Application of Multilayer Perceptrons for Response Modeling

A. Nachev and M. Hogan
Business Information Systems, J.E. Cairnes School of Business & Economics, NUI, Galway, Ireland

Abstract - This study explores the predictive abilities of multilayer perceptrons used for response modeling in direct marketing campaigns. We extend previous studies discussing how neural network design affects the model performance and also propose a simplified architecture, which outperforms the ones used before. We explore the variance in the neural network behaviour due to the randomness factor and validate the figures of merit by a rigorous testing procedure not applied in the previous studies. The model performance is estimated and analyzed using accuracy, ROC, AUC, lift, and precision-recall. We also compare multilayer perceptrons with logistic regression, naive Bayes, linear discriminant analysis, and quadratic discriminant analysis.

Keywords: direct marketing, data mining, neural networks, multilayer perceptron.

1 Introduction

Marketing considers two main approaches to communicate with potential buyers: mass marketing, which targets the audience by broadcasting channels, and direct marketing, which uses one-to-one communication with potential buyers in the form of mailing, phone calls, delivery of promotional materials, etc. Nowadays, with increased competition, particularly in the financial and bank sector, direct marketing is a highly appreciated approach due to its efficiency, less waste of resources and cost-effectiveness. A well-executed direct marketing campaign can provide a positive return on investment by motivating customers to respond to the promoted services or goods. One of the preparatory stages ensuring a high return on investment consists of carefully processing the data available in order to generate an appropriate selection of potential buyers.

Formally, the direct marketing task is to use customers' historical purchase data in order to identify those who are most likely to respond positively to a new product/service.

In the terminology of data mining, this is a classification task: having historical data, which consists of attributes and class labels, a well trained model can classify, i.e. split a set of potential customers into two classes: the ones to be contacted, because of their high probability of response; and those not to be contacted, because they are unlikely to make a purchase. The term 'high' is subject of control — that is the cutoff point used to map probabilities into yes/no labels.

Many data mining and machine learning techniques have been involved to build decision support models capable of predicting the likelihood if a customer will respond to the offering or not. These models can perform well or not that well depending on many factors, an important of which is how training of the model has been planned and executed.

Recently, neural networks have been studied in [5, 6, 8, 14] and regarded as an efficient modelling technique. Decision trees have been explored in [5, 6, 8, 13]. Support vector machines are also well performing models discussed in [8, 11, 14]. Many other modelling techniques and approaches, both statistical and machine learning, have been studied and used in the domain.

In this paper, we focus to the neural network models, discussing the factors, which affect their performance and capabilities to predict. We extend the methodology used in [5, 6, 8] addressing issues, such as optimisation of the model hyper-parameters, data validation techniques and metrics used to estimate models behavior, and also propose an architecture, which outperforms the ones used before. The remainder of the paper is organized as follows: section 2 provides an overview of the neural networks, in particular multilayer perceptrons; section 3 discusses the dataset used in the study, its features, and the preprocessing steps needed to prepare the data for experiments; section 4 presents and discusses the experimental results; and section 5 gives the conclusions.

2 Multilayer Perceptron

A variety of neural network models are used by practitioners and researchers for clustering and classification, ranging from very general architectures applicable to most of the learning problems, to highly specialized networks that address specific problems. Each model has a specific topology that determines the layout of the neurons (nodes) and a specific algorithm to train the network or to recall stored information. Among the models, the most common is the multilayer perceptron (MLP), which has a feed-forward topology. Typically, an MLP consists of a set of input nodes that constitute the input layer, an output layer, and one or more layers sandwiched between them, called hidden layers (Figure 1). Nodes between subsequent layers are fully connected by weighted connections so that each signal travelling along a link is multiplied by its weight $w_{ij}$. Hidden and output nodes receive an extra bias signal with value 1 and weight $\theta$. The input layer, being the first
Layer, has size (number of nodes), which corresponds to the size of the input samples. Each hidden and output node computes its activation level by:

\[ s_i = \sum_j w_{ij}x_j + \theta \] (1)

and then transforms it to output by an activation function. The MLPs we use in this study work with the logistic activation function:

\[ f_j(s_i) = \frac{1}{1 + e^{-\beta s_i}} \] (2)

where \( \beta \) is slope parameter. The overall model is given in the form:

\[ y_i = f_i(w_{i,\theta} + \sum_{j=1}^{I_H} f_j(\sum_{n=0}^{I} w_{m,n}x_n + w_{m,\theta})w_{i,n}) \] (3)

where \( y_i \) is the output of the network for node \( i \), \( w_{ij} \) is the weight of the connection from node \( j \) to \( i \) and \( f_j \) is the activation function for node \( j \). For a binary classification, there is one output neuron with logistic activation function.

