uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Dajie Zhou

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Systems Science)

GRADE / DEGREE

Department of Systems Science

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Adding Probability to Neural Network Prediction of NICU Mortality

TITRE DE LA THÈSE / TITLE OF THESIS

Professor Monique Frize

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Professor Tet Yeap

Professor Mustapha Yagoub

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Adding Probability to Neural Network Prediction of NICU Mortality

By Dajie Zhou

A thesis submitted to the Faculty of Graduate and Postdoctoral Studies

in partial fulfillment of the requirements for the degree of

Master of Science, Systems Science

University of Ottawa

June 10th, 2006

Your file Votre référence
ISBN: 978-0-494-18489-9
Our file Notre référence
ISBN: 978-0-494-18489-9

Table of Content

Acknowledgments

My sincerest appreciation goes to my graduate advisor Dr. Monique Frize. Thank you for offering me the opportunity to become a member of our research group. Without your guidance, patience, encouragement and support, I cannot enjoy the art of scientific research and accomplish this thesis work.

Many thanks to all the members of MIRG group, Dr. Robin C. Walker, Christophe Herry and Doaa Swailum Ibrahim, etc. I appreciate your friendship, feedback and help.

Most importantly, I would like to acknowledge the continuous support from my family: my mother Prof. Meijuan Wu and my father Zhongming Zhou. Thanks for your advices to maintain myself in an optimistic and positive attitude. A special thank to my mother's invention: a special functional food to keep me in excellent health during stressful studies.

Glossary

Hypothesis: Any verbal statement is a hypothesis. A hypothesis in data mining/machine learning/statistics context is a concept/cause-effect relationship/explanation of some phenomena that can be tested or argued by the analysis on data. Statistically, there exists null hypothesis and alternative hypothesis (the alternative to null hypothesis). A null hypothesis is the first statement that we made from empirical observation or experiment results. A hypothesis space is a predefined space for potential hypotheses.

Goodness-of-Fit Test: Goodness of fit represents the conformity of the observed data's distribution with the posited distribution. The goodness-of-fit test is a statistical test in which the validity of the null hypothesis is tested. The Hosmer-Lemeshow goodness of fit test compares observed and expected number of outcomes over all categorized deciles.

P-value: P-value measure of how much evidence we have against the null hypotheses. The smaller the p-value, the stronger evidence we can reject the null hypothesis. It assesses how likely to get certain result that correspondence the observed samples, assuming that the null hypothesis is true.

Statistical Significance: A test result is not statistically significant when P-value based on a null hypothesis is less than the critical value α= 5% or 1%,

Constant Predictor: Constant predictor classifies all patterns in a data set to the category with the highest a priori probability of the data set.

MIRG. Medical Information technology Research Group located in Ottawa, ON. Principle investigators: Dr. Monique Frize, P.Eng., O.C. (Carleton University and University of Ottawa) and Dr. C. Robin Walker, MBBS, FRCPC (Children's Hospital of Eastern Ontario and University of Ottawa).

Maximum Likelihood Estimation: The maximum likelihood estimation is a statistical method used to infer the underlying probability distribution of a population on a given data set.

Neonatal Mortality: Defined by the World Health Organization as death occurring before 28 days of age.

Acronyms

ANN. Artificial neural network

APACHE. Acute physiology and chronic health evaluation

CBR. Case-based reasoner

KNN. K-nearest neighbour algorithm

MLP. Multi-layer Perceptorn

NICU. Neonatal intensive care unit

NTISS. Neonatal therapeutic intervention scoring system

NBC. Naïve Bayes Classifier

ROC. Receiver operating characteristic curve

SNAPPE-II. Score for neonatal acute physiology' with Perinatal Extension-II

ANN-PPF. Artificial neural network based probability prediction framework

List of Tables

List of Figures

Abstract

It has been shown that neural networks can be trained to predict clinical outcomes of a neonatal intensive care unit (NICU). This thesis expands past research and presents a neural network approach that predicts the probability of NICU mortality. The resulting neural network models are able to not only classify the dichotomous outcome (i.e. survival or death) but also estimate the probability of death.

The general conditions necessary for the neural network to estimate probabilities are discussed in the thesis. The neural network estimation of mortality probability includes the following steps: modeling the neural network cost function with the likelihood function; deriving the gradient ascent training algorithm to perform the maximum likelihood estimation; developing the neural network models with the NICU data; evaluating performance by the Receiver Operating Characteristic Curve and Hosmer-Lemeshow test.

These neural network based probability estimation models applied as the mortality prognostic tools are presented. For this purpose, two approaches for improving the models' sensitivity are suggested: adjustment of the cost function and the cutoff point. Both of them were tested and results are discussed.

Chapter 1 Introduction

1.1 Motivation

All parents expect to have healthy babies. The neonatal intensive care unit (NICU) is equipped with advanced neonatal medical devices, professional physicians and health care staff to provide special treatment for sick newborns. Babies are usually admitted to the NICU if they are born prematurely, or face illness with high risk of death, or if complications occur during delivery. The estimation of the neonatal mortality risk in the NICU has implications with the provision of aggressive patient care and management of medical resources [Stevens et al. 1994]. The timely, objective and accurate judgment of the mortality risk not only helps to control the in-hospital mortality rate but also reflects the quality of the hospital services in obstetric and neonatal care.

For this purpose, researchers and physicians explored and developed various approaches to improve diagnosis in the NICU. Scoring systems such as NTISS[Gray et al. 1992], CRIB/CRIB-II[INN 1993, Parry et al. 2003], SNAP/SNAPPE-II[Richardson et al. 2001] were developed as decision support tools to study infants' severity of illness and risk of death. These scores were calculated from the sum of sub-scores assigned by physicians based on patients' physiological parameters, such as birth weight, blood pressure, etc. A high score usually represents higher risk of death. The availability of reliable and valid instruments to measure the severity of illness allows the conduction of unbiased comparisons in benchmarking [Field et al. 2002] and quality of care studies. Moreover, they can better define populations of neonates within clinical trials, outcome evaluations, or resource utilization studies [Pollack et al. 2000]. Although scoring systems are widely used by NICU physicians to evaluate patients' potential outcomes in order to decide on the course of therapy, they do not directly predict the

probability of mortality. Estimating mortality probability provides physicians with numerical comparison on outcomes and the illness of severity.

With the development of computer technologies and computing capabilities, artificial intelligence systems have been increasingly adopted as medical decision support tools since the 1990s. Various approaches such as Bayesian Network, Neural Network, Principal Component Analysis (PCA) and Decision Tree were introduced by researchers and computer engineers and scientists to analyse medical data and assist in predicting outcomes related to certain clinical conditions [Onisko et al. 1999, French et al. Long et al. 1993]. Artificial Neural Networks (ANNs), with a strong ability to learn from data and deal with noisy data [Haykin 1997], are gradually accepted by researchers to develop decision support tools for studying medical outcomes.

The Medical Information technology Research Group (MIRG), located in Ottawa, Canada, and founded by Dr. Monique Frize, a professor of Carleton University and University of Ottawa, has been engaged in research on using ANNs to predict clinically significant outcomes since the early 90s [Frize et al. 2000]. Frize et al. [1995] performed studies on estimated duration of artificial ventilation, as well as estimates of mortality and length of stay in an adult intensive care unit. For the purpose of simulating the manner in which physicians work, an artificial neural network model can be trained to estimate various clinical outcomes. The simulation process is like a clinician's consideration of potential patient outcomes: "And for this particular patient, this is what I think will happen"[Frize et al 1995, 1998]. Artificial Neural Network models with a weight elimination cost function and log-sensitivity stopping criterion identified the most influential risk factors for predicting NICU mortality and were shown to outperform conventional scoring systems (e.g. SNAP-II) [Ennett PhD thesis 2003].

In MIRG's previous work, ANN-based clinical outcome prediction models directly classified predicted outcome into distinct classes. For example, the results of mortality prediction models categorized the patient into two populations, namely death and survival, which represent high and low risk patients respectively. However, they cannot further discriminate the risk (i.e. severity of illness) within the same populations. Incorporating information to stratify patients into more groups according to the mortality risk will help to allocate medical resources and improve neonatal care effectively. The probability of mortality represents the level of uncertainty of death, which is an indicator for the severity of illness. An ANN model of probability estimation would be of great interest and flexibility to decision makers to overview the populations based on the detailed levels of illness severity. In addition, it is useful for a decision support tool to provide probabilistic information of medical outcomes since physicians are familiar with statistical methods and data. Finally yet importantly, for researchers who are interested in classifying a dichotomous outcome, this kind of probability model can lead to the development of a mortality-screening tool by assigning appropriate cutoff points within the probability range.

Therefore, we need an approach within the ANN framework to estimate probabilities of outcomes. This new method is to supplement MIRG's ongoing study on predicting NICU outcomes.

1.2 Problem Definition

Predicting probability of NICU mortality requires development of new artificial neural network models. The output of these models is able to approximate the probability of mortality given the inputs using NICU patients' physiological variables. The training data and

test data for model development came from the Canadian Neonatal Network database (CNN), which is introduced in section 4.1. The CNN database contains patients' physiological data and outcomes (e.g. death or survival) but not probability information.

From traditional understanding, if an ANN is trained to predict probability, training data should include probabilistic information. For a training set without any probability, a brutal-force way is that we can use frequency analysis to add probabilities to the training set. However, this approach needs extra data processing. A desired method should directly train ANNs from samples without any probability attributes, and the resulting ANNs should be able to estimate probabilities.

Another issue in this research is that the outcome distributions in the medical database are highly imbalanced. For example, the patients who survive always outnumber the ones who die. Since we are interested in developing mortality probability estimation models, we need to know if these highly imbalanced distributions affect the generation of these models. Artificial neural networks usually have difficulty to learn rare outcomes in a skewed data set. A neural network model trained from an imbalanced data set usually suffers low sensitivity (i.e. poor performance on the prediction of the rare outcomes). MIRG members used over-sampling and down-sampling approaches to neutralize the imbalanced distributions in training sets. These sampling methods did improve the sensitivity of ANN classifiers. However, some questions may arise: will this approach work well in this probability problem? Are there alternative solutions? We need to answer them in this research.

1.3 Assumption

An important assumption in this thesis is the independency of clinical variables. "Although some variables are not guaranteed to be conditionally independent each other in any clinical scenario, it seems that there are no strong dependencies between attributes (i.e. medical variables) in the data used by human experts because attributes are usually properly defined" [Kononenko 1993]. Therefore, in this thesis, we can assume that all medical variables are conditionally independent each other.

1.4 Research Objectives

The primary objective of this research is to look for a method to train ANNs to provide the probability of NICU mortality. The resulting probability prediction models should have good sensitivity and specificity when we use them as mortality-screening tools.

Several issues must be addressed. First, necessary conditions and assumptions for a neural network to estimate probability should be discussed. Second, because a training algorithm is the core of neural network learning, a new weight-update algorithm is required. Third, as there are over 80 variables in the CNN database, we need to identify a set of principal neonatal mortality predictors among these variables. The set of predictors is employed as ANN inputs. In addition, ANN simulation results need to be validated in order to ensure success of the presented method. The Receive Operating Characteristic curve [Fawcett 2003] measures the models' discrimination ability. The model's calibration is showed by the results from the Hosmer-Lemeshow goodness-of-fit test [Hosmer & Lemeshow 1989]. The calibration indicates if a model-predicted distribution represents the true distribution over the observed

data. Last, solutions for obtaining a good sensitivity without affecting probability estimation are necessary.

1.5 Thesis Outline

The thesis is divided into six chapters as follows:

Chapter 1: Introduction presents the motivation, the problem statement and the objectives of this research.

Chapter 2: Background covers the relevant information from medical decision support systems, artificial neural networks, Bayesian learning framework to understand the problem context and solutions. This chapter introduces many important concepts that lead to our problem solutions.

Chapter 3: Alternative Approaches provides an introduction and proposal of a variety of solutions for estimating the probability of NICU mortality. The reason of including these methods is that they all tend to solve the same problem. However, some solutions may not be restricted within the neural network framework. To some extent, these methods supplement the ANN-based solutions developed in our research. Moreover, these methods could be of interest for future research.

Chapter 4: Methodology provides a description of research methods used. This chapter includes an introduction of the research database and variables, a method of modeling ANN cost function by Bayesian likelihood function, the derivation of a weight-update algorithm, neural network performance measures, a prototype of the simulation program, three approaches for improving ANN sensitivity (sampling, adjustment of cost function, moving the cutoff point). In addition, other issues, such as neural network parameters, data preprocessing, network training mode, stopping criteria, are presented.

Chapter 5: Analysis of Results presents experimental results and discussions based on these results.

Chapter 6: Conclusion summarizes the conclusion, main contributions, limitations and the future work.

Chapter 2 Background

In this chapter, some clinical scoring systems are first reviewed in section 2.1. Artificial neural network as a powerful decision support tool is presented in section 2.2. This section includes not only a summary of work done by MIRG but a background introduction of ANN learning. Based on the problem in the thesis, general conditions necessary for an ANN to estimate the posterior probability are discussed. These conditions unveil the linkage between the traditional neural network learning and some important concepts in the Bayesian framework. Concepts and methods within the Bayesian learning framework are introduced in the section 2.3. The research approach is proposed in the last section.

2.1 Clinical Mortality Prediction Model in Literature

Scoring systems are the earliest tool assisting physicians to assess diseases and conditions of in-hospital patients. Researchers developed various scoring models to assess severity of illness and the risk of death for NICU patients. The following scoring systems are the most popular models used in the NICU environment.

The Neonatal Therapeutic Intervention Scoring System (NTISS) [Gray et al. 1992] is a therapy-based severity of illness assessment index to measure the risk of mortality. It is a modified form of the TISS score [Cullen et al 1974, 1983] used in NICU. NTISS variables are in binary value form (i.e present or absent). The final score is calculated only from the most intense intervention in a therapeutic category. NTISS scores the most intense level for each therapy during a 24 hour scoring period, thus some scoring categories are mutually exclusive. The first scoring period begins at the time of admission (i.e.time of first vital signs collection), although other 24-hour scoring periods are feasible. When scores are computed for sequential

days, attention must be paid to whether scoring is based on initiation of a therapy or simply the presence/continuation of the therapy.

CRIB (clinical risk index for babies)[INN 1993], widely used in Europe, is used to point severity of illness of infants with birth weight less than 1550g or gestational age less than 31 weeks. It has the disadvantage of using some data that can be determined by the clinician [Gagliardi et al. 2004]. CRIB-II [Parry et al. 2003] is an update of CRIB, which is scored from five items: sex; birth weight; gestational age; worst base excess; temperature at admission and use logistic equation to predict death rate.

SNAP (score for neonatal acute physiology) [Richardson et al. 1993b], developed and mainly adopted in United States and Canada, can be applied for infants with all birth weight and gestational age. SNAP consists of 37 variables to measure risk of mortality. The more severe the condition is, the higher the points are scored. SNAP has close relation with nursing workload, length of hospital stay and therapeutic intensity [Richardson et al. 1993a]. SNAPPE (score for neonatal acute physiology—prenatal extension) [Richardson et al. 1993b] uses SNAP variables plus birth weight, small for gestational age (SGA < 5^{th} percentile) and 5 minute Apgar score < 7 to assess mortality risk. As other scoring systems, SNAP and SNAPPE were designed to analyze groups of patients, not for predicting individual outcomes or for decisions of the individual healthcare [Richardson et al. 1993a]. CRIB and SNAPPE are the most commonly used scores, and their performance has been extensively validated [Gagliardi et al. 2004]. However, both scoring systems were developed based on the data from almost 20 years. The mortality rate in the old days is higher. These systems can no longer provide the accurate estimation of the mortality risk nowadays. SNAP-II and SNAPPE-II, as simplified neonatal illness severity and mortality risk scores, were developed in 2001[Richardson et al. 2001], which

use six and nine variables respectively collected over 12 hours from admission instead of the original SNAP/SNAPPE over 24 hours. In experiments of SNAPPE-II, the area under the receiver operator characteristic curve was .91 +/- 0.01. Goodness-of-fit (Hosmer-Lemeshow) Chi statistics was 0.90. Both SNAPII and SNAPPEII are simple, accurate, and robust across populations based on clinical studies and applications[Richardson et al. 2001].

The Mortality Index for Neonatal Transportation (MINT) score [Broughton et al. 2004] was developed to evaluate risk of neonates who need transport to another hospital based on the information collected from a source hospital. The 7-variable (Apgar score at 1 minute, birth weight, presence of a congenital anomaly, and infant's age, pH, arterial partial pressure of oxygen, and heart rate at the time of the call) model was developed that obtained the results of the area under ROC curves of 0.82 and 0.83 for the derivation and validation cohorts, respectively. The 7 variables were then used to calculate the MINT score, which gave areas under ROC curves of 0.80 for neonatal and prenatal death.

2.2 Clinical Decision Support System and Artificial Neural Network

2.2.1 Clinical Decision Support System

We make decisions every day. The important decisions are often difficult to make and require both a great deal of information and an advanced level of decision support [Marakas 1999]. "A decision support system couples the intellectual resources of individuals with the capabilities of the computer to improve the quality of decisions" [Gorry et al 1989]. A computer-based decision support system functions as a decision maker experiencing a decision-making process in real life. It requires technologies in the field of artificial intelligence (AI). As an interdisciplinary field of computer science and cognitive psychology, AI focuses on the model

and simulation of human thinking, reasoning, learning, and solving processes. Although those processes in human brains are too sophisticated to be characterized in today's computer technology, the development of AI techniques, such as fuzzy logic classifiers, artificial neural networks, genetic algorithm, decision trees, Bayesian learning, represents steps closer to the goal.

In the clinical environment, decisions (i.e. diagnosis) are made according to patients' historical profiles, personal descriptions, clinical features and real-time physiologic parameters. They are often surrounded with context of imprecision, uncertainty and vagueness. Numerical and textual information generated by advanced medical devices are always interwoven together. Redundant, erroneous or even missing data are common. These result in difficulties to define rigorous patterns of symptoms and distinguish medical outcomes. Diagnosis may vary from different clinicians and largely depends on their personal experience and ability in handling data. Clinical decision support systems (CDSS) help clinicians make healthcare decisions by the analysis of patient physiological variables [Payne 2000]. These systems usually consist of computer programs for improving clinical diagnosis based on information provided by the physicians or nurses [Barnett et al. 1987] in order to reduce medical errors and improve patient care [Trowbridge]. As an advocator of exploring clinical decision support systems, MIRG is engaged in improving patient care through the development of computer information systems to support integrated artificial intelligence applications. Frize et al [1993] combined an expert shell with case-based reasoning abilities and a graphical user interface to display closely matching patient cases in both an adult and a neonatal intensive care unit. The idea is for the system to imitate a physician's approach in "remembering similar past cases" in order to establish a differential diagnosis and/or to determine a course of therapy [Frize & Walker. 2000].

