ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 13-18

Storing Many-to-many mappings on a Feed-forward Neural Network Using Fuzzy Sets

Roelof K. Brouwer

University College of the Cariboo

email rkbrouwer@ieee.org

Abstract

Feed-forward networks are generally trained to represent functions or many-to-one (m-o) mappings. In this paper however a feed-forward network with modified training algorithm is considered to represent multi-valued or one-to-many (o-m) mappings. The o-m mapping is viewed as an m-o mapping where the values corresponding to a value of the independent variable are sets. Thus the problem of representing a o-m mapping has been converted into a problem of training a network to return sets rather than vectors. The resulting o-m mapping may have variable multiplicity leading to sets of variable cardinality. The crisp sets of variable cardinality in turn are replaced by fuzzy sets of fixed cardinality by adding elements, called “do not cares” which have membership values of zero. Since the target outputs of the feedforward network are now sets of fixed cardinality and the actual output of a feedforward network is a vector the training algorithm is modified to take into account the fact that order should be removed as a constraint when the error vector is calculated. Results of simulations show that the method proposed is quite effective.

Keywords feed forward neural networks, multi-layer perceptron, relations, one-to-many mappings, many-valued functions, back propagation, fuzzy set

1 Introduction

Functions are many-to-one in the sense that an input produces a unique output. One-to-many (o-m) valued-ness however is common in everyday experience and thereby in data that represents observations since we may perceive different things from the same data.

In [KY1998] it was demonstrated that the bottleneck network could learn many-valued functions. The authors studied the problem of recalling the hand configurations required to grasp an object based upon its image. In [KY1998 , HSMMY1999] the relaxation method for recall described in [UFSK1995] is replaced by successive iteration. The result of this was that the recall process was accelerated four-fold. The authors of another paper [TN1998] discuss a very similar method for using a recurrent neural network for approximating a certain version of o-m mappings.

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 13-18

Bishop [Bish1995] 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 [LL2001] 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.
The paper is organized in the following manner. First some work published by others was briefly described. This is followed by an illustration of multi-valued-ness and a brief description of the algorithm. A simulation describing the results of applying this approach is then described. The paper ends with a description of some of the problems with the method proposed here.

2 Storing a many-many mapping on a feed-forward Network

The method for storing a o-m mapping on a (Feed-forward Network) FFN is based on interpreting the o-m mapping as a mapping with sets as dependent variable. The outputs of a feed-forward network however are of fixed dimension and order is significant. Sets on the other hand are unordered.
An example of a many-many mapping with varying multiplicity is the mapping from ages of parents and number of years married to ages of their children as depicted in Table 1 below.

Age of Husband	Age of Wife	Number of Years Couple is Married	Ages of children
22	19	1	{0.8}
30	27	4	{}
37	35	10	{1.5, 2.8, 8.7}
45	47	15	{10.8}
54	50	20	{18.4, 6.9,12.2}
60	55	35	{20.8,18.1,25.4, 30.6}

Table 1 Example of many-many mapping with varying multiplicity

There are problems in implementing this mapping on a feed forward network both in terms of training and also retrieval. If we set the number of output units equal to the maximum cardinality of the target output sets we still have a problem during retrieval because it has to be known how much of the output is applicable.
A potential solution for the problem of varying cardinality is to have another output unit present the cardinality of the output. The cardinality indicates how many of the output elements to include in the set defined by the output. Thus (3, 4.5, 6.8, 9.2, 8.7) corresponds to the set {4.5, 6.8, 9.2}. During training the last element in the preceding case is a "do not care." The solution suggested above has been dealt with in great detail in another paper by the author.

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 13-18

There is another way of dealing with the problem of variable cardinality and that is to use fuzzy sets [Pedr 1995] of fixed cardinality to represent crisp sets of variable cardinality. A crisp set {} can be represented by a fuzzy set of greater cardinality by adding any number of elements whose membership values are 0. Thus the set {4.6, 5.2, 6.4} can be made into an equivalent set of ordered pairs with cardinality of 5 as {(1, 4.6), (1, 5.2), (1, 6.4), (0, D_0), (0, D_1)}. D_0 and D_1 may be any values and stand for "do not cares". (1,4.6) means that 4.6 is included in the set. The first value in each pair is a membership value and indicates the degree to which the second value in the pair is included in the fuzzy set. During training of the feed forward network the target pairs may be in any order to facilitate training.

During use of the network for function value determination the output of the feed-forward network will be a fuzzy set but the target output however is a crisp set. This discrepancy is rectified by converting the fuzzy set to a crisp set using a threshold, t.

The gradient method of training requires that an error vector be calculated. This means that the difference between two membership functions has to be determined. Since the membership function for the target is crisp the difference may be calculated as follows.