The MLPs we use in this study work with the logistic activation function:

\[ f_j(s_i) = \frac{1}{1 + e^{-\beta s_i}} \] (2)

where \( \beta \) is slope parameter. The overall model is given in the form:

\[ y_i = f_i(w_{i,\theta} + \sum_{j=1}^{I_H} f_j(\sum_{n=0}^{I} w_{m,n}x_n + w_{m,\theta})w_{i,n}) \] (3)

where \( y_i \) is the output of the network for node \( i \), \( w_{ij} \) is the weight of the connection from node \( j \) to \( i \) and \( f_j \) is the activation function for node \( j \). For a binary classification, there is one output neuron with logistic activation function.

![Figure 1. Architecture of an MLP neural network with one hidden layer.](image)

The purpose of the hidden layer(s) is to extend the neural network abilities to learn. Several hidden layers can be placed between the input and output layers, although for nearly all problems, one hidden layer is sufficient. In that case, the hidden layer simply converts inputs into a nonlinear combination and passes the transformed inputs to the output layer. The most common interpretation of the hidden layer is as a feature extraction mechanism. That is, the hidden layer converts the original inputs in the problem into some higher-level combinations of such inputs. Using two or more hidden layers rarely improves the model, and it may introduce a greater risk of bad learning (local minima).

The number of input and output nodes is defined by the problem. The number of hidden nodes to use is far from clear. There is no theory yet to tell how many nodes are needed to solve a problem. Often, that number is chosen empirically by experiments. If an inadequate number of nodes are used, the network will be unable to model complex data, and the resulting fit will be poor. If too many nodes are used, the training time may become excessively long, and, worse, the network may overfit the data. When overfitting occurs, the network fits the training data extremely well, but it generalizes poorly to new, unseen data.

The training algorithm we use for the MLP is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [2, 9]. The BFGS method approximates the Newton's method, a class of hill-climbing optimization techniques. The algorithm stops when the error slope approaches zero or after a maximum of epochs.

For neural network, we adopt the popular multilayer perceptron, as coded in the R nnet package.

### 3 Dataset and Preprocessing

The direct marketing dataset used in this study was provided by Moro et al. [8], also available in [1]. It consists of 45,211 samples, each having 17 attributes, one of which is the class label. The attributes are both categorical and numeric and can be grouped as:

- demographical (age, education, job, marital status);
- bank information (balance, prior defaults, loans);
- direct marketing campaign information (contact type, duration, days since last contact, outcome of the prior campaign for that client, etc.)

A summary the attributes and their description is presented in Table 1.

The dataset is unbalanced, because the successful samples corresponding to the class 'yes' are 5,289, which is 11.7% of all samples. There are no missing values. Further details about data collection, understanding, and initial preprocessing steps can be found in [8].

The majority of our experiments used the dataset for training neural networks. They, however, process numeric data only in a fairly limited range, usually [0,1]. This presents a problem, as the dataset we use contains both numeric values out of the usual range and non-numeric. The data transformations needed sort that out are part of the data preparation stage of the CRISP-DM project model [3]. We did two transformations: mapping non-numeric data into binary dummies and normalization/scaling into the [0,1] interval.