2.2.2 Artificial Neural Networks

In general, artificial neural networks are computer programs simulating human brain's cognitive processes. Neural networks consist of parallel-distributed and inter-connected computational units to store experiential knowledge from a learning process in order to classify new patterns [Haykin 1999]. The simple computational unit is analogue to the neuron in the brain. The knowledge learned by the network is represented by synaptic weight between two computational units. The leaning process is to adjust the synaptic weights to reduce the error between desired output and actual output as much as possible when the network exposed to environmental information (i.e. input data).

An ANN-based Decision Support System possesses many advantages for the clinical environment. First, developing an ANN model does not need prior information and frequent consultation with physicians. Obtaining knowledge from a domain expert may be time and economic inefficiency [Marakas 1999]. Second, ANNs have the better ability to deal with noisy or incomplete data than traditional statistical or AI methods [Marakas 1999]. Last but most important, ANNs can usually find solutions of complex problem including pattern classification, mapping input data and outcomes with non-linear relationship [Haykin 1999]. Medical data is often believed to have nonlinear relationships between input variables and outcomes, so the neural network is a good choice for data analysis in the medical environment [Baxt 1994].

ANNs vary in their distinct architectures and learning processes. From an architecture perspective, ANNs include single layer networks, multilayer feed-forward neural networks (i.e. with one or more hidden layers between input layer and output layer), recurrent networks (i.e.

feed-forward networks with feedback loops) and associative networks, etc. ANN learning has supervised learning and unsupervised learning, proposed by Rosenblatt (1958) and Hebb (1949) respectively. The former is mainly used in pattern recognition and classification; the latter is usually applied in pattern association. Tasoulis et al. [2004] used self-organizing learning, a type of unsupervised learning, to identify association among different clinical variables collected from ECG and their significance to the outcome of inner bleeding. In this thesis, both multilayer and single layer networks were developed and the learning process was supervised learning.

2.2.2.1 Basic Principals of ANN

Model of a Neuron

Fig 2.1 shows a single layer neural network with one neuron as a computational unit connected with input synapses and output synapse. It is a simple form to explain how an ANN works. Each input synapses is assigned with a weight *Wi*. Input signals *Xi* were connecting to the neuron by multiplying corresponding *Wi*. A linear combiner in the neuron sums the input signals multiplied by the weight from each synapses. A bias term Bk is a threshold for the sum of inputs on which the neuron is fired [Penny & Frost 1996]. Bias has the effect of increasing or lowering the sum of neuron inputs. Usually, in neural computing, bias can be considered as an additional term of synaptic weight with an input signal value of 1 or -1. The neuron output is calculated by an activation function of the sum.

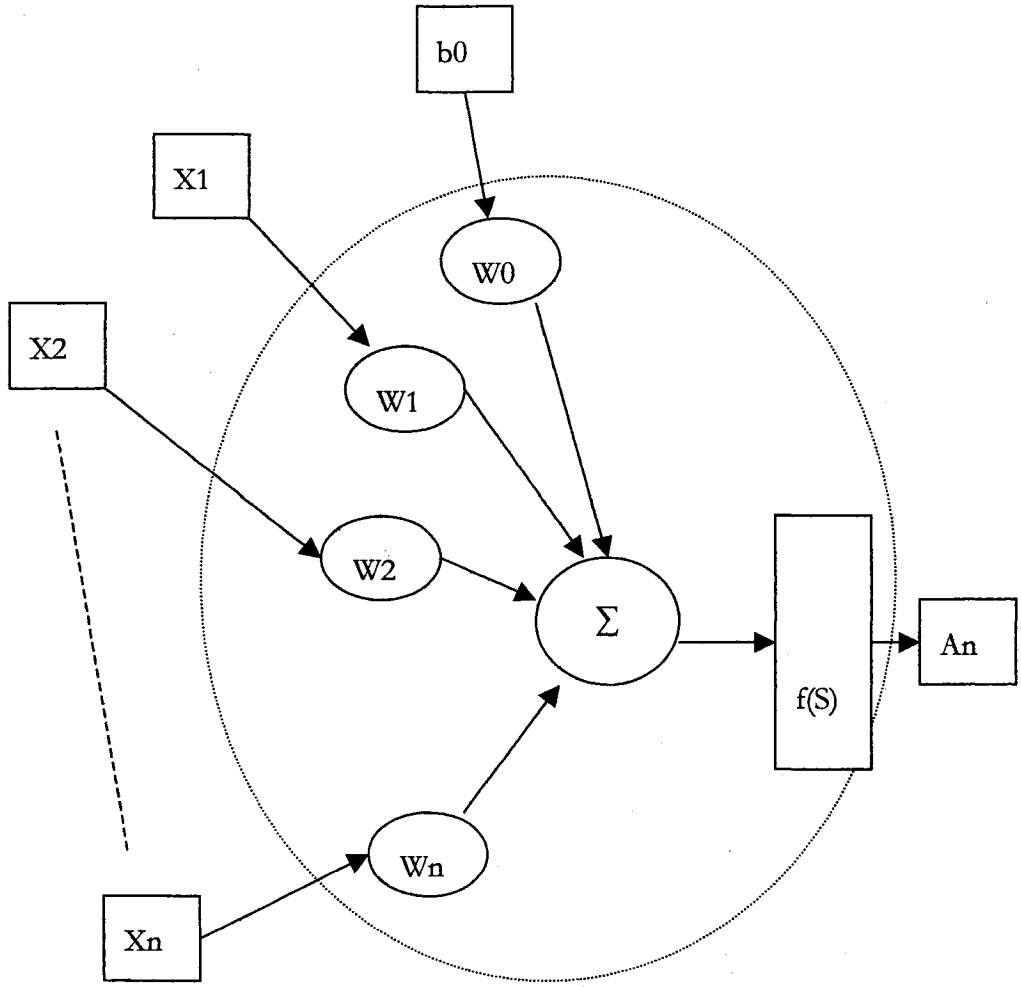

Fig 2.1 Example of Single Layer Neural Network [Ennett, 2003]

Activation Functions

Sigmoid, hard-limit, hyperbolic tangent functions are commonly used activation functions of ANNs. The sigmoid, the continuous, differentiable and graphing S-shaped output curve with outputs between 0 and 1, is defined as a monotonically increasing function that exhibits a balance between linear and nonlinear behavior [Mennon et al 1996]. The hyperbolic tangent

(and its derivative) was used in most MIRG's ANN models. The output of the hyperbolic tangent function ranges from –1 to 1, which offers a sharp sigmoidal transition at zero and results in very few output values near zero, so a nonzero output is mostly frequent [Fausett 1994]. Hyperbolic tangent function can be applied to learning algorithms for classification of two distinct outcomes (e.g. death and survival). The activation functions used in this thesis are sigmoid functions with a slope parameter of 1 because a probability value ranges from 0 to 1.

Multilayer Feed-forward ANN

Fig 2.2 is an example of a typical multilayer feed forward network. This kind of neural network is generalized from a single layer neural network and referred to as a multilayer perceptron model (MLP). The network consists of one input layer, one or more hidden layers and an output layer. Units from one layer connect to every unit of the next layer. Input signals propagate through the network in a forward manner. The Networks can either be partially connected to only some units in a preceding layer or fully-connected to all units. A two layer feed-forward network without hidden layers can be considered to represent a linear relationship between inputs and outputs, whereas neural networks with one or more hidden layer are applied to represent non-linear relationship. Usually, a three layer neural network (i.e. one hidden layer) can approximate any bounded non-linear mapping between input signals and outcomes.

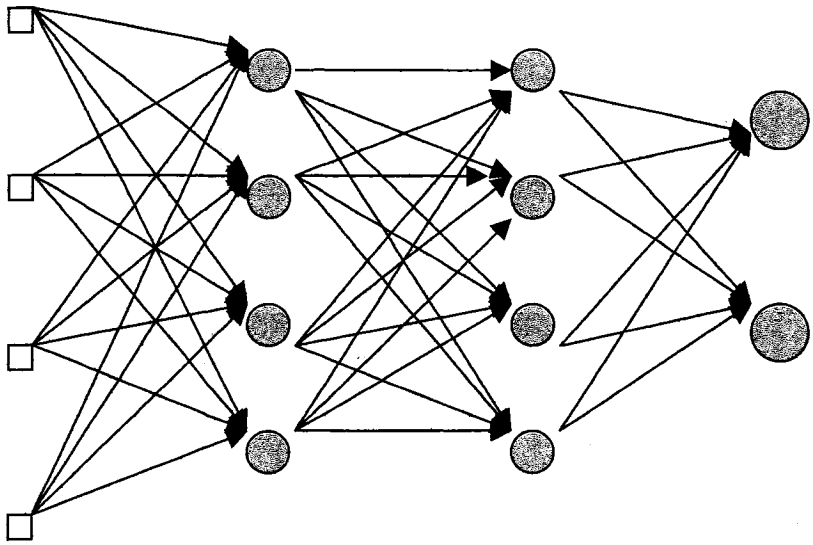

Fig 2.2 Feed-forward Neural Network of a Multilayer Perceptron Modal [Haykin 1999]

Back-propagation Algorithm

MLP feed-forward neural networks are trained in a supervised manner by the well-known *backward error propagation* (back-propagation) algorithm [Haykin, 1999; Fausett, 1994]. The algorithm was inherited from the least-mean square (LMS) algorithm. The back-propagation algorithm consists of two-way signal passes: the first is a forward pass where synaptic weights are fixed and outputs are calculated from sum of units' transfer functions in an orderly way from the first layer to the output layer; a backward pass updates weights according to an *error-correction rule* in order to reduce the error between actual outputs and desired outputs.

The basic idea behind back-propagation algorithm is to use gradient-descent move in error space for every iteration, so as to decrease the error faster and search for the point in the error space with a (global) minimum of the mean squared error. The back-propagation algorithm computes as follows:

Step 1: calculate the output of neuron j, input signals for the output neuron come from a previous layer.

Step 2: if the neuron j is a neuron in the output layer, compute the error signal of desired output and actual output.

$$\xi(n) = 1/2\sum e_j{}^2(n)$$

Step 3: compute the local descent δj, if the neuron is the output neuron, the local descent is $\delta_j = e_j(n)\varphi'_j(v_j(n))$. Otherwise, the local descent in the neurons of the layer just before the output neurons is

$$\delta_j(n) = \varphi'_j(v_j(n))\sum_k \delta_k(n)w_{kj}(n)$$

Step 4: update the weight according to the *delta rule.*

$$\Delta W_{ji}(n) = \eta\delta_j(n)y_i(n)$$

Weights are updated on a *pattern-by-pattern* manner until all the training samples are processed. Although other algorithms such as *conjugate gradient descent* and *Levenberg-Marquardt* [Bishop, 1995; Shepherd, 1997] are substantially faster for many problems, *backward error propagation* is still popular for its many advantages such as computational efficiency [Ampazis].

Sequential Training Mode versus Batch Training Mode

In practice, an entire presentation of training examples in a back propagation process is called an *epoch.* Training examples should be randomized in their order of presentation at the beginning of each *epoch.* This randomization tends to make the search in error space stochastic

over training cycles so as to avoid bias from training [Haykin, 1999]. A training process can be carried out in two ways:

Sequential Mode is the most popular way in the back-propagation process. The error is computed when every training sample is processed. Synaptic weights are updated based on each sample. A *Sequential Mode* back-propagation algorithm has several advantages. It is mostly used in feed-forward neural network learning. The algorithm is simple to implement. It is appropriate to train with large data sets with high redundant samples, which will make the learning process take advantage of such redundancy to classify some difficult patterns (e.g. rare outcomes).

Back-propagation algorithm with a *Batch Mode* computes errors after processing entire training samples. Thus, weights are adjusted after every epoch. This way of training provides an accurate estimate of the gradient vector. This makes it relatively easy to research the theoretical condition for algorithm convergence. Furthermore, the *batch mode* has fewer difficulties to parallelize than the *sequential mode.*

Adaptive Network Parameters

The purpose of back-propagation is to find a minimum point in error surface by gradient descent movement in multi-dimensional weight space. In each back-propagation iteration, a step to final convergence of minimum error point is made. A large step may result in quick convergence while it may also overstep a solution. Therefore, a set of adaptive network parameters is defined to adjust a learning process.

Learning rate η can vary according to network error comparison between two consecutive epochs. A larger *learning rate* will increase the step size for updating weights, but too large weight changes may result in a network becoming oscillatory or unstable. A smaller *learning rate* will result in a smoother trajectory in learning processes but may cause less improvement in reducing errors.

Momentum causes several steps of movement in a fixed direction, which avoids a learning process falling in local minima and avoids instability of the learning process. As a modified *delta rule* of the equation Eq 2.1 [Rumelhart et al 1986a], the *momentum* is implemented by adding a fraction of the last weight change to current weight update. The fraction term *a*, which can be a positive or negative number, usually ranges from 0 to 1. It will change according to a comparison of current-to-previous iteration errors. When the *momentum* is set to zero, current weight change operates without the *momentum*. Otherwise, current weights and biases are calculated as the modified *delta rule*. When the *momentum* is one, current weight change is equal to previous weight change and the gradient resulted from that is ignored [Demuth & Beale 1997].

$$\Delta W_{ji}(n) = \alpha \Delta W_{ji}(n-1) + \eta \delta_j(n) y_i(n) \qquad [\text{Eq } 2.1]$$

Weight Decay and Weight Elimination

In practice, a neural network with optimal size may learn less noise from training data and result in better generalization performance [Haykin, 1999]. *Network pruning*, a procedure where the structures of MPL networks are optimized by eliminating certain synaptic weights to balance reliability of training data and goodness of the neural network models [Haykin, 1999]. *Weight decay* [Hinton, 1989] and *Weight-elimination* [Weigend et al. 1991] are implemented by

adding a penalty term to a risk function. *Weight decay* forces excess weights that have little or no impact on network to reduce to zero for improving network performance. *Weight elimination* can identify small and unreliable weights. *Weight elimination* eliminates them from neural networks. MIRG used *weight elimination* approach to prune network structures. The networks obtained the same or better generalization performance before pruning. [Frize et al 2001].

Stopping Criteria

Training stopping criteria should be carefully determined. The lack of enough training will result in a large predictive error. Over training may cause a neural network to overfit on training examples so the neural network will be incapable of generalization. Training can stop at the point when a given number of epochs elapses, or when the error reaches an acceptable level, or when the error stops improving.

MIRG developed another criterion to work with an automated network training process called the maximum *logarithmic-sensitivity index* stopping criterion[Scales 2001, Ennett et al. 2002, 2003]. The *logarithmic-sensitivity index* attempts to achieve optimal sensitivity and specificity classification while slightly favouring higher sensitivity.

$$\log\text{-sensitivity index} = -\text{sensitivity}^{n} * \log 10(1 - \text{sensitivity} * \text{specificity})$$

Training usually stops when there is no log-sensitivity index increase within the last 500 epochs.

Sample Sets with Rare Outcomes

Sometimes, a training set can be highly skewed, which means some of outcomes are rare (e.g. a death outcome). If the network is trained by such skewed data set, the resulting classifier may lack the ability to recognize rare patterns even if the correct classification rate is high. Thus, under these situations, rebalancing data sets is often required. To perform the method, instances of rare outcomes are taken out from the original training set and duplicated to create an artificial training set. The neural network classifier trained by the new data set will have a better performance on classifying rare outcomes. [Ennett 1999, Frize 2000, 2003].

2.2.2.2 General Conditions for ANN Approximating Probabilities

Basically, there are two objectives of neural network learning. One is to directly output classification results. The other is to model outputs as posterior probabilities of class membership. Previous ANN models developed by MIRG and models in most clinical research trials from literature were based on the first objective. They are classification neural network models. Classification neural network models do not make probabilistic decisions on the network outputs unless further post processing the outputs to approximate the posterior probabilities [Titsias et al.]. Comparing with the first one, the second objective is more powerful and relates to the Bayesian framework. However, not all the algorithms can train the neural network to estimate the probabilities.

Bishop [1995] discussed general conditions and underlying assumptions necessary for a neural network to produce an output with probability information. He interpreted that networks trained by minimizing the cross-entropy or sum-of-squares error function can estimate the posteriori probability of class membership. For the sum-of-squares error function, errors between predicted outcomes and desired outcomes should follow a standard normal

distribution. Under this assumption, a neural network learning process from a Bayesian perspective is to find a maximum likelihood hypothesis that minimizes the squared errors between predicted outcomes and observed outcomes [Mitchell 1997]. After converging, the trained neural network can approximate the posterior probability of class membership. However, Gaussian distribution of errors is not guaranteed in many scenarios. It is an ideal condition for network training. It is also a prerequisite for the sum-of-square error function relating to the concept of maximum likelihood. Therefore, with the assumption of Gaussian distribution of errors, a feed-forward MLP or a radius basis function network using the sum-of-square cost function can be trained to predict the probability when the activation functions of output units are sigmoid functions.

A cross-entropy based error measure is based on the concept of entropy. The entropy was first developed by the physicists in the context of thermodynamic equilibrium and later introduced into information theory [Shannon 1948]. The concept of entropy is applied in many disciplines. In probability theory, suppose that a random variable x is drawn from a distribution with the probability mass function *(p.m.f.)* $P(x)$. P_k is the probability of presence of the random variable in the *kth* observation, provided the total number of observations is close to infinity. The entropy is defined in the Equation Eq 2.2:

$$S=-\sum P_k \ln P_k \quad \text{[Eq 2.2]}$$

In term of information theory, S represents the expected amount of information obtained from the observations of the random variable x. Therefore, $-\ln P_k$ is regarded as the information gained from the *kth* presence of the variable x. However, the true distribution of $P(x)$ is often unknown. We use a known distribution such as Y(x) to replace the true

distribution. Thus, the information from the *kth* presence of the variable x under the new distribution is $-\ln Y_k$. Since the random variable x was drawn from the true distribution, the expected amount of information S obtained is given by the equation Eq 2.3:

$$S=-\sum P_k \ln Y_k \quad [Eq2.3]$$

It is easy to show that the replacement distribution $Y(x)$ equal to the true distribution $P(x)$ results in the smallest value of the cross-entropy [Bishop 1995]. When the cross entropy is applied to neural network training, a new cost function of the neural network can be derived from the Eq 2.3. Bishop [1995] gave the cross entropy cost function in the form of equation Eq 2.4:

$S=-\sum\sum t^n_k \ln y_{nk}$ [Eq 2.4] where N is a set of data point drawn independently from a unknown distribution. C is the number of classes. x^n is the *nth* sample in the data set. y_{nk} is the predicted probability given the input x^n belongs to the k class. tnk is the true probability given the input xn belongs to the k class.

