THE MOVING TARGETS TRAINING ALGORITHM FOR CLASSIFICATION NEURAL NETWORKS

A. LIKAS and I.E. LAGARIS

19-99

Preprint no. 19-99/1999

Department of Computer Science
University of Ioannina
451 10 Ioannina, Greece

The Moving Targets Training Algorithm for Classification Neural Networks

Aristidis Likas
Department of Computer Science
University of Ioannina
45110 Ioannina
Greece
E-mail: arly@cs.uoi.gr

Isaac E. Lagaris
Department of Computer Science
University of Ioannina
45110 Ioannina
Greece
E-mail: lagaris@cs.uoi.gr

Abstract

The neural network approach to classification based on feedforward networks requires an encoding of the problem classes in terms of target vectors corresponding to each class. All the proposed encoding schemes assume that the target vectors are specified at the beginning and remain fixed during training. We propose the moving targets approach to classification using feedforward neural networks. By allowing the target values to vary, the method achieves to significantly ease the training especially in the case of small network architectures where the optimization task is hard. For these architectures, comparative experiments using several datasets indicate that the resulting classifiers perform better than those obtained via the traditional method which assumes fixed target values.

1 Introduction

The conventional neural network approach to classification based on feedforward networks requires an encoding of the problem classes in terms of target vectors corresponding to each class [1, 7]. This is necessary, since MLPs actually perform regression, therefore in order to use them for classification the target vectors for each class must be specified. In general, class encoding aims at separating the classes by maximizing the distance between the target vectors corresponding to each class. The classification of an input pattern is based on the nearest neighbor principle: the network output vector is compared with the corresponding target vectors and the pattern is classified to the class whose target vector has the minimum distance from the corresponding network output vector.

The most common approach to target encoding is the 1-out-of-K encoding for a problem with K classes. Other types of target encoding also have been proposed such as error-correcting codes [2] and "thermometer" coding [7]. A common characteristic of all approaches is that the target vectors (codes) are fixed from the beginning and do not change during training.

In the proposed approach, the above restriction is relaxed. There is no reason why the class target vectors should remain fixed as long as the distance between them does not become very small. In this spirit, we propose the *moving targets* approach, where the target encoding is *parametric* and the adjustment of the corresponding parameters is part of the training process. This adds several advantages the main one being the added flexibility for the training. The parameters of the network are adjusted so that the outputs move towards the targets, but also the targets may move towards the outputs. However, care must be taken so that the target vectors remain distinct and the discrimination among the classes is guaranteed.

To implement the idea of moving targets we propose an appropriate formulation of the error function to be minimized in order to train the network. We consider the case of two classes C_1 and C_2 and one network output. The extension to the case of many classes and many outputs is straightforward. We show that given the distance c between the target values t_1 and t_2 corresponding to each of the two classes, the values of t_1 and t_2 are adjusted during training and can be specified from the parameters of the network (weights and biases) using appropriate formulas. Section 2 presents the method and also provides the formulas for computing the derivatives of the proposed error function with respect to the network parameters. Section 3 provides comparative results using well-known datasets and, finally, Section 4 contains conclusions and some directions for further research.

2 The Moving Targets Method

Consider a classification problem with two classes C_1 and C_2 and a training set with n patterns $(x_1, \ldots, x_n)$, with $x_i \in R^d$. Let X_1 and X_2 denote the sets containing the elements of X belonging to C_1 and C_2 respectively and let n_1 and n_2 denote the cardinality of those sets.