Non-numeric categorical variables cannot be used as they are. They must be decomposed into a series of dummy binary variables. For example, a single variable, such as education having possible values of "unknown", "primary", "non-numeric".
"secondary", and "tertiary" would be decomposed into four separate variables: unknown - 0/1; primary - 0/1; secondary - 0/1; and tertiary - 0/1. This is a full set of dummy variables, which number corresponds to the number of possible values. Note, however, that in this example only three of the dummy variables need - if the values of three are known, the fourth is also known. For example, given that these four values are the only possible ones, we can know that if the education is neither unknown, primary, nor secondary, it must be tertiary. Thus we map a categorical variable into dummies, which are one less than the number of possible values. Using reduced number of dummies we converted the original dataset variables into 42 numeric variables altogether, which is 6 less than the 48 variables used in [5, 6]. There are two benefits of that: first, the neural network architecture becomes simpler and faster; secondly, in some modeling algorithms, such as multiple linear regression or logistic regression, the full set of dummy variables will cause the algorithm to fail due to the redundancy. The model building utility we used converts categorical variables to binary dummies without redundancy.

<table>
<thead>
<tr>
<th>#</th>
<th>Name (type)</th>
<th>Description: values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>age (numeric)</td>
<td>type of job: &quot;blue-collar&quot;, &quot;admin.&quot;, &quot;student&quot;, &quot;unknown&quot;, &quot;unemployed&quot;, &quot;services&quot;, &quot;management&quot;, &quot;retired&quot;, &quot;housemaid&quot;, &quot;entrepreneur&quot;, &quot;self-employed&quot;, &quot;technician&quot;.</td>
</tr>
<tr>
<td>2</td>
<td>job (categorical)</td>
<td>marital status: &quot;married&quot;, &quot;divorced&quot;, &quot;single&quot;</td>
</tr>
<tr>
<td>3</td>
<td>marital (categorical)</td>
<td>&quot;unknown&quot;, &quot;secondary&quot;, &quot;primary&quot;, &quot;tertiary&quot;</td>
</tr>
<tr>
<td>4</td>
<td>education (categorical)</td>
<td>has credit in default? &quot;yes&quot;, &quot;no&quot;</td>
</tr>
<tr>
<td>5</td>
<td>default (binary)</td>
<td>average yearly balance, in euros</td>
</tr>
<tr>
<td>6</td>
<td>balance (numeric)</td>
<td>has housing loan? &quot;yes&quot;, &quot;no&quot;</td>
</tr>
<tr>
<td>7</td>
<td>housing (binary)</td>
<td>has personal loan? &quot;yes&quot;, &quot;no&quot;</td>
</tr>
<tr>
<td>8</td>
<td>loan (binary)</td>
<td>contact communication type: &quot;unknown&quot;, &quot;telephone&quot;, &quot;cellular&quot;</td>
</tr>
<tr>
<td>9</td>
<td>contact (categorical)</td>
<td>last contact day of the month</td>
</tr>
<tr>
<td>10</td>
<td>day (numeric)</td>
<td>last contact month of the year: &quot;Jan&quot;, &quot;Feb&quot;, &quot;Mar&quot;, &quot;Dec&quot;</td>
</tr>
<tr>
<td>11</td>
<td>month (categorical)</td>
<td>last contact duration, in seconds</td>
</tr>
<tr>
<td>12</td>
<td>duration (numeric)</td>
<td>number of contacts performed during this campaign and for this client</td>
</tr>
<tr>
<td>13</td>
<td>campaign (numeric)</td>
<td>number of contacts performed before this campaign and for this client</td>
</tr>
<tr>
<td>14</td>
<td>previous (numeric)</td>
<td>outcome of the previous marketing campaign: &quot;unknown&quot;, &quot;other&quot;, &quot;failure&quot;, &quot;success&quot;</td>
</tr>
<tr>
<td>15</td>
<td>outcome (categorical)</td>
<td>output variable (desired target): &quot;yes&quot;, &quot;no&quot;</td>
</tr>
</tbody>
</table>

The second data transformation we did is related to normalization/scaling. This procedure attempts to give all data attributes equal weight, regardless of the different nature of data and/or different measurement units, e.g. day (1-31) vs. duration in seconds (0-4918). If the data are left as they are, the training process gets influenced and biased by some ‘dominating’ variables with large values. In order to address this, we did normalization (z-scoring) according to the formula:

\[
x_{i}^{new} = \frac{x_{i} - \mu}{\sigma},
\]

where \( \mu \) is the mean and \( \sigma \) is the standard deviation of the variable in question. After the transformation, each variable has zero mean and unit standard deviation. In order to scale down the values mapping them into [0,1], we used the linear transformation

\[
x_{i}^{new} = \frac{x_{i} - a}{b - a},
\]

where \([a,b]\) is the range of values for that variable.

