A Hybrid Network for Input that is both Categorical and Quantitative

Roelof K. Brouwer
Department of Computing Science
University College of the Cariboo
Kamloops, BC, Canada, V2C 5N3
rkbrouwer@ieee.org

Abstract

The data on which a MLP (multi-layer perceptron) is to be trained to approximate a continuous function may include inputs that are categorical in addition to numeric or quantitative inputs. An approach examined in this paper is to train a hybrid network consisting of an MLP and a encoder with multiple output units; a separate output unit for various combinations of values of the categorical variables. Input to the feed forward sub network of the hybrid network is restricted to truly numerical quantities.

Results show that the method discussed here of separating numerical from quantitative is quite effective.

1 Introduction

Often the data on which a multiplayer *perceptron (MLP)* is to be trained to approximate a continuous function has inputs that are categorical rather than numeric or quantitative. A MLP [1,2] with connection matrices that multiply input values and sigmoid functions that further transform values however represents a continuous mapping in all input variables. The values of categorical variables even if they are ordinal should not be passed through the MLP as if they were continuous in nature since Kolmogorov's theorem is only proven for a network that is used to represent a continuous mapping. The underlying function to be represented may really be several distinct functions with the values of the categorical variable labeling these functions. As an alternative then we could consider a separate MLP for various combinations of values for the categorical variables found in the training data and expected during prediction.

Another method that makes use of several networks in combination is the mixture of experts [3]. Other work using neural networks in case where categorical variables are present is by Barton and Burgess [4]. Burgess [5] describes a methodology, based upon the statistical concept of analysis of variance (ANOVA), which can be used both for non-linear model identification and for testing the statistical significance of categorical inputs to a neural network Bishop [6,7] introduces a new class of neural, network models obtained by combining a conventional neural network with a mixture density model. The complete system is called a Mixture Density Network. Lee and Lee [8] also address the problem of multi-value regression estimation with neural network architecture.

This paper is organized as follows. It commences with a description of a neural network construction and its amended training algorithm. This is followed by the results of simulations that demonstrate the validity of the approach suggested in this paper.

2 Alternative Approach

2.1 Introduction

A commonly used approach is to convert the categorical values into real numbers and to feed them together with the other numerical input into a single MLP without distinction from the numerical input. Let us instead consider a more general approach as shown in Figure 1. In this case the categorical input is treated separately from the numerical input.

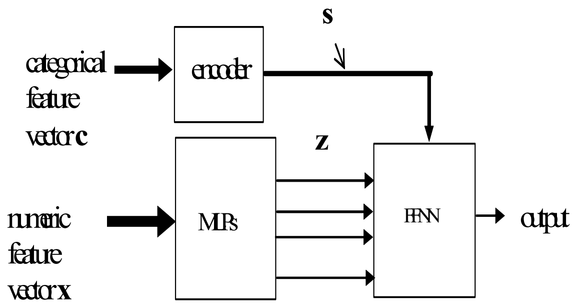

Figure 1 General network with categorical input treated seperately

If we represent the output of the neural network by **z** then the complete network calculates the output as

$$y = f(\mathbf{z}, \varphi(\mathbf{w},\mathbf{s})) \qquad (1)$$

with $\varphi(\mathbf{w},\mathbf{s}) = (\varphi_i(\mathbf{w},\mathbf{s}_i) \quad i = 0,1..h\text{-}1)$ and where $\varphi_i(\mathbf{w},\mathbf{s}_i)$ could be Boolean functions with values from $\{0,1\}$. The parameter vector **w** would be trainable. $\mathbf{s}_i$ $i = 0,1..h\text{-}1$ are elements of $\{0,1\}$ and are the components of output of the encoder network. We may now consider various options since we have choices regarding the φ function performed by the FFNN, the encoder and the MLPs.