Definition

Let $(x_0, x_1) \in \{0,1\}\times\Re$ and $(y_0, y_1) \in [0,1]\times\Re$ x_0 and y_0 are membership values while x_1 and y_1 are set members

Let $t \in (0,1]$ represent threshold

The difference between the two pairs, $(x_0, x_1)$and (y_0, y_1) is defined as

$(t - y_0, x_1 - y_1)$ for $x_0=1$

$(t- y_0, 0)$ otherwise

The distance between the two pairs is the Euclidian length of the difference vector.

For the distance between a crisp and fuzzy membership function we then take the square root of the sum of the squares of the distances between the individual pairs. Since order in a set is not defined an order may be selected to reduce the distance between the two sets. A ranking of pairs may be based on lexicographic order defined as follows. This way the membership value plays a more dominant role.

Definition

$(a,b)<(c,d)\Leftrightarrow(a<c)\vee(a=c)\wedge(b<d)$

If we have to find the difference between two membership functions of unequal length we expand the shorter one so that it is the same length as the other.

3 Architecture and Training Algorithm

Let us now consider the feed-forward network to be used and its training algorithm. The network used for training consists of an input layer, an output layer and one hidden layer. The output units in the even positions have sigmoid activation functions while the other output units have identity activation functions.
The training algorithm the gradient method. The gradient of the output with respect to all unknowns is found by feeding the gradients forward as described in [BROU1997]. $\mathbf{out}^{(2)}$ is the actual output of the network. ∇ stands for gradient while C represent the cost or error function to be minimized. $\mathbf{W}^{(k)}$ k=1,2 are the connection matrices for the layers.
If we use the cost function traditionally used in back propagation

$$C = .0.5 * +/(\mathbf{out}^{(2)}\text{-}\mathbf{d})*(\mathbf{out}^{(2)}\text{-}\mathbf{d})$$

we get

$$\nabla_{\mathbf{out}(2)}C = (\mathbf{out}^{(2)}\text{-}\mathbf{d})$$

$$\nabla_{\mathbf{W}}{}^{(2)}\ C{=}.(\mathbf{out}^{(2)} - \mathbf{d}) * f\ \grave{}\ (\mathbf{W}^{(2)}\ .\ \mathbf{out}^{(1)},\text{-}1) * / \ \mathbf{out}^{(1)},\text{-}1 \quad (1)$$

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

+/ calls for summation over the first dimension of the array that follows as operand. */ represents the outer product. $\mathbf{W}^{(2)'}$ is $\mathbf{W}^{(2)}$ with the last column removed. Note that the last column of the connection matrices now includes the bias terms and $\mathbf{out}^{(0)}$ and $\mathbf{out}^{(1)}$ are appended with –1. * “1_ means that each row of $\mathbf{W}^{(2)'}$ 'is multiplied by the vector f ` ($\mathbf{W}^{(1)}$. $\mathbf{out}^{(0)}$,-1) .
A membership function is used to convert sets of variable cardinality to constant cardinality. Thus the crisp target sets to be incorporated on the feed forward network are converted into fuzzy sets all of which are of the same cardinality. The actual output is interpreted by having the output units in the even positions produce the membership values and the units in the odd positions output the set members. The members in the fuzzy set can be ordered in any way without changing the fuzzy set as long as membership values and member values are bound together. To obtain an error vector for a particular output vector and target membership function, the pairs in the target vector will be ordered so as to reduce the error and consequent modification in weight vectors required. To effect this the target fuzzy set pairs will be ordered in the same way as the corresponding actuals using lexicographic ordering.
For example let the actual output be (0.86 8.96 0.52 2.54 0.73 5.23). The values in the even positions are interpreted as membership values. Let the target output be the set {8.87,9.24} which when expanded into a fuzzy set becomes {(1, 8.87), (1, 9.24), (0,-0.23)} The lexicographic ranking for the actual is 2, 0, 1. This means that the target should be {(0,-0.23), (1, 8.87), (1, 9.24)} . Strictly speaking the gradient of the cost function with respect to output is not just the error vector because the error vector also changes if the ranking of the output changes since then the target matching to the output will change. However this aspect is not taken into account here.

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 13-18

4 Simulations

The data used for testing the algorithm is derived from the inverse of the many to one function x+ 0.5*sin 2πx. The relation graphed below is the inverse of this function. A similar function was used by [Bish1994] in their report. The inverse is single valued and 3-valued depending upon the value of the independent variable.

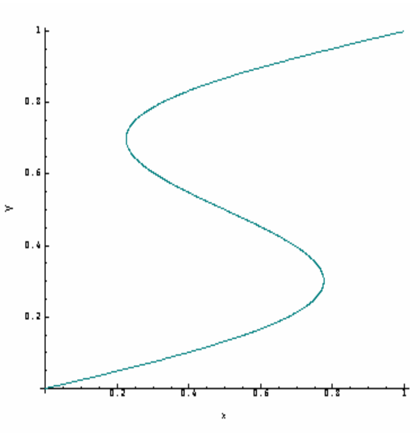

Figure 1 Inverse of x+ 0.5*sin 2πx.