The two transformations were applied to each of the variables independently and separately.

### 4 Experiments and Discussion

In order to build models for direct marketing application and compare their characteristics with those discussed in other studies [8, 5, 6], we used the same dataset as before and did experiments consistently. We, however, extended the methodology addressing the following issues:

- **Optimization of neural network architecture.** Given, the direct marketing task is a classic binary classification problem, which cannot be described as consisting of two or more clearly separable feature extraction stages, we could expect that the two-hidden layers architecture proposed in [5, 6] could be simplified to one hidden layer. We also reduce the input layer size by 6 nodes using alternative mapping of categorical variables into binaries, as discussed above. Simpler architectures are always preferable as they can be built and trained easily and run faster. One-hidden layer MLPs have proven both theoretically and empirically to be universal approximators for the majority of classification tasks.

- **Validation and testing.** With reference to the data preparation and evaluation stages of CRISP-DM, we extend the testing methodology by partitioning the data into training, validation and test sets. Using validation and test sets in a double-testing procedure helps to avoid overestimation of the model performance. If validation set is used only, it is involved in the model building process for benchmarking during the hyper-parameter optimization. Thus, the validation data become compromised for the purposes of model evaluation, because they have been ‘seen’ before and the model hyper-parameters have adapted to the validation data. Using validation data for testing usually leads to overestimation of the model performance. Further to [8, 5, 6], which use one-test
procedure with validation set (despite called test in [8]), we adopt double-testing.

- **Randomness and 'lucky' set composition.** With reference to the CRISP-DM data preparation stage, sampling procedure plays important role in model building. The most common partitioning procedure for training, validation, and test sets uses random sampling. Although, this is a fair way to select a sample giving each member of the original dataset equal chance of being selected, this is not an exact replica, in miniature, of the original dataset, and, as a consequence, does not preserve exactly the statistical parameters of the original dataset. In cases where data saturation is not sufficient for a good training, some 'lucky' draws train the model much better than others. Thus, the model instances show variance in behavior and characteristics influenced by the randomness. In order to address this issue and further to [8, 5, 6], we used a methodology, which combines cross-validation (CV), multiple runs over random selection of the folds and initial weights, and multiple runs over random selection of partitions.

All experiments were conducted using R environment [4, 10, 12]. In machine learning applications, classification performance is often measured using accuracy as the figure of merit. For a given operating point of a classifier, the accuracy is the total number of correctly classified instances divided by the total number of all available instances. Accuracy, however, varies dramatically depending on class prevalence. It can be a misleading estimator in cases where the most important class is typically underrepresented, such as the class of 'yes' of those who respond positively to the direct marketing. For these applications, sensitivity and specificity can be more relevant performance estimators.

In order to address those accuracy deficiencies, we did Receiver Operating Characteristics (ROC) analysis [7].

In a ROC curve, the true positive rate (TPR), a.k.a. sensitivity, is plotted as a function of the false positive rate (FPR), a.k.a. 1-specificity, for different cut-off points. Each point on the ROC plot represents a sensitivity/specificity pair corresponding to a particular decision threshold. A test with perfect discrimination between the two classes has a ROC plot that passes through the upper left corner (100% sensitivity, 100% specificity). Therefore the closer the ROC plot is to the upper left corner, the higher the overall accuracy of the test. The area under the ROC curve (AUC) is a common measure for the evaluation of discriminative power. AUC represents classifier performance over all possible threshold values, i.e. it is threshold independent.

For the sake of consistency with the previous studies, we used 98 % of the dataset for training and validation, split randomly in ratio 2/3: 1/3. The rest of 2% were retained for test. Search of optimal NN architecture was made exploring models with one hidden layer of size from 0 to 13. Experiments showed that sizes above 13 are far from optimal and not considered. In order to validate the results and reduce the effect of lucky set composition, each architecture was tested 300 times: internally, the fit algorithm runs 10 times with different random selection of training and validation sets and initial weights. For each of those set compositions, the 3-fold cross-validation creates 3 model instances and average results. We iterated all that procedure 10 times per architecture recording and averaging accuracy and AUC. Table 2 and Figure 2 show the results.

