Chapter 4

METHODOLOGY

This section describes the models used in the experiments of breast cancer survival prediction, namely the statistical and neural network models. Both the models used the same data set, and the comparisons are made based on these data set.

4.1 The Data Set

In this research, data was collected from University Malaya Medical Centre (UMMC), Kuala Lumpur form 1993 to 2002. The available data was taken from the patients' very first diagnosis, and this amounted to around 1001 cases. Table 4.1 represent the prognosis indicators used for the analysis:

Table 4.1: Prognosis Indicator

Name	Description
AGE	Patient's age in year at time first diagnosis.
RACE	Ethnicity (Chinese, Malay, Indian and Others)
STG	Stage (how far cancer has spread anatomically)
T	Tumour type (the extent of the primary tumour)
N	Lymph node type (amount of regional lymph node involvement)
M	Metastasis (presence or absence)
ER	Estrogen receptor (negative or positive)
GD	Tumour grade
PT	Primary treatment (type of surgery performed)
AC	Adjuvant Chemotherapy
AR	Adjuvant Radiotherapy
AT	Adjuvant Tamoxifen

The dataset has to be cleaned of errors and be made free from redundancy in order to be useful for the final analysis. This is to guarantee the accuracy of the results. The collection of patients' records may be incomplete and include erroneous submissions. Thus, incomplete records and records with empty fields need to be removed from the dataset.

In survival analysis, survival time of an individual is very important. Many patients become untraceable as their death are not reported or some divert to other hospitals for further treatment. To counter this problem, the records of patients lost in the follow-up procedures are sent to the Registration Department of Malaysia to ascertain whether the patients are still alive. Subsequently, the dates of the patients demise are recorded to obtain their survival time.

After cleansing, remaining dataset were subsequently used in the Kaplan-Meier analysis and Backpropagation network analysis. A brief explanation on Kaplan-Meier method can be found in the next section.

4.2 Statistical Model

Statistical methods are usually used to model and predict survival data with the ability to handle censored data. Survival curves are used in estimating patient's survival with a certain disease progression such as type and stage of cancer. Part of the work done in this project was to aggressively utilise the Kaplan-Meier method for analysis purpose in the area of breast cancer.

The Kaplan-Meier method, also called the Product Limit Estimator, is a model where a clear interval is created for each occurrence of death. It involves computing the number

of people who died at a certain time, divided by the number of people who were still in the study at that time and multiply these probabilities by any earlier computed probabilities (Simon, 2002).

The Kaplan-Meier survival curve is graphically illustrated, comparing a number of curves for different subject groups. Figure 4.1 provides a visual representation of Kaplan-Meier analysis of breast cancer by the stage of the cancer. The results carried out using Kaplan Meier model is discussed in Appendix B.

