
PERGAMON

Neural Networks 15 (2002) 881–890

Neural Networks

www.elsevier.com/locate/neunet

A feed-forward 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

Received 24 January 2001; accepted 29 April 2002

Abstract

The data on which a multi-layer perceptron (MLP) is to be trained to approximate a continuous function may have inputs that are categorical rather than numeric or quantitative such as color, gender, race, etc. A categorical variable causes a discontinuous relationship between an input variable and the output. A MLP, with connection matrices that multiply input values and sigmoid functions that further transform values, represents a continuous mapping in all input variables. A MLP therefore requires that all inputs correspond to numeric, continuously valued variables and represents a continuous function in all input variables. The way that this problem is usually dealt with is to replace the categorical values by numeric ones and treat them as if they were continuously valued. However, there is no meaningful correspondence between the continuous quantities generated this way and the original categorical values. Another approach is to encode the categorical portion of the input using 1-out-of-*n* encoding and include this code as input to the MLP.

The approach in this paper is to segregate categorical variables from the continuous independent variables completely. The MLP is trained with multiple outputs; a separate output unit for each of the allowed combination of values of the categorical independent variables. During training the categorical value or combination of categorical values determines which of the output units should have the target value on it, with the remaining outputs being 'do not care'. Three data sets were used for comparison of methods. Results show that this approach is much more effective than the conventional approach of assigning continuous variables to the categorical features. In case of the data set where there were several categorical variables the method proposed here is also more effective than the 1-out-of-*n* input method.

Keywords: Feed forward neural networks; Multi-layer perceptron; Nominal input; Indicator variables; Categorical variable regression; Anova

1. Introduction

Often the data on which a multi-layer perceptron (MLP) is to be trained to approximate a continuous function has inputs that are categorical rather than numeric or quantitative. Examples of categorical variables are gender, race, region, type of industry, different types of species of fish, etc. A categorical variable causes a discontinuous relationship between an input variable and the output. A quantitative variable on the other hand is one that assumes numerical values corresponding to the points on a real line, e.g. the kilowatt-hours of electricity used per day or the number of defects in a product. An example of a data set containing a combination of quantitative and categorical data is the 'Boston housing data' data set (Harrison & Rubinfeld, 1978) which is used for training here. In this case there are two categorical independent variables: one is the Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) and the other is the index of accessibility to radial highways. Another example is the data used for predicting the age of abalone. In this case one of the independent variables is categorical with values of M for male, F for female and I for infantile.

Categorical variables may be broken up into two types: ordinal and non-ordinal. Ordinal variable values may be ordered. A non-ordinal categorical variable will be a label of a category without ordering property as is true in the case of the abalone data where no ordering can be placed on the values M, F and I. The Charles River dummy variable above also has this property.

A MLP (Rumelhart, Hinton, & Williams, 1986; Wasserman, 1993; Werbos, 1974) with connection matrices that multiply input values and sigmoid functions that further transform values, however, represents a continuous mapping in all input variables. A MLP requires that all inputs correspond to numeric, continuously valued variables and represents a continuous function in all input variables. One way of dealing with the problem of categorical variables, which seems intuitively unacceptable, is to replace the

* Tel.: +1-250-828-5219.
E-mail address: rkbrouwer@ieee.org (R.K. Brouwer).

0893-6080/02/$ - see front matter
PII: S0893-6080(02)00090-4

categorical values by numeric ones and treat them as if they were continuously valued. Thus if the set of values for a categorical variable was the set {red, blue, green} the values red = 1, green = 2 and blue = 3 would have been used. However, there is no meaningful correspondence between the continuous quantities generated this way and the original categorical values. This type of encoding imposes an ordering on the values that does not exist. To ensure the generalization capability of a neural network, the data should be encoded in a form that allows for interpolation. Therefore categorical variables should be distinguished from the continuous independent variables. One way of dealing with this is to use 1-out-of-*n* coding on inputy, e.g. for the three colours we get, red = (100), green = (010) and blue = (001). This means one input unit for each categorical value. In this paper, however, the encoded values are not input to the MLP but are used to select 1-out-*n* output units. This is somewhat similar but not quite the same as the more drastic solution of having a separate network for each combination of categorical values.

Bishop (1994, 1995) 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. This network can in theory represent arbitrary conditional probability distributions in the same way that a conventional neural network can represent arbitrary functions. The neural network component of the MDN has three groups of output units for each of the Gaussian kernel functions. These units output a mean, variance and multiplier for each conditional probability density in the mixture of densities. The multiplier for including each density in the mixture is the probability that a particular density out of the mixture is applicable. Each combination of categorical values could have its own density.

Lee and Lee (2001) also address the problem of multi-value regression estimation with neural network architecture. They confine the multi-regression problems to those mapping vectors to a scalar with a single input vector mapping to several scalars. They propose a modular network approach such that each module handles only a single output rather than a set of outputs. A decision then has to be made as to which modules, since the number of outputs for a single input varies, produce the correct output values for the given input.

The remainder of the paper is organized as follows. It commences with a discussion of some neural network approaches. Next follows 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. Finally we have the summary including a comparison of the neural network approach to the statistical approach based on using indicator variables, a generalization of the approach in this paper and a description of future work.

2. Neural network approaches

2.1. Introduction

The values of categorical variables even if they are ordinal should not be passed through the MLP as if they were continuous in nature. Kolmogorov's theorem is only proven for a network that is used to represent a continuous mapping. The underlying mapping 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 each combination of values for the categorical variables, **c**, found in the training data and also expected during prediction. A method that makes use of several networks in combination is the mixture of experts (Jacobs, Jordan, Nowlan, & Hinton, 1992).

This means that the categorical feature vectors are used as function identifiers rather than as values of a quantitative variable. A combination of values of the categorical variables then serves as a label of a function in the remaining numeric input variables. In the case of abalone data[1] three separate networks, each with a single output, would be trained. In case of the Boston housing data (Harrison & Rubinfeld, 1978) up to 12 networks would be required depending upon the allowable number of combinations of categorical values. This is an extreme approach and may mean that some of the functions as a consequence have little or no data for training. Other approaches are based on the other extreme where we use only one MLP to represent the continuous mapping and the non-continuous mapping. In between the two extremes one has approaches in which there is more than one MLP but not a separate one for each combination of values of categorical variables.