2.2 The FNNN

The first simplification is to set $\varphi_i(\mathbf{w},\mathbf{s}_i) = \mathbf{s}_i$ and let f be the dot product. The output of the combined network is the dot product of the output of the MLP, **z**, with the output of the encoder, **s** i.e

$$y=\mathbf{z}.\mathbf{s} \qquad (2)$$

If **s** is a 1 of n vector with the position of the 1 identifying the output unit of the MLP then **s** behaves like a selector by selecting the ouput that applies.

2.3 The MLPs

As far as the MLPs are concerned we could consider a separate MLP for each combination of values for the categorical variables, **c**, found in the training data and expected during prediction. In statistical terms the ANN consists of several MLP's with a dedicated MLP for each regression equation as shown in Figure 2. Each regression equation corresponds to a subset of different combination of values of categorical variables.

$$y=(\mathbf{W}^{(1)}.(f(\mathbf{W}^{(0)}.\mathbf{x})).\mathbf{s} \qquad (3)$$

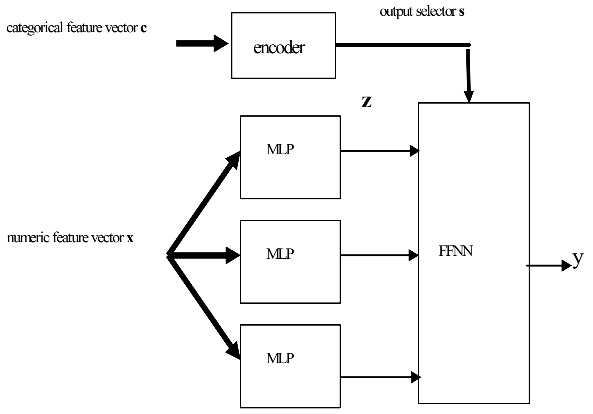
categorical feature vector **c**
encoder
output selector **s**
z
MLP
MLP
MLP
numeric feature vector **x**
FFNN
y

Figure 2 Separate MLP's

The nominal input through selector, **s**, then determines which MPL has the correct result for the continuous input. The encoder will generate a 1 of n binary vector, **s**, that is combined with the output of the MLP in a dot product. The FNN selects the MLP that provides the correct output. For example each of the three MLP's in Figure 2 represents a different regression equation. This means that the categorical feature vectors are used as function identifiers rather than as values of a quantitative variable. In the case of one category variable with 3 values 3 separate networks each with a single output unit would be trained. In case of the Boston housing data discussed in this paper up to 12 networks would be required depending upon the allowable number of combinations of categorical values.

The problem with the preceding method of including categorical input separately is that the number of output units required for the network is equal to the number of categorical feature values or even combinations of feature values (not the number of categorical features). The training data is effectively segmented with a segment for each categorical feature value. This may leave some segments with very few training elements and some with no training elements at all. It is quite possible that some categorical feature value combinations do not occur in the training data set because they are not at all possible in reality.

However a collection of MLPS, each with the same input and each with one output unit is equivalent to a single MLP that is not fully connected between the hidden layer and the output layer. Each output unit will only be connected to the subset of hidden units that correspond to the same component MLP.

If we now allow the single MLP to be fully connected from hidden layer we get a MLP with more expressive power, because of additional degrees of freedom, to represent each of the two regression equations We may reduce the number of hidden units. Based on this the approach examined in this paper is to train a single MLP with multiple outputs; a separate output unit for each rule or condition.

The transfer function corresponding to the complete feed-forward network that accepts both the quantitative and categorical input is then

$$y=(\mathbf{W}^{(1)}.(f(\mathbf{W}^{(0)}.\mathbf{x})).\mathbf{s} \qquad (4)$$

The matrices $\mathbf{W}^{(0)}$ and $\mathbf{W}^{(1)}$ correspond to the connection matrices for the first and second layer respectively in the MLP. For simplicity sake the additional unit in the input layer and hidden layer of the MLP sub-network that has a constant input of –1 to be multiplied by a weight representing bias has been left out.