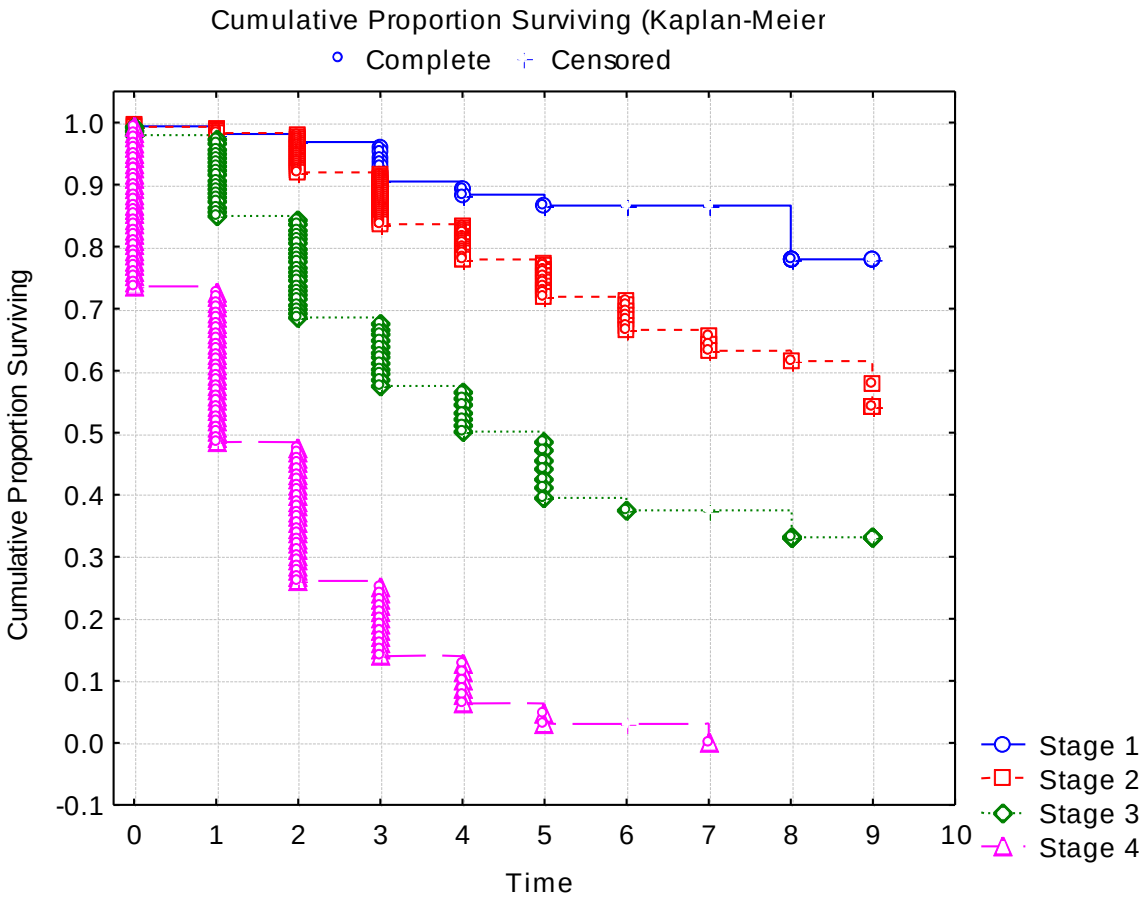

Figure 4.1: Kaplan-Meier Survival Curve of Breast Cancer by Stage

4.3 Neural Network Models

In medical research, multi layer perceptrons that use backpropagation training are most commonly used (Burke et al., 1997). Backpropagation networks are networks where

signals travel in one direction from input neuron to an output neuron without returning to its source. Backpropagation network consists of at least three layers of units: an input layer, at least one hidden layer and an output layer. The output from the input layer is connected as an input into the hidden layer. The output from the hidden layer however is connected as an input into the output layer to produce output.

Matlab, which stands for Matrix Laboratory, is a very powerful and comprehensive suite of various tools for mathematical calculation, visualization, implementation and simulation of neural networks. There are numerous useful toolboxes with generic functions in Matlab such as signal processing, simulations, projection, control systems, fuzzy logic, wavelets and neural networks (Demuth & Beale, 2000). The Neural Network Toolbox that provides comprehensive support for many proven network paradigms is used in this project.

4.3.1 Transfer Functions

The transfer function is a function used to transform the activation level of a neuron into an output signal. The behaviour of the ANN depends on both the weights and the activation function that is specified for the neuron. More importantly, each layer has its own transfer function. Usually, in the hidden layers of networks the sigmoid functions are used followed by the linear transfer function in the output layer. The Matlab neural network toolbox provides a variety of transfer functions. The sigmoid transfer functions used in this project are the tan-sigmoid in each hidden layer while the linear function is purelin.

4.3.2 Training Algorithms

In neural network training there are two different styles of training, namely, incremental and batch training. In incremental training the weights and biases of the network are updated after the presentation of each single training sample, while in batch training the weights and biases of the network are updated after all of the inputs have been presented.

Training algorithms are mathematical procedures used to automatically adjust the network's weights and biases. The training algorithm dictates a global algorithm that affects all the weights and biases of a given network. There are several different training functions supported by Matlab. The following describes the training algorithms used in this project.

4.3.2.1 Resilient Backpropagation

The sigmoid transfer function also known as "squashing" function is often used as an activation function for neural networks. They are typically used in the hidden portion of multi layer networks. These functions compress an infinite input range into a finite output range, where their slopes approach zero as the input increases. Problems arise when sigmoid functions are applied to the steepest descent, since the gradient can have a very small magnitude and therefore, cause small changes in the weights and biases even though they are far from their optimal values. The Resilient Backpropagation (Rprop) training algorithm eliminates this effect by modifying the update-value for each weight according to the behaviour of the sequence of partial derivatives in each dimension (Riedmiller & Braun, 1993). If the derivative is zero, then the update value remains the same. Whenever the weights are oscillating, the weight change will be reduced. If the weight continues to change in the same direction for several iterations,

then the magnitude of the weight change will also be increased. **Trainrp** is a network training algorithm that updates weight and bias values according to the Rprop.