2.2. Approaches

An approach examined in this paper, which we will refer to as the separation approach, is to train a single MLP with one hidden layer and with additional output units; a separate output unit for each possible instance of the categorical feature vector. Input to the MLP is restricted to the numerical components of the feature vectors. The values for the categorical components bypass the MLP but are used to select one of the appended set of output units to provide the value of the dependent variable. During training the instance of the categorical feature vector portion of the input vector determines which of the output units should have the target value on it with the remaining outputs being treated as irrelevant. The error for gradient descent learning is the square of the difference between the target value on the

[1] 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.

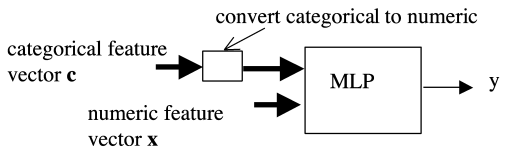

Fig. 1. No special treatment for categorical input.

output unit corresponding to the categorical feature vector instance, **c**, and the actual output for that unit. Figs. 1 and 2 serve to contrast the two methods. The method depicted in Fig. 1 is the usual method of converting the categorical values to numbers and then treating them as if they were continuous. Fig. 2 depicts the separation method. The approach of having a separate network for each categorical value is obvious and is not depicted below.

In general, in our approach the encoder will generate a binary vector **s** that is combined with the output of the MLP in a dot product. For example let us assume that we have two categorical features the first with values *M*, *F*, and *I* and the second with values 'soft', and 'hard'. The six values of **c**, assuming that all combinations are feasible, are (*M*,soft), (*M*,hard),…,(*T*,hard), etc. The corresponding set of six binary vectors to represent the categorical data could be set to (1,0,0,0,0,0), (0,1,0,0,0,0),…,(0,0,0,0,0,1). In this case there is a 1 in only one position and the binary vector, **s**, acts like a selector that selects only one of the outputs from the MLP. This is 1-out-*n* encoding. However, the code vector is not input to the MLP.

For comparison of the number of parameters in the cases above, weights in this case, let n be the number of numeric inputs, m the number of categorical inputs, p the number of combinations of categorical values and h be the number of elements in the hidden layer. The number of parameters to be determined in the two approaches is $(n+m+1)h+(h+1)$ for the single-output case versus $(n+1)h+(h+1)p$ for the multi-output case. The 1 in the preceding expressions is due to a bias on each unit that is represented by a weight. In the case of the abalone data the differences are $10\times5+6$ versus $9\times5+6\times3$ or 56 versus 63. By comparison the number of parameters required if a separate network were to be built for each categorical value is $(9\times5+6)3$ or 153.

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. $\mathbf{y}=\mathbf{z}\cdot\mathbf{s}$. The transfer function corresponding to the complete feed-forward network that accepts both the quantitative and categorical input is then

$$\mathbf{y}=(\mathbf{W}^{(2)}(\mathbf{f}(\mathbf{W}^{(1)}\mathbf{x}))\mathbf{s} \tag{1}$$

The matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ correspond to the connection matrices for the first and second layer, respectively, in the MLP. The implied products here are dot products. In the method emphasized in this paper **s** is a 1-out-of-*n* binary vector. 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 has been left out.

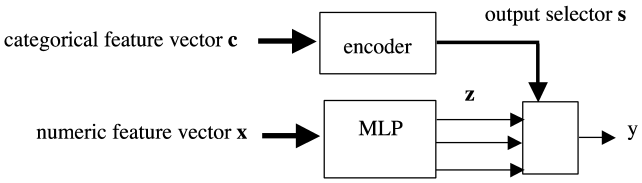

Fig. 2. Categorical input and numeric input are segregated (separation approach).

3. Training algorithm used for the hybrid network

The hybrid network to be trained consists of a one hidden layer MLP which is modified to allow the encoded categorical value to modify the output of the MLP. The learnable parameters are the connection matrices from the input layer to the hidden layer and the hidden layer to the output of the MLP. In the detailed algorithm below, based on the training algorithm described by Brouwer (1997), $f(x)=1/(1+e^x)$ with $f'(x)=e^x/(1+e^x)^2$. **I** is the identity matrix of the required dimension. ' $*$ /' is the vector cross product. Note that component-wise multiplication of arrays is defined here as long as the dimensions of one of the arrays form a suffix of the dimensions of the other array. Thus the component-wise multiplication between a 3×4 matrix and a $3\times4\times6$ matrix is defined. '·' is the dot product. $\mathbf{out}^{(0)}$ is the input excluding the categorical input. $\mathbf{W}^{(1)}$ is the connection matrix between the first layer and the hidden layer, while $\mathbf{W}^{(2)}$ is the connection matrix between the hidden layer and the output layer of the MLP for the numeric input. In the following algorithm C is the cost function or error function, μ the learning rate, and **d** is the target output.

1. initialize
$\mathbf{W}^{(k)}$ $k=1,2$ { values are randomly selected from $[-1,1]$}
2. modify

$$\Delta\mathbf{W}^{(k)}=-\mu_1\nabla_{\mathbf{W}}^{(k)}C \qquad k=1,2$$

With a cost function of

$$C=.0.5\times(\mathbf{s}\cdot\mathbf{out}^{(2)}-\mathbf{d})^2$$

{**s** is the categorical feature vector in binary coded form }

for one input vector the gradients are

$$\nabla_{\mathbf{W}}^{(k)}C=(\mathbf{s}\cdot\mathbf{out}^{(2)}-\mathbf{d})*\mathbf{s}\cdot\nabla_{\mathbf{W}}^{(k)}\mathbf{out}^{(2)} \qquad k=1,2$$

The only part different here from the method described in Brouwer (1997) is the inclusion of the vector **s** in the expressions for the gradients.

