Chapter 4

Methodology

The aim of this chapter is to describe the methodology, which will be used in subsequent chapters to comparatively assess the application of MLP type Neural Networks, of varying model complexity and a Logistic Regression model to clinical decision making problems from the field of Psychiatry. This methodology arises directly from the discussions and discourses of Chapters 2 and 3. The methodology consists of the following components:

- LD and MLP Neural Network Training
- Model Selection and Model Comparison
- Significance tests
- Estimating the future performance a model

4.1 LD and MLP Neural Network Training

Model Implementations

The primary aim of empirical studies, carried out in this thesis, is to compare a Logistic Discriminant model (LD, which is effectively a Logistic Regression) with Multi-Layer Perceptron (MLP) models.

The LD model can be implemented as MLP model with zero hidden units, where the inputs are directly connected to a single logistic output unit. As well there is an additional bias unit, which always fires a value of one, which also connects to the output unit (see Chapter 2 and Appendix 2 for a detailed exposition).

MLP models contain a layer of logistic hidden units which are intermediate between the inputs and a single logistic output unit. In this case, inputs connect with these hidden units. The hidden units, in turn, connect with a single logistic output unit. As well there is an additional bias unit, which always fires a value of one, which also connects to the hidden units and the output unit (see Chapter 2 and Appendix 2 for a detailed exposition).

The number of hidden units in a model, determine the complexity of that model. An LD model, has zero hidden units and is the least complex model. It is a linear model, which partitions an input space into two regions (one for cases belonging to one class and the other for cases belonging to the other class) using a single linear function.

MLP models are more complex, with their complexity indexed by the number of hidden units they contain. The least complex MLP model contains 2 hidden units (because having one hidden unit is identical to having zero hidden units). An MLP with 2 hidden units partitions the input space into two regions using a non-linear boundary, implemented by the joint boundary created by two linear functions (one each for each hidden unit). The next more complex MLP model has 3 hidden units, and partitions the input space into two regions using a non-linear boundary constructed from 3 linear functions. An MLP with four hidden units construct a non-linear boundary from 4 linear functions, an MLP with five hidden units construct a non-linear boundary from 5 linear functions, and so on. As the number of hidden units increases, so does the number of linear functions used to construct the decision boundary. As the number of hidden units increases, the complexity (number of possible twists and turns) of the constructed boundary also increases. See Chapter 2 and Appendix 2 for a detailed exposition.

Model complexity, in general can be measured by the number of adjustable parameters in that model. As the number of hidden units in an MLP model increases, so does the number of parameters. For a given input dimension (number of inputs), an LD model has the fewest number of parameters, an MLP model with 2 hidden units has just more than twice the number of parameters of the LD model, an MLP model with 3 hidden units has just over three times the number of parameters of the LD model, and so.

The tradeoff between bias and variance, discussed in chapter 2, has a decisive impact here. As we increase complexity (in blocks which are multiples of the input dimension) from LD to MLP2, to MLP3, etc, error due to bias decreases and error due to variance increases. Within the bounds of fixed size training dataset, there will be a point, on the complexity continuum (gradated by whole numbers of hidden units), where shifting from one complexity level to the next highest complexity level, results in a poorer partitioning (in terms generalised classification accuracy) of the input space. Our general aim is to attempt to identify this point on the complexity continuum. However at a fixed size training dataset, particularly with kinds of datasets available in psychiatry (that is relatively small sized datasets which have relatively large number of input variables), this point is likely to be at the low end of the continuum. MLP models, with many hidden units and therefore a very large number of parameters are highly likely to overfit on small datasets and therefore have poor classification accuracy when applied to future cases. To avoid this possibility, we are limiting our consideration of models to the LD model and one MLP model with 3 hidden units, in our investigations of clinical datasets reported in later chapters

Software

All models, LD and MLP are implemented using the NevProp 4.0, software [Goodman 1999]. This package was designed specifically for the application of MLPs to bio-medical problems [Goodman & Harrel 2001].

