A Successive Projection Method for Multilayer Neural Networks

Keiji Tatsumi and Masao Fukushima

Graduate School of Information Science
Nara Institute of Science and Technology
Ikoma, Nara 630-01, Japan

March 23, 1995

Abstract

Error back-propagation (BP) is one of the most popular ideas used in learning algorithms for multilayer neural networks. In BP algorithms, there are two types of learning schemes, on-line learning and batch learning. The on-line BP has been applied to various problems in practice, because of its simplicity of implementaion. However, efficient implementation of the on-line BP usually requires an *ad hoc* rule for determininig the learning rate of the algorithm. In this paper, we propose a new learning algorithm derived from the successive projection method (SPM) for solving a system of nonlinear inequalities. Although SPM can be regarded as a modification of on-line BP, the former algorithm determines the learning rate (step-size) adaptively based on the output for each input pattern. Some simulation results on pattern classification problems indicate that the proposed algorithm is more effective than the standard on-line back-propagation algorithm.

1 Introduction

The multilayer neural network (NN) is an artificial neural network which realizes several or many pairs of input and output patterns. Error back-propagation (BP) [10] is one of the most popular ideas used in learning algorithms for such networks. There are basically two types of learning schemes in BP algorithms. One is called "batch" back-propagation, and the other is called "on-line" back-propagation. Many speed-up algorithms have been proposed for both types of BP algorithms. Accelerated methods of batch BP have been designed by exploiting various optimization methods such as the conjugate gradient method and quasi-Newton methods [1, 4, 5, 12]. The on-line BP has been applied to various problems in practice, because of its simplicity of implementation. However, efficient implementation of the on-line BP usually requires an *ad hoc* rule for determining the learning rate of the algorithm. In this paper, we propose a new learning algorithm derived from the successive projection method (SPM) for solving a system of nonlinear inequalities. This algorithm is an extension of the one proposed in [13], which can only be applied to networks with a single output unit. Although SPM can be regarded as a modification of on-line BP, the former algorithm determines the learning rate (step-size) adaptively based on the output for each input pattern. Some simulation results on pattern classification problems indicate that the proposed algorithm is more effective than the standard on-line back-propagation algorithm.

This paper consists of six sections. In Section 2, we give some definitions and notations for the multilayer neural network and briefly describe the error back-propagation method. In Section 3, we present a learning algorithm based on the successive projection method (SPM) and in Section 4 we discuss the relation between SPM and on-line BP. In Section 5, we show computational results with SPM and on-line BP. Finally Section 6 concludes the paper.

2 Multilayer neural network

We consider a multilayer neural network (NN) in which each unit is governed by the following equations:

$$y_j^k = f(u_j^k), \tag{1}$$

$$u_j^k = \sum_i w_{ij}^{k-1} y_i^{k-1} - \theta_j^k, \tag{2}$$

where w_{ij}^{k-1} is a weight from unit i in layer $k-1$ to unit j in layer k, θ_j^k is a threshold value at unit j in layer k, and f is the sigmoid function given by

$$f(u) = \frac{1}{1+\exp(-\lambda u)}, \tag{3}$$

where λ is a positive constant. The multilayer neural network consists of an input layer, an output layer and some hidden layers. The layers are numbered $1, 2, \cdots, K$, where layer 1 is the input layer and layer K is the output layer. The units in the intput layer and in the output layer are respectively called intput units and output units.

It is convenient to consider the threshold value θ_j^k a weight for the constant input -1 from a dummy unit. The input and output of the neural network are represented as vectors $\boldsymbol{x}$ and $\boldsymbol{z}$, respectively. The weights, including the threshold values, are represented as a vector $\boldsymbol{w} = (w_{11}^1, w_{12}^1, \cdots, w_{ij}^k, \cdots)^T$. Note that $\boldsymbol{z}$ is regarded as a function of $\boldsymbol{w}$ and $\boldsymbol{x}$, namely $\boldsymbol{z} = \boldsymbol{z}(\boldsymbol{w}, \boldsymbol{x})$. Throughout, the numbers of units in the input and output layers are denoted M and N, respectively.