Algorithm for calculating the gradient of the final output, assuming two processing layers. Gradients are passed forward.

Table 1
Encoding method 1

$\mathbf{c}_0$	$\mathbf{c}_1$	$\mathbf{s}_0$	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_3$	$\mathbf{s}_4$	$\mathbf{s}_5$
d	g	1	0	0	0	0	0
d	h	0	1	0	0	0	0
e	g	0	0	1	0	0	0
e	h	0	0	0	1	0	0
f	g	0	0	0	0	1	0
f	h	0	0	0	0	0	1

$\mathbf{out}^{(0)} = \mathbf{x}$ {the input to the MLP excluding the categorical input}
$\nabla_{\mathbf{W}}^{(1)}\mathbf{out}^{(0)} = \mathbf{0}$
$\mathbf{out}^{(1)} = f(\mathbf{W}^{(1)}\cdot\mathbf{out}^{(0)})$
$\nabla_{\mathbf{W}}^{(1)}\mathbf{out}^{(1)} = f'(\mathbf{W}^{(1)}\cdot\mathbf{out}^{(0)}) * \mathbf{I} * /\mathbf{out}^{(0)}$
$\mathbf{out}^{(2)} = f(\mathbf{W}^{(2)}\cdot\mathbf{out}^{(1)})$
$\nabla_{\mathbf{W}}^{(1)}\mathbf{out}^{(2)} = f'(\mathbf{W}^{(2)}\cdot\mathbf{out}^{(1)}) * \mathbf{W}^{(2)}\cdot\nabla_{\mathbf{W}}^{(1)}\mathbf{out}^{(1)}$
$\nabla_{\mathbf{W}}^{(2)}\mathbf{out}^{(2)} = f'(\mathbf{W}^{(2)}\cdot\mathbf{out}^{(1)}) * \mathbf{I} * /\mathbf{out}^{(1)}$

If we include the units with constant input of -1 and if we substitute for $\nabla_{\mathbf{W}}^{(2)}\mathbf{out}^{(2)}$ in the expression for $\nabla_{\mathbf{W}}^{(2)}C$ we get the expression

$$\nabla_{\mathbf{W}}^{(2)'}C = (\mathbf{s}\cdot\mathbf{out}^{(2)} - \mathbf{d}) * \mathbf{s} * f'(\mathbf{W}^{(2)'}\cdot(\mathbf{out}^{(1)}, -1)) * /(\mathbf{out}^{(1)}, -1)$$

$\mathbf{W}^{(2)'}$ is $\mathbf{W}^{(2)}$ with an additional column of bias terms. The expression for $\nabla_{\mathbf{W}}^{(1)}C$ after some substitutions becomes

$$\nabla_{\mathbf{W}}^{(1)'}C = +/(\mathbf{s}\cdot\mathbf{out}^{(2)} - \mathbf{d}) * \mathbf{s} * f'(\mathbf{W}^{(2)'}\cdot(\mathbf{out}^{(1)}, -1)) * \mathbf{W}^{(2)} * \text{“}1\_f'(\mathbf{W}^{(1)'}\cdot(\mathbf{out}^{(0)}, -1)) * /(\mathbf{out}^{(0)}, -1)$$

‘ + /’ implies summation over the first dimension of what follows. $\mathbf{W}^{(1)'}$ is $\mathbf{W}^{(1)}$ with an additional column of bias terms. An expression of the form $\mathbf{A} * \text{“}1\_\mathbf{a}$ means that each row of $\mathbf{A}$ is multiplied component-wise by the vector $\mathbf{a}$ rather than each column. Note the small difference in expression between the algorithm for the case where categorical input is converted to numeric and used as input to the MLP and the case where categorical input is treated separately. If we replace code vector, $\mathbf{s}$, by the scalar 1 then the algorithm above reverts back to the algorithm for the former. Of course there is a large difference due to the fact that in one case the categorical variables are segregated from the numeric ones.

Note that we may choose not to use sigmoid functions in the output layer. If the outputs selected by $\mathbf{s}$ are to be combined linearly through a vector $\mathbf{v}$ we replace $\mathbf{s}$ by $\mathbf{s} * \mathbf{v}$ wherever $\mathbf{s}$ appears in the expressions above. However, if there is no sigmoid transformation in the MLP output layer, $\mathbf{v}$ can be absorbed in $\mathbf{W}^{(2)'}$ without loss of generality as we will see later. Therefore it only makes sense to include another set of parameters by means of $\mathbf{v}$ if there is a non-linear transform in the output layer.

If $\mathbf{v}$ is included because a sigmoid is used in the output layer we require the following additional gradient

$$\nabla_{\mathbf{v}}C = (\mathbf{sv}\cdot\mathbf{out}^{(2)} - \mathbf{d}) * \mathbf{s}\cdot\mathbf{out}^{(2)}$$

The other gradients become

$$\nabla_{\mathbf{W}}^{(2)'}C = (\mathbf{sv}\cdot\mathbf{out}^{(2)} - \mathbf{d}) * \mathbf{sv} * f'(\mathbf{W}^{(2)'}\cdot(\mathbf{out}^{(1)}, -1)) * /(\mathbf{out}^{(1)}, -1)$$

and

$$\nabla_{\mathbf{W}}^{(1)'}C = (\mathbf{sv}\cdot\mathbf{out}^{(2)} - \mathbf{d}) * \mathbf{sv} * f'(\mathbf{W}^{(2)'}\cdot(\mathbf{out}^{(1)}, -1)) * \mathbf{W}^{(2)} * \text{“}1\_f'(\mathbf{W}^{(1)'}\cdot(\mathbf{out}^{(0)}, -1)) * /(\mathbf{out}^{(0)}, -1).$$

The complete algorithm is then after replacing $\mathbf{s}$ by $\mathbf{sv}$ which is equal to $\mathbf{s} * \mathbf{v}$