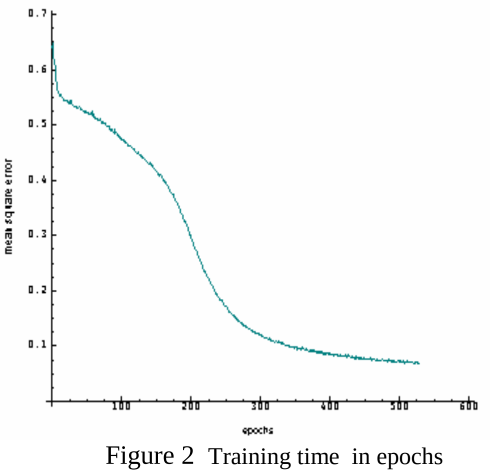

Figure 2 Training time in epochs

After 2500 epochs the mean square error was 0.75. The testing mean square error was 0.77. Table 2 shows a small subset of the test results. mv is the membership value. Column v is the column of set values whose membership value is in the preceding column. Note ordering of the pairs in the target is irrelevant.

			Target y							Actual y		
x	mv	v	mv	v	mv	v	mv	v	mv	v	mv	v
0.204	1	0.05	0	1507	0	1081	1	0.160	0	_0.208	0	0.443
0.764	1	0.267	1	0.337	1	0.942	1	0.962	1	0.251	1	0.327
0.724	1	0.229	1	0.379	1	0.932	1	0.849	1	0.254	1	0.483
0.684	1	0.204	1	0.406	1	0.921	1	0.674	1	0.292	1	0.627
0.884	1	0.972	0	1513	0	1637	1	0.954	0	0.393	0	_0.055
0.604	1	0.168	1	0.45	1	0.899	1	0.292	1	0.416	1	0.833
0.924	1	0.982	0	1515	0	1427	1	0.958	0	0.421	0	_0.132
0.564	1	0.153	1	0.47	1	0.888	1	0.143	1	0.475	1	0.889
0.364	1	0.092	1	0.566	1	0.818	1	0.186	1	0.557	1	0.835
0.484	1	0.127	1	0.507	1	0.863	1	_0.012	1	0.559	1	0.923
0.404	1	0.103	1	0.546	1	0.835	1	0.062	1	0.583	1	0.881
0.004	1	0.001	0	1425	0	1843	1	0.002	0	_0.744	0	0.126

Table 2 The target output and actual output for some of the test data.

5 Summary

It has been shown how the gradient descent training algorithm for feed forward networks can be modified so that a feed forward network can be used to represent a one-to-many mapping where the multiplicity is variable. The method proposed here is less complex than some of the other methods proposed in the literature.
Its drawback is that the one-to-many valued-ness must be explicit in the training data and it will not do to have almost one-to-many valued-ness in the training data. This might be the case where errors have converted the one-to-many data into many-to-one data. If the latter is true the data has to be pre-processed to bring out the one-to-many property.

Acknowledgements

I thank the Government of Canada for providing financial support through an NSERC grant.

6 References

[Bish1995] C. M. Bishop *Neural Networks for Pattern Recognition* Clarendon Press Oxford 1995
[Brou1997] R. K. Brouwer "Training a feed-forward network by feeding gradients forward rather than by back-propagation of errors." Neurocomputing 16(1997) 117-126
[Hech1989] R. Hecht-Nielsen *Neurocomputing*. New York: ,Addison-Wesley, (1989).
[HN1999] L. U. Hjorth I. T. Nabney. "Regularization of Mixture Density Networks". *Artificial Neural Networks*, 7-10 September (1999)
[HSMMY1999] K. Hiraoka, T. Shigehara, H. Mizoguchi, T. Mishima, S. Yoshizawa, S "On the overfitting of the five-layered bottleneck network." *Proceedings. ICONIP '99. 6th International Conference on Neural Information Processing*, Volume: 1, (1999) 234 -239
[KY1998] Hiraoka Kazuyuki and Shuji Yoshizawa. "Recalling of Many-Valued Functions by Successive Iteration on Bottleneck Networks." *ICONIP'98 5th International Conference on Neural Information Processing* (1998)
[LL2001] K. Lee and T. Lee. "Design of Neural Networks for Multi_value Regression. *International Joint Conference on Neural Networks* (2001) 93-98
[Pedr 1995] W. Pedrycz Fuzzy Sets Engineering CRC Press 1995
[Shiz1994] M. Shizawa "Regularization Networks for Approximating Multi-Valued Functions: Learning Ambiguous Input-Output Mappings from Examples" *IEEE International Conference on Neural Networks* (1994)
[TN1998] Y. Tomikawa and Kenji Nakayama. "Approximating Many Valued Mappings Using a Recurrent Neural Network" *Proceedings of 1998 IEEE World Congress on Computational Intelligence* (1998).