Minimizing the cross-entropy error function results in neural network outputs approximating the posterior probabilities of class membership. Two assumptions lie in this approach. First, a training set should be sufficient large. Second, samples in the training set are considered independent each other. It is easy to conclude that the cross-entropy method is more general and flexible than the sum-of-square error approach since it does not take into consideration the prior distribution of errors. In addition, as suggested, both methods closely relate to the maximum likelihood concept and can be interpreted in a Bayesian learning framework.

2.3 Bayesian Learning and Maximum Likelihood Estimate

Bayesian Learning, also called Bayesian Reasoning, is a probabilistic approach to classify patterns. With this approach, samples are classified to corresponding patterns according to their posterior probabilities of class membership. Given a sample, a posterior probability for it belonging to a class is first calculated by Bayesian Reasoning. The sample will be classified to a class on which it has the highest posterior probability. Bayesian learning discriminates patterns based on probability distributions. It is also a building basis for many learning algorithms to handle probabilities when those algorithms do not explicitly manipulate probabilities by themselves. Michie et al [1994] performed a detailed study with naïve Bayes classifier and other learning algorithms such as neural networks, which showed that the Bayes classifier is competitive or outperformed other methods.

Bayesian learning came from Bayes theorem (by Tomas Bayes, an 18th century clergyman), which is one of the important concepts in probability theory. Bayes theorem is given by:

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

P(h) is called 'a *priori probability*' of class *h*, which represents background information of the distribution of a training set. *P(D)* can be considered as the probability distribution of training data *D*. *P(D | h)* is the conditional probability distribution of data *D* classified as *h* class. The formula is used to compute the 'a *posteriori probability*', *P(h | D)*, which is the conditional probability distribution of the h class given the observed data *D*. In general, Bayes methods process sample data to update a prior distribution into a posterior distribution [Ramoni et al. 1999]. Probabilistic in nature, Bayesian methods are widely used in clinical applications since physicians are more familiar with statistical methods. Kononenko[1993] used Bayesian learning

methods in identifying primary tumor, the prognostics of recurrence of breast cancer, and diagnosis of thyroid diseases.

2.3.1 Maximum a Posteriori and Maximum Likelihood

In Bayesian context, there is a notion called hypothesis space. A hypothesis space contains a group of hypotheses which may be considered as possible solutions of a problem. Normally, given training data, a Bayesian learning process is to search for the most probable hypothesis in the hypothesis space. Such a hypothesis is called maximum a posteriori (MAP) hypothesis [Mitchell 1997]. The MAP hypothesis is defined by the formula below:

$$h_{MAP} = \arg\max_{h \in H} P(h \mid D) = \arg\max_{h \in H} P(D \mid h)P(h)$$

We can assume that each hypothesis in the hypothesis space has the same *a priori probability* P(h), So the MAP is determined only by the term $P(D \mid h)$. The hypothesis that maximizes the $P(D \mid h)$ is called maximum likelihood (ML) hypothesis. Thus, the learning process is changed to find a ML hypothesis with the training data.

2.3.2 Maximum Likelihood Estimation

In addition to using Bayesian learning to classify the patterns, sometimes we are interested in understanding the statistical characteristics of a population. A probability mass function (*p.m.f.*) and a probability density function (*p.d.f.*) are used to model the statistical nature of a population. These functions reflect the probabilistic relationship between random variables and their corresponding outcomes. The *p.m.f.* is applied on the probability distribution of discrete variables. The *p.d.f.* is to describe the distribution of continuous variables. However, the

distribution of a population is often unknown. In other words, the *p.m.f.* or *p.d.f.* is unknown but we can obtain a set of samples from the population. To investigate the statistical nature of the population, we usually adopt *maximum likelihood estimation* approaches on the samples to approximate an underlying probability distribution function (i.e. *p.d.f.* or *p.m.f.*).

The *maximum likelihood estimation* is a simple, effective method in probability theory to estimate the unknown distribution of a population from observed samples. Suppose that the distribution depends on an unknown *p.d.f./p.m.f.* specified by a set of unknown parameters θ. A function form of the *p.d.f./p.m.f.* is denoted by $f(x; \theta)$. Because the θ can be any possible value, θ is in a parameter space Ω. Suppose that $X1, X2, X3, \ldots Xn$ are random samples independently drawn from the distribution. In probability theory, the joint probability mass/probability density for obtaining these random samples is the product of the probability mass/probability density on every sample:

$$f(X1,X2,X3\ldots Xn; \theta)=\prod f(Xi; \theta) \quad i=1\ldots n; \text{ [Eq 2.5]}$$

$X1,X2,X3,\ldots Xn$ can take values from values x1,x2,x3,...xn. Therefore, the equation Eq2.5 is rewritten as:

$$L(\theta)=f(X1=x1, X2=x2, X3=x3\ldots Xn=xn; \theta)= \prod f(Xi=xi; \theta) \quad i=1\ldots n; \quad \text{[Eq 2.6]}$$

A reasonable estimate of θ is the value that maximizes the joint probability in Eq2.6 to obtain the observed samples. $L(\theta)$ is called the likelihood function. Suppose that when $\theta=\theta'$, where $L(\theta)$ reaches the maximum. θ' is the *maximum likelihood estimator* of the parameter θ. A process to find the θ' is the maximum likelihood estimation.

For a fairly large sample set, the maximum likelihood estimation approach will have an approximate Gaussian distribution with a center of true parametric values. Hence, when a sample size increases, the *maximum likelihood estimator* tends to converge to the true parameter θ. In other words, the *maximum likelihood estimator* is more biased in a small sample size than in a large one.

The maximum likelihood estimation is an approach of the parametric probability estimation, where the function form of a *p.d.f.* .or *p.m.f.* is known. The purpose of the parametric estimation is to approximate a true distribution by adjusting a set of parameters in a pre-defined function form (e.g. *Gaussian* or *Exponential distribution*). On the other hand, the neural network approximation is in general a non-parametric estimation, where the function form of a *p.m.f.* or *p.d.f.* cannot be identified. However, we know that a three-layered feed-forward neural network (with one hidden layer) can approximate any bounded continuous functions of inputs when the number of hidden nodes is sufficiently large. The number of hidden nodes required for ANN experiments in this thesis is discussed in section 4.4. Therefore, it is reasonable to apply the maximum likelihood estimation on ANNs to approximate a probability function by changing network parameters such as the number of hidden neurons, weights and bias. In this case, neural network learning based on the maximum likelihood estimation is to identify the set of network parameters that result in ANN outputs approximating probability of NICU mortality.

2.4 Research Approach in the Thesis

As aforementioned, the general conditions for a neural network approximating probabilities depends on the form of ANN cost functions. Sum-of-square error function and cross-entropy error function both have underlying assumptions. When errors between predicted and target outcomes follow a Gaussian distribution, a neural network with the sum-of-square error function is able to estimate probabilities. The cross-entropy function works well with a large training set and assumes all the patterns in the set independent from each other. Both approaches can be explained in the Bayesian framework.

The maximum likelihood estimation is a foundation of many methods in statistics to estimate an unknown distribution of a population. For our problem, it was applicable to combine the maximum likelihood estimation and neural network learning to approximate probabilities of clinical outcomes. In order to do so, cost functions in neural network training should be modeled as likelihood functions. Comparing with the sum-of-square cost function, the maximum likelihood method is more flexible and free from any prior knowledge of an error distribution. Moreover, the maximum likelihood method is more intuitive than the cross-entropy approach from a probability perspective.

A term should be clarified here: the *predicted probability*. We explain it under a following scenario. When a patient's outcome is predicted as death, we recall the data of all previous patients and find those patients with the similar physiological parameters or symptoms of disease. If previous patients all died, we can say the new patient has a 100% probability of death based on past experience. On the other hand, if we find that 90% patients in previous cases died but others survived, we say the new patient has a 90% probability of death.

To estimate probabilities, neural network training in this thesis is a process of maximizing the likelihood cost function. Traditional neural network based on the gradient decent search on the error surface is to seeking a point of minimum of the global error. In our case, on the contrary, the global maximum of the likelihood function was the aim of neural network learning. Therefore, a new weight update algorithm with gradient-ascent search for a neural network with a single binary output unit was developed. It is presented in chapter 4.

The new training algorithm based on gradient ascent was applied in two-layer and three-layer MLP feed-forward neural networks. The training process was a batch mode, because the calculation of joint probability $L(\theta)$ depends on the product of each sample's marginal probability $f(x|\theta)$ according to equation Eq 2.6. Network parameters such as learning rate were redefined and tested according to the requirements of batch training. The stopping criterion of neural network training used log-sensitivity index for the purpose of consistency with previous research, although other criteria such as the maximum log likelihood are also available. The resulting ANN models were able to predict probability of NICU mortality, which met the objective of this research, adding probability to ANN prediction of clinical outcomes.

Chapter 3 Alternative approaches for the development of probability models

In this chapter, we present some approaches in addition to the artificial neural network for estimating the probability of clinical outcomes. Although not restricted within the ANN framework, they may lead to better solutions to some problems in future. All the approaches are derived from the Bayesian framework so they are able to approximate the posterior distribution of a population.

3.1 Naïve Bayes Classifier

Naïve Bayes Claasifier (NBC)[Han et al 2001] assumes variables to be discrete and conditionally independent of each other, although it can use both continuous and discrete values. A NBC classifies patterns based on their estimated posterior probabilities of class membership. The following steps describe the classification process of *Naïve Bayes Learning*:

Suppose the input variables of a NBC denoted by $<A_1, A_2, .., A_i, ... A_n>$. A_i is one of the variables in the NBC model. For each pattern in a training set, A_i takes certain value a_i. Hence, $<a_1, a_2, \ldots a_n>$ corresponds to a pattern from the training set. We use the following formula to calculate the posterior probability of the membership of each class. Based on the probabilities of each class, we classify the pattern to a class with the highest posterior probability.

$$V_{MAP} = \arg\max_{V_j \in V} P(V_j \mid a1, a2, \ldots an) \quad \text{[Eq 3.1]}$$

Since all the variables in a NBC model are conditionally independent of each other, the posterior probability of class membership for a pattern is equal the joint posterior probability

based on each variable. The joint probability can be calculated from the product of the marginal posterior probability of class membership based on each variable. Therefore, a new formula for a *naïve Bayes classifier* is written as following :

$$V_{NB} = \arg\max_{Vj \in V} \prod_i P(a_i \mid Vj) P(Vj)$$

V_{NB} is the target value output by the *naïve Bayes classifier*. The classifier requires the estimation of $P(V_j)$ and $P(a_i \mid V_j)$ based on their frequencies in training data. Although some variables are not guaranteed to be conditionally independent each other in any scenario, the variables defined by domain experts usually have no strong dependencies and we can consider them independent [Kononenko 1993]. Therefore, in a clinical context, we can always assume that all physiological variables are conditionally independent in order to apply a NBC.

When using naïve Bayes classifier to estimate the posterior probability of clinical outcomes, comprehensive frequency analysis is required. Given a particular outcome, the analysis will count the frequencies of any possible value of each variable. For example, the conditional probability $P(A_i=s \mid V_j)$ is equal to the frequency of variable A_i on s with a corresponding outcome V_j dividing total frequencies on all the possible values of variable A_i with the outcome V_j. Suppose a set of physiological variables used in a NBC denoted by <A1,A2,...An>. For a patient instance in a training set, these variables take values denoted by <s1,s2,...sn>, the probability of the patient death can be calculated by the following formula.

$$P(death \mid A1 = s1, A2 = s2 ... An = sn) = \frac{P(A1 = s1, A2 = s2, ... An = sn \mid death) P(death)}{P(A1 = s1, A2 = s2, ... An = sn)}$$

$$= \frac{\prod_{i=1}^{n} P(Ai = si \mid death)}{\prod_{i=1}^{n} P(Ai = si)} \text{P(death)}$$

For a discrete variable A_i, the probability of $P(A_i=s_i \mid \text{death})$ and $P(A_i=s_i)$ is easy to compute. However, for a continuous variable, the probability is hard to obtain. To solve the problem, two solutions are suggested. A brutal force way is to do a frequency analysis on every value of the variable from a training set. A drawback is that the training set may not be able to cover all possible values of the variable. Therefore, when the variable of a new pattern contains any unseen values of the training set, the naïve Bayes classifier will not work. Another solution is to incorporate some domain knowledge. The potential values of a continuous variable can be divided into different groups. Each group represents a range of values. The frequency analysis is based on the range of a corresponding group.

3.2 Logistic Regression

Logistic regression is a widely used statistical approach to model and describe the probabilistic relationship between an outcome and a set of variables. An outcome in logistic regression should be either binary or dichotomous. The result of a logistic regression equation can be interpreted as the posterior probability of class membership. For example, for the following form of logistic regression model

$$E(y) = \frac{e^{b0+b1X1+b2X2+b3X3}}{1+e^{b0+b1X1+b2X2+b3X3}}$$ <x1, x2,x3> are input variables, E(y) is the dependent variable calculated by the logistic transformation equation. We can treat E(y) as an estimated outcome.

The range of E(y) is from 0 to 1. <b0,b1,b2,b3> are a collection of linear coefficients that adjust the contribution of each variable for the target outcome. These coefficients of a logistic regression model functions as the weights and bias of a single-layer neural network model. In a logistic regression model, a target outcome y is written as: $y=E(y)+\varepsilon$. Here ε is regarded as the error between the observed outcome y and a predicted outcome E(y). If y=1, $\varepsilon=1-E(y)$ with the probability of E(y). If y=0, $\varepsilon=-E(y)$ with the probability of 1-E(y).

A logistic regression model is derived by the maximum likelihood estimation method. In other words, a set of fixed coefficients <b0,b1,b2,b3> is defined by the criterion of maximizing the probability of obtaining training data. Logistic regression is being increasingly used in the areas of health research. "In 1989, over 30% of the articles published in the American Journal of Public Health employed some form of logistic regression modeling" [Hosmer et al 1991]. The MPMII, SNAP-II/SNAPPE-II models are all developed by logistic regression. From the form of a logistic transformation equation, the generation process of a logistic regression model is analogous to single-layer neural network learning. In addition, the area under ROC and the goodness-of-fit test are the performance measures of a logistic regression model.

3.3 Kernel-based Methods

When estimating the probability density function of a distribution, it is intuitive to think about using the following equation as an estimator for the true probability density p(x):

$p'(x) = \frac{Kn/n}{Vn}$, suppose that there is Kn points of a total number of n point falling into a region with a volume of Vn.

To satisfy the convergence from p'(x) to p(x), three conditions are required:

$$\lim_{n\to\infty} Vn = 0$$

$$\lim_{n\to\infty} Kn = \infty$$

$$\lim_{n\to\infty} Kn/n = 0$$

Basically, there are two common methods to obtain the estimator p'(x). They are the *kernel-based method* and the *K-nearest neighbour* algorithm. The kernel-based method specifies the volume Vn as a certain function of n. The *K-nearest neighbour* algorithm defines Kn as a certain function of n. This method is presented in the next section.

The *kernel-based method* (i.e. the *Parzen Window* method [Parzen, 1962]) is a non-parametric estimation approach to estimate the probability density or probability mass function for an unknown distribution. To do so, the method first defines a kernel function (or a Parzen Window) W.

Suppose that x_1, x_2, ..., x_N is a sample of a random variable x. The Parzen window approximation of the probability density function of the variable x is

$$p'(x) = \frac{1}{n}\sum_{i=1}^{n} W(x - x_i) \quad \text{[Eq 3.2]}$$

Here W is called a kernel, namely, a certain probability density function. W can use any form of existing functions. Suppose that W is a Gaussian density function with zero mean and σ^2 variance. From the equation Eq 3.2, the estimation of probability density on x is actually the average of a linear superposition of all Gaussian density functions with the mean of the corresponding sample x_i. Although many other density functions besides the Gaussian can be applied as the kernel, the Gaussian is usually used since it is easy to manipulate and derive. The

choice of the variance σ^2 is also very important, because it defines the size of a window. If the size of a window is too small, the density estimation error on each sample can be very low. However, as we have already known in neural network training, a very low error in every training sample may cause overfitting. In a classification or probability estimation problem, the generalization ability of a model is critical. In addition, it is showed that an estimated distribution will converge to a true distribution when the size of a training set grows to infinity regardless the window's width. Therefore, for a limited size of training data, the window's width should be appropriately adjusted as a smooth factor through the training and testing processes. One advantage of the *Parzen window* method is that it requires no prior information from the problem and data(it is a non-parametric estimation approach). Because it approximates the probability density of the data, the *Parzen Window* method can be used either alone or in concert with other approaches such as a neural network. An example is shown in the section on probabilistic neural networks to predict the probability of continuous outcomes (e.g. length of stay).

3.4 K-nearest Neighbour

As presented in the previous section, the K-nearest neighbour is another kernel-based method and belongs to a non-parametric estimation approach. The K-nearest neighbour algorithm assumes all patterns in a training set correspond to points in a n-dimensional space, where n can be the number of variables of a prediction model [Mitchell 1997]. We use the standard Euclidean distance to measure the nearest neighbours of a pattern. The algorithm assigns the class label of a pattern to the majority outcomes out of its K nearest neighbours. More precisely, let a pattern x described by a vector form: <a1, a2, a3... an>, the distance between x

and any other patterns (e.g. y=<b1, b2, b3,... bn>) in the training set is calculated by formula Eq 3.3.

$$d(x,y)=\sqrt{\sum_{i=1}^{n}[a_i(x)-b_i(y)]^2} \quad \text{[Eq 3.3]}$$

For the each pattern y in the training set, we calculate its distance to the pattern x. Pick out k nearest patterns of x according to the distance. For example, we have a two-class problem, i.e. a class label is represented by either 0 or 1, K1 is the number of patterns out of K with a class label of 1. K0 is the number of patterns out of K with a class label of 0. We can take K1/K as the probability of the pattern x with the class 1 and K0/K as the probability of the x with the class 0.

