model

In this section, we introduce an efficient method for modeling of the hazard function with Bayesian artificial neural networks. Our goal is to predict $\lambda(r, j, \boldsymbol{x}_i)$ in Eq. (4.1) using three of the Bayesian methods which we discussed in Chapter 3. This new ANN model has several output nodes, each of which corresponds to a different time interval. This structure is similar to the ANN model used by Mani *et al.* [76].

4.4.1 Data Preprocessing

Prior to using the proposed ANN model, data need to be preprocessed. This process can be explained using a simple example. Consider three subjects, called A, B and C who have been observed for J number of years. Suppose, we have information about their risk factors x_1 and x_2, survival time and whether they are being censored or not, as given in Table 4.1. We have considered two competing risk types, R_1 or R_2, for each subject, where they can decease due to one of that reason. The "censor" variable indicates whether a subject has lost follow up somewhere during the study period or has been alive until the end of a study. Hence, for all deceased subjects during the study period, it is set to zero. As can be seen, subject A and B have deceased due to risk types, R_1 and R_2 after 3 and 4 years respectively. According to Table 4.1, subject C has lost follow-up after 2 years.

Table 4.1: Sample data

Subject	x_1	x_2	**Survival Time**	**Risk Type**	**Censored**
A	1	0	3	R_1	0
B	1	1	4	R_2	0
C	1	1	2	R_1 or R_2	1

Table 4.2: Preprocessed data

Subject	x_1	x_2	R_1	R_2	h_1	h_2	h_3	h_4	h_5	...	h_J
A	1	0	1	0	0	0	1	1	1	...	1
B	1	1	0	1	0	0	0	1	1	...	1
C	1	1	1	0	0	0	0	0	0	...	0
C	1	1	0	1	0	0	0	0	0	...	0

In order to use the new ANN model, this information needs to be preprocessed as given in Table 4.2. Since, there are four inputs, covariates x_1 and x_2 and two indicator variables R_1 and R_2, we need to create an ANN with 4 inputs. Censored subjects like C, can be exposed to any of the competing risks and hence, his information is presented twice into the model as given in Table 4.2. If we assume a constant hazard for each year, then there are J number of output nodes in the ANN. i.e., if a subject is alive or censored, then h_j=0, if a subject is deceased, then h_j=1 as in,

$$h_j = \begin{cases} 0, \text{ subject is alive or censored} \\ 1, \text{subject is decreased in the } j^{th} \text{ time interval due to the } r^{th} \text{ risk .} \end{cases}$$

4.4.2 Network Training

In developing the proposed ANN model, we used the hyperbolic tangent and the exponential activation functions in the hidden and the output layers. The proposed ANN structure is represented in Fig. 4.1. The network output, $y(j|r, \boldsymbol{x})$, gives the hazard for each time interval j, as in Eq. (4.7),

$$y(j|r, \boldsymbol{x}) = \lambda(r, j, \boldsymbol{x}) = exp\left(b_{jh}^{(2)} + \sum_{h=1}^{K} w_{jh}^{(2)} \tanh\left(b_{hl}^{(1)} + \sum_{l=1}^{d} w_{hl}^{(1)} x_l\right)\right). \tag{4.7}$$

where $j = 1, 2..., J$. Moreover, $x_1, ..., x_d$ are the inputs, and $w_{hl}^{(1)}$ and $w_{jh}^{(2)}$ are the hidden and output layer weights.

In order to train the network, we used both ML and Bayesian methods which we discussed in Chapter 1 and 3. For the Bayesian methods, we used a zero mean Gaussian prior for the weight distribution. During the training, we minimized the regularized canonical error function given by Eq. (4.8), where α is the non-negative weight decay parameter. As per [43], we trained several ANN models with weight decay values with

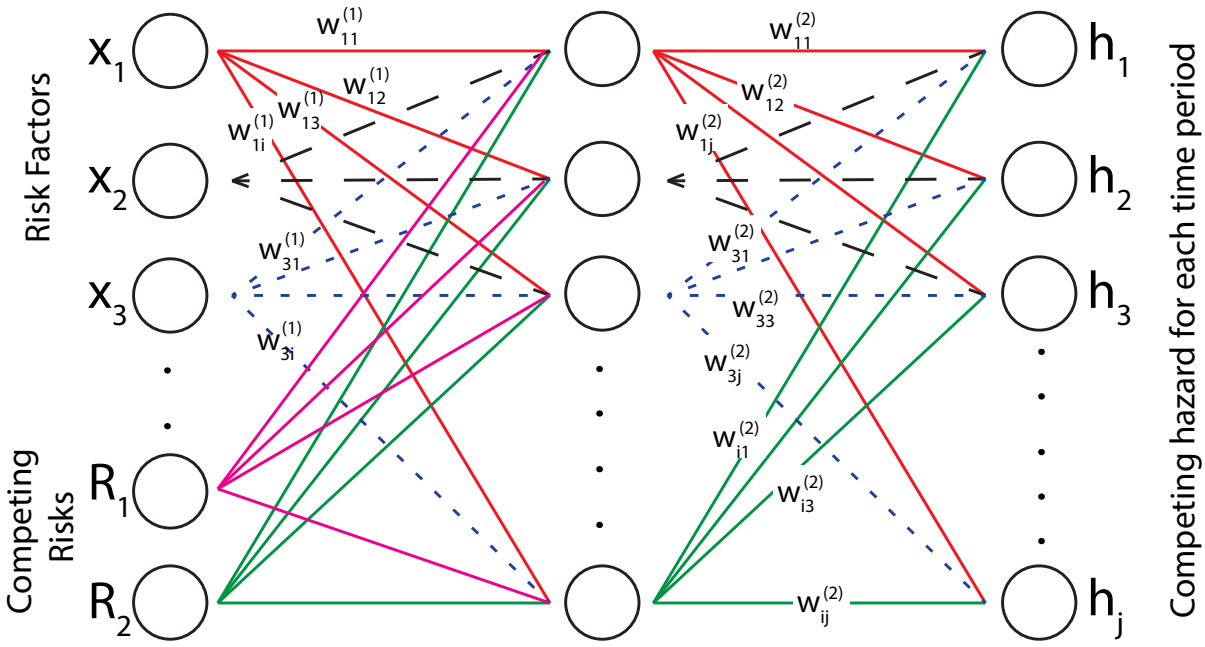

Figure 4.1: The proposed ANN model

$\{0.01, 0.025, 0.5, 0.075, 0.1\}$.

$$E = -\sum_{i=1}^{N}\sum_{j=1}^{J}\Big(h_j \log\big(y(j|r,\boldsymbol{x_i})\big) - y(j,r,\boldsymbol{x_i})\Big)\tau_{ij} + \frac{\alpha}{2}\Big(\sum_{h=1}^{K}\sum_{l=1}^{d}\big(w_{hl}^{(1)}\big)^2 + \big(b_h^{(2)}\big)^2 + \sum_{j=1}^{J}\sum_{h=1}^{K}\big(w_{jh}^{(2)}\big)^2 + \big(b_j^{(2)}\big)^2\Big) \tag{4.8}$$

In the ML approach, we used a 5-fold cross validation technique to find the optimal number of hidden nodes in each network. The optimal network for each decay value is chosen based on the minimum average validation error, and that is used for hazard predictions using the testing data. Automatic relevance determination prior is used to determine the relative importance of the risk factors. When using the HMC and Hybrid Bayesian methods, we used a 5000 burn-in period, prior to sampling.

$$S_i(j) = exp\Big(-\Big(\sum_{j=1}^{J} y(j|r,\boldsymbol{x}_i)\Big)\Big) \tag{4.9}$$

The corresponding survival probabilities are obtained using Eq. (4.9). We evaluated the models using several error measurements calculated based on the predicted median survival time and the actual survival time of the non-censored subjects in the testing data.

4.4.3 The Lung Cancer Data

Table 4.3: Lung cancer patient information

	Male	**Female**
Cause of Death		
Lung	13029(64%)	10303(58%)
Other	2724(13%)	1928(11%)
Censored	4767(23%)	5511(31%)
Age at Diagnosis		
45-49 years	635(3%)	705(4%)
50-54 years	1320(6%)	1161(7%)
55-59 years	2206(11%)	1747(10%)
60-64 years	3208(16%)	2515(14%)
65-69 years	3757(18%)	3127(18%)
70-74 years	3723(18%)	3086(17%)
75-79 years	3187(16%)	2837(16%)
80-84 years	1793(9%)	1826(10%)
85+ years	691(3%)	738(4%)
Stage of the Cancer		
Localized	5536(27%)	5525(31%)
Regional	7028(34%)	5816(33%)
Distant	7956(39%)	6401(36%)
Histology Type		
Adeno	9162(45%)	10056(57%)
Squamous	8492(41%)	5054(28%)
Large Cell	917(4%)	691(4%)
Small-cell	1949(10%)	1941(11%)
Total	20520	17742

The data for our study are selected from the Surveillance, Epidemiology and End Results (SEER) program [77], and it contains details of 38,262 white lung cancer patients data who have been diagnosed from 2004 to 2009. Among these, 23,332 subjects were deceased due to lung cancer and 4,652 were deceased due to some other causes. The rest were considered as censored due to missing information or lost in the follow-up.

In our analysis, four risk factors were used: age at diagnosis, tumor size, histology and the stage of cancer. As can be seen from Table 4.3, a higher amount of patients were between the ages of 65-75 and most of them had distant metastasis. The majority of the patients were diagnosed with adeno or squamous cell carcinoma. The overall median follow-up time for males was 1.33 years and 2 years for females, while median tumor size is about 38 mm and 32 mm for the two groups respectively.

We found that the survival time between males and females to be significantly different from each other, which was already a known fact [78], and hence, two separate analyses were conducted. In order to develop the piecewise constant hazard model, we partitioned the total follow-up time into six disjoint intervals, each with a 12-month period. For our analysis with GEE and ANN models we have used SAS and MATLAB.