Error Function

For classification problems the Cross-Entropy Error function, is a better choice of Training Error criteria (the measure that is minimised by training) than is Mean Square Error, a criteria more traditionally used in regression problems (Hastie et al 2001). Thus in all the subsequent studies the training criteria for all Neural Network models (including the Logistic Discriminant model) will be the Cross-Entropy error function (see Appendix 2 for an extended discussion and a definition).

Optimisation Algorithm

In this thesis, we will use the QuickProp algorithm [Falhman, 1987], for both the LD and MLP classifiers to find an optimal set of weights for each classification problem. A more detailed exposition of optimisation algorithms (including QuickProp) for MLP type neural networks is contained in Appendix 2.

Weight Decay

Weight Decay is a technique, applied during optimisation training, which reduces towards zero, individual weights which do not contribute to model. This reduces the effective complexity of the model, by eliminating unneeded (in terms of the model) weights. In the studies reported later in this thesis a weight decay of - 0.01 was used to train all models. See Chapter 2 and Appendix 2 for a detailed exposition.

Stop Training Criteria

As outlined in chapter 2 (see Figure 2.7) the most ideal point on the training iterations continuum at which to stop training, is the turning point at which Test Set Error stops decreasing and begins to increase. In order to estimate the location of this point, with respect to training, the following training procedure will be employed to train all models (LD and MLP):

1. Randomly split the Full Dataset into a Training Dataset (75%) and a ‘hold out’ Test Dataset (25%).

2. Train the model, on the Training Dataset, using cross-entropy error as the training criteria. Continue training for up to 1,000 iterations (of the QuickProp algorithm), and simultaneously measure cross-entropy error on the ‘hold out’ Test Dataset.

3. Determine the minimum value of the ‘hold out’ Test Dataset Cross-Entropy error and identify the corresponding value of Training Dataset Cross-Entropy error associated with this point in training. Call this value the *Training Target Value*.

4. Repeat steps 1 to 3 above 5 times. That is create five 25% hold out sets and train on the remaining 75% five times. This results in five *Training Target Values*. Average these to arrive at an *Average Training Target Value*.

5. Train the model again on the Full Dataset, and stop training when Cross-Entropy Error measured on this full dataset is equal to the *Average Training Target Value*.

The training procedure described above is implemented as an optional algorithm in the Nevprop 4.0 software used in this thesis. The procedure is referred to as “Hold-Out training” in the Nevprop Manual (Goodman, 1999).

4.2 Comparison of Linear and Non-Linear Classification

Our hypothesis, which we wish to test in respect of a number of different clinical datasets, is that an MLP generated non-linear model will classify better a Logistic Regression generated linear model. This immediately raises a number of questions such as: which MLP generated non-linear model (amongst several) do we want to use as our comparator to a Logistic Regression generated model? and what measure of classifier performance will we use as our basis for comparison?

Model Comparison using Bootstrapping

Bootstrapping is a resampling procedure, which can be used to derive an estimate of the accuracy of a classifier when it is applied to future cases. Therefore it can be used in place of a test set. In the studies reported in later chapters we will draw 100 Bootstrap samples (by sampling with replacement) from the original training dataset. Each bootstrap sample has the same N as the original training dataset, but it is not an identical copy of the original training dataset. It will contain duplications of some cases from the original training dataset (because once sampled a case is replaced and can be resampled) and it will not contain some cases (because they not chosen in N samplings). On average each bootstrap derived sample will contain 63% of the cases in the original training dataset [Goodman 1999]