The probability estimation described in the previous paragraph depends on the value of K. Recall the non-parametric estimation theory in the section 3.3. We know that as the K and the total number of training samples n grows to infinity, where k << n, $\lim_{n\to\infty} Kn/n=0$, an estimated distribution will converge to the true distribution[Duda et al. 2001]. However, although we can get a relatively large training set, in practice, the training set can still be limited. Therefore, the value of K determines the performance of the estimation. A large value of K may result in including too much noisy data. A small value of K may not be able to describe the true nature of the distribution. In general, the selection of K relates to the model's generalization ability.

An advantage of the K-nearest neighbour method is that it is intuitive and its result is easy to explain. However, the performance of a model depends on the value of K. There are two methods to find the optimal value of K: a brute force approach and a divide-and-conquer

approach. The brute force method searches for the appropriate values of K by testing all possible values from 0 to the size of the entire training set. This method is certainly computationally complex and inefficient. For the second method, the K chosen in the training and testing process is according to the divide-and-conquer searching algorithm. This method has less computational complexity and is faster than the first one. However, for a very large training set, both methods are inefficient.

Some modification to the K-nearest neighbour can neutralize the negative effect caused by an inappropriate selection of K. The distance-weighted nearest neighbour algorithm [Mitchell 1997] is one of them. In this approach, the decision is made by the weighted sum of K neighbours' outcomes. The weight of each neighbour can be given by the inverse square of its distance to the pattern x. The purpose of the algorithm is to make near neighbors play more effect on the final decision than far neighbours. It is very important for a large K value, where the majority members of K nearest neighbours may have longer distance than the minority members. It is unreasonable to approximate either class membership or the probability only based on majority votes.

3.5 Probabilistic Neural Networks

A probabilistic neural network (PNN) was first introduced by Specht [1988, 1990]. This type of neural networks classifies patterns by the probabilistic approach based on Bayes formula and Parzen Window estimation. A probabilistic neural network consists of three layers: an input layer in which input nodes are corresponding to the dimension of a pattern; a hidden layer including the same number of hidden unites as the number of patterns in a training set; an output layer where output nodes represent class membership of a pattern. Every pattern in

the training set forms a hidden unit of the probabilistic neural network. The vector of a pattern is normalized to set as the weight vectors from the input units to the hidden unit of the pattern. The activation functions of hidden units are Parzen Window functions that usually take the form of a Gaussian function. Each output unit corresponds to a class. A hidden unit only connects to an output unit according to the class label of the pattern represented by the hidden unit. Therefore, the input layer to the hidden layer is fully connected, but the hidden layer to the output class layer is partially linked. "Each pattern unit contributes to its associated class unit a signal which is equal to the probability generated by a Gaussian function centered on the associated training point" [Duda et al. 2001]. An output node calculates the estimation of a conditional probability density on its corresponding class. The advantage of probabilistic neural networks is fast learning speed. It requires only a single pass to finish training. The decision hyper-surfaces of a PNN are approaching Bayes-optimal decision boundaries as the number of training samples grows [Gorunescu]. Because a PNN actually contains the entire set of training patterns, it is space consuming. PNNs allow true incremental learning where new training data can be added at any time without requiring retraining of the entire network [Wasserman, 1993]. Jaimes et al. [2005] constructed a probabilistic neural network (PNN) with 10 neurons in the input layer, 368 in the hidden layer and two in the output layer to predict the mortality of suspected sepsis ICU patients. Model calibration was measured using the Hosmer-Lemeshow goodness-of-fit test and discrimination was evaluated by area under ROC of 0.8782.

3.6 Bayesian Network

A Bayesian network (i.e. Bayesian belief network) represents the joint probability for a set of variables [Mitchell 1997]. The cause-effect/cause-result probability relationship between

variables and outcomes is described by a directed acyclic graph. Each variable/outcome forms a node in the graph. A network arc directs a parent-child relation between two nodes. The arc also represents that a variable is conditionally independent of any other non-child variables in the network. [Mitchell 1997]. In each node, a conditional probability table attached to the corresponding variable based on its probability distribution given the observed outcomes of its parent nodes. Thus, a Bayesian network provides a path that shows a sequence from cause to result based on their probability relationship. The outcome of any variables can be inferred by the conditional probability given the observed values of other variables. In a clinical context, a Bayesian network is more suitable to model a clinical pathway [Johnson 1997] than the direct prediction of a particular outcome from a set of non-sequenced variables. Unlike a neural network, an output produced by a Bayesian network is interpretable. Another important advantage of the Bayesian network is its ability to process missing values. The development of a Bayesian network may depend on the guidance from domain experts. However, data relationship is very complicated in some cases, hence, it is even difficult to identify a cause-effect/parent-child path from expert opinions. In these cases, the K2 algorithm is applied to develop the structure of a Bayesian network from training samples. Micalowski et al [2006] developed a Bayesian network framework that models a clinical pathway of post-operative interventions in an ICU. The network predicted the length of stay in a hospital based on pre-defined clinical activities.

Chapter 4 Methodology

This chapter provides the steps of research method in this thesis. The chapter consists of the following sections:

Step 1: Database and variables

Step 2: Modeling ANN cost functions and deriving a training algorithm

Step 3: Selecting the stopping criterion and adaptive network parameters

Step 4: Development of ANN mortality probability prediction model

Step 5: Performance evaluation procedures

Step 6 Development of ANN training tool in Matlab™

Step 7: Probability models for imbalanced datasets

4.1 Step 1: Database and Variables

The patient data used in this research are from the Canadian Neonatal Network(CNN) database. The CNN data were collected by the Canadian Neonatal Network from January 8, 1996 to October 31, 1997. The database contains patient records from seventeen NICUs across Canada during a two-year period.

The original database contained 20488 patient records, which can be categorized by Day 1 (admission to a NICU), Day 3, Day 14 and Day 28 (or discharge) based on the data collection time in a NICU. In our experiments, the admission data (Day 1), collected within the first 12 hours, were used.

Patient outcomes such as death were recorded in the Canadian Neonatal Network database. Neonatal mortality is defined by the World Health Organization as death occurring before 28 days after birth [Health Canada 2000]. The database only contains mortality and survival

records for patients who stay in hospitals for 28 days. This means that the true in-hospital mortality rate may be slightly higher than the rate reported in the database. The database also excluded moribund babies since they usually only receive comfort measures but not aggressive care [Gray et al. 1992]. Another reason for removing moribund babies is that some studies with the SNAP scoring system showed that no tests were performed and no monitoring values were recorded for these infants, which can lead to inaccurate estimation of mortality risk [Richardson et al. 1993a]. In addition, infants with missing data about mortality, birth weight, small for gestational age status and Apgar score at five minutes, were removed as well.

The incomplete values in the database were imputed by Colleen M. Ennett in her Phd research [Ennett, 2003]. These missing values in the CNN database were caused either by insufficient time and resources for collecting all these information or by no needs to collect it on less sick babies. Neural networks cannot process data with missing values. To solve this problem, Ennett [Ennett, 2003] used a hybrid approach which combined artificial neural networks and a case-based reasoner implemented by Haley Engine to impute the missing values. The missing values in a record were imputed by the mean value of the 10 nearest records to the missing-value record in an Euclidian vector space. The weights for calculating these nearest cases were extracted from a two-layer artificial neural network. According to Ennett's results, these artificially imputed values could approximate the true clinical data and allow to develop ANN mortality prediction models. After all missing values were imputed, the new database contained 19,427 patients. The number of records in the new database was slightly less than the original one, because any records with five or more missing values were excluded from the hybrid approach for lack of information. The records with less missing values in the original database, which were considered having a higher mortality risk, were extracted from the new database. This resulted in a subset containing 5102 records. The 5102 subset has a mortality

rate of 9.5% (484 deaths) whereas the whole new database has a mortality rate of 3.75% (727 deaths in 19427 records). Therefore, from mortality distributions in these two sets, we know that the 5102 set contains sicker babies than the 19427 set.

Probability estimation models were developed based on both sets. Hence, we are able to tell if the difference of prior mortality distribution affects the performance of the probability prediction models.

The original CNN database contains 80 variables. In order to develop ANN models, we use two sets of variables from these 80 variables: a 9-variable set from SNAPPEII Scoring System(shown in table4-1), a 13-variable set developed by Ennet [Ennet 2003] (shown in table 4-2). Both sets of variables have been shown to be good mortality risk indicators for NICU mortality estimation.

Table 4-1. SNAPPE-II Variables

Physiological Parameter	Variable name
Birth weight	bthwt
Small for gestational age	sga
Apgar score at 5 minutes	apgar5
Presence of seizures	seizure
Lowest temperature	ltempf
Lowest mean blood pressure	lbloodp
Lowest serum pH	lserum
Lowest urine output	lurine
Lowest pO2/FiO2 ratio	po2fio2r

Table 4-2. 13 Variables by Ennett

Physiological Parameter	Variable name
Birth weight	bthwt
Small for gestational age	sga
Apgar score at 5 minutes	apgar5
Highest respiratory rate	hrespr
Lowest temperature	ltempf
Highest pCO2 reading	hpco2
Lowest serum pH	lserum
Lowest urine output	lurine
Lowest pO2/FiO2 ratio	po2fio2r
Lowest platelet count	lplt
Highest sodium concentration	hsodium
Lowest glucose concentration	lgluc
Highest mean blood pressure	hbloodp

Table 4.3 and 4.4 shows the missing value ratio Database [Ennett, 2003] of SNAPPE-II variables and the 13 variables respectively.

Table 4-3. Missing Values Ratio of SNAPPE-II Variables

Physiological Parameter	Missing in 5102 set	Missing in 19427 set
Birth weight	0.0	0.0
Small for gestational age	0.0	0.0
Apgar score at 5 minutes	0.0	0.0
Presence of seizures	0.0	0.0
Lowest temperature	0.0	0.3
Lowest mean blood pressure	0.0	15.9
Lowest serum pH	0.0	39.9
Lowest urine output	0.0	52.7
Lowest pO2/FiO2 ratio	0.0	63.9

Table 4-4. Missing Values ratio of 13 Variables

Physiological Parameter	Missing in 5102 set	Missing in 19427 set
Birth weight	0.0	0.0
Small for gestational age	0.0	0.0
Apgar score at 5 minutes	0.0	0.0
Highest respiratory rate	0.3	0.3
Lowest temperature	0.0	0.3
Highest pCO2 reading	40.1	40.1
Lowest serum pH	0.0	39.9
Lowest urine output	0.0	52.7
Lowest pO2/FiO2 ratio	0.0	63.9
Lowest platelet count	28.2	28.2

Highest sodium concentration	59.8	59.8
Lowest glucose concentration	13.9	13.9
Highest mean blood pressure	12.7	12.7

Since ANNs require data to be prepared in a certain format in order to produce accurate results, the raw data should be normalized. All the input values are set within the range from -1 to 1. Each record consists of 9/13 input variables and 1 outcome variable. All the data on these variables must be normalized. This was done by taking each input variable in a column, subtracting the mean of that column, and then dividing by three times the standard deviations. This approach called the linear rescaling assumes that all the variables are independent from each other [Bishop, 1995].

4.2 Step 2 Modeling Cost Functions and Deriving a Training Algorithm

As discussed in Chapter 2, in order to train the neural network perform to the maximum likelihood estimation, we first needed to model an ANN cost function as the likelihood function with the mortality outcomes of NICU patients. Some researchers discussed the modeling method for different scenarios [Hosmer et al. 1989][Mitchell 1997]. Here, we summarize them according to the scenario for our problem. Suppose that the training data D composed of patient records $\{<x_1, y_1>, <x_2, y_2>.....<x_m, y_m>\}$. x_m represents the values of a set of input variables for the *mth* patient and y_m corresponds to the outcome, which takes values of 1 or 0 (death or survive). We can rewrite the *likelihood function* in the following form:

$L(\theta)=f(<x_1, y_1>, <x_2, y_2>.....<x_m, y_m>;\theta)= \prod f(x_i, y_i|\theta)$, where $f(<x_i, y_i>; \theta)$ denotes the probability of y_i, given the input as x_i in the *ith* patient record.

The conditional probability that $y_i=1$ given x_i will be denoted as $P(y_i=1 \mid x_i)$. This value can be given by the neural network output $O(x_i)$. It follows that $P(y_i=0 \mid x_i)=1-O(x_i)$. We further rewrite $f(<x_i, y_i> \mid \theta)$ in the *likelihood function* as

$$f(< x_i, y_i > \mid \theta) = O(x_i)^{yi}[1 - O(x_i)]^{(1-yi)} \quad (2)$$

Since we assume the observations of patient records are independent, the likelihood function on the training data with m records is obtained as

$$\prod_{i=1}^{m} O(x_i)^{yi}[1 - O(x_i)]^{(1-yi)} \quad (3)$$

We write the equation (3) in a log-likelihood form:

$$L(\theta) = \ln[L(\theta)] = \sum_{i=1}^{m} y_i * \ln O(x_i) + (1 - y_i) * \ln(1 - O(x_i)) \quad (4)$$

Equation (4) is the likelihood function that is also the cost function of a neural network. After the likelihood function was obtained, the neural network training is a process to find a set of parameters (hypothesis) that maximize the likelihood function in the parameter space (hypothesis space). The maximum likelihood estimation is based on two conditions: the samples in the training set are considered independent each other; when the training set is sufficiently large, the maximum likelihood estimation of the probability will converge to the true probability. Therefore, the *maximum likelihood hypothesis* is written as follows [Mitchell 1997]:

$$h_{ML} = \arg\max_{h \in H} \sum_{i=1}^{m} di \ln O(xi) + (1 - di) \ln(1 - O(xi))$$

Traditional neural networks based on the gradient decent search on the error surface to seek a point of the minimum of the global error. In our case, on the contrary, the global maximum of the likelihood is the aim of the neural network learning. Therefore, a new weight update technique with the gradient ascent search for two and three layer neural networks is desired.

The right hand side of the equation (4) is rewritten as a simple form G(O, D). Now, we can use gradient ascent (i.e. the learning rate is positive) to derive neural network weight updating rule so as to maximize the G(O, D). Mitchell [1997] presented the derivation of the weight-updating rule for a single layer neural network. The weight update equation is written as follows:

$$\Delta W_{jk} = \eta \sum_{i=1}^{m} (di - O(xi)) x_{ijk} \quad (5)$$

x_{ijk} is the kth synapses input to jth output neuron when the neural network is taking the ith training sample. The weight updating equation (5) is for a single-layer neural network. All the input neurons connect to one output neuron in such a neural network. $O(xi)$ is the neural network output given the input vector is x_i.

A multilayer perceptron model, which consists of at least one hidden layer, represents a nonlinear mapping between inputs and outputs. In a MLP model, a neuron can be either an output neuron or a hidden neuron. If the neuron is the output one, the weight updating formula on the neuron is the same as equation (5). We rewrite equation (5) as:

$$\Delta W_{jk} = \eta \sum_{i=1}^{m} (di - O(xi)) x_{ijk} \quad (6)$$

In equation (6), X_{ijk} is the synapse input from hidden neuron k to output neuron j when a neural network is taking the ith training pattern.

If a neuron is a hidden one, according to the principle of back-propagation algorithm, the weight updating formula can be determined recursively by the output neuron with which the hidden neuron directly connects. Consider the situation depicted in Fig 4.1

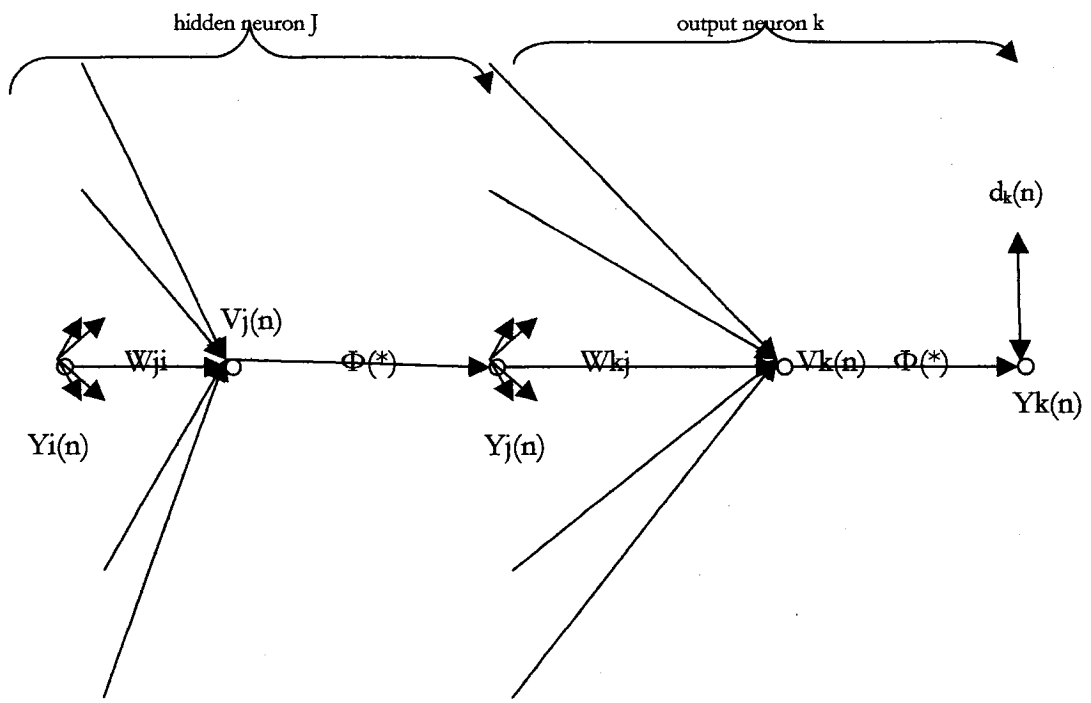

Fig 4.1 A Hidden Neuron and an Output Neuron in a Backward Error Propagation Context [Haykin 1999]

Term explanation:

Yi(n): the nth input on neuron i or ith dimensional variable of input vector

Wji: synapse weight from neuron i to neuron j

Vj(n): $\sum WjiYi$ the sum of all weighted input values

Φ(*): sigmoid activation function

Yj(n): Φ(Vj(n)) the output value of hidden neuron j

Wkj: synapse weight from hidden neuron j to output neuron k

Vk(n): $\sum$WkjYj the sum of all weighted input values from hidden layers

Yk(n): neuron network output for nth input pattern (i.e. the output value of neuron k)

$d_k(n)$: desired outcome of nth input pattern.