4.3.2.2 Levenberg-Marquardt Backpropagation

Levenberg-Marquardt works by making the assumpti03ons that the underlying function being modeled by the neural network is linear (Statsoft, 2003). This algorithm is designed specifically to minimize the sum-of squares error function. Based on this calculation, the minimum value can be determined exactly in a single step. The calculated minimum is then tested, and if the error is smaller, the algorithm moves the weights to the new point. This process is repeated iteratively on each generation. Levenberg-Marquardt therefore compromises between the linear model and a gradient-descent approach. Successful steps are accepted and lead to a strengthening of the linearity assumption, which is approximately true near to a discernible minimum. Unsuccessful steps are rejected and lead to a more cautious downhill step. Thus, Levenberg-Marquardt continuously switches its approach and can make very rapid progress.

Levenberg-Marquardt uses the update formula:

$$\Delta \mathbf{W} = - (\mathbf{Z}^{\mathbf{T}}\mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^{\mathbf{T}} \boldsymbol{E}$$

where E is the vector of case errors, and Z is the matrix of partial derivatives of errors with respect to the weights. When the scalar λ is zero, this is just Newton's method, using the approximate Hessian matrix. When λ is large, this becomes gradient descent with a small step size.

Trainlm is a network training function that updates weight and bias values according to the Levenberg-Marquardt method.

4.3.2.3 Scaled Conjugate Gradient Backpropagation

The conjugate gradient algorithms require a line search at each iteration. This line search is computationally expensive, since it requires the network to respond to all training inputs that are computed several times for each search. The scaled conjugate algorithm was designed to avoid the time-consuming line search. This algorithm combines the model-trust region approach used in Levenberg-Marquardt algorithm with the conjugate gradient approach. **Trainscg** is a network training function that updates weight and bias values according to the scaled conjugate gradient method.

4.3.2.4 One Step Secant Backpropagation

The one step secant (OSS) method is an attempt to bridge the gap between the conjugate gradient algorithm and the quasi-Newton algorithm. This algorithm does not store the complete Hessian matrix but assumes that at each iteration the previous Hessian was the identity matrix with the added advantage that the new search direction can be calculated without computing a matrix inverse value. **Trainoss** is a network training function that updates weight and bias values according to the one step secant method.

4.3.3 Pre-processing

Pre-processing is a process that convert inputs and targets into a form understandable or acceptable by the neural network. Pre-processing the network inputs and targets will make the neural network training more efficient (Abdul-Kareem et al., 2001). There are a few techniques in this field and two specific techniques used in this project are Principal Component Analysis (PCA) and “One of N” representation.

4.3.3.1 Principal Component Analysis

Principal Component Analysis (PCA) is a procedure used to reduce the dimension of the input vectors. The Matlab Neural Network Toolbox provides this pre-processing technique. The PCA technique has three effects:

- it orthogonalises the components of the input vectors so that they are uncorrelated with each other,
- it orders the resulting orthogonal components (principal components) so that those with the largest variation comes first, and
- it eliminates those components that contribute the least to the variation in the data set.

4.3.3.2 One of N Representation

The "One of N" pre-processing technique is a process that transforms the input variable into binary values. This method implies that each categorical value is handled as a separate input (Berry and Linoff, 1997). For example, race, which can be either Malay, Chinese, Indian or Others, would be represented by four inputs. Table 4.2 illustrates the "One of N" representation. The "One of N" technique is ideal for use with binary outputs in the form of 1s and 0s. The problem with this technique is that if the numbers of categories grow, the transformation of the binary digit increases. A system has been developed to convert the input into a binary format by using visual basic programming.

Table 4.2: One-of-N Representation

Variable	Category	Transformed				
Race	Malay	1	0	0	0	1000
	Chinese	0	1	0	0	0100
	Indian	0	0	1	0	0010
	Others	0	0	0	1	0001

4.3.4 Survival Intervals

The survival time is classified according to predefined intervals. Data of patients were divided in sets according to the survival time (1 year, 2 years, etc.) in different periods after diagnosis of breast cancer. The training process is done in each of the intervals of survival time. 1 indicates the patient is dead while 0 means the patient is still alive for a particular survival interval. Table 4.3 represents the example of survival intervals. Column 3 of the table indicates a patient's actual survival of 4 year 3 months. This patient would then be given a "0" to indicate "alive" for intervals Year 1, Year 2, Year 3 and Year 4 and a "1" to indicate "dead" for Year 5, Year 6, Year 7, Year 8 and Year 9.