$\Delta\mathbf{W}^{(1)'} = -\mu_0 * (\mathbf{sv}\cdot\mathbf{out}^{(2)} - \mathbf{d}) * \mathbf{sv} * f'(\mathbf{W}^{(2)'}\cdot(\mathbf{out}^{(1)}, -1)) * \mathbf{W}^{(2)} * \text{“}1\_f'(\mathbf{W}^{(1)'}\cdot(\mathbf{out}^{(0)}, -1)) * /(\mathbf{out}^{(0)}, -1)$.
$\Delta_{\mathbf{W}}^{(2)'} = -\mu_1 * (\mathbf{sv}\cdot\mathbf{out}^{(2)} - \mathbf{d}) * \mathbf{sv} * f'(\mathbf{W}^{(2)'}\cdot(\mathbf{out}^{(1)}, -1)) * /(\mathbf{out}^{(1)}, -1)$
$\Delta\mathbf{v} = -\mu_2 * (\mathbf{sv}\cdot\mathbf{out}^{(2)} - \mathbf{d}) * \mathbf{s}\cdot\mathbf{out}^{(2)}$

3.1. Encoding of categorical input

Each feature vector is divided into two components with one containing the quantitative features and the other the categorical or qualitative features. Let us call the latter the categorical feature vector. Table 1 shows the method of encoding the categorical feature vector used in the simulations in this paper. Assume that there are two categorical features with values d, e, and f for category 0 and values g and h for category 1. The categorical input vector is $\mathbf{c} \in \{d, e, f\} * \{g, h\}$. The encoded form of $\mathbf{c}$ is $\mathbf{s}$.

This is 1-out-of-n encoding. However, the resulting code value is not used as input to the MLP but bypasses the MLP and is applied to the output of the MLP. This approach has been explained in detail in the preceding paragraph. The output unit selected is unique to a combination of categorical values. In this case exactly one output unit is selected as the result.

Various other encoding schemes are possible but are not compared in the simulations in this paper. However, they will be briefly described. Rather than having one output unit for each member in the Cartesian product of the categorical features another method is to have one output unit for each categorical value rather than for each combination of categorical values. For this method the binary code is 1 out of n_i for each value for each categorical feature. In case of a third method each value within each categorical feature is assigned a binary number. A binary number for each categorical feature vector may be used in a fourth method.

Each of the methods above requires fewer output units for the MLP module than the method preceding it. This means less representational power but representational power may have to be sacrificed in case of insufficient training data for a large number of degrees of freedom due to additional parameters.

4. Using the network for prediction

Let us consider alternate ways of viewing the network when it is used for prediction after training has been used to find the parameters. Note that Eq. (1) can also be written as

$$\mathbf{y} = (\mathbf{W}^{(2)\mathrm{T}}\mathbf{s})\cdot\mathbf{f}(\mathbf{W}^{(1)}\mathbf{x}) \quad (2)$$

$\mathbf{W}^{(2)\mathrm{T}}$ is the transpose of the connection matrix from the hidden layer to the output layer of the MLP. The complete network can now be described as an MLP whose weight vector from hidden layer to the single output is $\mathbf{w}(\mathbf{s}) = \mathbf{W}^{(2)\mathrm{T}}\mathbf{s}$. The weight vector therefore is a function of categorical input and of a learnable connection matrix. Since the number of values for vector $\mathbf{s}$ is the number of combinations of values of categorical features, which is the same for all the encoding methods described previously, the number of weight vectors is the same in all four encoding methods briefly described earlier. Only in method 1, however, is each weight vector a single column of $\mathbf{W}^{(2)\mathrm{T}}$. In the remaining methods weight vectors are sums of columns of $\mathbf{W}^{(2)\mathrm{T}}$. Only in the case of the first method do the vector values of $\mathbf{s}$ make up an independent set. Now let $\mathbf{S}$ be a matrix in which the columns are the values of $\mathbf{s}$. Each column of $\mathbf{S}$ corresponds to a different combination of categorical values. In encoding method 1 $\mathbf{S}$ is equal to the identity matrix. If we now let $\mathbf{W}^{(2)'\mathrm{T}} = \mathbf{W}^{(2)\mathrm{T}}\mathbf{S}$ and $\mathbf{s} = \mathbf{St}$ we can write Eq. (2) as

$$\mathbf{y} = (\mathbf{W}^{(2)'\mathrm{T}}\mathbf{t})\cdot\mathbf{f}(\mathbf{W}^{(1)}\mathbf{x}) \quad (3)$$

and

$$\mathbf{y} = (\mathbf{W}^{(2)'}\mathbf{f}(\mathbf{W}^{(1)}\cdot\mathbf{x}))\mathbf{t} \quad (4)$$

Now $\mathbf{t}$ is a vector with just one non-zero entry equal to 1 in all 4 encoding methods. $\mathbf{t}$ selects a single column from $\mathbf{S}$. Each method now has the same coding scheme for prediction with the values of $\mathbf{t}$ making up the columns of the identity matrix. However, the rows of $\mathbf{W}^{(2)'}$ are not independent for encoding methods 2–4.

Let us next compare the transformation by our hybrid multi-output network and the one in the case where a separate MLP is used for each combination of categorical values. In the latter case the prediction function is

$$\mathbf{y} = \mathbf{w}(\mathbf{c})\mathbf{f}(\mathbf{W}(\mathbf{c})\cdot\mathbf{x}) \quad (5)$$

This shows that the connection matrix from the input layer to the hidden layer and the weight vector are both dependent on the categorical vector, $\mathbf{c}$, in some unknown way determined during training. In the former case the weight vector is also dependent upon $\mathbf{c}$ but in a very specific way ($\mathbf{s}$ is dependent upon $\mathbf{c}$ and the weight vector is $\mathbf{W}^{(2)\mathrm{T}}\cdot\mathbf{s}$). The connection matrix is not dependent upon $\mathbf{c}$. Thus more representational power is obtained by having a separate MLP for each categorical vector as is expected.