Table 2 Validated performance metrics of neural networks with $H$ hidden nodes and architecture 42-H-1, where accuracy (ACC) and area under ROC curve (AUC) values are average of 300 model instances of that architecture. $ACC_{\text{max}}$ and $AUC_{\text{max}}$ are maximal values obtained.

<table>
<thead>
<tr>
<th>$H$</th>
<th>ACC</th>
<th>$ACC_{\text{max}}$</th>
<th>AUC</th>
<th>$AUC_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>89.514</td>
<td>92.040</td>
<td>0.895</td>
<td>0.937</td>
</tr>
<tr>
<td>1</td>
<td>89.011</td>
<td>92.150</td>
<td>0.898</td>
<td>0.910</td>
</tr>
<tr>
<td>2</td>
<td>89.849</td>
<td>93.143</td>
<td>0.903</td>
<td>0.924</td>
</tr>
<tr>
<td>3</td>
<td><strong>90.489</strong></td>
<td><strong>93.143</strong></td>
<td>0.906</td>
<td><strong>0.939</strong></td>
</tr>
<tr>
<td>4</td>
<td>90.289</td>
<td>91.107</td>
<td>0.908</td>
<td>0.913</td>
</tr>
<tr>
<td>5</td>
<td>90.250</td>
<td>90.606</td>
<td>0.910</td>
<td>0.921</td>
</tr>
<tr>
<td>6</td>
<td>90.090</td>
<td>90.701</td>
<td>0.912</td>
<td>0.919</td>
</tr>
<tr>
<td>7</td>
<td>90.285</td>
<td>90.606</td>
<td>0.913</td>
<td>0.919</td>
</tr>
<tr>
<td>8</td>
<td>90.025</td>
<td>90.700</td>
<td><strong>0.915</strong></td>
<td>0.923</td>
</tr>
<tr>
<td>9</td>
<td>90.049</td>
<td>90.505</td>
<td>0.915</td>
<td>0.922</td>
</tr>
<tr>
<td>10</td>
<td>90.050</td>
<td>90.505</td>
<td>0.915</td>
<td>0.920</td>
</tr>
<tr>
<td>11</td>
<td>90.091</td>
<td>90.800</td>
<td>0.913</td>
<td>0.918</td>
</tr>
<tr>
<td>12</td>
<td>89.528</td>
<td>90.300</td>
<td>0.913</td>
<td>0.923</td>
</tr>
<tr>
<td>13</td>
<td>90.120</td>
<td>90.403</td>
<td>0.914</td>
<td>0.918</td>
</tr>
</tbody>
</table>

Models with 42-3-1 architecture show best average accuracy of 90.489%. They outperform the 48-20-15-1 architecture from [5]. There were also certain model
instances, which have higher accuracy (\(\text{ACC}_{\text{max}}\) in the Table 2). Models with 42-8-1 architecture have best average AUC of 0.915. Certain model instances achieved AUC=0.939.

The variance and instability of results can be explained by insufficient saturation of data for training. The model can't be trained well to discriminate between classes, particularly to recognize the under-presented 'yes' class. Nevertheless, experiments show that given the data saturation, a 42-3-1 neural net can be trained to reach accuracy 93.143%, which significantly outperforms the 48-20-15-1 one proposed in [5] and [6].

Figure 3 shows ten colour curves, each of which is a plot of a 42-8-1 model trained and validated by the 98% dataset. The colors represent different cutoff points with color bar shown on the right side of the box. The black curve is average of the 10 curves. The variance of TPR is depicted by the standard deviation bars.

![ROC of NN trained by 98% of Data](image)

Figure 3. ROC curves of 10 neural network models with 42-8-1 architecture. Colors show values of the cutoff points applied. Black line represents average values of the 10 models. Standard deviation bars measure variance.