4.5 Results

For both males and females, we created a training data set (70%) and a testing data set (30%). The training set was used to train the models while the testing dataset was used to evaluate the prediction accuracies of the proposed models.

We started our analysis by developing Poisson regression models. However, according to the deviance and the Pearson chi-square statistics, none of those models showed adequate results [79] as they are susceptible to correlated observations. Trying several other models, i.e., a Poisson model with an overdispersion parameter and a negative binomial model resulted with the same conclusion. Therefore, we chose an alternative method, generalized estimating equations (GEE).

Using GEE, we created two different statistical models for males and females, which are given in Tables 4.4 and 4.5. We can see that, for each 10 mm increase in the tumor size, the hazard rate for males increases by 4% and by 2% for females. In general, as patients get older, their lung cancer hazard rates get increased. Furthermore, we can see that the patients diagnosed with small cell carcinoma have the highest hazard compared to other histology types. For males, their hazard is 48% higher than the patients with adeno cell

carcinoma. For females, it is about 41%. Over time, the hazard rates seem to increase rapidly for males than females. Applying these two models, we were able to predict the hazard and to obtain the corresponding survival probabilities for our lung cancer testing data.

Table 4.4: Analysis of GEE parameter estimates : males

Risk Factor	Parameter Estimate	Standard Error	95% Confidence Limits		Z	$Pr > \|Z\|$
Intercept	-2.4794	0.0571	-2.5913	-2.3676	-43.45	< 0.0001
tumorsize	0.0044	0.0002	0.0040	0.0049	18.8	< 0.0001
Age 50-54	0.0266	0.0625	-0.0960	0.1491	0.43	0.6707
Age 55-59	0.0545	0.0584	-0.0599	0.1690	0.93	0.3503
Age 60-64	0.1274	0.0556	0.0185	0.2363	2.29	0.0219
Age 65-69	0.1309	0.0546	0.0239	0.2379	2.4	0.0165
Age 70-74	0.3099	0.0542	0.2037	0.4161	5.72	< 0.0001
Age 75-79	0.3622	0.0545	0.2554	0.4689	6.65	< 0.0001
Age 80-84	0.5036	0.0565	0.3929	0.6144	8.91	< 0.0001
Age 85+	0.7313	0.0643	0.6053	0.8573	11.38	< 0.0001
Histology Large-cell	0.3149	0.0469	0.2230	0.4067	6.72	< 0.0001
Histology Small-cell	0.3908	0.0289	0.3341	0.4475	13.5	< 0.0001
Histology Squamous	0.1832	0.0222	0.1397	0.2267	8.25	< 0.0001
Stage Distant	1.3825	0.0268	1.3299	1.4351	51.54	< 0.0001
Stage Regional	0.5644	0.0272	0.5111	0.6177	20.74	< 0.0001
t	0.1436	0.007	0.1298	0.1574	20.39	< 0.0001

Table 4.5: Analysis of GEE parameter estimates : females

Risk Factor	Parameter Estimate	Standard Error	95% Confidence Limits		Z	$Pr > \|Z\|$
Intercept	-2.1196	0.0571	-2.2315	-2.0076	-37.1	< 0.0001
tumorsize	0.0028	0.0002	0.0024	0.0032	15.21	< 0.0001
Age 50-54	0.0410	0.0614	-0.0793	0.1613	0.67	0.5041
Age 55-59	0.0396	0.0576	-0.0732	0.1525	0.69	0.4915
Age 60-64	0.0868	0.0553	-0.0216	0.1952	1.57	0.1164
Age 65-69	0.0903	0.0548	-0.0171	0.1977	1.65	0.0995
Age 70-74	0.2664	0.0545	0.1597	0.3732	4.89	< 0.0001
Age 75-79	0.3601	0.0551	0.2521	0.468	6.54	< 0.0001
Age 80-84	0.4971	0.0579	0.3836	0.6107	8.58	< 0.0001
Age 85+	0.6319	0.068	0.4985	0.7653	9.29	< 0.0001
Histology Large-cell	0.2131	0.0419	0.1311	0.2952	5.09	< 0.0001
Histology Small-cell	0.3452	0.0304	0.2856	0.4048	11.35	< 0.0001
Histology Squamous	0.1293	0.0192	0.0915	0.167	6.72	< 0.0001
Stage Distant	1.2672	0.0251	1.218	1.3165	50.41	< 0.0001
Stage Regional	0.4875	0.025	0.4384	0.5365	19.49	< 0.0001
t	0.1023	0.0072	0.0882	0.1164	14.25	< 0.0001

Next, we created the ANN models with different learning techniques, ML and Bayesian. In each situation, we created several ANN models by varying the number of hidden nodes from 3 to 13 and also used different weight decay values. As mentioned earlier, the optimal networks in the ML method are selected based on the minimum average validation error. In the Bayesian approach, we used the minimum of regularized cost function to find the best set of models. By using each optimal network, we predicted the hazard and corresponding survival probabilities for the testing data. In order to evaluate the prediction accuracies of different ANNs and GEE, we used the actual survival times and their predicted median survival times of non-censored subjects in the same data set. For a better comparison, we calculate several prediction errors, including the root mean square error (RMSE), mean absolute error (MAE), mean percentage error (MPE), and relative squared error (RSE) as given in Tables 4.6 and 4.7.

Table 4.6: Model evaluation for males

Male		**RMSE**	**MAE**	**RSE**	**MPE**	**Data Count**
	GEE	4.0986	3.5155	8.4539	-2.5349	4659
Alpha 0.01	ML	2.3253	1.6900	2.7210	-0.6645	4659
	Evidence	1.5001	1.1147	1.1325	-0.1407	4659
	HMC	1.4942	1.1106	1.1237	-0.1423	4659
	Hybrid	1.4658	1.0922	1.0813	-0.1480	4659
Alpha 0.05	ML	2.2693	1.6412	2.5916	-0.6125	4659
	Evidence	1.5004	1.1164	1.1330	-0.139	4659
	HMC	1.4943	1.1106	1.1237	-0.1423	4659
	Hybrid	1.4655	1.0918	1.0809	-0.1483	4659
Alpha 0.075	ML	2.2144	1.6174	2.4676	-0.5819	4659
	Evidence	1.4898	1.1046	1.117	-0.1524	4659
	HMC	1.4813	1.1008	1.1042	-0.1378	4659
	Hybrid	1.4659	1.0913	1.0813	-0.1530	4659

As per above tables, we can see that the Bayesian approach tends to provide better predictions compared to both GEE and ML methods, for both genders. Although the predictions from Bayesian ANN show negative MPE values, which indicates underestimations of the survival, that is significantly less than of other models. The smallest error values were found with the Hybrid Bayesian approach which was trained using a weight decay

Table 4.7: Model evaluation for females

Female		**RMSE**	**MAE**	**RSE**	**MPE**	**Data Count**
	GEE	4.3146	3.8683	8.6342	-2.9081	3568
Alpha 0.01	ML	2.5209	1.8927	2.9475	-0.8038	3568
	Evidence	1.6075	1.1881	1.1985	-0.3199	3568
	HMC	1.5926	1.1752	1.1764	-0.3352	3568
	Hybrid	1.5899	1.1725	1.1734	-0.3465	3568
Alpha 0.05	ML	2.4737	1.8529	2.8383	-0.7844	3568
	Evidence	1.6027	1.1836	1.1915	-0.3220	3568
	HMC	1.5933	1.1799	1.1775	-0.325	3568
	Hybrid	1.5894	1.1718	1.1718	-0.3615	3568
Alpha 0.075	ML	2.4969	1.8700	2.8916	-0.7757	3568
	Evidence	1.6193	1.2023	1.2162	-0.3228	3568
	HMC	1.5930	1.1780	1.1770	-0.3234	3568
	Hybrid	1.5896	1.1760	1.1720	-0.3255	3568

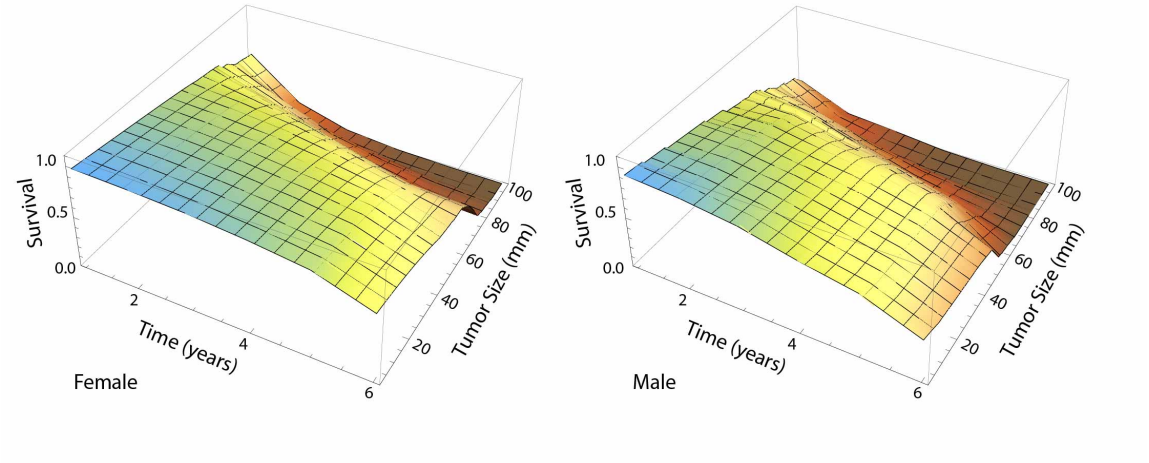

Figure 4.2: Tumor size vs survival probabilities for females and males

value of 0.05, for both genders. Further analysis on patients' hazard and survival was carried out using those two models.