5. Simulations

Following are the results of doing simulations to permit comparisons of the approaches discussed previously. The three approaches that are compared are (1) conventional method: this means replacing category values by real numbers and using the values with the other values as input; (2) 1-out-of-*n*: this means encoding the categorical value combinations using 1-out-of-*n* encoding and using the coded values with the others as input; (3) separation method: this is the same as method 2 except that the codes are not input to the MLP but are used to modify the output of the MLP.

5.1. Simulation 1: Boston housing data

The data for this simulation is concerned with housing values in suburbs of Boston (Harrison & Rubinfeld, 1978). 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 $1000s. Fig. 3 depicts a comparison between method 1 (conventional) and method 3 (separation) which is the proposed method. In each case the upper curve corresponds to test results and the lower to training results. The hidden layer consisted of five units in both cases. Both connection matrices are initialized with random values in $[-1,1]$. The number of training vectors is 303 and the number of test vectors is also 303. The RMSE before training is not plotted as it is quite large and there is a large improvement after just one training epoch. The connection matrices are modified after every presentation of an input vector. For each epoch a random permutation of all the training vectors is used for training.

This demonstrates that the preferred method is the separation method proposed in this paper. Since in the preceding situation the separation method had an advantage with more learnable parameters we will next even the score by having four hidden units in case of the conventional method and two in case of the separation method. The results are given in Fig. 4. Again the upper curve for each pair of curves corresponds to progress made on test data.

The number of learnable parameters are $61 = 14\times4 + 5\times1$ and $35 = 13\times2 + 3\times3$, respectively. Thus 75% more parameters were required in the conventional method. Yet the RMSE is approximately 14% higher after 95 epochs in the conventional case.

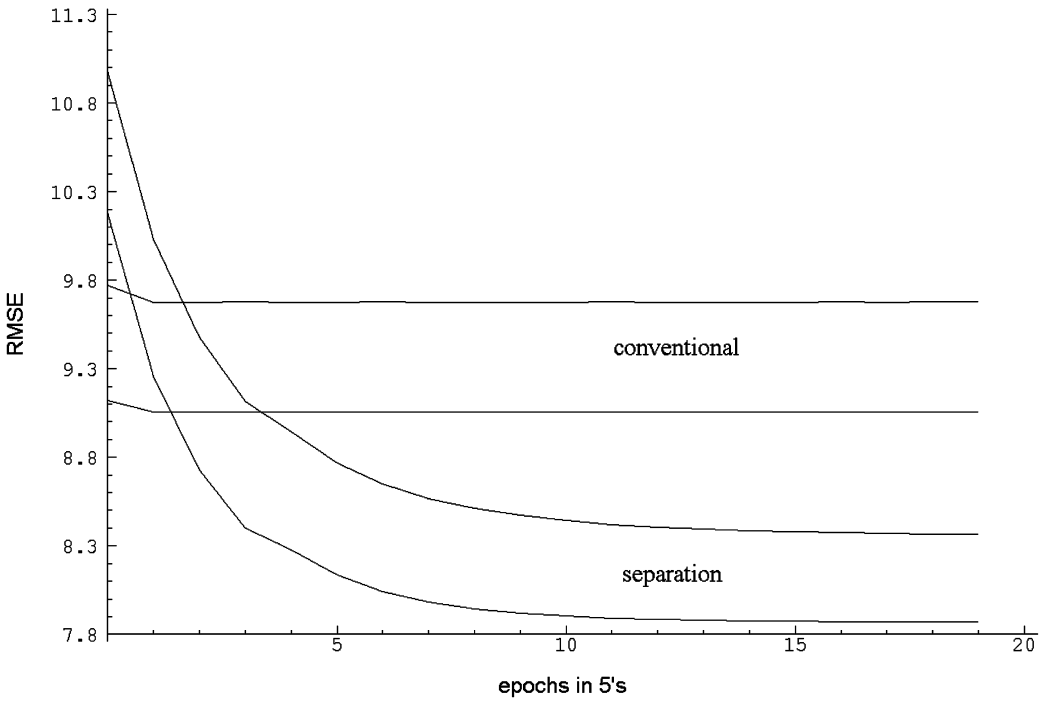

Fig. 3. Boston housing data set training performance five hidden units in each case.

In Fig. 5 a comparison is made between method 2 where 1-out-of-n encoding is used to provide categorical input representation and the separation method. The learning rate used was 0.001. The number of hidden units in both cases is 4. Again the separation method is the clear winner even though the other method has more learnable parameters. The following explanation may be considered in general by comparing the dimensions of the individual connection matrices as shown in Table 2. Let m be the number of continuous variables, n the number of binary components to represent all combinations of the categorical values, and h is the number of hidden units.

Table 2 shows that even though the separation method has fewer parameters in total it has more parameters in the second connection matrix and fewer parameters in the connection matrix between the input and the hidden layer.

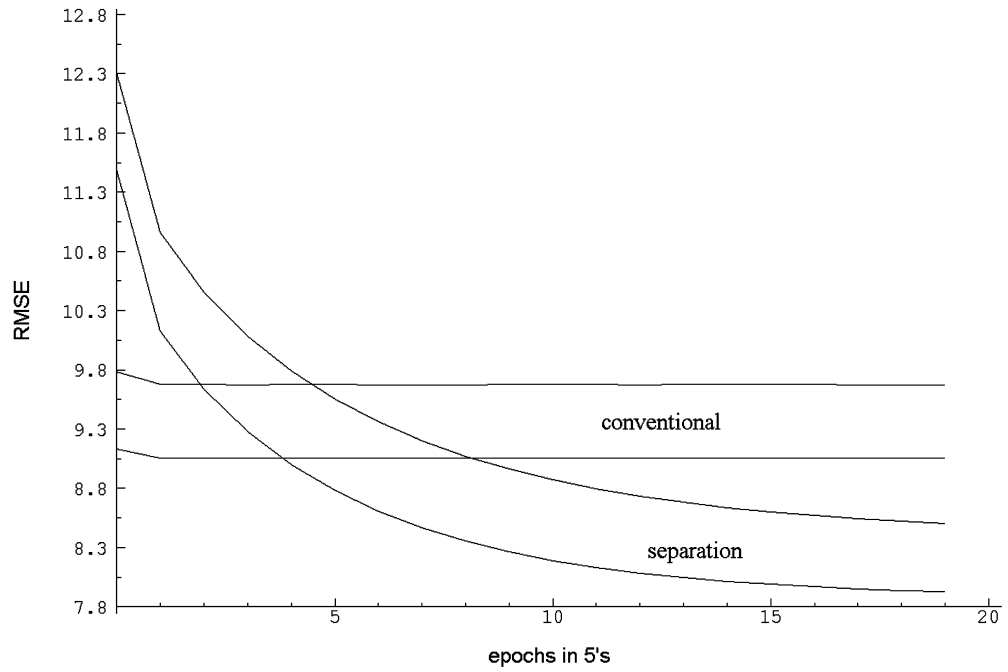

Fig. 4. Boston housing data set training performance with different number of hidden units.