According to the chain rule of calculus, we calculate the partial derivative $\partial G(D,O)/\partial Wji$ and regard this as the gradient vector of the hidden neuron:

$$\frac{\partial G(D,O)}{\partial Wji} = \sum_{n-1}^{m} \frac{\partial G(O,D)}{\partial O(Xn)} \bullet \frac{\partial O(Xn)}{\partial V_k(n)} \bullet \frac{\partial V_k(n)}{\partial Yj} \bullet \frac{\partial Yj}{\partial V_j(n)} \bullet \frac{\partial V_j(n)}{\partial Wji}$$

O(Xn) is the nth output of a neural network so it is equal to $\Phi(V_k(n))$. In addition, Φ is a sigmoid function and its derivative is *Yk(n)(1- Yk(n))* (i.e. *O(Xn)*(1-O(Xn))*). Therefore:

$$\frac{\partial O(Xn)}{\partial V_k(n)} = \varphi'(V_k(n)) = O(Xn) * (1 - O(Xn))$$

$$\frac{\partial G(O,D)}{\partial O(Xn)} = \sum_{n=1}^{m} \frac{\partial(d_n \ln O(Xn) + (1-d_n)\ln(1-O(Xn)))}{\partial O(Xn)} = \sum_{n=1}^{m} \frac{d_n - O(Xn)}{O(Xn)(1-O(Xn))}$$

$$\frac{\partial V_k(n)}{\partial Yj} = Wkj \quad \text{and} \quad \frac{\partial Yj}{\partial V_j(n)} = \varphi'(V_j(n))$$

$$\frac{\partial Vj(n)}{\partial Wji} = Yi(n)$$

Thus, the gradient equation can be rewritten as:

$$\frac{\partial G(D,O)}{\partial Wji} = \sum_{n=1}^{m} (d_n - O(Xn)) * Wkj * \varphi'(V_j(n)) * Yi(n) \quad (7)$$

Weight updating formula for a hidden neuron should be:

$$\Delta Wji = \eta \sum_{n=1}^{m} (d_n - O(Xn)) * Wkj * \varphi'(V_j(n)) * Yi(n) \quad (8)$$

η is the ANN learning rate.

Equation (8) is the weight updating formula from an input neuron i to a hidden neuron j. To update the weights in a three-layer neural network, we should use equation (6) and (8) together. The new weight updating formulas are based on a gradient ascent method instead of a traditional gradient-descent approach. The weight updating rule for hidden neurons was derived in the same manner as the conventional back-propagation algorithm, although it is not based on minimizing error between desired output and real output. Here we call the new training algorithm a *revised back-propagation* (it is certainly not the backward error propagation). The new algorithm perform gradient search to maximize the likelihood function of training data, whereas the traditional back-propagation algorithm seek to minimize the error in training data.

4.3 Step 3: Selecting Stopping Criterion and Network Parameters

The new weight-updating rule in the research is based on gradient ascent, but our previous neural network training is based on gradient decent. Hence, stopping criteria such as ASE (average squared error) are not appropriate. The logarithmic-sensitivity index described in chapter 2 attempts to achieve optimal sensitivity and specificity. The index is slightly favouring sensitivity, which will be used to determine the point at which to stop the training. High sensitivity represents good classification performance on death, whereas specificity shows classification performance on a survival outcome. The ideal values of both should be 1. The following equations are used for calculation of the sensitivity and specificity.

$$sensitivity = \frac{true\_positives}{true\_positive + false\_negatives}$$

$$specificity = \frac{true\_negatives}{true\_negatives + false\_positives}$$

True_positive represents the number of death outcomes correctly classified by ANN. *True_negative* represents the number of survival outcomes correctly classified by the ANN. *False_positive* represents the number of survivals classified as death. *False_negative* represents the number of deaths classified as survivals.

The outputs of a mortality probability estimation model are from 0 to 1. This is different from previous models developed by MIRG. In previous models, 0 was the cutoff point for separating a death or survival outcome. In the models developed in this research, we use 0.5 as the cutoff point. An ANN output>0.5 would be classified as a death whereas an output<0.5 would be regarded as a survival.

The new weight updating equation in section 3.2 shows the gradient vector is calculated by the joint probability of every input sample in a training set. Therefore, the training mode of ANNs in our experiment is the batch mode, not the sequence mode. In MIRG's previous experiments, the stopping criterion was defined to be 500 epochs after the logarithmic sensitivity index value shows no improvement. This criterion was used in this research.

The values of network parameters such as a learning rate should be set well. Wilson et al [2000] explained that batch training was much slower than sequence training. Hence we should use a small learning rate. To avoid training oscillating around local optima, the learning rate is defined according to the length of steps that a gradient vector makes.

4.4 Step 4: Development of ANN Mortality Probability Prediction Models

As aforementioned, both SNAPPE-II variables and the new set of 13 variables were employed

as the network inputs and the ANNs were trained by the training sets of 5103 records and 19427 records respectively. For each set of these variables, a 2-layer feed-forward ANN model with no hidden nodes and 3-layer feed-forward ANN models, each of which contains one hidden layer with the hidden nodes number from 2 to 2n+1, were developed. The n is the number of input variables. The maximum number of hidden units was defined according to the Kolmogorov superposition theory in an ANN context. It suggests that the number of hidden units should be no more than two times the number of inputs units plus one [Beiu 2004]. Therefore, a total of 19 and 26 neural network models with different number of hidden nodes were built in the experiments from each training set. There was only one output unit for each neural network model. The target values of the output unit were coded as 0 (i.e. death) and 1 (i.e. survival) respectively. The typical structure of a three-layer neural network model in the experiments is shown in Fig 4.2.

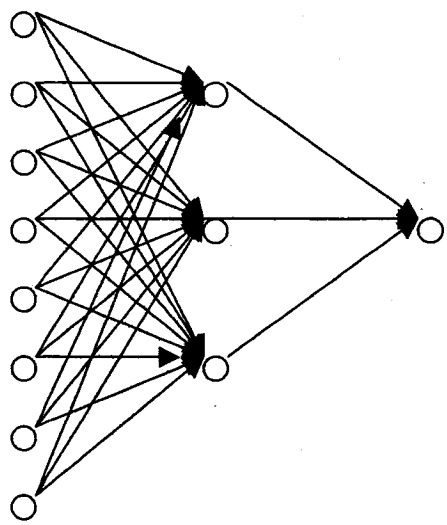

Fig 4.2 The Typical Structure of a Three-layer Network in the Research

4.5 Step 5 Performance Evaluation Procedures

The area under Receive Operating Characteristic Curve (ROC) and the Hosmer-Lemeshow goodness-of-fit test were two performance measures for the ANN models in this thesis. The ROC was used to evaluate a models' discrimination ability. The Hosmer-Lemeshow (Goodness of Fit) test was used to assess calibration. In other words, it measured if the probability values outputted by a neural network represent a true probability of mortality based on our samples.

Discrimination refers to the ability of a model to distinguish those who die from those who survive. A model with perfect discriminating ability assigns higher probabilities of death to the babies who are going to die than for the babies who are going to survive. Receive Operating Characteristic Curve (ROC) graph is a useful tool to evaluate this ability. A Receiver Operating Characteristics curve shows a model's performance as a trade-off between specificity and sensitivity. The model's sensitivity and specificity can be calculated by a two by two confusion matrix (contingency table)[Fawcett 2003]. The ROC curve is plotted according to the coordinate of a false positive rate (1-specificity) versus a true positive rate (sensitivity) while a threshold parameter (the cutoff point to separate a dichotomous outcome) varies. The curve always goes through two points in a coordinate (0,0 and 1,1). In a mortality prediction scenario, 0,0 means that a classifier classify every sample as death (i.e. the sensitivity is 0 and the specificity is 1). The point of 1,1 is where every pattern is classified as survival(i.e. the sensitivity is 1 and the specificity is 0). Fig 4.3 shows typical examples of ROC curves. To compare discrimination abilities in different models, a common method in ROC graph is to calculate the area under ROC [Bradley 1997; Hankey et al.,1982]. The area under ROC ranges from 0 to 1. In the mortality probability estimation problem, the area under ROC value with

one indicates that the model perfectly classifies the death and the survive populations, whereas a value of 0.5 corresponds to a random guess. A good model should have the area under ROC larger than 0.5. A larger area usually shows a higher discrimination ability of a model than a small area does. A model with the area under ROC larger than 0.7 is usually considered a good discrimination performance. Although some alternative approaches of the ROC such as the Cost Curve claim to perform more accurate to evaluate a model's performance [Drummond et al], the area under the ROC is mostly used for measuring a model's discrimination ability in clinical studies.

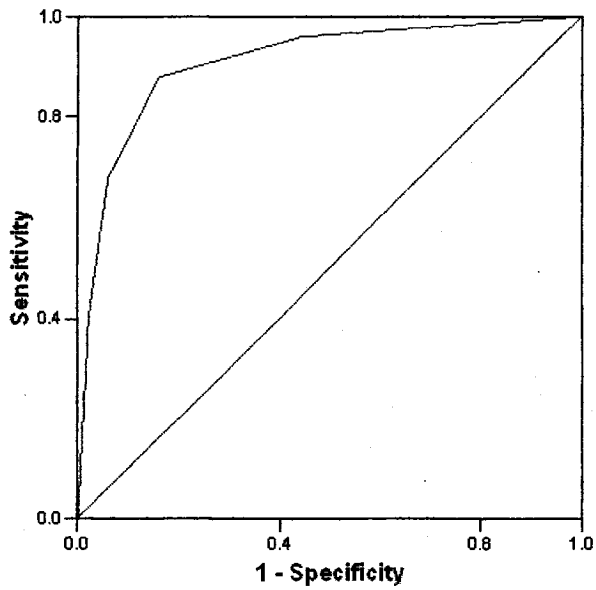

Fig 4.3 Examples of ROC curves (the diagonal line is representing the random guess)

The calibration of a model "evaluates the degree of correspondence between the estimated probabilities of mortality produced by a model and the actual mortality of patients" [Lemeshow & Gall 1994]. It indicates if there is good correspondence between the observed and expected number of outcomes over all estimated probability levels. For example, if the

probability for a death outcome is 0.6 calculated by the model, 60 deaths will be likely observed within a set of 100 samples for this type of patients. In this thesis, a model's calibration was evaluated by the Hosmer-Lemeshow (HL) goodness-of-fit test [Hosmer, Lemeshow 1989].

The Hosmer-Lemeshow test divided all data into g (e.g. g=10) groups based on the estimated probabilities from the smallest to the largest. In each group, we calculated the number of a model estimated mortality and real mortality from the observed data. Any group with the expected deaths no greater than 5 would be merged with one of its neighbour groups, since the small number of expected outcomes would bias the P-value and affect the assessment of the model's goodness of fit. The Hosmer-Lemeshow test measures the calibration by calculating the Pearson Chi-square statistics $\hat{C}$ (defined in equation (9)) across the groups by the estimated probabilities. The sum of the statistics follows a Chi-squared distribution, with the degrees of freedom of the number of groups g minus 2. A P-value > 0.05 implies that there is no significant difference between a predicted distribution and a true distribution. That is, the goodness of fit is acceptable and the model is able to approximate the probabilities for a mortality outcome. A P-value close to 1.00 indicates a model has a perfect goodness of fit [Richardson et al. 2001].

$$\hat{C} = \sum_{k=1}^{g} \frac{(o_k - n_k' \overline{\pi}_k)^2}{n_k' \overline{\pi}_k (1 - \overline{\pi}_k)} \quad (9)$$

n_k' is the number of patterns in the kth group

$\overline{\pi}_k$ is the average estimated probability.

o_k is the number of observed death in kth group

4.6 Development of the ANN Training Tool

In early ANN simulation experiments, MIRG researchers used an automated neural network research tool for developing clinical prediction models [Frize et al 2000]. The tool adjusted the network parameters (e.g. learn rate, momentum, weight decay constant etc.) automatically without user intervention during an entire training process. It greatly simplified experiment steps and freed users from trivial manual work. Rybchynski [Rybchynski 2005] developed a new research tool (ANN RFW) as a replacement of the previous one by adding new features. The ANN RFW is able to automatically find the optimal set of experiment parameters that result in the best performance of ANN training. The verification data sets are included in Rybchynski's tool to test the ANN performance on unseen data. MIRG researchers currently use the ANN RFW in neural network simulation experiments.

Both existing ANN research tools were built based on the traditional ANN training rule (i.e. back propagation algorithm, gradient descent search, etc). They are not suitable for developing a probability estimation model, because the training algorithm in this research is gradient ascent and the cutoff point is changed. Therefore, a new program was developed for the experiments here. It supplements the previous tools. For the consistency with the previous work and the purpose of seamlessly integrating the new program into the existing ANN research tools, the program was also coded in Matlab™. The input interface of loading data files, the interface of the experiment parameter configuration and the operational starting point remain the same as the previous tools, although the computational logic is different. MIRG members who are familiar with the previous tools will have no difficulty in using the new program for training probability models. The integration of the new program with the existing ANN tools will benefit the user from developing probability prediction model along with

classification models at the same time. We call the new program MIRG's ANN-PPF (probability prediction framework) tool. Fig 4.4 shows the flow chart of the new program.

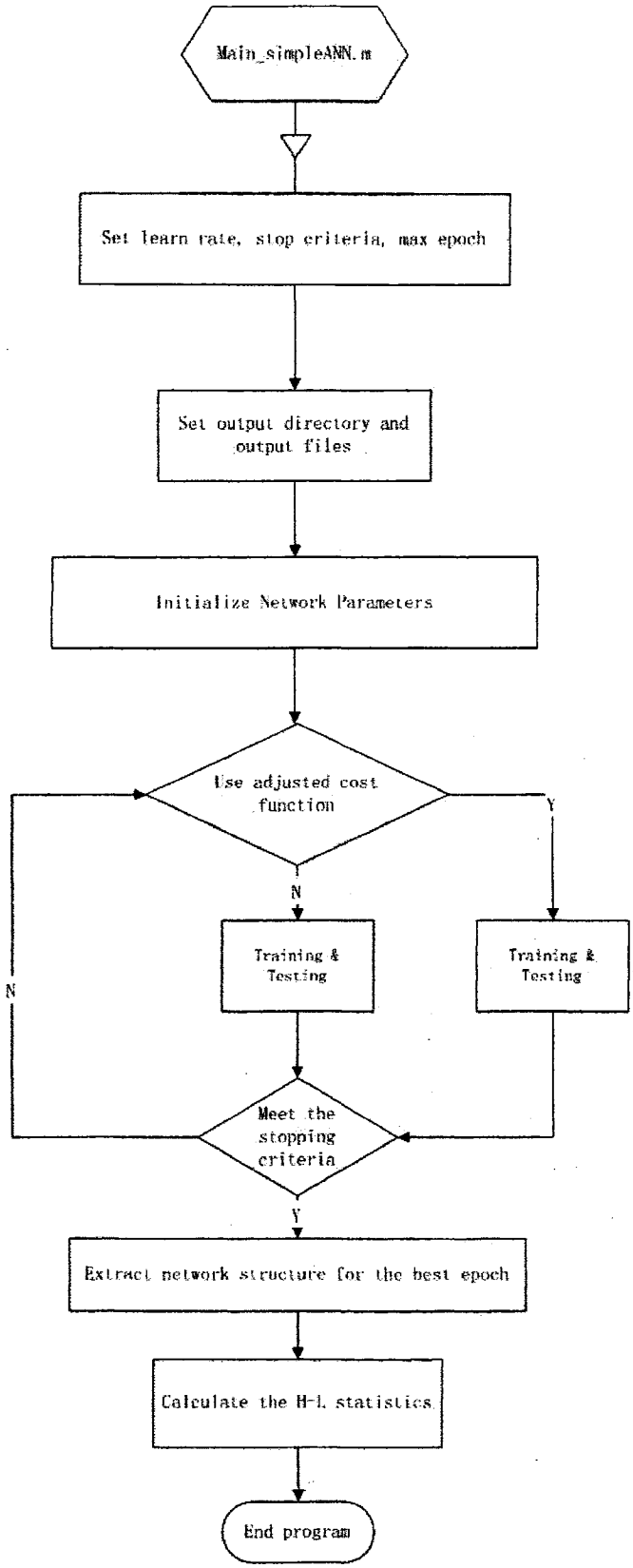

Fig 4.4 Flow chart of the ANN-PPF tool in the research

4.7 Probability Models for Imbalanced Datasets

Mortality ratio in both the 5102 data set and the 19427 data set is low. Since it is difficult for an ANN to classify rare outcomes in an imbalanced data set, high specificity and low sensitivity may result. For data with the linear relationship, ANN performance will not suffer from low sensitivity. However, medical data are considered non-linear. Therefore, an imbalanced data set will affect our results. However, to some extent, we need a probability model to be a tool for mortality prognostic screening (e.g. classifying survival and death cases directly). Ideally, to ensure confidence of using a model for such a scenario, a specificity value of close to 100% and a sensitivity value of at least 50% would be acceptable by users [Ennett, 2003]. Two approaches of creating artificial training set to increase ANN sensitivity were used in previous studies: over-sampling the samples of a rare outcome, down-sampling the samples of a dominant outcome. Both methods are based on changing the prior distribution of a rare outcome, which are not available for developing a probability estimation model. Because the ANN approach in this thesis provides direct estimates of the posterior probabilities, the prior distribution adjustment of a training set will result in the failure to calculate the posterior probability in a test set with a different prior distribution. A method to increase sensitivity without affecting the probability estimation is desired.

Two interesting attempts towards the solution of this problem were studied. The first was to change a neural network's original cost function by adding a scalar term and training the neural network with the new cost function. The other was done by adjusting the classification cutoff point of an existing probability estimation model. The following section explains the principles of both methods. The validation of both methods is presented in Chapter 5.

The first method is based on balancing the error contribution from both classes (We suppose there are only two classes to classify). The weight update depends on the cost function (i.e. the error function) of a neural network. For a two-class problem, the cost function can be written as

$E(w)=E(w1)+E(w2)$ (10) where $w1$, $w2$ are the two classes. $E(w)$ is a neural network error term(i.e. the total cost). $E(w1)$ is the error term caused by the patterns from class $w1$. $E(w2)$ is the error term caused by the patterns from class $w2$. Suppose that $w1$ is the rare class and $w2$ is the dominant class in a training set.