2.4 The Encoder Network

Next the encoder network operation will be described. The encoder network accepts the categorical input portion of the total input and produces a selector vector, **s**. The dot product of this 1 of n selector vector with the output of the MLP, **z**, produces the final output, y. The encoder network converts the categorical input into a partitioned 1 of n code as illustrated in (5). The coded categorical input is then compared to entries in a table, **S**.

For a particular categorical variable a category is represented by a binary vector using 1-of-n encoding. (n is the number of categories). Each category for a category variable is then represented by a position in a binary vector. These 1 of n arrays are combined to form a 1-dimensional array of 1-dimensional arrays. The benefit of this representation is that conditions for rules are easier to express as subsets of coded inputs. An example is shown in (5).

$$[(0, 1, 0), (1, 0), (1, 0)] \tag{5}$$

A condition for selection an output unit would then be expressed as

$$[(1, 1, 0), (0, 1), (1, 0)] \tag{6}$$

There may be several conditions or rules like this with each condition corresponding to a regression equation. Each rule consists of a condition and the identifier of a regression equation. An example of a set of rules, **S**, is in Table 1.

Table 1 An example of S

Condition	1 of n representation of output unit identifier and selector **s**
[(1,0,0), (1, 1), (0, 1)]	0 0 1
[(0,1,0), (0, 1), (1, 0)]	0 0 1
[(0,0,1), (1, 0), (0, 1)]	0 0 1
[(1,0,1), (0, 1), (1, 0)]	0 1 0
[(0,1,0), (1, 0), (0, 1)]	0 1 0
[(0,0,1), (1, 0), (1, 0)]	0 1 0
[(1,1,0), (1, 0), (1, 0)]	1 0 0
[(0,1,1), (0, 1), (0, 1)]	1 0 0

2.5 Training

Training is done by using gradient descent as is normally done on a feed forward network with some additional operations that will be described next.

There are several output units for the MLP while in the training data there is only one scalar output value for each input. To obtain an error value for training, the output unit of the MLP that produces the value closest to the target output for the given training pattern is used to update the connection matrices, $\mathbf{W}^{(0)}$ and $\mathbf{W}^{(1)}$, of the MLP. It is important to note that the connection matrix, $\mathbf{W}^{(1)}$, between the hidden layer and the output layer has only the row corresponding to the winning output unit modified. The categorical input concatenated with the identifier of the output unit producing the smallest error, through selector **s**, is then either added to a table, **S**, or replaces an entry in the table or does nothing to the table. Note that this in general will be a many to one mapping. As a final step in training, table, **S**, is reduced by combining rows that contain the same value for **s** whenever possible. The entries in the first column will then be conditions.

Formally let t be the target output of the entire network, y the output of the entire network and **z** the output of the MLP. Then

$$e=y-t \tag{7}$$

and

$$\mathbf{e} = (\mathbf{z} - t) \tag{8}$$

Note that bolded **e** is a vector and un-bolded e is a scalar.

$$e= y-t = \mathbf{s}.\mathbf{z}-t = \mathbf{s}.(\mathbf{z}-t) = \mathbf{s}.\mathbf{e} \tag{9}$$

A performance measure we may use during training is as in eqn. (10)

$$C = \frac{(y-t)^2}{2} = \frac{e^2}{2} \tag{10}$$

Then

$$\nabla_{\mathbf{U}} C = e \nabla_{\mathbf{U}} y \tag{11}$$

U represents any parameter of the network to be learned. Now

$$\nabla_{\mathbf{U}} y = \mathbf{s}.\nabla_{\mathbf{U}} \mathbf{z} + \mathbf{z}.\nabla_{\mathbf{U}} \mathbf{s} \tag{12}$$