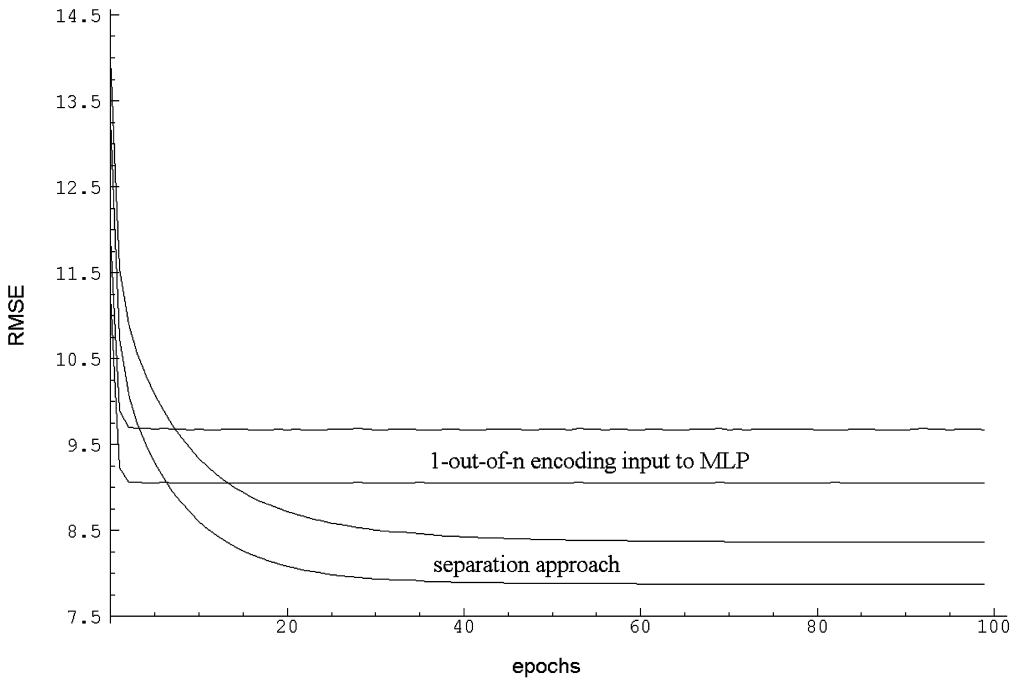

Fig. 5. A comparison of methods 2 and 3.

These parameters are more difficult to learn in back propagation because of the passing of the error through the hidden layer. The categorical inputs are not intended to be used in interpolation and therefore need not be passed through the hidden layer as they are in the 1-out-of-*n* method. They should bypass the hidden layer and be used as selector as in the separation method.

5.2. Simulation-abalone data

The data for this simulation is data that is used for predicting the age of abalone based on physical measurements.[1] The number of attributes is 8 with all but one being continuously valued. Fig. 6 shows the difference in training progression over 1000 epochs between a network which uses the values {0,1,2} for the categorical values and our multi-output network that uses the categorical input to select the output unit to use, i.e. the separation method. The output data is standard normalized before training. The error measure is the root mean square error (RMSE) as before. The graph shows both the testing RMSE and the training RMSE for the conventional approach and the separation approach. It may be noted that testing progress follows training progress over epochs very closely. Two thousand out of 4177 feature vectors were used for training while 2177 were used for testing.

The number of units in the hidden layer was 3. The learning rate used was 0.001. No special methods were used to accelerate training. Note again the improvement obtained by using the separation method.

6. Representing several functions by a single network

The partitioning of the data according to combinations of categorical values with each element of the partition corresponding to a different function suggests that we may attempt to represent several functions simultaneously by a single MLP. Following is an experiment to see if that is feasible. Three separate functions are stored in a single network; binary arithmetic operators $+$, $*$, and /.

The network that is used has no sigmoid function in the output layer. The training data consists of 300 elements and the testing of 300 elements. The values of both of the independent variables are from 0 to 20. The RMSE as a function of training epochs is shown in Fig. 7.

Note again that the progress on the test set follows the progress on the training set quite closely. The

Table 2
Comparison of parameters corresponding to the connection matrices for the 1-out-of-*n* encoding method and the separation method

	Dimensions of first connection matrix	Dimensions of second connection matrix
1-out-of-*n* encoding	$(m+n+1)\times h$	$(h+1)\times 1$
Separation method	$(m+1)\times h$	$(h+1)\times n$
Difference in number of parameters	$n\times h$	$(1-n)\times(h+1)$

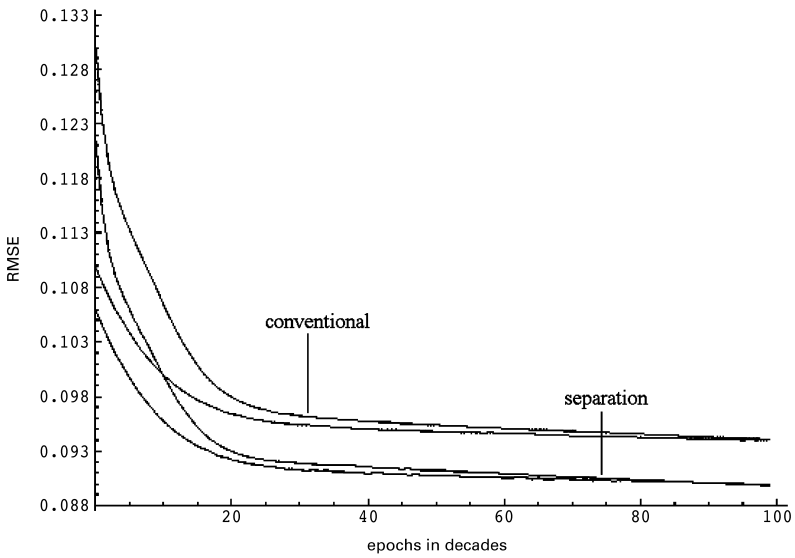

Fig. 6. Abalone data set training performance.