Figure 4.2 depicts the variation in the survival probabilities among males and females patients according to different tumor sizes while keeping the other categorical risk factors in their mode categories. We can see that, as the tumor size increases men tend to have a lesser survival probability compared to females.

It is a known fact there is a significant variations between the hazard rates among the different histology types for different genders. Figure 4.3 represents the variation

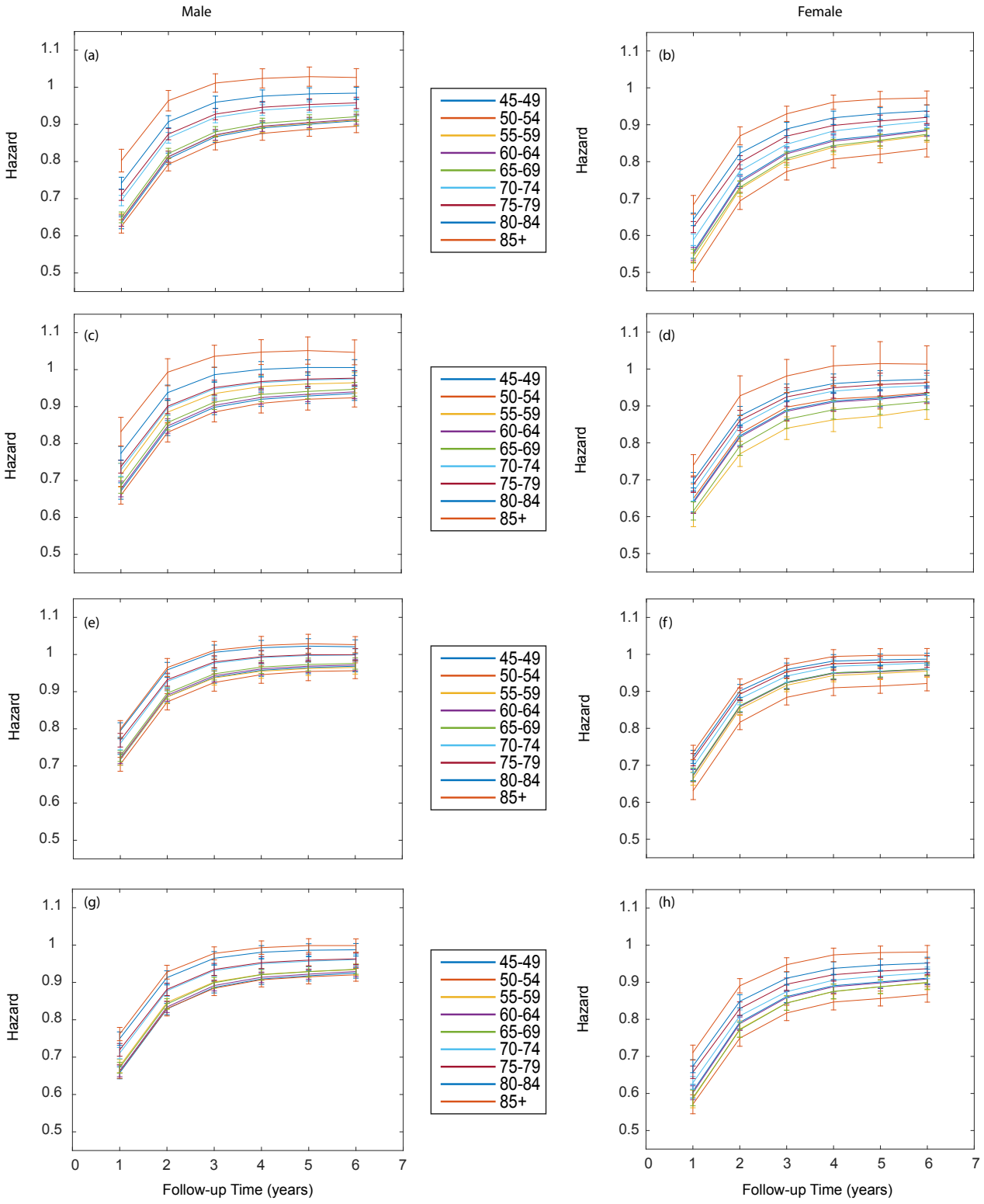

Figure 4.3: Hazard variation for males and females for different histology types (a) Male-Adeno (b) Female-Adeno (c) Male-Large cell (d) Female-Large cell (e) Male-Small cell (f) Female-Small cell (g) Male-Squamous cell (h) Female-Squamous cell

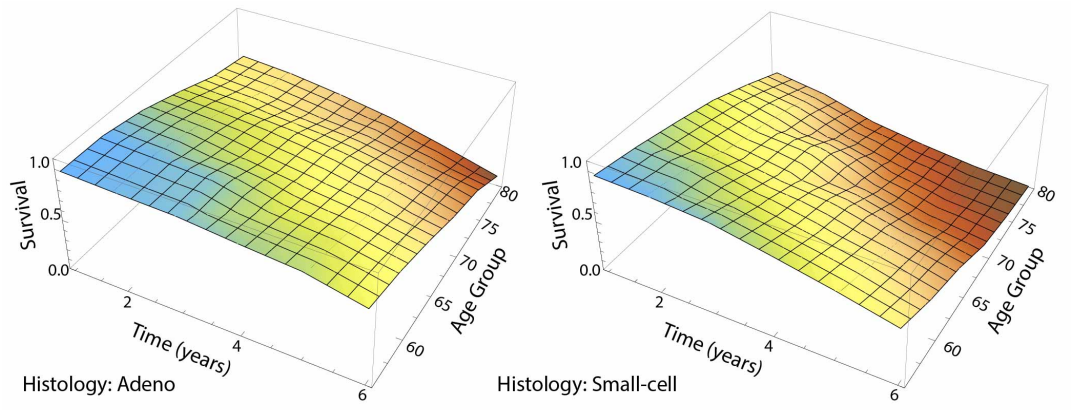

Figure 4.4: Survival probabilities of females with different histology types

in the hazard rates we obtained from our models according to the patients' age group and histology types, for both males and females. The left panel of Figure 4.3 shows the hazard for males with their histology types, adeno, large cell, small cell and squamous cell carcinoma, respectively. The right panel shows the hazard for females for the same histology types. Error bars were created to represent one standard deviation from our mean hazard predictions. Unlike in the GEE approach, these error bars do not effected by the underestimated standard errors of parameter estimates. For all the histology types, males have a higher hazard than females. Moreover, a higher hazard can be seen for the patients who diagnosed with small cell carcinoma. In fact, this is the most dangerous lung cancer out of the four we have considered in our analysis. Older patients show a relatively higher hazard in both genders.

Figure 4.4 manifests the variation in the survival for different age groups, for the patients who diagnosed with small cell carcinoma and adeno cell carcinoma. It further confirms the fact that patients with small cell carcinoma have a significantly lower survival probability compared to the other group. This pattern remains the same for all the age groups.

Figure 4.5 represents the variation in the survival probabilities of males according to the age group and the same histology types, adeno and small cell carcinoma. Similary to

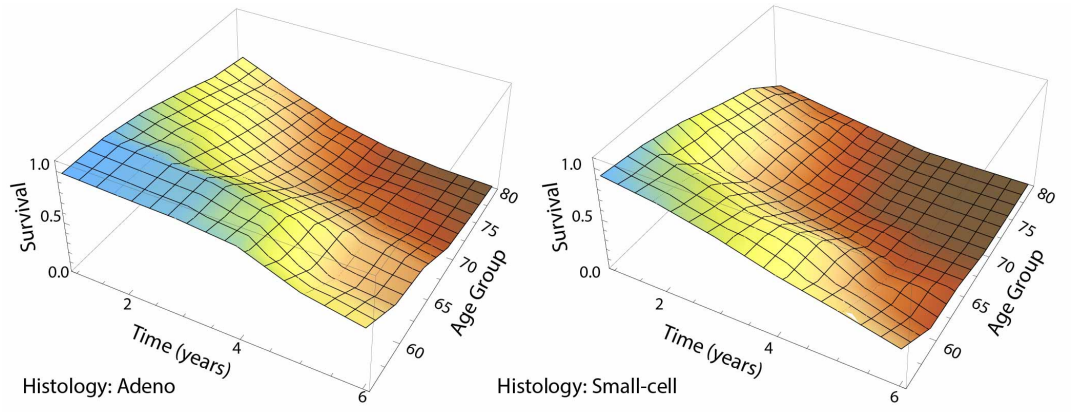

Figure 4.5: Survival probabilities of males with different histology types

females, we can see the same survival patterns as for males. However, in overall, men tend to have lower survival probabilities compared to females [80].

We used ARD prior to identify the relevant importance of the risk factors into the network. Table 4.8 summarizes the rankings of those risk factors based on these hyper-parameter values. Risk factors with smaller hyperparameters are highly contributing to the model outcome. Tumor size and distant metastasis are the top two key factors which highly contribute to the Male ANN model. For females, the most contributing key factors include the distant metastasis and being in the age group of 65. These rankings confirm the fact that our findings have a faithful agreement between the true nature of the lung cancer survival.

4.6 Conclusions and Contributions

We introduce a new neural network model with Bayesian learning to develop the piece-wise constant hazard model. In developing the proposed ANN model, we used the nonlinear Poisson regression model which we presented in Chapter 3. With our new model, we were able to improve the prediction accuracies over conventional methods.

Table 4.8: Relative importance of risk factors for hazard prediction

	Males		Females	
Rank	**Alpha**	**Risk Factor**	**Alpha**	**Risk Factor**
1	0.4892	Tumor Size	0.2179	Distant
2	0.9462	Distant	0.5864	AgeGroup 65
3	1.9458	Age Group 50	1.0550	Age Group 55
4	2.4891	Regional	1.1020	Squamous
5	4.8110	Age Group 55	1.6206	Large cell
6	5.7267	Age Group 80	2.1013	Age Group 85
7	6.7499	Large cell	2.3808	Small cell
8	7.5830	Age Group 75	2.5596	Tumor Size
9	11.1670	Age Group 70	3.2416	Age Group 50
10	13.8046	Squamous	3.8491	Age Group 60
11	16.9652	Age Group 85	4.6063	Age Group 80
12	18.9110	Small cell	6.2623	Age Group 75
13	550.3511	Age Group 65	6.3303	Regional
14	1097.8433	Age Group 60	8.9294	Age Group 70

During the training, network parameters were trained using the back propagation algorithm. In order to compute the Hessian matrix, we have used a special algorithm developed by Pearlmutter [81], using a similar approach like in Nabney [20].