U is replaced by the connection matrices $\mathbf{W}^{(0)}$ and $\mathbf{W}^{(1)}$. **s** is a function of the categorical input, **q**, and does not depend on the connection matrices in the trained network even if it does during training as we will see later. Therefore the second term on the right side in (12) can be dropped. Then

$$\nabla_{\mathbf{W}^{(0)}}\mathbf{z} = (\mathbf{W}^{(1)} \quad \mathbf{W}^{(0)}.\mathbf{x} \times \mathbf{I} \otimes \mathbf{x} \tag{13}$$

$$\nabla_{\mathbf{W}^{(1)}}\mathbf{z} = \quad \otimes \quad ^{(0)} \tag{14}$$

We get

$$\mathbf{s}.\nabla_{\mathbf{W}^{(0)}}\mathbf{z} = (\mathbf{s}.\mathbf{W}^{(1)} \times \quad ^{(0)} \quad \otimes \mathbf{x} \tag{15}$$

$$\mathbf{s}.\nabla_{\mathbf{W}^{(1)}}\mathbf{z} = \mathbf{s} \quad \otimes \quad ^{(0)} \tag{16}$$

$$= \mathbf{s} \otimes \quad ^{(0)} \tag{17}$$

f(x) is equal to tanh(x). Note that component-wise multiplication of arrays is defined as long as the dimensions of one of the arrays form a prefix of the dimensions of the other array. Thus the component wise multiplication between a 3*4 array and a 3*4*6 array is defined. **I** is the identity matrix with the same rank as **s**.

The complete training algorithm consists of applying the following:

$$\mathbf{z}=\mathbf{W}^{(1)}.\mathrm{f}(\mathbf{W}^{(0)}.\mathbf{x}) \tag{18}$$

$$\mathbf{s}_{\mathrm{pmin}(\mathbf{z}\text{-t})}=1 \quad \text{and} \quad 0 \ \text{otherwise} \tag{19}$$

pm in (**z**-t) means position of maximum in **z**-t.

$$\Delta\mathbf{W}^{(0)} = -\mu_0 \times \mathrm{e} \times (\mathbf{s}.\mathbf{W}^{(1)} \times \quad ^{(0)} \quad \otimes \mathbf{x} \tag{20}$$

$$\Delta\mathbf{W}^{(1)} = -\mu_1 \times \mathrm{e} \times \mathbf{s} \otimes \quad ^{(0)} \tag{21}$$

In addition to the above the table **S** is updated at each step.

3 Simulations

Following are the results of carrying out simulations to permit comparisons of the approaches discussed previously.

3.1 Simulation 1 -Boston Housing Data

The data for this simulation is from the StatLib library that is maintained at Carnegie Mellon University. The creator is Harrison, D. and Rubinfeld, D.L. 'Hedonic prices and the demand for clean air', J. Environ. Economics & Management, vol.5, 81-102, 1978. Date: July 7, 1993. The data set is available at http://www.ics.uci.edu/~mlearn/ MLRepository.html. It is concerned with housing values in suburbs of Boston. The feature vectors consist of 12 continuous attributes, one non-ordered binary-valued attribute and one ordered categorical attribute. Thus we have two kinds of categorical features. The value to be predicted is the median value of owner-occupied homes in $1000's.

The first categorical variable has the values 0 and 1 while the second has the values 1, 2, 3, 4, 5, 6, 7, 8, and 24. Using 1 of n encoding the first variable requires 2 bits and the second 9 bits for a total of 11 bits. The total number of combinations for categorical input is 2*9=18 although only 15 show up in the simulation. The number of hidden units per output unit is set at 5. The mean square error on training data before training was 1051.2. While the mean square error on test data before training was 1034.8.The learning rate was 0.0001 for both connection matrices. The number of output units made available is 15 while the number actually used is 11. Figure 3 shows the progress in training. The error on the test data is very close to the error on the training error at each epoch.