RMSE values for testing after 5000 epochs are 22.3523 and 11.13613. The values used to distinguish the three functions are 0, 1, and 2. In the conventional method these values are used as input to the MLP while in the second case the coded forms of these values are used to select the correct output.

To show the feasibility of loading three functions on a single network the results should still be compared to training a separate network for each of the functions above. Table 3 is a comparison of the method proposed in this paper called the separation method and the method of dedicating a network to each of the functions to be represented. If h is the number of hidden units the number of parameters are $3(3h + h + 1) = 12h + 3$ for the case where three separate networks are used and $3h + 3(h + 1) = 6h + 3$ for the separation method.

The columns in Table 3 have the following interpretations.

1. RMSE for addition network.
2. RMSE for multiplication network.
3. RMSE for division network.
4. Average RMSE over the three networks.
5. Number of hidden units in each of the separate networks.

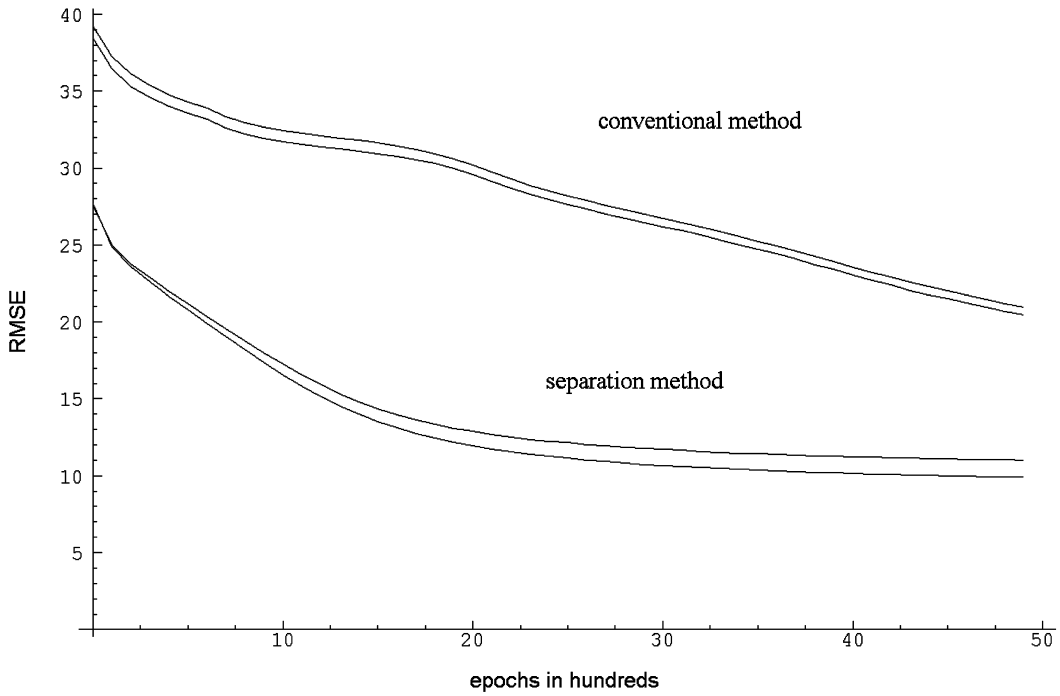

Fig. 7. Training performance in representing three functions on a single network.

Table 3
Comparison of RMSEs on testing

Dedicated network for each function						Separation method		
C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9
0.7892	20.9322	0.4877	7.4030	5	63	11.4267	5	33
1.2221	43.2865	1.1746	15.2278	2	27	8.43366	6	39
1.2221	43.2865	1.1746	15.2278	2	27	9.84887	4	27

6. Total number of parameters for the three networks.
7. RMSE for proposed separation method applied to a single network.
8. Number hidden units for proposed method.
9. Total number of parameters for proposed method.

In the first row the number of hidden units is kept the same which means that in case of three separate networks there are many more parameters in total and thus more degrees of freedom. Thus it is expected that the RMSE will be smaller for the same number of training epochs. In the third row the total number of parameters is the made the same by allowing for more hidden layer units in the case of the separation method. This time the separation method produces a smaller RMSE than the average for the three networks. This demonstrates that it is feasible to represent several functions by a single network with a separate output unit for each of the functions to be represented. In fact if the number of parameters is kept the same the separation method produces a much better result.

7. Summary and further work

We have demonstrated a very effective method for treating categorical features when training an MLP by segregating the categorical features from the quantitative features. The quantitative part of the feature vector will be processed by an MLP with additional output units. These outputs are then combined with the coded form of the categorical part of the feature vector. In some instances the most appropriate way is to have a separate function and therefore a separate MLP for each categorical feature vector instance but that may not be feasible if there is insufficient data. Adding several output units to 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 to several MLPs is limited, however.