Our contributions to this Chapter can be summarized as follows.

1. A new piecewise constat hazard model is developed with a Bayesian ANN model.

2. With the new ANN model, we have obtained better survival predictions over the other conventional methods.

3. Uncertainties in the hazard predictions can be captured with error bars.

4. The relative importance of the risk factors in predicting the patients' survival can be identified with the automatic relevance determination prior.

CHAPTER 5

BAYESIAN ARTIFICIAL NEURAL NETWORK FOR VULNERABILITY PREDICTION

5.1 Introduction

Operating system vulnerabilities are constant threats to software developing companies and their customers. In past few years, severe security vulnerabilities had lead to disclose sensitive information of general public to unauthorized personnel and had reported a wide impact on their daily lives. We know that almost all the software organizations are interested in developing secure operating systems (OS). Nevertheless, it is impractical to develop an OS without any vulnerabilities [82].

Accurate prediction of future vulnerabilities can help OS companies to make necessary strategic and operational plans. This includes maintenance scheduling, assessing current security risks, estimating the necessary resources needed for handling potential security breaches and secure deployment of operating systems.

Here in this chapter, our goal is to develop a vulnerability prediction model using a special type of artificial neural network called, recurrent neural network (RNN). This model is trained using an online training algorithm based on Hybrid Monte Carlo sampling. In order to develop the proposed model, we have used the vulnerabilities recorded in the Linux operating system.

5.2 Literature Review

Despite the effort given by the software developers in developing secure systems, vulnerabilities are found in each software. The discovering process of these vulnerabilities is a hard and a costly procedure [83]. As a solution, vulnerability prediction models have been

introduced to assist the vulnerability identification process. However, most of these models are based on the source codes of the software and hence require knowledge in specific software languages. Due to this, those models have limited usage [84].

For the first time in 2015, Roumani *et al.* have developed a vulnerability prediction model using a linear time series approach, ARIMA [85]. However, they do not have utilized the nonlinear approaches like artificial neural networks (ANN) and support vector regression (SVR). As per our knowledge, there is no any other study which has utilized a time series approach to predict the future vulnerabilities. Though one may not anticipate, there can be a timely variations in vulnerability identification process. Therefore, we utilized a time series approach to study the vulnerability identification patterns.

Just like for ordinary regression, artificial neural networks are widely used for time series predictions. A wide variety of applications of these can be found in market predictions, meteorological and network traffic forecasting [86–88]. Most of these studies have used feed-forward ANN models in a sliding window format over the input sequence. Time series prediction with SVR spans over many practical application areas from financial market to electric utility load forecasting [89]. SVR models can be easily implemented due to the convex optimization process associated in determining its model parameters.

We have developed a vulnerability forecast model for the Linux operating system. It is one of the oldest open source OS in the market. The popularity of the Linux OS has been significantly increased mainly because it has a zero maintenance cost and relatively high reliability with fewer security risks.

5.3 Vulnerability Data for Linux Operating System

The vulnerability data for the Linux OS are extracted from the National vulnerability database. This is the US government repository that integrates publicly available vulnerability information (refer Appendix A for the calculation procedure of CVSS vulnerability scores). For our analysis, we have used the monthly vulnerability data recorded from January 2001 to December 2016.

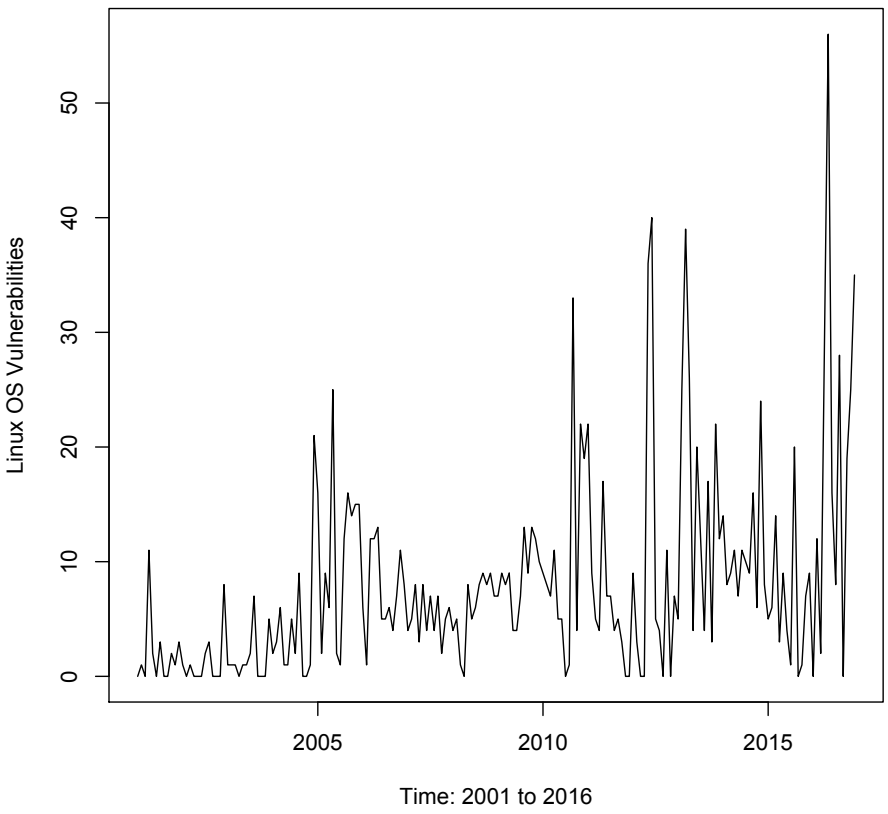

Figure 5.1: Vulnerability patterns for Linux OS from 2001 to 2016

Figure 5.1 shows the pattern of the vulnerabilities associated with the Linux OS. It does not depict any specific seasonal pattern. However, we can see a significant increment in the recorded number of vulnerabilities over time, especially a peak can be seen for the recent years 2014 to 2016. This was due to some severe vulnerabilities detected in Linux OS, which is known as "Heartbleed".

5.4 Methodology

Our target here is to develop a nonlinear time series prediction model with Bayesian neural networks. For the comparison purposes, we have also created time series prediction

models with ARIMA and SVR. Here, we present the details about those methods using a time series $\{x_t, x_{t-1}, ..., x_1\}$.

5.4.1 Auto-Regressive Integrated Moving Average (ARIMA) Model

ARIMA model is the most general class of time series prediction model which was developed by Box and Jenkins in 1970 [90]. In order to use this method, we need to make any non-stationary time series "stationary" by differencing. If necessary, a nonlinear transformation can be applied to make data stationary.

In order to identify whether a time series is stationary, we can use a partial autocorrelation function (PACF) and an autocorrelation function (ACF). The idea is to identify the presence of auto regressive (AR) and moving average (MA) components in the residuals. If there are enough spikes outside the insignificant zone of the ACF or PACF, we can conclude that the residuals are not random. This implies that there is information available in residuals to be extracted by AR and MA models. The analytical form of the ARIMA(p,d,q) model is given by,

$$\Phi_p(B)(1-B)^d x_t = a + \Theta_q(B)\epsilon_t, \tag{5.1}$$

where B is the backshift operator,

$$\Phi_p(B) = 1 - \Phi_1 B - \Phi_2 B^2 - .... - \Phi_p B^p, \tag{5.2}$$

is the AR operator with order p and,

$$\Theta_q(B) = 1 - \Theta_1 B - \Theta_2 B^2 - .... - \Theta_q B^q, \tag{5.3}$$

is the MA oprator with order q. ϵ_t is the error term and it is distributed as $\mathcal{N}(0, \sigma^2)$.

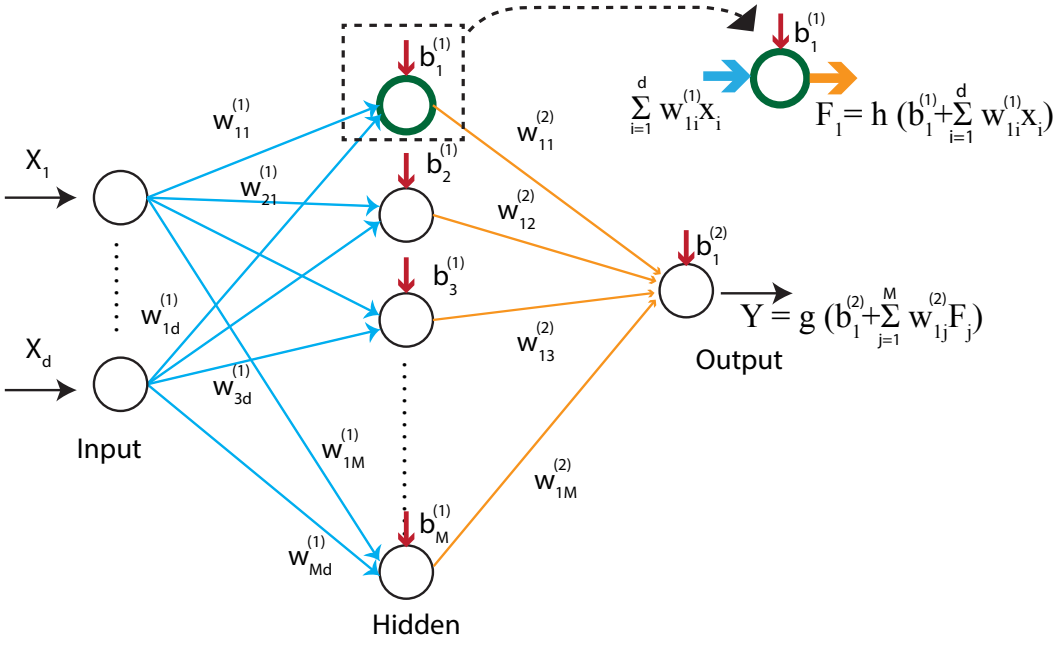

Figure 5.2: A feed-forward ANN model for time series prediction

5.4.2 Artificial Neural Network

On our study, we have considered two types of neural networks. Feed-forward neural networks and recurrent neural networks. Figure 5.2 represents the operational structure of a three layer feed-forward ANN model for a time series model with one output node.