Equation (10) represents neural network training errors consisting of the errors caused by both classes. Since the $W2$ is the dominant class, the error term $E(w2)$ will contribute more to the overall error term E(w). Besides, neural network weight update is based on a batch mode. Thus, the weight update is usually a gradient method depending on the sum (in a back-propagation case) or the product (in our case) of the error of an entire epoch instead of the pattern-by-pattern approach. The direction of the weight update will be closer to the direction of the error vector $E(w2)$. That explains why the neural network will have difficulty to learn from the rare class, which results in low sensitivity. Fig 4.5 is an illustration of the error vectors in a two-class problem.

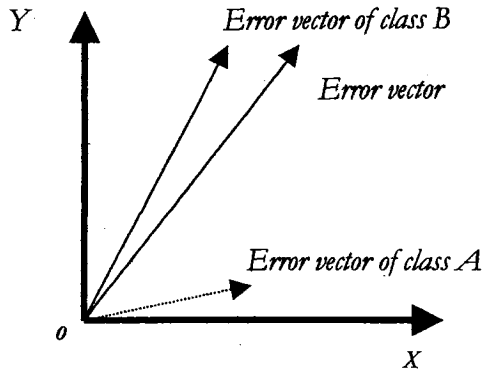

Fig 4.5 A Two-class Example for the Error Vectors (class B is the dominant class and class A is the rare class)

If we balance the errors from the two classes, the gradient vectors will move towards the direction via the error surface of the rare class. As a result, the neural network will learn more from the patterns of the rare class before a training process stops. Sensitivity will increase. This can be done by adding a scalar a to the cost function, $E(w)=E(w1)+ aE(w2)$ (11) where $0< a <=1$.

Equation (11) represents how the new error vector $E(w)$ will decrease some effect from the error $E(w2)$ of the dominant class. When a takes the value of 1, the new cost function is the original one. While the new cost function applied on the weight update formulas derived in this thesis, the direction of the weight update can be adjusted towards the rare class. If the value of a approaches 0, the sensitivity of a model will be close to 1 and the specificity will reduce to 0. In other words, a can be treated as a trade-off between sensitivity and specificity.

The second method, moving a cutoff point, is much simpler than the first one. It takes place after ANN models are trained. The initial cutoff point of ANN training is 0.5, the ANN

output >0.5 would be classified as a death whereas the output <0.5 would be regarded as a survival case. The cutoff point is applied in order to calculate the sensitivity and the specificity in the logarithmic-sensitivity index during training and testing phases. Therefore, after the training stopped, sensitivity and specificity are calculated based on the cutoff point 0.5. When the cutoff point is moving, the sensitivity and the specificity of an ANN model will vary. Although performance measures such as sensitivity and specificity may not make much sense from the view point of the probability estimation, they are able to underline some thresholds for recognizing babies with high risk or low risk. Thus, we may use a probability estimation model as a classification model by setting up the cutoff point.

In our experiments, ANNs were first trained with the cost function derived in section 4.2. Then, both methods suggested in the section 4.7 for improving the model's sensitivity were applied. The simulation results are presented in the section 5.4 and 5.5.

Chapter 5 Analysis of Results

5.1 Comparison of SNAPPE-II Models and 13-variable Models

As aforementioned, a total of 19 ANN models of SNAPPE II variables and 27 ANN models of 13 variables were trained and tested from a training set and test set of total 5102 cases. The performances of these ANN models were assessed by the following measures: the test logarithm sensitivity index, sensitivity, specificity, the area under ROC, correct classification rate (CCR). Except for the area under ROC, other measures were all based on a cutoff point of 0.5. Any output with a probability over 0.5 was classified as a death, otherwise it was a survival. Table 5.1 shows performance of 1 single-layered and 18 double-layered neural networks using SNAPPE-II variables. Table 5.2 shows the performance of 1 single-layered and 26 double-layered neural networks employed with the 13 input variables. Both tables list logarithmic likelihood (Log-likelihood) values, which show the models' likelihood based on training data. The reason that we list logarithmic likelihood is to provide a reference for future research. In future research, we may use Log-likelihood as a stopping criterion. Except Log-likelihood, other performance measures are based on the results of the test set.

Number of Hidden Unit	Logarithmic sensitivity	Best Epoch	Sensitivity	Specificity	CCR (%)	ROC Area	Log Likelihood
0 (Single)	0.023125	486	0.21875	0.98767	91.5344	0.8528	-814.472
2	0.032302	1866	0.25625	0.98313	91.4756	0.8554	-790.401
3	0.045604	2975	0.3	0.98443	92.0047	0.8583	-771.265
4	0.039688	2742	0.28125	0.98637	92.0047	0.8616	-760.178
5	0.03403	2265	0.2625	0.98313	91.5344	0.8554	-787.561

6	0.045568	3036	0.3	0.98378	91.9459	0.8629	-741.559
7	0.039534	6289	0.28125	0.98313	91.7108	0.8595	-705.555
8	0.035921	2641	0.26875	0.98572	91.8283	0.8584	-768.604
9	0.023125	431	0.21875	0.98767	91.5344	0.8506	-804.647
10	0.024558	458	0.225	0.98767	91.5932	0.8502	-801.542
11	0.026001	657	0.23125	0.98637	91.5344	0.849	-799.565
12	0.030743	2620	0.25	0.98637	91.7108	0.8666	-749.134
13	0.030719	2018	0.25	0.98572	91.652	0.8611	-768.954
14	0.041477	3571	0.2875	0.98313	91.7695	0.8597	-735.007
15	0.043545	2901	0.29375	0.98443	91.9459	0.8627	-743.946
16	0.034161	4009	0.2625	0.98637	91.8283	0.8598	-731.109
17	0.037762	1707	0.275	0.98572	91.8871	0.865	-773.369
18	0.037791	2072	0.275	0.98637	91.9459	0.8672	-754.179
19	0.03403	3364	0.2625	0.98313	91.5344	0.8552	-736.026

Table 5.1 ANN Performance Results of SNAPPE-II Variables

Number of Hidden Unit	Logarithmic sensitivity	Best Epoch	Sensitivity	Specificity	CCR (%)	ROC Area	Log Likelihood
0 (Single)	0.027757	126	0.23899	0.98184	91.2404	0.869	-758.078
2	0.037837	968	0.27673	0.97601	91.0641	0.8672	-758.958
3	0.046237	3984	0.30189	0.98444	92.0635	0.869	-717.084
4	0.043905	2291	0.2956	0.9799	91.5932	0.8683	-742.795
5	0.04394	1906	0.2956	0.98054	91.652	0.8744	-712.875

6	0.034336	708	0.26415	0.97925	91.2404	0.8652	-743.255
7	0.049988	3867	0.31447	0.97471	91.2992	0.8694	-686.516
8	0.044079	2293	0.2956	0.98314	91.8871	0.8761	-690.154
9	0.041919	1136	0.28923	0.98054	91.5932	0.8734	-733.108
10	0.064695	4585	0.3522	0.97925	92.0635	0.8751	-689.745
11	0.05035	2906	0.31447	0.98054	91.8283	0.877	-690.498
12	0.049907	6431	0.31447	0.97341	91.1817	0.8747	-629.02
13	0.03986	2138	0.28302	0.9786	91.358	0.8713	-691.834
14	0.043801	2454	0.2956	0.97795	91.4168	0.8755	-679.551
15	0.046018	2258	0.30189	0.98054	91.7108	0.8773	-708.681
16	0.058957	5497	0.33962	0.97017	91.1229	0.8778	-655.334
17	0.047734	6295	0.30818	0.97341	91.1229	0.8723	-656.635
18	0.057296	4486	0.33333	0.98054	92.0047	0.8743	-676.454
19	0.07186	4515	0.37107	0.96952	91.358	0.8756	-620.042
20	0.059394	6144	0.33962	0.97601	91.652	0.8725	-635.852
21	0.059345	4664	0.33962	0.97536	91.5932	0.8814	-657.547
22	0.059297	3498	0.33962	0.97471	91.5344	0.8794	-641.15
23	0.054565	2959	0.32704	0.97536	91.4756	0.8791	-688.308
24	0.043421	3393	0.2956	0.97082	90.7701	0.8816	-666.529
25	0.054698	3153	0.32704	0.9773	91.652	0.876	-610.599
26	0.054256	6485	0.32704	0.97082	91.0641	0.8746	-642.859
27	0.054167	6984	0.32704	0.96952	90.9465	0.8724	-613.053

Table 5.2 ANN Performance Results of 13 Variables

The statistical comparison between SNAPPE-II models and 13-variable models is shown in table 5.3.

	Logarithmic Sensitivity (%)	Sensitivity (%)	Specificity (%)	CCR (%)	ROC Area	Log Likelihood
SNAPPEII	3.47±0.71	26.31±2.58	98.52±0.16	91.73±0.18	0.86±0.005	-765±29.5
13-Variable	4.96±0.97	31.14±2.86	97.68±4.29	91.46±0.34	0.87±0.004	-681±43.4

Table 5.3 Mean and Standard Deviation of Performance Comparison between Two Input Variable Sets (the constant predictor of test set is 90.5)

From table 5.3, it is clear that ANN models with 13 variables, after training, result in slightly larger sensitivity than models with SNAPPE-II variables. Since ANN training in the thesis is to maximize the likelihood cost function, the global maximum of the likelihood will result in a logarithmic likelihood value of 0. Table 5.3 shows that average convergence to the maximum likelihood of an ANN with the 13-variable inputs is better than an ANN with SNAPPE-II variables. The average areas under ROC curve of both variable sets are over 0.85. Therefore, both variable sets are capable to develop ANN models with good discrimination ability.

The initial comparison between two variable sets is from a perspective of classification and discrimination ability. Since we are more interested in the performance of probability estimation, the next comparison is from the results of the Hosmer-Lemeshow test. Every ANN model was assessed by the following measures: Hosmer-Lemeshow Chi-square statistics (C-value) of the training, test and entire 5102 sets; degree of freedom; P-value of the Hosemer-lemeshow test. Table 5.4 shows the Hosmer-Lemeshow test results of models with

SNAPPE-II variables. Table 5.5 shows the Hosmer-Lemeshow test results of models with the 13 variables.

Number of Hidden Units	C-value		Degree of Freedom		P-value	
0 (Single)	train	9.50	train	7	train	0.22
	test	4.85	test	6	test	0.56
	5102	32.32	5102	8	5102	<0.01
2	train	16.93	train	6	train	<0.01
	test	4.54	test	6	test	0.60
	5102	12.10	5102	6	5102	0.05
3	train	2.23	train	6	train	0.89
	test	4.44	test	6	test	0.62
	5102	3.20	5102	7	5102	0.87
4	train	3.84	train	7	train	0.80
	test	4.51	test	6	test	0.61
	5102	3.91	5102	8	5102	0.87
5	train	10.94	train	5	train	0.05
	test	5.15	test	6	test	0.52
	5102	8.78	5102	6	5102	0.19
6	train	7.31	train	7	train	0.40
	test	13.88	test	7	test	0.05
	5102	8.42	5102	8	5102	0.39
7	train	18.69	train	8	train	0.02
	test	7.42	test	6	test	0.28
	5102	13.60	5102	8	5102	0.09
8	train	3.89	train	7	train	0.79
	test	3.44	test	6	test	0.75
	5102	4.30	5102	7	5102	0.74
9	train	2.56	train	6	train	0.86
	test	4.11	test	6	test	0.66
	5102	3.97	5102	6	5102	0.68
10	train	6.02	train	6	train	0.42
	test	5.95	test	6	test	0.43
	5102	7.46	5102	7	5102	0.38
11	train	7.00	train	6	train	0.32
	test	2.12	test	6	test	0.91
	5102	6.76	5102	6	5102	0.34
12	train	14.85	train	8	train	0.06
	test	11.30	test	7	test	0.13
	5102	13.87	5102	8	5102	0.09
13	train	7.79	train	7	train	0.35
	test	3.54	test	7	test	0.83
	5102	5.93	5102	7	5102	0.55
14	train	11.56	train	8	train	0.17
	test	14.23	test	6	test	0.03
	5102	9.34	5102	8	5102	0.31
15	train	10.53	train	8	train	0.23
	test	15.81	test	8	test	0.04
	5102	4.83	5102	8	5102	0.78
	train	15.88	train	8	train	0.04

16	test	9.17	test	7	test	0.24
	5102	7.83	5102	8	5102	0.45
17	train	5.98	train	8	train	0.65
	test	11.15	test	8	test	0.19
	5102	2.09	5102	8	5102	0.98
18	train	11.17	train	8	train	0.19
	test	7.14	test	7	test	0.41
	5102	6.03	5102	8	5102	0.64
19	train	5.12	train	8	train	0.74
	test	11.29	test	7	test	0.13
	5102	4.46	5102	8	5102	0.81

5.4 Hosmer-Lemeshow test results for the models with SNAPPE-II variables

Number of Hidden Units	C-value		Degree of Freedom		P-value	
0 (Single)	train	4.84	train	8	train	0.77
	test	15.96	test	7	test	0.03
	5102	13.32	5102	8	5102	0.10
2	train	8.20	train	6	train	0.22
	test	3.32	test	5	test	0.65
	5102	9.26	5102	6	5102	0.16
3	train	6.49	train	8	train	0.59
	test	15.92	test	7	test	0.03
	5102	11.88	5102	8	5102	0.16
4	train	3.15	train	7	train	0.87
	test	4.73	test	6	test	0.58
	5102	4.01	5102	7	5102	0.78
5	train	2.49	train	8	train	0.96
	test	2.03	test	7	test	0.96
	5102	1.74	5102	8	5102	0.99
6	train	4.44	train	7	train	0.73
	test	4.47	test	6	test	0.61
	5102	5.77	5102	7	5102	0.57
7	train	8.52	train	8	train	0.38
	test	17.72	test	7	test	0.01
	5102	9.01	5102	8	5102	0.34
8	train	15.23	train	8	train	0.05
	test	13.80	test	6	test	0.03
	5102	16.84	5102	8	5102	0.03
9	train	2.18	train	8	train	0.98
	test	3.83	test	7	test	0.80
	5102	2.79	5102	8	5102	0.95
10	train	7.80	train	8	train	0.45
	test	7.11	test	7	test	0.42
	5102	4.43	5102	8	5102	0.82
11	train	6.81	train	7	train	0.45
	test	8.19	test	6	test	0.22
	5102	14.75	5102	8	5102	0.06
12	train	4.99	train	8	train	0.76
	test	17.26	test	7	test	0.02
	5102	6.38	5102	8	5102	0.60

13	train	12.35	train	8	train	0.14
	test	14.60	test	7	test	0.04
	5102	10.50	5102	8	5102	0.23
14	train	3.72	train	8	train	0.88
	test	15.85	test	7	test	0.03
	5102	3.55	5102	8	5102	0.90
15	train	4.47	train	8	train	0.81
	test	3.24	test	7	test	0.86
	5102	3.82	5102	8	5102	0.87
16	train	13.55	train	8	train	0.09
	test	31.31	test	8	test	<0.01
	5102	9.46	5102	8	5102	0.31
17	train	16.37	train	8	train	0.04
	test	17.37	test	7	test	0.02
	5102	9.46	5102	8	5102	0.31
18	train	9.07	train	8	train	0.34
	test	5.53	test	7	test	0.60
	5102	7.61	5102	8	5102	0.47
19	train	11.25	train	8	train	0.19
	test	22.13	test	7	test	<0.01
	5102	8.56	5102	8	5102	0.38
20	train	9.58	train	8	train	0.30
	test	23.90	test	8	test	<0.01
	5102	13.28	5102	8	5102	0.10
21	train	14.00	train	8	train	0.08
	test	15.70	test	7	test	0.03
	5102	10.07	5102	8	5102	0.26
22	train	8.66	train	8	train	0.37
	test	15.74	test	7	test	0.03
	5102	5.06	5102	8	5102	0.75
23	train	9.41	train	8	train	0.31
	test	8.28	test	7	test	0.31
	5102	6.40	5102	8	5102	0.60
24	train	18.20	train	8	train	0.02
	test	19.02	test	7	test	<0.01
	5102	12.77	5102	8	5102	0.12
25	train	14.33	train	8	train	0.07
	test	36.44	test	7	test	<0.01
	5102	7.99	5102	8	5102	0.43
26	train	11.85	train	8	train	0.16
	test	33.17	test	8	test	<0.01
	5102	7.06	5102	8	5102	0.53
27	train	15.28	train	8	train	0.05
	test	35.54	test	8	test	<0.01
	5102	11.66	5102	8	5102	0.17

Table 5.5 Hosmer-Lemeshow test results for the models with 13 input variables

From table 5.4, we see that SNAPPEII models with hidden neuron number of 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 17, 18 and 19 fit the observed data well (i.e. P-value of training, test and 5102 sets >0.05). Table 5.5 shows that the 13-variable models with hidden unit number of

2,3,5,6,9,10,11,15,18, and 23 fit the observed data well. Among these structures, the 13-variable ANN model with 5 hidden neurons obtained a large P-value in training (0.96 with 8 degree of freedom), test (0.96 with 7 degree of freedom) and 5102 (0.99 with 8 degree of freedom) sets. Table 5.6 shows a contingency table of the Hosmer-Lemeshow test on this structure.

Probability Group	Expected Death	Observed Death	Expected Survival	Observed Survival	Total Number of the Group
<0.1	102.8	98	3713.2	3718	3816
0.1-0.2	74.7	74	446.3	447	521
0.2-0.3	65.8	59	201.2	208	267
0.3-0.4	62.7	60	118.3	121	181
0.4-0.5	40.1	41	49.9	49	90
0.5-0.6	42.9	42	35.1	36	78
0.6-0.7	40.1	40	21.9	22	62
0.7-0.8	35.7	36	12.3	12	48
0.8-0.9	25.3	25	4.7	5	30
>0.9	8.5	9	0.5	0	9

Table 5.6 Hosmer-Lemeshow Test Contingency Table of the 13-variable Model with 5 Hidden Neurons on 5102 Record Set (H-L C-value=1.74; Df=8; P-value=0.99)

These results mean that both SNAPPEII variables and the 13 variables can be employed as ANN inputs and trained to obtain mortality probability prediction models with the method presented in this thesis. The SNAPPEII score cannot be used to calculate the probability of mortality directly. However, our experimental results show that a SNAPPE-II mortality probability prediction model can be developed by ANN training. A SNAPPEII mortality probability prediction model, in concert with the SNAPPE-II scoring system, may provide more information for clinical decision makers. In addition, from the comparison of P-value of

each structure, an ANN with the 13-variable inputs will be more likely to obtain a better probability prediction model. Therefore, it shows that the 13 variables are preferable as ANN inputs for development of mortality probability prediction models. In the next section, we use the 13-variable set to develop mortality probability prediction models with 19427 patient records from CNN database. The mortality distribution of the 19427 data is close to a true distribution of the CNN NICUs.