We wish to use the training set X in order to construct a multilayer perceptron (MLP) suitable for the above classification problem. More specifically, we assume an MLP with d inputs, arbitrary number of hidden layers and one output. Let p denote the set of network parameters (weights and biases) p_j $(j = 1, \ldots, P)$ and $N(p, x)$ denote the output of the network for an input pattern x. We encode with t_1 the target output value for the patterns of class C_1 and with t_2 the corresponding value for the patterns of class C_2. The training is performed by minimizing the error function

$$E(p, t_1, t_2) = \sum_{x_i \in X_1} (N(p, x_i) - t_1)^2 + \sum_{x_i \in X_2} (N(p, x_i) - t_2)^2 \qquad (1)$$

with respect to the network parameters p. Let $p^\star$ denote the optimal values for the network parameters. Then, the resulting network $N(p^\star, x)$ can be used as a minimum distance classifier: if $|N(p^\star,x)-t_1| < |N(p^\star,x)-t_2|$ then x is classified as a C_1 class member, otherwise as a C_2 class member.

Suppose now that t_1 and t_2 are treated as variational parameters and hence, are allowed to vary during the training process just like the network parameters p. These two extra parameters enhance the degrees of freedom and it is expected that the minimization task will be easier. Since at the minimum we have that $\partial E/\partial t_1 = \partial E/\partial t_2 = 0$, one readily obtains:

$$t_1(p) = \frac{1}{n_1}\sum_{x_i\in X_1} N(p,x_i) =< N(p) >_1 \tag{2}$$

and, similarly,

$$t_2(p) = \frac{1}{n_2}\sum_{x_i\in X_2} N(p,x_i) =< N(p) >_2 \tag{3}$$

This means that the optimum value for t_i is the average of the network outputs for the training patterns of class C_i. Using these values, we eliminate t_1 and t_2 in eq. (1), which becomes

$$E(p) = n_1(< N(p)^2 >_1 - < N(p) >_1^2) + n_2(< N(p)^2 >_2 - < N(p) >_2^2) \tag{4}$$

The minimum of the above error function is zero and occurs iff $N(p^\star,x)=a$, with a being a constant. In this case $t_1 = t_2 = a$ and the solution is of no interest for classification problems. To avoid this trivial solution and at the same time retain the 'mobility' of the output targets, we reformulate the training problem by adding a proper constraint:

$$\text{minimize } E(p,t_1,t_2)$$
$$\text{subject to: } |t_2 - t_1| > 0$$

We can assume without loss of generality that $t_1 < t_2$ and write

$$t_2 = t_1 + c \tag{5}$$

with $c > 0$. In this case the error function takes the form:

$$E(p,t_1,c) = \sum_{x_i\in X_1}(N(p,x_i)-t_1)^2 + \sum_{x_i\in X_2}(N(p,x_i)-t_1-c)^2 \tag{6}$$

with $c > 0$. This constrained minimization problem can be treated by a host of applicable methods. Note that this formulation contains the classical formulation where $t_1 = 0$ and $t_2 = 1$ by setting $t_1 = 0$, $c = 1$.

Requiring the partial derivative of E with respect to t_1 equal to zero we get that

$$t_1(p,c) = \frac{1}{n}(< N(p) > -n_2 c) \tag{7}$$

and

$$t_2(p,c) = \frac{1}{n}(< N(p) > + n_1 c) \tag{8}$$

where

$$< N(p) > = \frac{1}{n}\sum_{k=1}^{n} N(p, x_k) \tag{9}$$

Consequently, we get the following expression for the error function

$$\begin{aligned} E(p,c) &= \sum_{x_i \in X_1} (N(p,x_i) - < N(p) > + \frac{n_2}{n}c)^2 \\ &+ \sum_{x_i \in X_2} (N(p,x_i) - < N(p) > - \frac{n_1}{n}c)^2 \end{aligned} \tag{10}$$

with the constraint that $c > 0$. One may employ one of the several applicable methods for inequality constrained minimization problems to handle the above problem, however we followed a much simpler approach.

Instead of having c as an optimization parameter we considered that c is a constant in the interval $0 < c < 1$. The choice of this interval is justified by the fact that we used MLPs with a sigmoid output unit, ie. $N(p,x) \in (0,1)$. Then we solved a series of *unconstrained* optimization problems (for several datasets) each considering a different constant value of c in the range $(0,1)$. We found that on average, better classifiers are obtained when $c \in [0.65, 0.85]$ and that there are no significant variations within this range. A value of $c = 0.75$ is recommended for general use. Once the minimization process has terminated the final target values t_1 and t_2 can be computed from equations (7) and (8).