The analytical form of the time series prediction model is given by,

$$x_t = g(b_1^{(2)} + w_k^{(2)} \sum_{k=1}^{M} h(b_k^{(1)} + \sum_{l=1}^{d} w_{kl}^{(1)} x_{t-l})). \tag{5.4}$$

where x_t is the total number of vulnerabilities reported in month t, d is the number of lags (number of vulnerabilities reported in the past d months) and the M is the number of hidden nodes, h and g are the activation functions associated with the hidden and the output nodes.

Recurrent neural networks (RNN) are another widely used neural network models where their outputs act as inputs by looping around. The architecture of an RNN is given in Fig

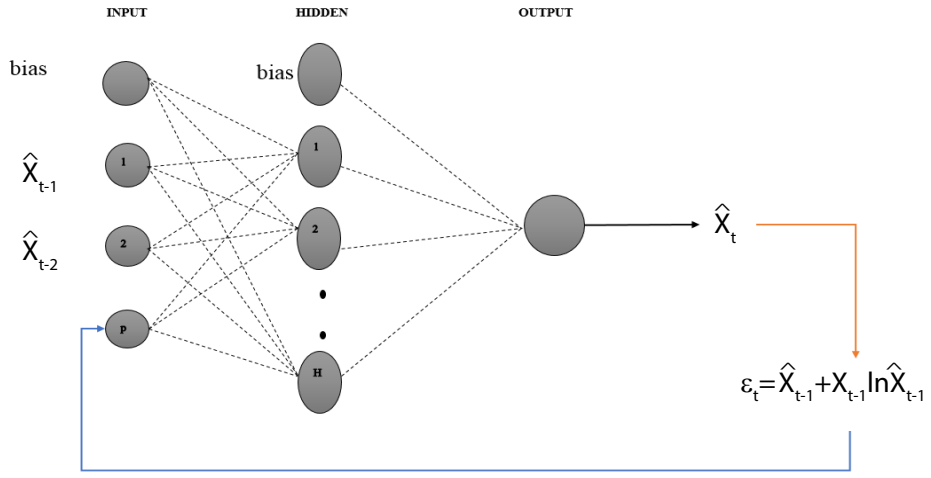

Figure 5.3: A recurrent neural network model for time series prediction

5.3. Unlike in feed-forward neural network, recurrent neural network passes the prediction error as an input to the network which makes them significantly better for time series predictions, specially when we need to add any moving average (MA) terms.

When using neural networks for time series prediction, a major challenging problem is that to find the appropriate number of inputs. There is no specific method to address this problem. Therefore, we had to rely on empirical methods. For our analysis, we tried a different number of inputs (lags) varying from 2 to 6 and a different number of hidden nodes from 3 to 13.

5.4.3 Support Vector Regression

Traditionally, support vector machines are used for classification. These learning algorithms have also been applied to general regression analysis to estimate a function. The application of support vector machines to general regression analysis case is called support vector regression (SVR) and is commonly used for many of the time series prediction applications.

The objective of a time series prediction is to find a function $f(x)$ such that, the predicted value of the time series at a future point in time is unbiased and consistent.

$$f(x) = w^T \phi(x) + bf(x) \tag{5.5}$$

If the data are not linear in its "input" space, the goal is to map the data x, to a higher dimension "feature" space via a kernel function $\phi(x)$, then perform a linear regression in the higher dimensional feature space [89]. The goal is to find "optimal" weights and threshold by minimizing,

$$\frac{1}{2} w^T w + C \sum_{i=1}^{L} (\varepsilon_i + \varepsilon_i^*), \tag{5.6}$$

with respect to the constraints,

$$\begin{aligned} y(x_i) - f(x_i) &\leq \epsilon + \varepsilon_i \\ f(x_i) - y(x_i) &\leq \epsilon + \varepsilon_i^* \\ \varepsilon_i^*, \varepsilon_i &\geq 0, \end{aligned} \tag{5.7}$$

where $y(x_i)$ is the actual targets, ϵ is the highest deviation from $y(x_i)$, C is the regularization parameter and ϵ_i, ϵ_i^* are the slack variables. More details about SVR can be found in [89].

In our analysis, we have implemented above discussed time series models using ARIMA, feed-forward neural networks and recurrent neural networks. For the evaluation purpose we have used the error measurements, mean absolute percentage error (MAPE), root mean square error (RMSE) and mean absolute error (MAE).

5.5 Results

In our data, there were several months with zero vulnerabilities. This may have happened due to some unrecorded vulnerabilities within the database. Therefore we first, followed liner imputation method to replace those zero vulnerabilities. Then we started our analysis by creating ARIMA models.

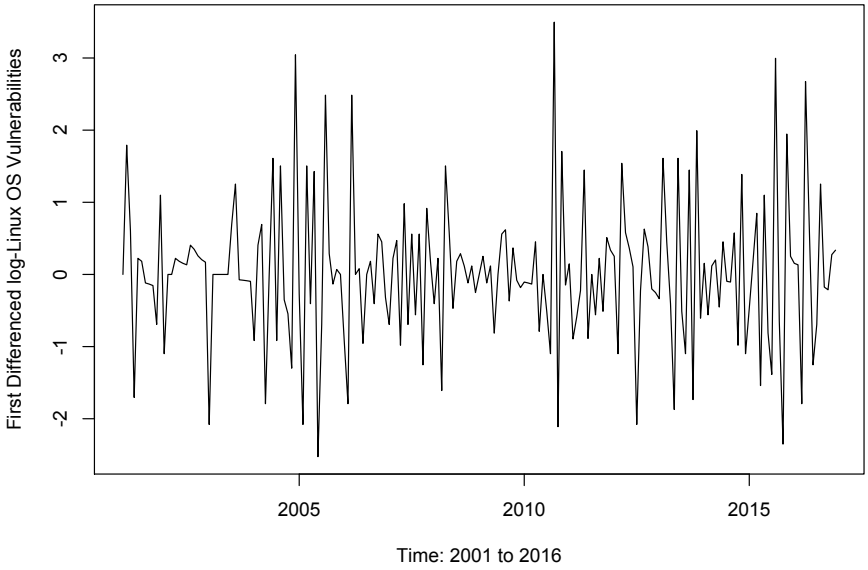

Figure 5.4: First differenced series after log transformation

Prior to the model building process, we first had to stabilize the variance of the nonstationary time series by applying a log transformation. We next applied the first differenced operator to the log transformed series, and it is given in Fig. 5.4.

The corresponding ACF and PACF plots of first differenced series are given in Fig. 5.5. These can be used to tentatively identify the possible AR and MA terms that are needed. After a careful evaluation, we found the best ARIMA model for the log transformed series is having 2 AR terms and 1 MA term with a drift. i.e., ARIMA(2,1,1). The estimated parameters and their standard errors are given in Table 5.1.

Table 5.1: ARIMA model estimates for Linux OS vulnerabilities

	AR(1)	**AR(2)**	**MA(1)**	**drift**
Estimate	0.1846	-0.0141	-0.94076	0.0076
Std. Error	0.0838	0.0822	0.0406	0.0047
AICc=430.77	RMSE=0.7768	MAPE=0.6057	Sample Avg. Res=0.0074	

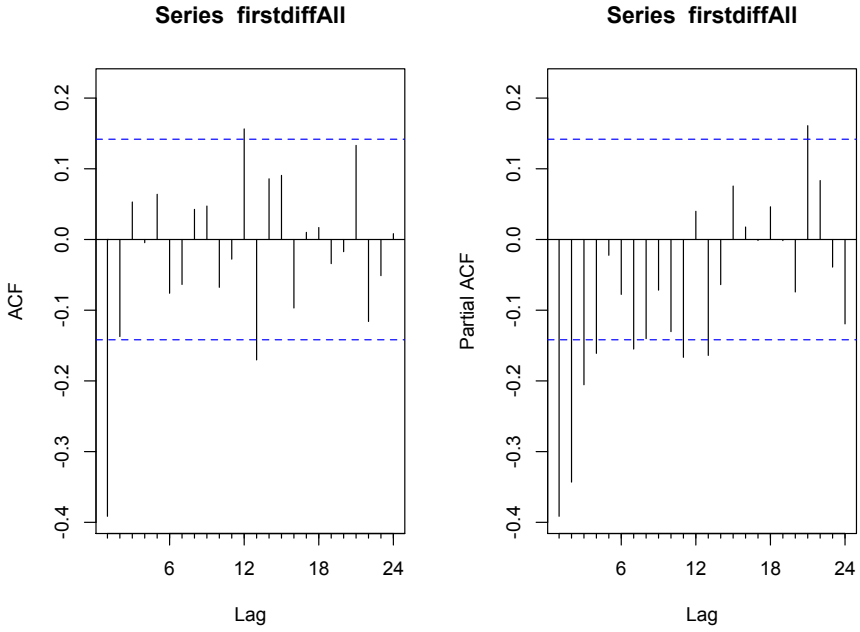

Figure 5.5: First differenced series after log transformation

As per the residual analysis, the sample average residual (SAR) for the training data is 0.0074 which indicates an underprediction from our model. Though we re-trained the model by subtracting SAR from the original data, it did not show a better improvement. Moreover, residuals seem to be randomly scattered, and a higher p-value in L-jung box test confirms the hypothesis that residual autocorrelation is zero. All these facts indicate that the fitted model is adequate.