Table 4.3: Survival Intervals

	Actual Survival					
	1 year 3 month	**3 year 9 month**	**4 year 3 month**	**5 year 9 month**	**7 year 10 month**	**9 year 2 month**
Survival Year						
Year 1	0	0	0	0	0	0
Year 2	1	0	0	0	0	0
Year 3	1	0	0	0	0	0
Year 4	1	0	0	0	0	0
Year 5	1	1	1	0	0	0
Year 6	1	1	1	0	0	0
Year 7	1	1	1	1	0	0
Year 8	1	1	1	1	0	0
Year 9	1	1	1	1	1	0

4.3.5 Post-processing

The performance of a trained network can be measured by performing a regression analysis between the network response and the corresponding targets. The regression

analysis returns three parameters namely m, b, and R. The m and b correspond to the slope and the y-intercept of the linear regression relating targets to the network outputs. A perfect fit, where the outputs exactly match the targets, would give a slope of 1 and an intercept of 0. The R-value (correlation coefficient) is a measure of how well the variation in the output is explained by the targets. If the R-value is equal to 1, the correlation between targets and outputs is almost perfect. The post-processing method is used in our project to test the training capability of the network. The PCA and post-processing procedure are automatically carried out using Matlab's prepca and postreg procedure respectively.

4.3.6 Generalisation

Generalization measures the ability of the network to correctly classify new unseen data. It is important that a trained neural network has the ability to generalise and not memorize the training in order to avoid overfitting. Overfitting can occur when the error on the training set is driven to a very small value, but when new data is presented to the network the error becomes large.

Early Stopping is a method which can improve the generalization of the Neural Network. In this technique the available data is divided into three subsets. The first subset is the training set which is used for computing the gradient and updating the network weights and biases.

The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error will normally decrease during the initial phase of training, as does the training set error. However, when the network begins to over fit the data, the error on the validation set will typically begin to increase.

If the error of the validation set begins to increase for a specified number of iterations, the training will automatically stop, and the weights and biases at the minimum value of the validation error are reinstated.

The test set error is not used during the training, but it is used to compare different models. It is also useful to plot the test set error during the training process. If the error in the test set reaches a minimum at a significantly different iteration number than the validation set error, this may indicate a poor division of the data set.

Early stopping can be used with any of the training functions by simply passing the validation data to the training function.

4.3.7 Elman Network

Elman network commonly is a two-layer network with feedback from the first layer output to the first layer input. This recurrent connection allows the Elman network to both detect and generate time-varying patterns. The connections are mainly feed forward but also include carefully chosen feedback connections that enable the network to remember "cues" from the recent past (Addison et al., 2003). The input layer is divided into two parts: the true inputs and the context units that hold a copy of the activations of the hidden units from the previous time steps. This allows the network to recognise sequences and also to produce short continuations of known sequences. The Elman network is used in this project to compare the training performance with that of the feed forward Backpropagation network.

4.3.8 Cross Validation

Cross Validation is a technique usually used to validate the data set. In this technique, the data is divided into several groups with each group containing approximately the same number of cases. In our cases the data is divided into five groups. One group is selected as a test set and another four groups is the training set. The training process is repeated five times, and each group was used once as a test set. Percent accuracy for the five groups is then calculated and the average of all groups is used as the result of the percent accuracy using cross validation.

4.4 Evaluation Methods

4.4.1 Percent Accuracy

The accuracy of the predictions is measured by the number of correctly predicted cases divided by all the cases in the study. A threshold is used to convert the outcomes of decimal number into '1' if the prediction outcome is equal or more than 0.5 or '0' if the prediction outcome is less then 0.5. Table 4.4 summarizes this problem.

Table 4.4: Percentage Accuracy

	Actual	Prediction	Outcome
	1	0.85589	1*
	0	0.69877	1
	0	0.15554	0*
	1	0.99881	1*
	1	0.51515	1*
	0	0.36455	0*
	1	0.74413	1*
	0	0.33333	0*
	1	0.58872	1*
	1	0.27112	0
Total	10 (cases)		8 (correctly)

* are predicted correctly

The percentage of accuracy is calculated as follows:

Percentage Accuracy = 8/10 * 100

= 80%

4.4.2 Chi-Square Test

Chi-square (χ^2) is a statistical test commonly used to compare observed data with data expected to obtain according to a specific hypothesis. The chi-square tests what scientists call the null hypothesis, which states that there is no significant difference between the expected and observed result. The p value will be considered to accept or reject the null hypothesis. If the p value for the calculated χ^2 is $p > 0.05$, the null hypothesis is accepted otherwise it is rejected.

4.5 Summary

The breast cancer data set was analysed using the Kaplan-Meier method and the Backpropagation neural network using the same data set. In the neural network method pre-processing of the data set was done to improve the efficiency of its training. A total of four different training algorithms namely, Resilient, Levenberg-Marquardt, Conjugate Gradient, and One Step Secant were used in the neural network experiments. Two different pre-processing techniques namely Principal Component Analysis and "One of N" are also used in the neural network experiments. The comparison between the results obtained from each of these experiments can be found in chapter 5.