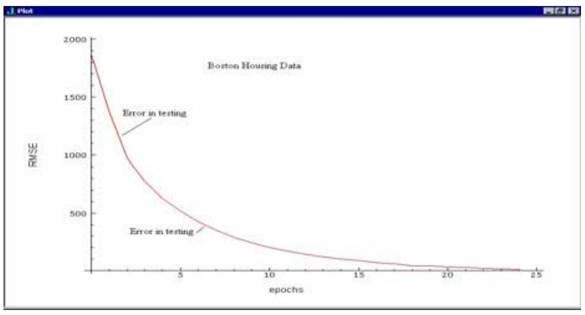

Figure 3

The table, **S**, obtained by training and sorted by the last column is

Table 2 Categorical input versus selector

R	Categorical input	**s**
1	[(0 1), (1 0 0 0 0 0 0 0 0)]	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
2	[(1 0), (0 0 1 0 0 0 0 0 0)]	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
3	[(1 0), (0 0 0 0 0 0 1 0 0)]	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
4	[(0 1), (0 0 0 0 1 0 0 0 0)]	0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
5	[(0 1), (0 0 0 0 0 1 0 0 0)]	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
6	[(1 0), (0 1 0 0 0 0 0 0 0)]	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
7	[(1 0), (0 0 0 0 0 0 0 0 1)]	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
8	[(1 0), (0 0 0 1 0 0 0 0 0)]	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
9	[(0 1), (0 0 0 0 0 0 0 0 1)]	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0	[(0 1), (0 0 1 0 0 0 0 0 0)]	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
1	[(1 0), (0 0 0 0 0 0 0 1 0)]	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
2	[(0 1), (0 0 0 1 0 0 0 0 0)]	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
3	[(1 0), (1 0 0 0 0 0 0 0 0)]	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
4	[(0 0), (0 0 0 0 0 1 0 0 0)]	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5	[(0 0), (0 0 0 0 1 0 0 0 0)]	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

The first column is there to identify the rows for the purpose of the reader. This table is reduced to

Table 3 Condition versus selector value

Condition	Selector
[(1 0), (0 0 0 0 1 1 0 0 0)]	1 0 0 0 0 0 0 0 0 0 0
[(1 1), (1 0 0 1 0 0 0 0 0)]	0 1 0 0 0 0 0 0 0 0 0
[(1 0), (0 0 0 0 0 0 0 1 0)]	0 0 1 0 0 0 0 0 0 0 0
[(0 1), (0 0 1 0 0 0 0 0 1)]	0 0 0 1 0 0 0 0 0 0 0
[(1 0), (0 0 0 1 0 0 0 0 0)]	0 0 0 0 1 0 0 0 0 0 0
[(1 0), (0 0 0 0 0 0 0 0 1)]	0 0 0 0 0 1 0 0 0 0 0
[(1 0), (0 1 0 0 0 0 0 0 0)]	0 0 0 0 0 0 1 0 0 0 0
[(0 1), (0 0 0 0 0 1 0 0 0)]	0 0 0 0 0 0 0 1 0 0 0
[(0 1), (0 0 0 0 1 0 0 0 0)]	0 0 0 0 0 0 0 0 1 0 0
[(1 0), (0 0 0 0 0 0 1 0 0)]	0 0 0 0 0 0 0 0 0 1 0
[(1 1), (1 0 1 0 0 0 0 0 0)]	0 0 0 0 0 0 0 0 0 0 1

3.2 *Simulation 2-Abalone Data*

The source of the data for this experiment is the Marine Resources Division, Marine Research Laboratories – Taroona, Department of Primary Industry and Fisheries, Tasmania GPO Box 619F, Hobart, Tasmania 7001, Australia. The data set is available at http://www.ics.uci.edu/~mlearn/MLRepository.html. The data is for predicting the age of abalone from physical measurements. The number of attributes is 8 with all but one being continuously valued. The output data is normalized before training. Of 4177 feature vectors 1/2 were used for training and the other 1/2 was used for testing.).
The simulation was done with 3 output units and 3 hidden units per output unit. The mean square error based on training data before training commenced was 2.328320995 and the mean square error on test data before training commenced was 2.502301048. The number of epochs allowed was 25. he learning rates for the two connection matrices was 0.0001. The actual number of output units used was 3. The training error and testing error at the end of 25 epochs was 0.232212 and 0.247859.