A statistical approach by comparison consists of using indicator variables in the case where some of the independent variables are categorical. Regression is then performed in the usual way with categorical variables

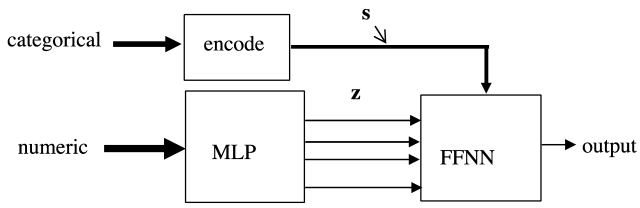

Fig. 8. More general network.

appearing as regressors, including complications such as several indicator variables and indicator variables that interact with other regressors (Neter, Wasserman, & Kutner, 1990). This statistical approach is analogous to using 1-out-of-n encoding of the categorical component for input to an MLP. Another statistical approach is to calculate a separate regression for each value of each categorical variable. This is analogous to using a separate MLP for each value of each categorical variable. Separate regressions and separate MLPs means that we have removed the categorical variable altogether and look at the data corresponding to the categorical variables as labeling of the different functions to be approximated.

The problem with the proposed separation 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 to the number of 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. Further work may therefore consist of the following generalization. The MLP here may have fewer output units than the total number of categorical feature value combinations. If we represent the output of the MLP by $\mathbf{z}$ then the FFNN in Fig. 8 below may calculate the output as

$$\mathbf{y} = \mathbf{z}\cdot\varphi(\mathbf{w}, \mathbf{s}) \tag{10}$$

with $\varphi(\mathbf{w}, \mathbf{s}) = (\varphi_i(\mathbf{w}, \mathbf{s}_i) i = 0, 1, \ldots, h-1)$ and where $\varphi_i(\mathbf{w},\mathbf{s}_i)$ are Boolean functions with values from {0,1}. The parameter vector $\mathbf{w}$ would be trainable. $\mathbf{s}_i$ $i = 0, 1, \ldots, h-1$ are elements of {0,1} and are the components of the coded value of the categorical input combination.

If we set $\varphi_i(\mathbf{w}, \mathbf{a}_i) = \mathbf{a}_i$ then we obtain the model used in the simulations in this paper. Combining the outputs of the MLP network through a dot product with weights other than values that would be learned does not improve things since the weights could be incorporated in the connection matrix since $(\mathbf{W}^{(2)'}\cdot(\mathbf{out}^{(1)}, -1))\cdot(\mathbf{v} * \mathbf{s})$ is the same as $(\mathbf{W}^{(2)'} * \mathbf{v})\cdot(\mathbf{out}^{(1)}, -1))\cdot\mathbf{s}$ by Lemma 1 in Appendix A. Note that $\mathbf{W}^{(1)'} * \mathbf{v}$ means that each column of $\mathbf{W}^{(1)'}$ is multiplied by $\mathbf{v}$

component-wise. Of course if the output layer of the FFNN contains sigmoid units the matter is different. We then get $f((\mathbf{W}^{(2)'}\cdot(\mathbf{out}^{(1)}))\cdot(\mathbf{v} * \mathbf{s})$ and it makes sense to calculate the vector $\mathbf{v}$ using learning.

Further work may consist in part of defining the above FFNN.

Acknowledgments

The support of a NSERC grant (Research Council of Canada) and the comments made by the referees is gratefully acknowledged.

Appendix A

Lemma 1. *Let* $\mathbf{A}$ *be a matrix and* $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ *be vectors* $(\mathbf{A}\cdot\mathbf{a})\cdot(\mathbf{b} * \mathbf{c})$ *is the same as* $((\mathbf{A} * \mathbf{b})\cdot\mathbf{a}).\mathbf{c}$ *and* $(\mathbf{A}^{\mathrm{T}}\cdot(\mathbf{b} * \mathbf{c}))\cdot\mathbf{a}$.

Proof. $(\mathbf{A}\cdot\mathbf{a})\cdot(\mathbf{b} * \mathbf{c})$ is $\sum_k \sum_l \mathbf{A}_{k,l} * \mathbf{a}_l * \mathbf{b}_k * \mathbf{c}_k$ is $\sum_l (\sum_k (\mathbf{A}_{k,l} * \mathbf{b}_k)\cdot\mathbf{c}_k) * \mathbf{a}_l$ which is $((\mathbf{A} * \mathbf{b})\cdot\mathbf{a})\cdot\mathbf{c}$.
$(\mathbf{A}\cdot\mathbf{a})\cdot(\mathbf{b} * \mathbf{c})$ is $\sum_k \sum_l \mathbf{A}_{k,l} * \mathbf{a}_l * \mathbf{b}_k * \mathbf{c}_k$ is $\sum_l (\sum_k ((\mathbf{A}^{\mathrm{T}}_{k,l}\cdot(\mathbf{b}_k * \mathbf{c}_k)) * \mathbf{a}_l$ which is $(\mathbf{A}^{\mathrm{T}}\cdot(\mathbf{b} * \mathbf{c}))\cdot\mathbf{a}$. □

References

Bishop, C. M. (1994). Mixture density networks. NCRG/94/004, available from http://www.ncrg.aston.ac.uk.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Clarendon Press.

Brouwer, R. K. (1997). Training a feedforward network by feeding gradients forward rather then by backpropagation of errors. *Neurocomputing*, *16*(2), 117–126.

Harrison, D., & Rubinfled, D. L. (1978). Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management*, *5*, 81–102. Date: 7 July 1993 (The data set is available at http://www.ics.uci.edu/~mlearn/MLRepository.html).

Jacobs, R. A., Jordan, M. L., Nowlan, S. J., & Hinton, G. E. (1992). Adaptive mixtures of local experts. *Neural Computation*, *3*(1), 79–87.

Lee, K., & Lee, T. (2001). Design of neural networks for multi-value regession. *International Joint Conference on Neural Networks*, 93–98.

Neter, J., Wasserman, W. W., & Kutner, M. H. (1990). *Applied linear statistical models*, IL, USA: Irwin, pp. 349–385.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, & PDP Research Group (Eds.), *Parallel distributed processing* (pp. 318–362). Cambridge, MA: MIT Press.

Wasserman, P. D. (1993). *Advanced methods in neural computing*. New York: Nostrand Reinhold (Van).

Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Doctoral Dissertation, Applied Mathematics, Harvard University, MA.