In order to efficiently minimize the error function (10) we need to compute the derivatives $\partial E / \partial p_j$ for any parameter p_j $(j = 1, \ldots, P)$ (weight or bias) of the network $N(p,x)$ and then employ an optimization algorithm that exploits the derivatives to perform minimization (e.g. gradient descent, quasi-Newton, conjugate gradients etc [4]).

To compute the error partial derivatives we first need to calculate the partial derivatives

$$d(k,j) = \frac{\partial N(p, x_k)}{\partial p_j} \tag{11}$$

for $(k = 1, \ldots, n)$ and $(j = 1, \ldots, p)$. The partial derivative $d(k,j)$ can be easily computed using the backpropagation technique in the case of MLPs. Then, for each parameter p_j we define the average

$$\bar{d}(j) = \frac{1}{n}\sum_{k=1}^{n} d(k,j) \tag{12}$$

Equation (10) can be written as

$$E(p) = \sum_{x_i \in X_1} e_1^2(p, x_i) + \sum_{x_i \in X_2} e_2^2(p, x_i) \tag{13}$$

where

$$e_1(p,x) = N(p,x) - \frac{1}{n}\sum_{k=1}^{n} N(p,x_k) + \frac{n_2}{n}c \tag{14}$$

and

$$e_2(p,x) = N(p,x) - \frac{1}{n}\sum_{k=1}^{n} N(p,x_k) - \frac{n_1}{n}c \tag{15}$$

Consequently, the partial derivative $\partial E/\partial p_j$ can be written as

$$\frac{\partial E}{\partial p_j} = 2\sum_{x_i \in X_1} e_1(p,x_i)(d(i,j) - \bar{d}(j)) + 2\sum_{x_i \in X_2} e_2(p,x_i)(d(i,j) - \bar{d}(j)) \tag{16}$$

3 Implementation and Experiments

To test the effectiveness of the proposed training method, we performed comparisons with the classical approach to target encoding (*fixed target approach*) where the targets are fixed and assume values $t_1 = 1$ and $t_2 = 0$. In the fixed target approach, if the MLP output for an input pattern x is $N(x) > 0.5$ then x is classified to class C_1, otherwise it is classified to class C_2. We have tested MLP architectures with one hidden layer and one sigmoid output unit. The architectures differ only in the number H of sigmoid units in the hidden layer.

To perform minimization of the error function, we have used the quasi-Newton BFGS method [4] provided by the MERLIN optimization environment [5] which has been found to be very effective in the case of MLP training [3].

In our series of experiments several datasets have been considered. For each dataset and for each MLP architecture (e.g for each H value) we have conducted 30 experiments with different initial weight values randomly selected in $(-1, 1)$. In every experiment both the fixed target and the moving target approach were tested starting from the same initial weight values. In this way, we were able to achieve a fair comparison of the training capabilities of both methods.

The first type of datasets we have used, concern the *parity* problems with the number of inputs d ranging from 3 to 8. Tables 1 and 2 display the average (over the 30 runs) percentage of correctly classified patterns as the number of hidden units H increases.

The second type of datasets were the *diabetes* datasets obtained from the Proben1 database [6]. The percentage (over the 30 runs) of correctly classified training/test patterns are displayed in Tables 3 and 4.

As experimental results indicate, the performance improvement obtained using the moving targets approach is significant when architectures of small size are employed. In such cases the number of adjustable parameters of the network is smaller and, therefore, it is more difficult for the optimization procedure to train the network adequately. Therefore, the flexibility

H	d=3	d=4	d=5	d=6
1	73	62	64	66
2	82	85	83	87
3	97	93	91	88
4	100	97	95	91
5	100	99	97	93
6	100	100	98	96
7	100	100	98	97
8	100	100	100	98

Table 1: Percentage of correctly classified training patterns using the moving targets method for parity datasets with d inputs and H units in the hidden layer.