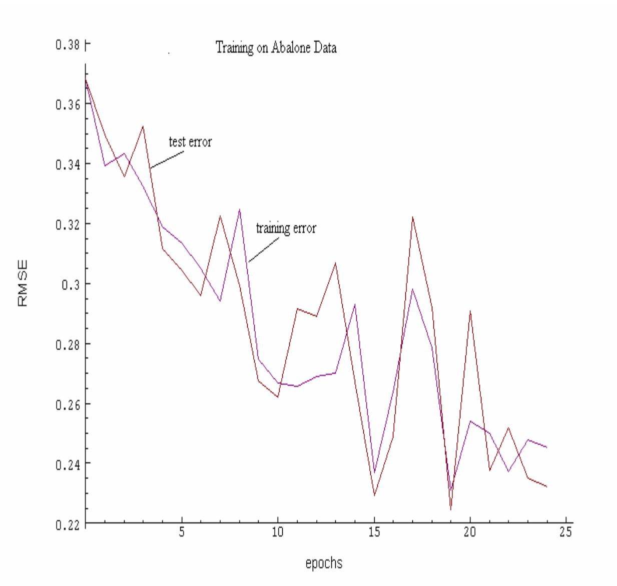

Figure 4 Training on Abalone Data

The table identifying the output unit corresponding to a category value is as shown in Table 4. Three output units were made available and 3 were utilized with one unit for each category variable

Table 4 Selector of output unit

Category	Category in 1 of n	Output unit	Output unit in 1 of n
M	0 0 1	1	0 1 0
F	0 1 0	0	1 0 0
I	1 0 0	2	0 0 1

4 Conclusion

We have shown how the categorical features may be segregated from the numeric features when using an MLP for prediction. The numeric feature vector is processed by an MLP with several outputs that are then combined with the coded form of the categorical feature vector. This is somewhat similar to the statistical approach except that non-linearities are used instead of linearities [9]. The only correct way is to have a separate function and separate MLP for each categorical feature vector but this is not feasible unless there is sufficient data for feature vector instance. Including several outputs in the MLP is an attempt to approximate the method of having a separate function approximator MLP for each categorical feature vector instance. Since the connection matrix from the input layer to the hidden layer is fixed this approximation is limited. Further work includes finding a Boolean function to replace the table, **S**.

5 References

[1] P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Doctoral Dissertation, Appl. Math., Harvard University, Mass. 1974.

[2] D. E. Rumelhart, G. E. Hinton, and J. L. McClelland. A general framework for parallel distributed processing. In *Parallel Distributed Processing*, pages 45-76, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Editors, The MIT Press, Cambridge, MA, 1986.

[3] R. A. Jacobs, M.L. Jordan S. J. Nowlan and G. E. Hinton Adaptive Mixtures of Local Experts *Neural Computation* 3/1 (1992) 79-87.

[4] J. G. Barton and A. Lees. "Comparison of shoe insole materials by neural network analysi"s. *Medical-and-Biological-Engineering-and-Computing.* 34 (6) Nov 1996, p 453-459.

[5] A. N. Burgess, "Non-linear model identification and statistical significance tests and their application to financial modeling" IEE, Stevenage, Engl. p 312-317 Proceedings of the 4th International Conference on Artificial Neural Networks 1995. Cambridge, UK

[6] C. M. Bishop "Mixture Density Networks" NCRG/94/004 Available from http://www.ncrg.aston.ac.uk

[7] C. M. Bishop *Neural Networks for Pattern Recognition* Clarendon Press Oxford 1995.

[8] K. Lee and T. Lee. "Design of Neural Networks for Multi_value Regession. *International Joint Conference on Neural Networks* (2001) 93-98.

[9] J. Neter, William Wasserman, W. and Michael H. Kutner, M. H. *Applied Linear Statistical Models* Irwin 1990 pp 349- 385.