After creating an ARIMA model, we then proceeded with support vector regression models and neural network models. As discussed earlier, we created both, feed-forward neural networks and recurrent neural networks. Each of these networks was trained using an online training method based on Hybrid Monte Carlo sampling. We have trained ANN models with 3,4 and 5 lags respectively. When we consider 3 lags for ANN-HMC, it means that we have used the last 3 months observations to trained the data. Where as for an RNN model, it means that we have considered last 2 months observations and 1 error term.

Table 5.2: Model evaluations

	Model	**SMAPE**	**RMSE**	**MAPE**	**MAE**
	ARIMA(2,1,1)	0.8234	19.1857	0.7529	14.5379
Lag 3	SVR	0.7716	18.9116	0.7347	13.8508
	ANN-HMC(3 AR)	0.8136	18.9917	0.7735	14.4190
	RNN-HMC(2 AR, 1MA)	0.7032	12.8586	0.8816	10.6879
Lag 4	SVR	0.7227	18.9869	0.7626	14.1552
	ANN-HMC(4 AR)	1.2445	20.0234	0.7798	16.0022
	RNN-HMC(3 AR, 1MA)	0.8032	18.8967	0.8832	14.7632
Lag 5	SVR	0.7805	18.7927	0.7944	14.0540
	ANN-HMC (5 AR)	1.3001	20.9987	0.7853	16.1156
	RNN-HMC(4 AR, 1MA)	0.8122	18.9010	0.8901	14.7734

Table 5.3: RNN Parameter Estimations for the Model with 3 Lags

$w_{kl}^{(1)}$	$b_k^{(1)}$	$w_k^{(1)}$	$b_1^{(2)}$
-4.14	13.73	5.11	10.29
-12.67	-17.96	14.85	
-25.93	20.44	-4.08	
-0.44	7.21	11.22	
-3.10	-3.39	19.87	
-1.36			
-14.18			
4.17			
8.75			
-12.75			
4.46			
0.64			
4.50			
14.36			
8.26			

This error term is calculated using,

$$\epsilon = \hat{x}_{t-1} + x_{t-1}\, ln\hat{x}_{t-1}. \tag{5.8}$$

Table 5.2 summarizes the model evaluations performed using testing data set for each method. From that, we can see that Bayesian RNN model based on HMC sampling has given the minimum SMAPE and RMSE values for the model predictions. Table 5.3 provides the estimated weight and bias values for the RNN model with 3 lags.

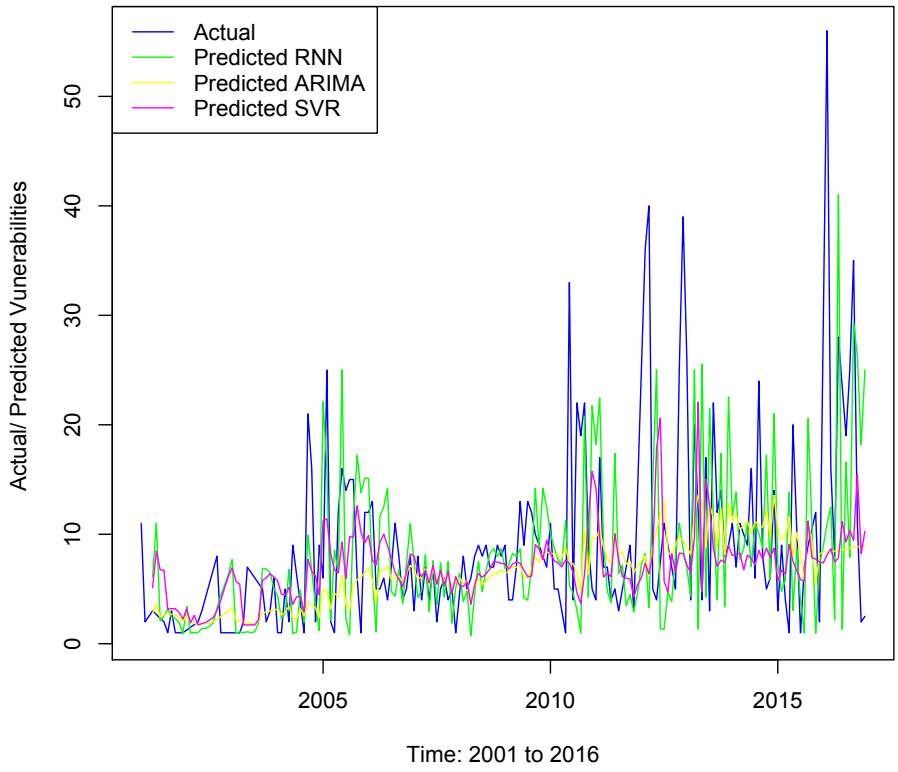

Figure 5.6: Forecasts of the vulnerability prediction model with the best recurrent neural network model

Table 5.4: Actual and predicted vulnerabilities for Linux OS for 2016

2016	Actual	ARIMA(2,1,1) (log scale)	ARIMA(2,1,1)	SVM	ANN	RNN	ARIMA Pred. Diff	SVR Pred. Diff	ANN Pred. Diff	RNN Pred. Diff
Jan	10.5	2.13	8.43	7.38	8.01	8.91	2.07	3.12	2.49	1.59
Feb	12	2.12	8.35	8.10	8.99	10.92	3.65	3.90	3.01	1.08
Mar	2	2.13	8.39	8.68	8.99	12.49	-6.39	-6.68	-6.99	-10.49
Apr	29	2.13	8.46	7.46	8.99	2.19	20.54	21.54	20.01	26.81
May	56	2.14	8.52	7.83	8.99	41.02	47.48	48.17	47.01	14.98
Jun	16	2.15	8.59	11.17	8.99	1.33	7.41	4.83	7.01	14.67
Jul	8	2.16	8.65	9.29	8.99	16.59	-0.65	-1.29	-0.99	-8.59
Aug	28	2.17	8.72	10.36	8.99	7.88	19.28	17.64	19.01	20.12
Sep	23.5	2.17	8.78	9.45	8.99	29.33	14.72	14.05	14.51	-5.83
Oct	19	2.18	8.85	15.48	8.99	26.32	10.15	3.52	10.01	-7.32
Nov	25	2.19	8.92	8.25	8.99	18.21	16.08	16.75	16.01	6.79
Dec	35	2.20	8.99	10.28	8.99	25.02	26.01	24.72	26.01	9.98

Figure 5.6 gives the model prediction over the time. It can be seen that predictions obtained from RNN follow the actual vulnerbility pattern as close as possible. Table 5.4 provides the actual and predicted vulnerabilities from all the methods, for the year 2016.

5.6 Conclusion and Contributions

In this chapter, we have implemented a vulnerability prediction model for Linux OS using artificial neural networks. The best prediction model was given by the recurrent neural network with 2 lags and 1 error term with online HMC learning technique.

It is clear that with the new vulnerability prediction model we can forecast the future vulnerabilities for the Linux OS with a high degree of accuracy. Our model findings can be helpful for the system administrators in arranging future software resources. Here, we summerize our contributions to this chapter.

1. We were able to develop an accurate vulnerability prediction model for the Linux OS.

2. We demonstrate the applicability of an online training algorithm for time series forecasts with neural network models.

3. We have successfully evaluated the performance of feed-forward neural networks and recurrent neural networks in the Bayesian setting.

4. The final predicted value on monthly basis are important to the information technology director in developing his strategic plans for the Linux OS. Based on the predicted value he can decide to redesign or leave it as in the OS.

5. The predicted value can be used to monitor the competitors OS for marketing purposes.

CHAPTER 6

OUTLOOK AND FUTURE RESEARCH WORK

Due to their flexibility in modeling, neural networks serve at the forefront of modern statistical learning. Incorporation of the Bayesian learning into neural networks has led to provide more appealing results. In this work, we have successfully demonstrated the applicability of those techniques in interdisciplinary research. More specifically, in the fields of health and computer science.

The proposed breast cancer diagnosis model is substantially important in assessing the potential risk of diagnosing with malignant breast cancer with a minimal cost. We would certainly be able to improve our predictions by adding more learning data, which is one of our targets to be achieved in the future.

The development of the nonlinear Poisson regression model with Bayesian neural networks opens a new era of hope for modeling count data as well as for survival modeling. More importantly, it provides a method to obtain the standard errors without being affected by the issue of overdispersion associated with the Poisson distribution. Unlike in the maximum likelihood method, the inclusion of automatic relevance prior in the Bayesian setting helps to identify the relative importance of the covariates in an ANN model. Moreover, the proposed method has been extended to develop a piecewise constant hazard model. This significantly improves the accurate evaluation of patients' survival times and potentially can be used by the medical practitioners to make important clinical decisions. We also have explored the possibility of applying the Bayesian neural networks in time series prediction.

During our research, we also have identified several other problems which might lead to future research work. One of that is to explore the possibility of developing a zero-inflated Poisson regression model with the Bayesian ANN to be served in a wide variety

of problems. Another interesting problem that we are planning to do in future is that to use ANN to estimate the commercial real estate appraisals. With such a model, we will be able to save a significant amount of time involved with the appraisal procedure. With that, we may be able to provide an accurate estimate by considering all the important effects which drive prices of the current market.

As we all know, applications of ANN are not just limited to one or two disciplines. It is likely that the Bayesian neural networks be one of the pioneering techniques in the modern era of Big data. We are happy to be a part of that, even with this small piece of work.

REFERENCES

[1] J. H. Friedman, "Data Mining and Statistics: What's the connection?", In Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics, 3 (1997).