5.2 Probabilistic Prediction Models with 13 Variables

With the set of 13 variables, 27 ANN models were trained with a training set extracted from the 19427 patient records. The performance of each model was assessed by the following measures: test sensitivity, test specificity, area under ROC of a test set, test correct classification rate. Table 5.7 shows these ANN performance results.

Number of Hidden Unit	Logarithmic sensitivity	Best Epoch	Sensitivity	Specificity	CCR (%)	ROC Area	Log Likelihood
0	0.022966	172	0.21739	0.99327	96.5714	0.8779	-1.31E+03
2	0.031582	413	0.25217	0.99343	96.7104	0.8834	-1.28E+03
3	0.030419	486	0.24783	0.99343	96.695	0.8827	-1.28E+03
4	0.03274	516	0.25652	0.99263	96.6486	0.8823	-1.27E+03
5	0.030413	1111	0.24783	0.99327	96.6795	0.8824	-1.25E+03
6	0.02705	495	0.23478	0.99247	96.556	0.8834	-1.28E+03
7	0.026026	690	0.23043	0.99376	96.6641	0.8855	-1.26E+03
8	0.036387	2643	0.26957	0.99103	96.5405	0.8739	-1.20E+03

9	0. 045917	2443	0. 3	0. 99007	96. 556	0. 8743	-1. 20E+03
10	0. 03899	2992	0. 27826	0. 99103	96. 5714	0. 8852	-1. 20E+03
11	0. 037647	3182	0. 27391	0. 99039	96. 4942	0. 8743	-1. 17E+03
12	0. 031588	851	0. 25217	0. 99359	96. 7259	0. 885	-1. 24E+03
13	0. 033921	2550	0. 26087	0. 99183	96. 5869	0. 8818	-1. 21E+03
14	0. 033946	1898	0. 26087	0. 99247	96. 6486	0. 8776	-1. 22E+03
15	0. 031576	1547	0. 25217	0. 99327	96. 695	0. 8756	-1. 27E+03
16	0. 040309	3023	0. 28261	0. 99055	96. 5405	0. 8737	-1. 17E+03
17	0. 036346	2826	0. 26957	0. 99007	96. 4479	0. 8828	-1. 20E+03
18	0. 037654	3758	0. 27391	0. 99055	96. 5097	0. 8746	-1. 17E+03
19	0. 044451	5178	0. 29565	0. 98975	96. 5097	0. 8786	-1. 08E+03
20	0. 045748	6258	0. 3	0. 98703	96. 2625	0. 8634	-1. 08E+03
21	0. 04178	1806	0. 28696	0. 99263	96. 7568	0. 889	-1. 19E+03
22	0. 039005	2392	0. 27826	0. 99135	96. 6023	0. 879	-1. 16E+03
23	0. 045846	5988	0. 3	0. 98879	96. 4324	0. 8561	-1. 00E+03
24	0. 043146	2044	0. 2913	0. 99199	96. 7104	0. 8718	-1. 16E+03
25	0. 046006	6047	0. 3	0. 99167	96. 7104	0. 85	-1. 05E+03
26	0. 033889	3936	0. 26087	0. 99103	96. 5097	0. 8852	-1. 17E+03
27	0. 040417	1580	0. 28261	0. 99279	96. 7568	0. 8844	-1. 20E+03

Table 5.7 ANN Performance Results of 19427 Data with 13 Input Variables

Table 5.8 lists Hosmer-Lemeshow test results (C-value, degree of freedom and P-value of training, testing, and the 19427 data set) of the 27 ANN models.

Number of Hidden Units	C-value		Degree of Freedom		P-value	
0 (Single)	train	6.55	train	8	train	0.59
	test	10.42	test	8	test	0.24
	19427	11.61	19427	8	19427	0.17
2	train	4.23	train	8	train	0.84
	test	21.3	test	7	test	<0.01
	19427	9.85	19427	8	19427	0.28
3	train	4.68	train	8	train	0.79
	test	18.85	test	7	test	<0.01
	19427	9.97	19427	8	19427	0.27
4	train	7.29	train	8	train	0.51
	test	12.29	test	7	test	0.09
	19427	7.39	5102	8	19427	0.50
5	train	4.83	train	8	train	0.77
	test	11.84	test	7	test	0.11
	19427	6.20	5102	8	19427	0.63
6	train	4.98	train	8	train	0.76
	test	14.91	test	7	test	0.04
	19427	8.67	19427	8	19427	0.37
7	train	5.14	train	8	train	0.74
	test	18.55	test	7	test	0.01
	19427	7.36	19427	8	19427	0.50
8	train	7.99	train	8	train	0.43
	test	22.74	test	8	test	<0.01
	19427	13.32	19427	8	19427	0.10
9	train	7.47	train	8	train	0.49
	test	30.99	test	7	test	<0.01
	19427	12.56	19427	8	19427	0.13
10	train	5.05	train	8	train	0.75
	test	19.35	test	7	test	<0.01
	19427	9.10	19427	8	19427	0.33
11	train	14.75	train	8	train	0.06
	test	52.82	test	8	test	<0.01
	19427	27.32	19427	8	19427	<0.01
12	train	7.54	train	8	train	0.48
	test	14.38	test	7	test	0.04
	19427	5.58	19427	8	19427	0.69
13	train	3.48	train	8	train	0.90
	test	29.17	test	7	test	<0.01
	19427	10.50	19427	8	19427	0.23
14	train	5.80	train	8	train	0.67
	test	20.08	test	7	test	0.01
	19427	12.43	19427	8	19427	0.13
15	train	4.35	train	8	train	0.82
	test	7.77	test	7	test	0.35
	19427	4.27	19427	8	19427	0.83
16	train	14.96	train	8	train	0.06
	test	39.80	test	7	test	<0.01
	19427	23.39	19427	8	19427	<0.01
17	train	5.85	train	8	train	0.66
	test	30.77	test	8	test	<0.01
	19427	16.10	19427	8	19427	0.04

18	train	8.99	train	8	train	0.34
	test	39.95	test	8	test	<0.01
	19427	15.65	19427	8	19427	0.05
19	train	10.35	train	8	train	0.24
	test	79.22	test	8	test	<0.01
	19427	22.48	19427	8	19427	<0.01
20	train	7.24	train	8	train	0.51
	test	96.24	test	8	test	<0.01
	19427	34.82	19427	8	19427	<0.01
21	train	6.15	train	8	train	0.63
	test	96.82	test	8	test	<0.01
	19427	25.09	19427	8	19427	<0.01
22	train	7.50	train	8	train	0.48
	test	50.54	test	8	test	<0.01
	19427	15.38	19427	8	19427	0.05
23	train	10.38	train	8	train	0.24
	test	104.21	test	8	test	<0.01
	19427	20.93	19427	8	19427	<0.01
24	train	4.12	train	8	train	0.85
	test	17.21	test	8	test	<0.01
	19427	5.95	19427	8	19427	0.65
25	train	13.03	train	8	train	0.11
	test	73.77	test	8	test	<0.01
	19427	21.83	19427	8	19427	<0.01
26	train	4.93	train	8	train	0.77
	test	33.68	test	8	test	<0.01
	19427	15.19	19427	8	19427	0.06
27	train	9.76	train	8	train	0.28
	test	16.71	test	8	test	<0.01
	19427	10.60	19427	8	19427	0.23

Table 5.8 Hosmer Lemeshow Test Results of ANN Models with 13 Input Variables

From table 5.7, a single layer ANN model and double layer ANN models with the hidden nodes' number of 4,5, 15, are considered a good fit on observed data. The P-values of training, testing and the 19427 data sets are larger than 0.05. These models are clinically acceptable to predict mortality probability. Among those models, ANN structure with 15 hidden units has the best goodness of fit result. Table 5.9 is a Hosmer-Lemeshow contingency table of this structure on 19427 data set. Fig 5.1 shows a ROC graph of this model on both training and test sets. For a probability model, higher sensitivity does not represent better goodness of fit. From table 5.7, we can see that ANN models with 9, 12, 14, 25 hidden nodes have a sensitivity of 0.3. This value is slightly higher than the sensitivity of other models. However, the goodness

of fit of these models are not acceptable. These models cannot be used to estimate the probability of mortality.

Probability Group	Expected Death	Observed Death	Expected Survival	Observed Survival	Total Cases in the Group
<0.1	219.223	219	17526.78	17527	17746
0.1-0.2	101.0847	106	614.9153	610	716
0.2-0.3	77.35324	74	239.6468	243	317
0.3-0.4	67.3008	71	124.6992	121	192
0.4-0.5	66.92561	57	83.07439	93	150
0.5-0.6	61.91809	60	52.08191	54	114
0.6-0.7	48.2483	46	26.7517	29	75
0.7-0.8	42.7586	41	14.2414	16	57
0.8-0.9	34.3627	35	6.6373	6	41
>0.9	17.7892	18	1.2108	1	19

Table 5.9 Hosmer-Lemeshow Contingency Table of an ANN with 15 Hidden Units on the 19427 Data Set (H-L C-value=4.27; Degree of Freedom=8; P-value=0.83)

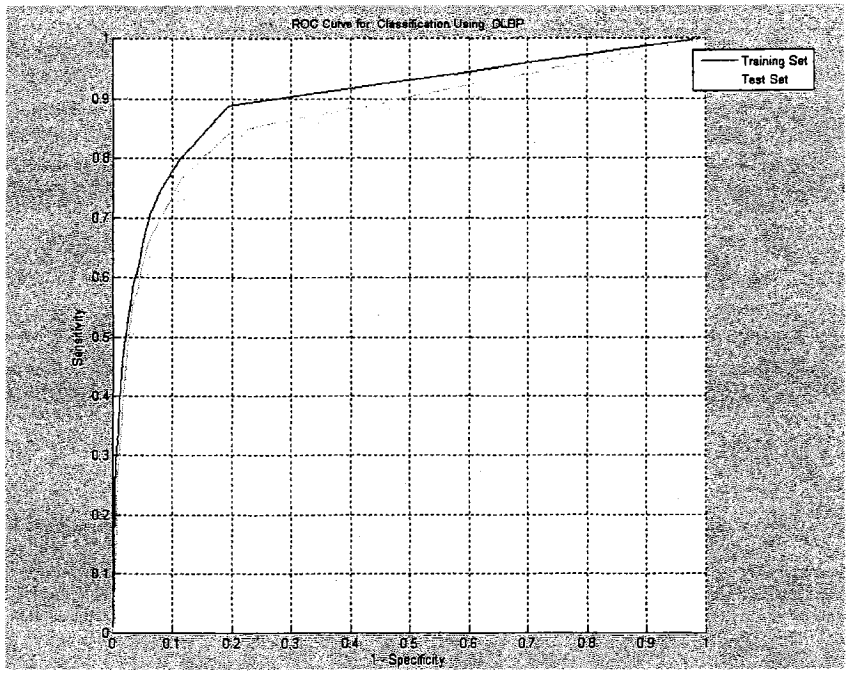

Fig 5.1 ROC Graph of the ANN model with 15 hidden units (upper curve is for the training set, the other curve is for the test set)

5.3 Comparison of Models developed from different Prior Distributions

Statistical comparison between 13-variable models developed by 5102 data set (mortality rate 9.5%) and models developed by 19427 data set (mortality rate 3.75%) is shown in table 5.10.

	Logarithmic Sensitivity (%)	Sensitivity (%)	Specificity (%)	CCR (%)	ROC Area
5102	4.96±0.97	31.14±2.86	97.68±4.29	91.46±0.34	0.87±0.004
19427	3.65±0.64	26.88±2.23	99.16±0.16	96.60±0.11	0.88±0.009

Table 5.10 Mean and Standard Deviation of Performance Comparison for 13-variable Models Developed by 5102 Data Set and 19427 Data Set (the constant predictor of the 5102 set is 90.5 and the constant predictor of the 19427 set is 96.4)

Table 5.10 shows that using the maximum likelihood estimation method encountered problems with imbalanced outcome distributions. The sensitivity is relatively low. Average sensitivity in the 19427 data set is slightly lower than average sensitivity in the 5102 data set. On the other hand, using the method presented in this thesis, we successfully obtained probability prediction models from training sets with different prior mortality distributions. Low sensitivity and imbalanced data set did not affect the development of a probability model as long as the prior distributions in training and test sets were consistent.

5.4 Comparison of Two Approaches for Improving Sensitivity

According to Bayes rules, the prior probability is a factor used to calculate the posterior probability. In training and test sets with inconsistent prior mortality distributions, a model developed by the training set is unable to approximate the true posterior probability of mortality in the test set. Therefore, inconsistent prior distributions between training and test sets will cause failure to develop a probability estimation model. The sampling approach currently used by MIRG alters the prior distribution of outcomes. Although it results in the improvement of test sensitivity, this method is yet unavailable for the problem in this thesis. For this purpose, we present two approaches to improving sensitivity without changing the prior distribution in a training set: adjustment of a cutoff point and adjustment of a neural network cost function. Table 5.11 presents the results of adjusting a cutoff point between 0 and 0.5. It uses a 13-variable ANN model with 15 hidden units as an example. The original position of cutoff point for the model was 0.5. We moved a cutoff point from 0.5 to 0.15. Sensitivity increased while values of the cutoff point decreased. This approach does not affect the model's probability estimation, because moving a cutoff point happens after model development and adjustment is within a predicted probability range.

Cutoff	0.5	0.4	0.35	0.25	0.15
Sensitivity %	25. 2	33.5	37.8	47.8	59.1
Specificity %	99. 3	98.7	98.3	97.7	95.9
CCR %	96.7	96.4	96.2	95.9	94.6

Table 5.11 The Adjustment of a Cutoff Point for Improving Test Sensitivity

The second approach, adjusting a cost function by adding a scalar *a* to the error term caused by a dominant outcome (i.e. the survival outcome), takes place during a model generation process. Therefore, whether ANN models trained by the second approach are able to estimate the mortality probability is unknown.

To train the neural network as the second method, we first add a scalar term to the ANN cost function. The new cost function is written as follows:

$$L(\theta) = \ln[L(\theta)] = \sum_{i=1}^{m} y_i * \ln O(x_i) + \alpha(1 - y_i) * \ln(1 - O(x_i)) \quad (1)$$

As the same manner of deriving the weight update formula in section 4.2, the weight update formula for updating weights and bias from hidden layer to output layer is written as follows:

$$\Delta W_{jk} = \eta \sum_{i=1}^{m} (di(1 - O(xi)) + \alpha(di - 1)O(xi))x_{ijk} \quad (2)$$

The weight update formula for updating weights from input layer to hidden layer is written as follows:

$$\Delta Wji = \eta \sum_{n=1}^{m} (d_n(1 - O(Xn)) + \alpha(d_n - 1)O(Xn)) * Wkj * \varphi'(V_j(n)) * Yi(n) \quad (3)$$

We use the two new weight update formulas in the ANN simulation tool to train neural networks. Table 5.12 still uses the aforementioned 13-variable ANN model with 15 hidden neurons as an example. It demonstrates that sensitivity was improved while *a* was reduced. Results show that the second approach can increase a model's sensitivity.

α=	1	0.6	0.45	0.25	0.15
Sensitivity %	25. 2	33.9	42.2	56.1	63
Specificity %	99. 3	98.8	98.2	97.2	94.8
CCR %	96.7	96.5	96.2	95.7	93.7

Table 5.12 The Adjustment of an ANN Cost Function for Improving Test Sensitivity

5.5 Evaluation of Goodness of Fit on the Second Approach

As aforementioned, changing a cutoff point does not affect goodness of fit. The goodness of fit will remain unchanged wherever the value of a cutoff point is. The second method, adding a scalar to a dominant error term in an ANN cost function, alters the cost function against the original likelihood function. Training neural networks with the altered cost function will not provide the rigorous maximum likelihood estimation on the training samples. Will ANN outputs approximate the posterior probability of class membership? We used α=0.3 as a scalar to an ANN cost function in order to increase test sensitivity to 50%. The 13 variables were employed as ANN inputs. 27 ANN structures were trained. Table 5.13 lists resulting ANN models.