The model, under examination, is trained on each the 100 bootstrap samples and this creates 100 sets of parameter estimates. The model is then applied to original full training dataset, 100 times, each time with a different bootstrap sample derived parameter set. For each bootstrap sample, two estimates of classification accuracy (measured as Area Under the ROC Curve - $\boldsymbol{A_z}$) are calculated for each parameter set. The first, we will call it B, is calculated as the $\boldsymbol{A_z}$ of the model (using bootstrap sample derived parameter set) applied to its own training set (the bootstrap sample). The second, we will call it C, is calculated as the A_z of the model (using the bootstrap sample derived parameter set) applied to the original full training dataset. That is we are using the original full training dataset as a test set for the bootstrap derived training dataset. On average, values of B for each bootstrap sample are more optimistic (higher) than the corresponding value of C. By subtracting C from B for each bootstrap sample, we can calculate a third value (B – C). This value is an estimate of the magnitude of the optimistic bias contained in the value for B. If we average the value of (B – C) across the 100 bootstrap samples, we derive an estimate of the optimistic bias (OB) of this model. We can then use this estimate of the optimistic bias of this model to derive an estimate of the classification accuracy of this model on future cases, by subtracting OB from the value of A_z obtained by training the model on the full training dataset and measuring it on the full training dataset. This gives us a corrected value for the A_z, which we will refer to later as 'bootstrap corrected A_z '. This bootstrap correction procedure is implemented as an algorithm in the NevProp 4.0 software, we are using to implement our models. Bootstrapping can provide an unbiased estimate of the cross-validation accuracy (that is accuracy for classifying future cases) associated with a model [Hastie et al 2001, Goodman et al 1996]. See Chapter 2 and Appendix 2 for a more detailed exposition of the bootstrap correction procedure.

Non-Linear Model Complexity Selection

We can vary the complexity of the MLP generated non-linear models, almost indefinitely, by varying the number of hidden units in an MLP, starting at 2 and proceeding towards infinity. For practical problems such as ours, we can apriori exclude models with a large numbers of hidden units (say 10 plus), because these models will each contain a very large number of parameters, which in turn will lead to overfitting. However this still leaves us with a choice from a range of models (with 2 to 9 hidden units) of increasing complexity, indexed by the number of hidden units. From our previous examination of the Bias-Variance Trade Off, we know that for each increment in complexity, error due to bias will decrease and error due to variance will increase, and that the change in error between models is a function of Bias and Variance only.

The aim of model selection is to choose a model (amongst two or more) that is likely to perform well on future cases. A spare (used only for model selection) large test dataset could be used as the basis for model selection. However in less data rich circumstances (the norm in psychiatric research), we need to find another way. If we use Bootstrap derived indices of accuracy as our basis for model selection, then we cannot later 're-use' these indices as a basis for comparing the 'chosen' model with another model(s). This is because the bootstrap derived indices have been used as part of the design process of the model, in that they have been used for the purposes of selecting the value of a model hyperparameter: the model complexity level.

To avoid these problems, in this thesis we going to use a training dataset derived score of a model(s)'s generalisation performance, the Akaike Information Criterion (AIC). Our discussion of the Bias-Variance Trade Off in Chapter 2 (see figure 2.8), indicates that at one point somewhere along a complexity dimension, classification error measured on a Test Dataset will be at a minimum, and that this point is the best criteria for selecting a model of a given degree of complexity, as the best model. The best model is the one which offers the best trade off between error due to Bias and error due to variance. The AIC is a single index or score which can be calculated for any model, based only training dataset derived values, which takes into account estimates of the impacts of both bias and variance.

$$\mathrm{AIC} = -2\log(L) + 2p \qquad \textbf{(4.1)}$$

Where $-\mathrm{Log}(L)$ is the negative log likelihood of the model

p is the number of parameters in the model

The first term, -2Log(L), is an estimate of the error due to Bias, whilst the second term, 2p, is an estimate of the error due to variance. As models increase in complexity the Bias term will become smaller, because the dataset is fitted better and the variance term will become larger, as the number parameters increases. We want models which trade off bias and variance to give the smallest overall error. Thus using AIC as a proxy for this, we will select the model which has the smallest AIC value.

For practical purposes in our studies we can calculate the AIC for all our models using the following procedures. The Nevprop 4.0 software we are using to implement our models, gives the average cross-entropy value for each model as a summary statistic. This value is equivalent to the value of the average negative log likelihood of the model on the training dataset (Goodman, 1999). Thus if we multiply this value by 2 and by the number of cases in the training set, we now have the value first term of the AIC for that model. The second term, the number of model parameters, is calculated by multiplying the number of inputs plus one by the number of hidden unit, then to this result adding the number of hidden units plus one and finally multiplying the last result by two.