H	d=3	d=4	d=5	d=6
1	50	50	52	55
2	64	68	70	75
3	85	81	82	84
4	97	89	90	91
5	99	95	92	94
6	100	98	96	97
7	100	99	98	97
8	100	100	99	98

Table 2: Percentage of correctly classified training patterns using the fixed targets method for parity datasets with d inputs and H units in the hidden layer.

H	diab1	diab2	diab3
1	78/76	79/74	78/75
2	80/77	81/72	81/76
3	82/77	84/74	84/78
4	84/75	86/73	86/77
5	87/76	90/73	88/76

Table 3: Percentage of correctly classified training/test patterns using the moving targets method for the diabetes datasets from the Proben database using H units in the hidden layer.

H	diab1	diab2	diab3
1	67/63	68/60	66/64
2	71/70	72/65	71/70
3	75/73	77/70	78/72
4	83/75	85/72	84/75
5	86/75	89/73	88/76

Table 4: Percentage of correctly classified training/test patterns using the fixed targets method for the diabetes datasets from the Proben database using H units in the hidden layer.

in optimization offered by the proposed method makes training substantially more efficient and the capability of the training method to avoid low quality solutions becomes very clear. As the size of the architecture increases, the two approaches tend to provide comparative performance results. Therefore, the moving targets method can be seriously considered as a technique for training feedforward classification networks.

Another important issue concerns the distance c between the the target values a and b corresponding to each class. As already noted, experiments indicate that the value of c must not be very large nor very small. The reason is, that since the output of the MLP is constrained to the interval $(0, 1)$, a large value (close to 1) reduces the flexibility for the targets t_1 and t_2 to be adjusted by the training procedure. On the other hand, a small value of c does not seem sufficient for effective discrimination between classes. Therefore, it is suggested that the value of c should be selected in the range $[0.65, 0.85]$ and we have used the value $c = 0.75$ in our experiments.

4 Conclusions

We have presented the moving targets approach to classification using feedforward neural networks. The method exploits the fact that the target values are allowed to change during training providing in this way greater flexibility to the minimization process which leads to more effective training. This advantage becomes very clear in the case of small architectures with few parameters where the minimization process is harder.

There is a lot of room for further research. Topics such as multi-class and multi-output extensions, different MLP architectures, use of different networks (eg RBF networks) await for further study and development. Moreover inequality constrained optimization methods may also prove useful in-spite the complexity of their implementation that may at first discourage their application. Our present results suggest that the moving targets idea is promising and certainly deserves further investigation.

References

[1] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[2] T.G. Dietterich and G. Bakiri, "Error-correcting Output Codes: A general method for improving multiclass inductive learning programs", in D. Wolpert *The Mathematics of Generalization: The Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning*, pp. 395-407, Addison-Wesley, 1995.

[3] A. Likas, D.A. Karras and I.E. Lagaris, "Neural Network Training and Simulation Using a Multidimensional Optimization System", *Int. J. of Computer Mathematics*, vol. 67, pp. 33-46, 1998.

[4] R. Fletcher, *Practical methods of Optimization*, second edition, Wiley, 1987.

[5] D.G. Papageorgiou, I.N. Demetropoulos and I.E. Lagaris, "Merlin-3.0, A Multidimensional Optimization Environment", *Computer Physics Communications*, vol. 109, pp. 227-249, 1998.

[6] Prechelt, L., *PROBEN1: A set of Neural Network Benchmark Problems and Benchmarking Rules*, Tech. Report 21/94, September 30, 1994, Fakultat fur Informatik, Universitat Karlshruhe, Germany.

[7] M. Smith, *Neural Networks for Statistical Modeling*, Van Nostrand Reinhold, 1993.