Num of Hidden	Log-sensitivity index	Sensitivity	Specificity	Test CCR	Test ROC	C-value for 19427 set

0 (single)	0.12759	0.47391	0.97486	95.7066	0.9034	680.18
2	0.16946	0.53478	0.96845	95.305	0.8968	655.07
3	0.15359	0.51304	0.97086	95.4595	0.8969	675.21
4	0.16377	0.52609	0.97262	95.6757	0.8982	635.15
5	0.16054	0.52174	0.97294	95.6911	0.8968	692.07
6	0.14173	0.49565	0.9731	95.6139	0.901	651.51
7	0.16705	0.53043	0.9723	95.6602	0.9046	769.81
8	0.16781	0.53043	0.97534	95.9537	0.9033	736.10
9	0.15432	0.51304	0.97406	95.7683	0.9026	740.81
10	0.16753	0.53043	0.97422	95.8456	0.9058	716.30
11	0.16818	0.53043	0.97678	96.0927	0.9062	750.61
12	0.16405	0.52609	0.97374	95.7838	0.9047	704.48
13	0.15758	0.51739	0.97422	95.7992	0.9005	722.78
14	0.15732	0.51739	0.9731	95.6911	0.8995	788.83
15	0.16741	0.53043	0.97374	95.7992	0.906	688.90
16	0.17429	0.53913	0.97374	95.8301	0.9041	686.52
17	0.15407	0.51304	0.97294	95.6602	0.9024	702.89
18	0.15739	0.51739	0.97342	95.722	0.9083	735.56
19	0.16448	0.52609	0.9755	95.9537	0.9038	657.74
20	0.19237	0.56087	0.97358	95.8919	0.8986	749.72
21	0.17794	0.54348	0.97422	95.8919	0.907	690.11
22	0.15436	0.51304	0.97422	95.7838	0.8982	671.48
23	0.16745	0.53043	0.9739	95.8147	0.9046	753.29
24	0.16412	0.52609	0.97406	95.8147	0.894	732.51

25	0.17136	0.53478	0.97582	96.0154	0.9059	649.26
26	0.16786	0.53043	0.9755	95.9691	0.9074	662.16
27	0.17103	0.53478	0.97454	95.8919	0.9061	685.51

Table 5.13 ANN performance results for adjustment of a cost function

From table 5.13, we know that sensitivity for most of ANN models increased to 50%. This is a significant increase compared to the results in table 5.7. However, Hosmer-Lemeshow C-statistics for these models are much greater than 15.51 (a C-statistics value when P-value=0.05 and degree of freedom=8). This means that P-value of each model is less than 0.05 and no models are able to estimate the NICU mortality probability. Therefore, although the outcomes' prior distributions in training and test sets are consistent, the approach of adjusting an ANN cost function cannot be used to develop an acceptable probability estimation model.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

A probability model has great utility in quantitating the severity of illness in the discussions between patients' families and healthcare service providers [Groeger et al 2003]. This thesis presents a method for estimating the probability of mortality in infants admitted to the NICU using physiological variables readily obtained on admission (first 12 hours). Artificial neural network models developed in the thesis not only classify distinct outcomes (survival or death), but also predict the probability of NICU mortality. General conditions for a neural network to estimate the posterior probability of class membership were discussed in chapter 2. These conditions closely relate to the concept of maximum likelihood in the Bayesian framework. The maximum likelihood estimation is a fundamental approach in Statistics. It is used for estimating an unknown distribution of a population with a set of observed data randomly drawn from the population. We apply it here to the context of neural networks. A neural network cost function is modeled as a likelihood function so that training is able to find the maximum likelihood. For three-layered neural networks, by referring to the traditional back-propagation algorithm, a weight update formula for updating the weights and bias of hidden neurons was derived in Chapter 4. Contrary to the traditional *backward error propagation* algorithm, the new weight update formula is based on gradient ascent. ANN models developed in our research were primarily evaluated by two performance measures: the area under ROC curve and the Hosmer-Lemeshow goodness-of-fit test. The first one shows a model's discrimination ability. It indicates if the model will assign a higher probability to the babies who are going to die than to the babies who are going to survive. The second one validates the goodness-of-fit for probability estimation based on the sample data that we obtained. Test results indicate if there exists correspondence between model predicted

outcomes and real outcomes. We discussed these two performance measures in detail in Chapter 4.

Data used in our research came from the CNN database. Two data sets with different mortality distributions were extracted from the CNN database: a 5102 data set (i.e. a data set contains 5102 patient's records extracted from the CNN database) with mortality ratio of 9.5% and a 19427 data set with mortality ratio of 3.7%. The first set contained sicker babies than the second one [Ennett 2003]. The mortality distribution of the second set is close to the distribution in real NICUs. Two sets of variables were employed as the neural network inputs: 8 SNAPPE-II variables and a set of 13 variables developed by MIRG researchers.

To determine which set of variables are better for the development of probability estimation models, we first apply both sets on the 5102 data set. Initial ANN experiment results with the 5102 data set showed that the 13-variable mortality predictors slightly outperformed SNAPPE-II variables in estimate mortality probabilities. Therefore, the 13-variable input set was further applied to the 19427 data set, the entire data set of the CNN database. ANNs were trained to predict mortality probabilities in 19427 data set. ANNs with hidden neuron number 0, 4, 5, 15 are able to estimate the probability of mortality in CNN database. Experimental results prove that the maximum likelihood method proposed in the thesis was successfully applied on CNN database for the development of probability estimation model of NICU mortality.

Experiments showed that ANNs trained with the maximum likelihood method were affected by the problem of imbalanced outcome distributions. A low mortality distribution in a training set resulted in a low sensitivity. For increasing sensitivity without changing prior distributions

in training sets, two approaches were discussed in Chapter 4 and 5. One is by adjusting a neural network cost function. The other is by moving a classification cutoff point.

Further experiments showed that both approaches significantly improved sensitivity. However, the method by adjusting a cost function could not develop any clinically acceptable probability prediction models. Hence, a model with high sensitivity may not result in good fit. The cutoff point approach can separate groups of dichotomous outcomes by setting the point within a range of estimated probabilities. An ANN-based probability prediction model trained from an imbalanced data set can apply this approach to obtain a good sensitivity and specificity, thus leading to a mortality-screening tool.

6.2 Contribution

Academic contributions are listed below based on thesis results.

1) This thesis supplements MIRG's previous ANN research and models by using a new approach that allows to estimate the probability of NICU mortality, replacing the death or survival dichotomous output. A gradient-ascent-based weight-update algorithm was derived. An ANN-PPF tool based on the formula for neural network simulation was developed and successfully used in this research. This provides new and important information for the physicians using our tools.

2) The thesis tested and confirmed that the 13-variable mortality predictor model developed by Ennett et al. perform better than the SNAPPEII variables developed by logistic regression and can be successfully used with this new ANN tool. This model, updated to output probability of mortality can be used either alone or in concert with

the diagnosis of clinicians may be very useful in processes related to NICU resource utilization and review and patient management.

3) We presented two approaches for improving sensitivity of ANN models. Although the adjusting cost function method cannot be applied on probability models according to the experimental results, it can replace MIRG's sampling approach for dealing with skewed data. This method could be of interest to MIRG' researchers in the future.

4) The thesis is the first application of maximum likelihood based ANN approach and Hosmer-Lemeshow goodness-of-fit test on NICU data to develop probability prediction models.

6.3 Future Work

Probability itself is an important numerical decision support tool in a clinical environment. The method described in this thesis can potentially be used for estimating many other NICU outcomes. Future work will add probability predictions to ventilation duration and to the occurrence of complications such as neuro-image abnormality, necrotizing entero-colitis, and broncho-pulmonary dysplasia. We intend to apply the maximum log-likelihood stopping criterion to ANN training.

Artificial neural networks are not the only choice for developing clinical probability prediction models. As we presented in Chapter 3, many approaches such as logistic regression, kernel -based methods, Bayesian networks, naïve Bayes classifier, etc. can be applied based on problems that we have. In chapter 3, in addition to the literature review, some ideas for using

these methods were proposed. The ultimate goal for our research is to develop reliable and usable clinical decision support systems and tools. Therefore, any potential solution should be compared and investigated in future studies.

References

1] Ampazis,N., Introduction to Neural Networks, http://iit.demokritos.gr/neural/intro/

2] Barnett GO, Cimino JJ, Hupp JA, Hoffer EP. An evolving diagnostic decision-support system. JAMA 1987;258:67-74.

3] Baxt WG. Complexity, chaos and human physiology: the justification for non-linear neural computational analysis. Cancer Lett 1994;77:85-93.

4] Beiu V, A novel highly reliable low-power nano architecture when von Neumann augments Kologorov in Proc. IEEE Application-Specific Systems, Architectures and Processors, 2004, pp. 167-177.

5] Bishop, C.M., Neural networks for pattern recognition, Oxford, Clarendon Press, 1995.

6] Bradley, A. P., The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, 30 (7), 1145-1159. 1997

7] Broughton, Simon J., Berry, Andrew, Jacobe, Stephen, Cheeseman, Paul, et al., The Mortality Index for Neonatal Transportation Score: A new mortality prediction model for retrieved neonates. PEDIATRICS Vol. 114 No. 4 October 2004, pp. e424-e428

8] Cullen DJ, Civetta JM, Briggs BA, et al. Therapeutic intervention scoring system: a method for quantitative comparison of patient care. Critical Care Med, 1974;2:57-60.

9] Demuth H, Beale M. Neural network toolbox user's guide, Version 3.0. The MathWorks Inc. Natick, MA, 1997.

10] Drummond, C, Holte, R C, What ROC curves can't do (and cost curves can) Proceedings of the ROC Analysis in Artificial Intelligence, 1st International Workshop. Valencia, Spain. August 22, 2004. pp. 19-26.

11] Duda RO, Hart PE, Stork DG, Pattern classification, New York Wiley, 2001.

12] Ennett, Colleen M., Imputation of missing values by integrating artificial neural networks and case-based reasoning, Phd. Thesis, Department of Systems and Computer Engineering, Carleton University, 2003

13] Ennett, Colleen M., Frize, Monique, Charette Elaine, Improvement and automation of artificial neural networks to estimate medical outcomes, Med Eng Phys 2004 321-328

14] Ennett CM, Frize M. Selective sampling to overcome skewed a priori probabilities with neural networks. Proc AMIA Symp 2000;(20 Suppl):225-9.

15] Ennett CM, Frize M, Scales N. Logarithmic-sensitivity index as a stopping criterion for neural networks. Proc IEEE EMBS-BMES 2002.

16] Ennett CM, Frize M, Scales N. Evaluation of the logarithmic-sensitivity index as a neural network stopping criterion for rare outcomes. Proc IEEE ITAB 2003.

17] Fawcett, Tom, ROC Graphs: notes and practical considerations for data mining researchers, HPL-2003-4, 20030117 External

18] Fausett L. Fundamentals of neural networks. Prentice-Hall: Englewood Cliffs, NJ, 1994

19] Field D , Manktelow B, Draper ES. Bench marking and performance management in neonatal care: easier said than done. Arch Dis Child Fetal Neonatal Ed 2002;87:F163 - 4.

20] French, Leon, Ngom, Alioune, Rueda, Luis, Fast protein superfamily classification using principal component null space analysis, School of Computer Science, University of Windsor. http://socr.uwindsor.ca/~french1/PCNSA.html

21] Frize M, Taylor KB, Nickerson BG, Solven FG, Borkar H. A knowledge-based system for the intensive care unit. Proc of the 15th Ann. Int. Conf. IEEE/EMBS, 1993; San Diego:677-678.

22] Frize, M.; Ennett, C.M.; Charette, E.; Automated optimization of neural networks in estimating medical outcomes Information Technology Applications in Biomedicine, 2000. Proceedings. 2000 IEEE EMBS International Conference on 9-10 Nov. 2000 Page(s):168 - 173 Digital Object Identifier 10.1109/ITAB.2000.892380

23] Frize M, Ennett CM, Stevenson M, Trigg HCE. Clinical decision-support systems for intensive care units using artificial neural networks. Med Eng Phys 2001 Apr; 23(3):217-25.

24] Frize M, Frasson C. Decision-support and intelligent tutoring systems in medical education. Clinical and Investigative Medicine, 2000 Aug;23(4):266-269.

25] Frize M, Wang L, Ennett C, Nickerson BG, Solven FG, Stevenson M. New advances and validation of knowledge management tools for critical care using classifier techniques. Proc. AMIA Annual Symposium 1998:553-558

26] Frize M, Solven FG, Stevenson M, Nickerson BG, Buskard T, Taylor K. Computer-assisted decision-support systems for patient management in an intensive care unit. Proc. Medinfo '95 1995; Vancouver:1009-1012. 14.

27] Gagliardi, L; Cavazza, A; Brunelli, A; Battaglioli, M; Merazzi, D; Tandoi, F; Cella, D; Perotti, G F; Pelti, M; Stucchi, I; Frisone, F; Avanzini, A; Bellu, R; and the NNL study group Assessing mortality risk in very low birthweight infants: a comparison of CRIB, CRIB-II, and SNAPPE-II. Archives of Disease in Childhood Fetal & Neonatal Edition. 89(5):F419-F422, September 2004.

28] Gorry, G.A., and M.S. Scott Morton, A framework for management information systems, Sloan Management Review 13(1):49-62, 1989

29] Gray JE, Richardson DK, McCormick MC, Workmann-Daniels K, Goldmann DA, Neonatal therapeutic intervention scoring system: a therapy-based severity-of-illness index. Pediatrics 1992 Oct;90(4):561-7.

30] Groeger JS, Glassman J, Nierman DM, et al. Probability of mortality of critically ill cancer patients at 72 h of intensive care unit (ICU) management Support Care Cancer (2003) 11:686–695 DOI 10.1007/s00520-003-0498-9

31] Han, J, Kamber, M, Data mining: concepts and techniques. (Chapter 7), pp.196-199 2001

32] Haykin, Simon, Neural networks : a comprehensive foundation, Upper Saddle River, N.J. Prentice Hall, c1999.

33] Hanley, J. A., & McNeil, B. J., The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology, 143, 29-36. 1982

34] Hosmer, David W., Lemeshow, Stanley, Applied logistic regression, Chapter 5, New York : Wiley, c1989.

35] Hosmer DW, Taber S, Lemeshow S, The importance of assessing the fit of logistic regression models: a case study, American Journal of Public Health, Vol 81, Issue 12 1630-1635, American Journal of Public Health, Vol 81, Issue 12 1630-1635, 1991

36] International Neonatal Network. The CRIB (clinical risk index for babies) score: a tool for assessing initial neonatal risk and comparing performance of neonatal intensive care units. Lancet 1993;342:193-8.

37] Jaimes F, Farbiarz J, Alvarez D, Martínez C Comparison between logistic regression and neural networks to predict death in patients with suspected sepsis in the emergency room. Crit Care. 2005; 9(2): R150-R156.

38] Keene AR, Cullen DJ. Therapeutic intervention scoring system: update 1983. Crit Care Med 1983;11:1-3

39] Kononenko, I, Inductive and Bayesian learning in medical diagnosis, Applied Artificial Intelligence, 7:317-337. 1993

40] Lemeshow S , Le Gall J-R., Modeling the severity of illness of ICU patients: a systems update. JAMA, 272(13):1049-1055. 1994

41] Lippmann RP, Moody J, Touretzky DS,. Advances in neural information processing systems (NIPS'90), Volume 3. Morgan Kaufmann: San Mateo. 1991;3:875-882.

42] Long, William J., Griffith, John L., Selker, Harry P., A comparison of logistic regression to decision-tree induction in a medical domain, 1993

43] Marakas, George M., Decision support systems in the twenty-first century, Upper Saddle River, N.J. : Prentice Hall, c1999.

44] Mennon,A.,K.Mehrotra,C.K. Mohan, and S. Ranka, Characterization of a class of sigmoid functions with applications to neural networks, Neural Networks, vol.9,pp.819-835 1996

45] Michie, D., Spiegelhalter, D.J.,Taylor, C.C., Machine learning, neural and statistical classification, (edited collection). New York: Ellis Horwood, 1994

46] Mitchell, Tom M, Machine learning. Chapter 6, New York : McGraw-Hill, c1997.

47] Neural Network **http://www.statsoft.com/textbook/stneunet.html#pnn**

48] Neural Network Theory and Matlab Application Implementation

49] Onisko, Agnieszka, Druzdzel, Marek J., Wasyluk, Hanna, A Bayesian network model for diagnosis of liver disorders, In Proceedings of the Eleventh Conference on Biocybernetics and Biomedical Engineering, pages 842-846, Warsaw, Poland, December 2-4, 1999

50] Parry G et al. CRIB II : an update of the Clinical Risk Index for Babies score. Lancet 2003;361(9371):1789-91

51] Payne TH. Computer decision support systems. Chest 2000;118:47S-52S.

52] Penny W, Frost D. Neural networks in clinical medicine. Medical Decision Making 1996;16:386-398.

53] Pollack MM, Koch MA, Bartel DA, *et al.* A comparison of neonatal mortality risk prediction models in very low birth weight infants. Pediatrics 2000;105:1051 – 7.

54] Scales, N. Mortality prediction in an ICU and coronary surgery database using neural networks. Term project, Department of Systems and Computer Engineering, Carleton University, Dec 2001.

55] Shepherd, Adrian J. Second-order methods for neural networks: Fast and reliable training methods for multi-layer perceptrons, Springer-Verlag, London. 1997

56] Specht, D.F. Probabilistic neural networks, Neural Networks, 1990 vol. 3, pp.109-118.

57] Stevens SM, Richardson DK, Gray JE, Goldmann DA, McCormick MC. Estimating neonatal mortality risk: an analysis of clinicians' judgments. Pediatrics. 1994 Jun;93(6 Pt 1):945-50

58] Ramoni M, Sebastiani, P, Bayesian methods for intelligent data analysis. In: Intelligent data analysis, an introduction. Berthold M, Hand D, editors. New York, NY: Springer, pp. 128–166. 1999

59] Richardson DK, Gray JE, McCormick MC, Workmann K, Goldmann DA. Score for neonatal acute physiology: a physiologic severity index for neonatal intensive care. Pediatrics 1993 Mar;91(3):617-23. [1993a]

60] Richardson DK, Phibbs CS, Gray JE, et al. Birth weight and illness severity: independent predictors of neonatal mortality. *Pediatrics* 1993;91:969–75. [1993b]

61] Richardson DK, Corcoran JD, Escobar GJ, Lee SK. SNAP-II and SNAPPE-II: simplified newborn illness severity and mortality risk scores. J Pediatr 2001;138:92-100.

62] Rumelhart DE, Hinton GE, Williams RJ. Learning internal representations by error propagation. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol 1:318-362. MIT Press: Cambridge, MA, 1986. [1986a]

63] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986;323:533-536. [1986b]

64] Rybchynski, DR. Design of an artificial neural network research framework to enhance the development of clinical prediction models, Master thesis. Department of Electrical Engineering, University of Ottawa, Ottawa Ontario, 2005

65] Tasoulis,D.K. Vladutu,L. Plagianakos,V.P. Bezerianos,A. Vrahatis M.N., Online neural network training for automatic ischemia episode detection
http://www.math.upatras.gr/~dtas/papers/TasoulisVPV2004.pdf

66] Titsias M, Likas A, A probabilistic RBF network for classification IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)-Volume 4 p. 4238

67] Trigg HCE. An investigation of methods to enhance the performance of artificial neural networks used to estimate ICU outcomes. Master's thesis, Department of Electrical Engineering, University of New Brunswick, Fredericton, NB, Jan 1997.

68] Trowbridge R, Weingarten S. Clinical decision support systems, Chapter 53
http://www.ahrq.gov/clinic/ptsafety/chap53.htm

69] Wasserman, P.D. Advanced methods in neural networks, Van Nostrand Reinhold, New York, (Chapter 3), pp.35-55. 1993

70] Weigend AS, Rumelhart DE, Huberman BA. Generalization by weight-elimination with application to forecasting.

71] Wilson, R., MartinezThe, T.R., Inefficiency of Batch Training for Large Training Sets, In Proceedings of the International Joint Conference on Neural Networks (IJCNN2000), Vol. II, pp. 113-117, July 2000.