4.3 Significance tests

The most frequently used criterion, in the medical literature, for comparing classifiers is A_z (the Area Under the ROC Curve, see Chapter 3). Once we have selected an MLP neural network model, on the basis of the AIC measure, then we will want to compare this MLP non-linear model to the Linear Logistic Regression model, on the basis of their bootstrap corrected A_z values, to determine if the selected non-linear MLP model offers us any improvement as a classifier of future cases beyond that which can be obtained using a linear Logistic Regression model.

Hanley & McNeil [1982,1983] discuss the use of ROC Curves to evaluate classifiers and present a test of statistical significance which can be used to compare the Areas under the ROC Curves derived by applying different classifiers to the same cases. Their test is used to compute a z score, which can then be converted to a probability, that the two Areas under the ROC Curves are the same. Hanley & McNeil [1983]'s formula for computing z is:

$$z = \frac{A_1 - A_2}{\sqrt{SE_1^2 + SE_2^2 - 2rSE_1SE_2}} \qquad \textbf{(4.2)}$$

Where A_1 and A_2 are the two Areas under the ROC Curves

SE_1 and SE_2 are the corresponding Standard Errors of the two Areas under the ROC Curves r is the correlation coefficient between the two Areas under the ROC Curves. A table for calculating r from case data is given in Hanley & McNeil [1983]. In order to calculate r we first need to calculate r_{pos} and r_{neg} , the correlations between individual case outputs of the two models, for the subset of cases which are positive (target value = 1) and the subset of cases which are negative (target value = 0), respectively

Hanley & McNeil [1982] show that The Standard Error of ROC Curve (SE_{ROC}) is related to the Standard Error of the Wilcoxon Statistic, and can be calculated as follows:

$$SE_{ROC} = \sqrt{\frac{AUC(1-AUC)+(N_P-1)(Q_1-AUC^2)+(N_N-1)(Q_2-AUC^2)}{N_N * N_P}} \quad \textbf{(4.3)}$$

Where $Q_1 = AUC/(2-AUC)$ and $Q_2 = 2AUC^2/(1+AUC)$

N_P = number of positive cases and N_N = number of negative cases

4.4 Determining the Future Performance of a Classifier

Once a set of candidate classifiers have been evaluated and one is chosen as the best (of the set), then we will want to know how well this classifier will operate in the real world. In our case, we have already made extensive use of our dataset for training (using the holdout method to determine the stop training point) and for model selection (using AIC). In particular, we cannot re-use, the bootstrap corrected A_z of our model, because it has already been used as the model comparison criteria. Thus, the bootstrap corrected derived A_z is part of the design parameters of our final model and therefore not independent of this model.

In such a case, we will need to obtain another independent dataset, apply our model to this independent dataset and use the performance of our model (which can be measured by various indices, see Chapter 3) on this independent dataset as an estimate of the future performance of our classifier. Due to the limitations of available data, this procedure could only be carried out with one of our clinical studies, the Autistic Disorder Diagnosis Study (Chapter 7).

4.5 Summary

The methodology used later in this thesis

1. Use NevProp 4.0 software package to implement LD and MLP models. This controls for differences due to different software idiosyncrasies.
2. Use Cross-Entropy as the error function for training
3. Use QuickProp as the optimization algorithm
4. Use weight decay during training
5. First train using 75% of training dataset. Use the remaining 25% of the training dataset as a 'holdout' Test Set, which is used to determine a stop training point in terms of a training error 'Target Value'. Repeat 5 times and average the obtained 'Target Values'.
6. Re-train using 100% of the training dataset. Stop training when training error reaches the 'Average Target Value'.
7. Use the above procedures to train a Logistic Regression model (no hidden units = Linear Discriminant (LD)) and eight MLP neural network model with 2 to 9 hidden units.
8. Use AIC to select one model from the eight MLP models as the 'best'
9. Statistically compare the bootstrap corrected A_z values of the Logistic Regression model and the selected MLP model using Hanley and McNeil's [1983] formula.
10. Optionally, use an independent test dataset, to estimate the future performance of the classifier with new cases.