[2] G. Shafer, "Why Should Statisticians Be Interested in Artificial Intelligence?", Glennshafer. Com, 1, (1990).

[3] R. C. Rabin, "For Women, a More Complicated Choice on Mammograms", The New York Times, Feb 11, (2014)

[4] T. Ayer, O. Alagoz, J. Chhatwal, J. W. Shavlik, C. E. Kahn, and E. S. Burnside, "Breast cancer risk estimation with artificial neural networks revisited: Discrimination and calibration", Cancer **116**, 3310 (2010).

[5] C. E. Floyd, J. Y. Lo, A. J. Yun, D. C. Sullivan, and P. J. Kornguth, "Prediction of breast cancer malignancy using an artificial neural network", Cancer **74**, 2944 (1994).

[6] R. K. Orr, "Use of an artificial neural network to quantitate risk of malignancy for abnormal mammograms", Surgery **129**, 459 (2001).

[7] Y. Wu, M. L. Giger, K. Doi, C. J. Vyborny, R. A. Schmidt, and C. E. Metz, "Artificial neural networks in mammography: application to decision making in the diagnosis of breast cancer", Radiology **187**, 817 (1993).

[8] C. M. Bishop, *Neural networks for pattern recognition*, Oxford University Press (1995).

[9] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer (2006).

[10] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators, Neural Networks", **2**, 359 (1989).

[11] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis", Doc. Anal. Recognition, Proceedings. Seventh Int. Conf., 958 (2003).

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation", In Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence, 318 (1986).

[13] P. E. Gill and W. Murray, *Practical Optimisation*, Academic Press (1981).

[14] D. J. C. MacKay, "The Evidence Framework Applied to Classification Networks", Neural Comput. **4**, 720 (1992).

[15] D. J. C. MacKay, "Information-based objective functions for active data selection", Neural Comput. **4**, 590 (1992).

[16] D. J. C. MacKay, "Bayesian methods for backpropagation networks". In E. Domany, J.L. van Hemmen, and K. Schulten (Eds.), *Models of Neural Networks III*, Chapter 6. New York: Springer-Verlag (1994).

[17] D. J. C. MacKay, "Bayesian non-linear modelling for the 1993 energy prediction competition". In G. Heidbreder (ed), *Maximum Entropy and Bayesian Methods*, Santa Barbera (1994).

[18] R. M. Neal, Bayesian Learning for Neural Networks. Ph.D. thesis, University of Toronto, Canada (1994).

[19] H. H. Thodberg, "Ace of Bayes: application of neural networks with pruning", Technical Report 1132E, The Danish Meat Research Institute, Maglegaardsvej 2, DK-4000 Roskilde, Denmark (1993).

[20] I. T. Nabney, *Netlab- Algorithms for pattern recognition*, Springer (2002).

[21] L. I. Aizerman, M.A., Braverman, E.M., Rozonoer, "Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning", Autom. Remote Control **25**, 821 (1964).

[22] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers", In Proceedings of Fifth Annual ACM Work. Comput. Learn. Theory, 144 (1992).

[23] V. N. Vapnik, *Statistical Learning Theory*, Wiley (1998).

[24] M. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine", J. Mach. Learn. Res. **1**, 211 (2001).

[25] M.H. Gail, L.A. Brinton, D.P. Byar, D. K. Corle, S. B. Green, C. Schairer, and J. J. Mulvihill, "Projecting individualized probabilities of developing breast cancer for white females who are being examined annually", Journal of the National Cancer Institute **81**, 1879 (1989).

[26] E. B. Claus, N. Risch and W. D. Thompson. "The calculation of breast cancer risk for women with a first degree family history of ovarian cancer", Breast Cancer Research and Treatment **28**, 115 (1993).

[27] S. M. Domchek, A. Eisen, K. Calzone, J. Stopfer, A. Blackwood, and B. L. Weber, "Application of breast cancer risk prediction models in clinical practice", Journal of Clinical Oncology **21**, 593 (2003).

[28] C. J. Van Asperen, M.A. Jonker, C. E. Jacobi, J. E. M. van Diemen-Homan,E. Bakker, M. H. Breuning, and G. H. de Bock, "Risk estimation for healthy women from breast cancer families: new insights and new strategies", Cancer Epidemiology, Biomarkers & Prevention **13**, 87 (2004).

[29] E. Amir, D. G. Evans, A. Shenton, F. Lalloo, A. Moran, C. Boggis, and A. Howell, "Evaluation of breast cancer risk assessment packages in the family history evaluation and screening programme", Journal of Medical Genetics **40**, 807 (2003).

[30] D. M. Euhus, A. M. Leitch, J. F. Huth and G. N. Peters. "Limitations of the Gail Model in the specialized breast cancer risk assessment clinic", The Breast Journal **8**, 23 (2002).

[31] P. Pantel, "Breast cancer diagnosis and prognosis", Report, Department of Computer Science, University of Manitoba, Canada (1998)

[32] M. S. Hung, M. Shanker, and M. Y. Hu, "Estimating breast cancer risks using neural networks", J. Oper. Res. Soc. **53**, 222 (2002).

[33] N. R. Cook, "Statistical evaluation of prognostic versus diagnostic models: beyond the ROC curve", Clin. Chem. **54**, 17 (2008).

[34] C. P. Utomo, A. Kardiana, and R. Yuliwulandari,"Breast Cancer Diagnosis using Artificial Neural Networks with Extreme Learning Techniques", Int. J. Adv. Res. Artif. Intell. **3**, 10 (2014).

[35] R. R. Janghel, A. Shukla, R. Tiwari, and R. Kala, "Breast cancer diagnosis using Artificial Neural Network models", The 3rd International Conference on Information Sciences and Interaction Sciences, 89 (2010).

[36] A. Abu-Hanna and P. J. F. Lucus, "Prognostic Models in Medicine: Ai and Statistical Approaches", Methods of Information in Medicine **40**, 1 (2001).

[37] P. Abdolmaleki, M. Yarmohammadi, and M. Gity, "Comparison of logistic regression and neural network models in predicting the outcome of biopsy in breast cancer from MRI findings", International Journal of Radiation Research **1**, 217 (2004).

[38] V. S. Bourd, S. Bonnevay, P. J. G. Lisboa, M. S. H. Aung, S. Chabaud, T. Bachelot, D. Perol, and S. Negrier, "Breast cancer predictions by neural networks analysis: a comparison with logistic regression", Conference Proceding IEEE Engineering in Medicine and Biology Society **2007**, 5424 (2007).

[39] S. Singh, J. Harini, and B. R. Surabhi. "A novel neural network based automated system for diagnosis of breast cancer from real time biopsy slides", Circuits, Communication, Control and Computing (I4C), International Conference on IEEE, (2014).

[40] D. J. Sargent, "Comparison of artificial neural networks with other statistical approaches", Cancer **91**, 1636 (2001).

[41] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review", J. Biomed. Inform. **35**, 352 (2002).

[42] G. Schwarzer, W. Vach, and M. Schumacher, "On the misuses of artificial neural networks for prognostic and diagnostic classification in oncology", Stat. Med. **19**, 541 (2000).

[43] B. D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press (1996).

[44] C. L. Huang, H. C. Liao, and M. C. Chen, "Prediction model building and feature selection with support vector machines in breast cancer diagnosis", Expert Syst. Appl. **34**, 578 (2008).

[45] H. X. Liu, R. S. Zhang, F. Luan, X. J. Yao, M. C. Liu, Z. D. Hu, and B. T. Fan, "Diagnosing breast cancer based on support vector machines", J Chem Inf Comput Sci **43**, 900 (2003).

[46] H. You and G. Rumbe, "Comparative Study of Classification Techniques on Breast Cancer FNA Biopsy Data", Int. J. Interact. Multimed. Artif. Intell. **1**, 5 (2010).

[47] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods", Adv. large margin Classif. **10**, 61 (1999).

[48] B. M. Gayathri and C. P. Sumathi, "Breast cancer detection using relevance vector machine", Int. J. Appl. Eng. Res. **10**, 37717 (2015).

[49] L. Wei, Y. Yang, R. M. Nishikawa, M. N. Wernick, and A. Edwards, "Relevance vector machine for automatic detection a of clustered microcalcifications", IEEE Trans. Med. Imaging **24**, 1278 (2005).

[50] W. E. Barlow, E. White, R. Ballard-Barbash, P. M. Vacek, L. Titus-Ernstoff, P. A. Carney, J. A. Tice, D. S. M. Buist, B. M. Geller, R. Rosenberg, B. C. Yankaskas, and K. Kerlikowske, "Prospective breast cancer risk prediction model for women undergoing screening mammography", J. Natl. Cancer Inst. **98**, 1204 (2006).

[51] W. D. Penny and S. J. Roberts, "Bayesian neural networks for classification: How useful is the evidence framework?", Neural Networks **12**, 877 (1999).

[52] D. M. Kline and V. L. Berardi, "Revisiting squared-error and cross-entropy functions for training neural network classifiers", Neural Comput. Appl. **14**, 310 (2005).

[53] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms", Pattern Recognit. **30**, 1145 (1997).

[54] C. P. Friedman and J. Wyatt, *Evaluation Methods in Biomedical Informatics*, Springer Science & Business Media (2006).

[55] Y. Xu, J. Kepner, and C. P. Tsokos, "Identify Attributable Variables and Interactions in Breast Cancer", J. Appl. Sci. **11**, 1033 (2011).

[56] https://en.wikipedia.org/wiki/Poisson-regression

[57] M. Kyri and O. Cokluk, "Using Multinomial Logistic Regression Analysis in Artificial Neural Network: An Application", Ozean Journal of Applied Sciences **3**, 259 (2010).

[58] M. J. Mathieson, "Ordinal models for neural networks. Neural networks in financial engineering", In: Refences A-PN, Y. Abu-Mostafa, J. Moody J, A. Weigend (eds), Proceedings of the third international conference on neural networks in the capital markets, 23 (1996).