Lift is probably the most commonly used metric to measure performance of targeting models in marketing applications. A model is doing well if the response within the target is much better than average for the population as a whole. In a cumulative lift chart (gains chart), the y-axis shows the percentage of true positive responses (tpr). Formally,

\[
tpr = \text{sensitivity} = \frac{TP}{TP + FN}. \quad (6)
\]

The x-axis shows the rate of positive predictions (rpp). This is percentage of customers contacted by direct mailing, which is a fraction of total customers.

\[
rpp = \frac{TP + FP}{TP + FP + TN + FN} \quad (7)
\]

Figure 4 shows the cumulative lift charts of 42-8-1 neural networks. Ten colored curves and standard deviation bars illustrate variance caused by the randomness in partitioning. The black line is average of the ten.

![Cumulative LIFT of NN](image)

Figure 4. Cumulative LIFT curve of 10 neural network models with 42-8-1 architecture. Colors show values of the cutoff points applied. Black line represents average values of the 10 models. Standard deviation bars measure variance.

A good way to characterize performance of a classifier is to look at how precision and recall change as threshold changes. This can be visualized by precision-recall curve (Figure 5). The better the classifier, the closer its curve is to the top-right corner of the graph. Formally,

\[
\text{precision} = \frac{TP}{TP + FP} \quad (8)
\]

\[
\text{recall} = \frac{TP}{TP + FN}. \quad (9)
\]

In terms of the direct marketing task, precision is the percent of correctly identified 'yes' customers (who purchase the product) among all reported as 'yes'; recall is the percent of correctly identified 'yes' customers among those who are
'yes' in the test set. Recall and precision are inversely related: as recall increases, precision decreases and visa versa. Figure 5 shows 10 colored curves, each representing a randomly built NN model. The black line averages the 10 models. Standard deviation bar visualize variance of the models precision as result of randomness. Colors of the curves and the color bar show the threshold values relevant to the cutoff points.

![Figure 5](image.png)

Figure 5. Precision - Recall of NN

In order to compare neural networks (NN) as a modeling technique with other techniques, we built models based on logistic regression (LR), naive Bayes (NB), linear discriminant analysis (LDA), and quadratic discriminant analysis (QDA).

Figure 6 shows ROC curves of the models in one plot. Generally, if two ROC curves do not intersect then one model dominates over the other. When ROC curves cross each other, one model is better for some threshold values, and is worse for others. In that situation the AUC measure can lead to biased results and we are not able to select the best model. Common practice is to compare crossing ROC curves by restricting the performance evaluation to proper subregions of scores. Choice of the method and threshold value is based on the task goals. The figure shows that the curves of NN and LR intersect one another, but in most of the regions NN outperform LR being closer to the top-left corner. This is particularly visible in the most north-west regions, there maximal accuracy is achieved. NN entirely dominate over LDA, NB, and QDA, which performance can be ranked in that order.

![Figure 6](image.png)

Figure 6. ROC curve of five models: Neural Network, Logistic Regression, Naive Bayes, Linear Discriminant Analysis, and Quadratic Discriminant Analysis. Each model runs with its optimal hyper-parameter values and size of the training dataset. Results are validated by 3-fold cross-validation.

5 Conclusions

Predictive models in direct marketing seek to identify customers/individuals most likely to respond to promotional solicitations or other intervention programs.

This paper presents a case study of data mining modeling techniques for direct marketing based on neural networks. We extend the studies of [5, 6, 8], addressing the following issues:

We explore how neural network design affect the model performance in order to find the optimal size of the hidden layer. We propose a simpler architecture with one hidden layer and three nodes. It achieves best accuracy and outperforms the 48-20-15-1 architecture proposed by [5]. When we measure the threshold independent figure of merit AUC, best performer is the 42-8-1 architecture.

We also did ROC analysis comparing neural nets, logistic regression, naive Bayes, linear and quadratic discriminant analysis techniques and found that neural nets prevail over logistic regression in most of the cutoff point intervals. Neural nets also entirely dominate over the rest of the techniques.

Another contribution of this study is validation of experimental results by a methodology that extends those of [5, 6, 8]. We used double-testing with both validation and
test sets and took into account performance variance and the effect of lucky set composition caused by randomness, not reported previously. Each model architecture was tested 300 times involving 3-fold cross-validation, random partitioning and iterations. We also did ROC, cumulative lift, and precision-recall analysis.

6 References