[59] N. Fallah, H. Gu, K. Mohammad, S. A. Seyyedsalehi, K. Nourijelyani, and M. R. Eshraghian, "Nonlinear Poisson regression using neural networks: A simulation study", Neural Comput. Appl. **18**, 939 (2009).

[60] M. Fornili, F. Ambrogi, P. Boracchi and E. Biganzoli, "Piecewise Exponential Artificial Neural Networks (PEANN) for Modeling Hazard Function with Right Censored Data", In: E. Formenti, R. Tagliaferri, and E. Wit, Eds., *Computational Intelligence Methods for Bioinformatics and Biostatistics*, Springer, 125 (2014).

[61] T. Sharaf, "Statistical Learning with Arti cial Neural Network Applied to Health and Environmental Data", Ph.D. thesis, University of South Florida, USA, (2015).

[62] A. Gelman and D. B. Rubin, "Inference from iterative simulation using multiple sequences", Statistical Science **7**, 457 (1992).

[63] D. R. Cox, "Regression Models and Life-Tables", J. of the Royal Stat. Soc. Series B (Methodological) **34**, 187 (1972)

[64] A. Wienke, *Frailty Models in Survival Analysis*. CRC Press (2010).

[65] M. Aitkin, N. M. Laird, and B. Francis, "A reanalysis of the Stanford heart transplant data", J. of the Amer. Stat. Assoc. **78**, 264 (1983).

[66] G. Han, M. J. Schell, and J. Kim, "Improved Survival Modeling in Cancer Research Using a Reduced Piecewise Exponential Approach", Stat. in Med. **33**, 59 (2014).

[67] M. S. Goodman, Y. Li, and R. C. Tiwari, "Detecting multiple change points in piecewise constant hazard functions", J. of Appl. Stat. **38**, 2523 (2011).

[68] D. Faraggi and R. A. Simon, "R. A neural network model for survival data", Stat. in Med. **14**, 73 (1995).

[69] E. Biganzoli, P. Boracchi, L. Mariani, and E. Marubini, "Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach", Stat. in Med. **17**, 1169 (1998).

[70] P. M. Ravdin and G. M. Clark, "A practical application of neural network analysis for predicting outcome of individual breast cancer patients", Breast Cancer Research and Treatment, 285 (1992).

[71] P. Borachi, E. Biganzoli, and E. Marubini, "Joint modelling of cause-specific hazard functions with cubic splines: An application to a large series of breast cancer patients", Comput. Stat. and Data Anal. **42**, 243 (2003).

[72] S. Diehl and W. Stute, "Kernel density and hazard function estimation in the presence of censoring", J. of Multivar. Anal. **25**, 299 (1988).

[73] J. M. Satagopan, L. Ben-Porat, M. Berwick, M. Robson, D. Kutler, and D. Auerbach, "A note on competing risks in survival data analysis", British Journal of Cancer **91**, 1229 (2004).

[74] T.R. Holford, "The Analysis of Rates and of Survivorship Using Log-Linear Models", Biometrics **(**36), 299 (1980).

[75] N. Laird and D. Olivier, "Covariance analysis of censored survival data using log-linear analysis techniques", Journal of the American Statistical Association **76**, 231 (1981).

[76] D. Mani, J. Drew, A. Betz, and P. Datta, "Statistics and data mining techniques for lifetime value modeling", Knowledge discovery and data mining, 94 (1999).

[77] Surveillance Epidemiology and End Results (SEER) Program, Research Data (1973-2009), Division of Cancer Control and Population Sciences, National Cancer Institute, Surveillance Research Program, Surveillance Systems Branch, (2012).

[78] J. P. Wisnivesky and E. A. Halm, "Sex differences in lung cancer survival: Do tumors behave differently in elderly women?", Journal of Clinical Oncology **25**, 1705 (2007).

[79] A. Pedan, "Analysis of Count Data Using the SAS System", In Proceedings of the Twenty-Sixth Annual SAS Users Group International Conference, Long Beach, 247 (2001).

[80] K. E. Osann, J. T. Lowery and M. J. Schell, "Small cell lung cancer in women: risk associated with smoking, prior respiratory disease, and occupation", Lung Cancer *28*, 1 (2000).

[81] B. A. Pearlmutter, "Fast Exact Multiplication by the Hessian", Neural Comput. **6**, 147 (1994).

[82] M Jimenez, M. Papadakis, and Y. L. Traon, "An Empirical Analysis of Vulnerabilities in OpenSSL and the Linux Kernel", Software Engineering Conference (APSEC) 23rd Asia-Pacific, 105 (2016).

[83] M. Howard and S. Lipner, *The Security Development Lifecycle*, Microsoft Press (2006).

[84] G. McGraw, "Automated code review tools for security", IEEE Computer **41**, 108 (2008).

[85] Y. Roumani, J. K. Nwankpa, and Y. F. Roumani,"Time series modeling of vulnerabilities", Comput Secur **51**, 32 (2015).

[86] T. Edwards, D. S.Tansley, N. Davey, and R. J. Frank,"Traffic Trends Analysis using Neural Network". In Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications, 157 (1997).

[87] D. W. Patterson, K. H. Chan, and C. M. Tan, "Time Series Forecasting with neural nets: a comparative study", In Proceedings of the international conference on neural network applications to signal processing, 269 (1993).

[88] S. Bengio, F. Fessant, and D. Collobert, "A Connectionist System for Medium-Term Horizon Time Series Prediction", In Proceedings of International Workshop Application Neural Networks to Telecoms, 308 (1995).

[89] N. Sapankevych and R. Sankar, "Time series prediction using support vector machines: a survey", IEEE Computational Intelligence Magazine **4**, 24 (2009).

[90] G. E. P. Box, G. M. Jenkins, "Time series analysis: forecasting and control", Holden-Day, San Francisco CA, (1970).

APPENDICES

Appendix A Draft CVSS v2.10 Equations (last revised 3-20-07)

1. $Base\ Score = (0.6 \times Impact + 0.4 \times Exploitability - 1.5) \times f(Impact)$
2. $Impact = 10.41 \times (1 - (1 - ConfImpact) \times (1 - IntegImpact) \times (1 - AvailImpact))$
3. $Exploitability = 20 \times AccessComplexity \times Authentication \times AccessVector$

$$f(Impact) = \begin{cases} 0, \text{ if Impact } 0 \\ 1.176, \text{ otherwise} \end{cases}$$

$$Access\ Complexity = \begin{cases} 0.35, \text{ if high} \\ 0.61, \text{ if medium} \\ 0.71, \text{ if low} \end{cases}$$

$$Authentication = \begin{cases} 0.704, \text{Requires no authentication} \\ 0.56, \text{ Requires single instance of authentication} \\ 0.45, \text{Requires multiple instances of authentication} \end{cases}$$

$$AccessVector = \begin{cases} 0.395, \text{Requires local access} \\ 0.646, \text{Local Network accessible} \\ 1, \text{Network accessible} \end{cases}$$

$$ConfImpact = \begin{cases} 0, \text{none} \\ 0.275, \text{partial} \\ 0.660, \text{complete} \end{cases}$$

Appendix A (Continued)

$$IntegImpact = \begin{cases} 0, \text{none} \\ 0.275, \text{partial} \\ 0.660, \text{complete} \end{cases}$$

$$AvailImpact = \begin{cases} 0, \text{none} \\ 0.275, \text{partial} \\ 0.660, \text{complete} \end{cases}$$

A.1 CVSS Temporal Equation

TemporalScore = BaseScore

1. Exploitability
2. RemediationLevel
3. ReportConfidence

$$Exploitability = \begin{cases} 0.85, \text{unproven} \\ 0.90, \text{proof-of-concept} \\ 0.95, \text{functional} \\ 1, \text{high} \\ 1, \text{not defined} \end{cases}$$

$$RemediationLevel = \begin{cases} 0.87, \text{official-fix} \\ 0.90, \text{temporary-fix} \\ 0.95, \text{workaround} \\ 1, \text{unavailable} \\ 1, \text{not defined} \end{cases}$$

$$ReportConfidence = \begin{cases} 0.90, \text{unconfirmed} \\ 0.95, \text{uncorroborated} \\ 1, \text{confirmed} \\ 1, \text{not defined} \end{cases}$$

A.2 CVSS Environmental Equation

$$Environmental\ Score = (Adjusted\ Temporal + (10 - AdjustedTemporal) \times Collateral\ Damage\ Potential) \times TargetDistribution$$

AdjustedTemporal = TemporalScore recomputed with the Impact sub-equation replaced with the following AdjustedImpact equation.

$$AdjustedImpact = Min(10, 10.41 \times (1 - (1 - ConfImpact \times ConfReq) \times (1 - IntegImpact \times IntegReq) \times (1 - AvailImpact * AvailReq)))$$

Appendix A (Continued)

$$CollateralDamagePotential = \begin{cases} 0, \text{none} \\ 0.1, \text{low} \\ 0.3, \text{low-medium} \\ 0.4, \text{medium-high} \\ 0.5, \text{high} \\ 0, \text{not defined} \end{cases}$$

$$TargetDistribution = \begin{cases} 0, \text{none} \\ 0.25, \text{low} \\ 0.75, \text{medium} \\ 1.00, \text{high} \\ 1.00, \text{not defined} \end{cases}$$

$$ConfReq = \begin{cases} 0.5, \text{low} \\ 1, \text{medium} \\ 1.51, \text{high} \\ 1, \text{not defined} \end{cases}$$

$$IntegReq = \begin{cases} 0.5, \text{low} \\ 1, \text{medium} \\ 1.51, \text{high} \\ 1, \text{not defined} \end{cases}$$

Appendix A (Continued)

$$AvailReq = \begin{cases} 0.5, \text{low} \\ 1, \text{medium} \\ 1.51, \text{high} \\ 1, \text{not defined} \end{cases}$$