The learning of the multilayer neural network (NN) means a process of changing weights $\boldsymbol{w}$ to realize given P input-output pairs

$$\begin{array}{ccc} \text{inputs vectors} & & \text{target vectors} \\ \boldsymbol{x}^1 & \longrightarrow & \boldsymbol{t}^1 \\ \boldsymbol{x}^2 & \longrightarrow & \boldsymbol{t}^2 \\ & \vdots & \\ \boldsymbol{x}^P & \longrightarrow & \boldsymbol{t}^P \end{array}$$

where each input vector $\boldsymbol{x}^p \in R^M$ is called a "pattern" and the corresponding target vector $\boldsymbol{t}^p \in R^N$ is called the "training signal" for pattern p. To measure a gap between the training signals and the outputs of NN, we define the output error function $e_s^p(\boldsymbol{w})$, which is associated with each output unit s and each pattern p, and the total error function $E(\boldsymbol{w})$ by

$$e_s^p(\boldsymbol{w}) = \frac{1}{2}(z_s^p(\boldsymbol{w}) - t_s^p)^2 \tag{4}$$

and

$$E(\boldsymbol{w}) = \sum_{p=1}^{P}\sum_{s=1}^{N} e_s^p(\boldsymbol{w}), \tag{5}$$

respectively. Here, for each pattern p, $\boldsymbol{z}^p(\boldsymbol{w}) = (z_1^p(\boldsymbol{w}), \cdots, z_N^p(\boldsymbol{w}))^T$ represents the output vector $\boldsymbol{z}(\boldsymbol{x}^p, \boldsymbol{w})$ of NN with weights being fixed at $\boldsymbol{w}$. Now, the learning of NN can be formulated as the following minimization problem:

$$\text{Minimize } E(\boldsymbol{w}). \tag{6}$$

In order to solve this problem, we may use the gradient of $E(\boldsymbol{w})$, which is given by

$$\nabla E(\boldsymbol{w}) = \sum_{p=1}^{P}\sum_{s=1}^{N} \nabla e_s^p(\boldsymbol{w}).$$

Specifically the update of weights $\boldsymbol{w}$ at the l-th iteration of a gradient-based learning algorithm may be written as

$$\boldsymbol{w}^{l+1} = \boldsymbol{w}^l - \eta \sum_{p=1}^{P}\sum_{s=1}^{N} \nabla e_s^p(\boldsymbol{w}^l), \tag{7}$$

where η is a positive step-size called the learning rate. This learning algorithm, which may be regarded as the steepest descent method for solving (6), is the error back-propagation (BP) method popularized by Rumelhart, Hinton and Williams [10].

In the same paper [10], Rumelhart, Hinton and Williams suggest the following alternative method:

$$\boldsymbol{w}^{l+1} = \boldsymbol{w}^l - \eta \sum_{s=1}^{N} \nabla e_s^p(\boldsymbol{w}^l), \tag{8}$$

where p is *an* input pattern, rather than the set of all patterns, which is chosen at the l-th iteration. In the latter method, once the gradient vector $\sum_{s=1}^{N} \nabla e_s^p(\boldsymbol{w}^l)$ for a pattern p is calculated, weights $\boldsymbol{w}$ are immediately updated by using it.

The methods (7) and (8) are respectively called batch learning and on-line learning. In this paper, we simply call them batch BP and on-line BP. On-line BP is preferred to batch BP, particularly when all training signals are not available at the start of learning, or a continuous adaptation to a stream of input-output signals is demanded [1, 3]. In addition, because of its simplicity of implementation, it is often employed in various practical problems. On the other hand, if high precision mapping is required, batch BP is more effective [3]. Both methods have been interpreted as an algorithm modeled after the steepest descent method for solving problem (6). With regard to batch BP, this interpretation is correct. In fact, if the learning rate η is chosen to be sufficiently small, then it is expected that the sequence $\{\boldsymbol{w}^l\}$ of weights generated by this method converges to a local minimum of problem (6). But it does not seem appropriate to regard on-line BP as the steepest descent method for problem (6), because a single update in batch BP is not necessarily close to the aggregation of updates for all patterns in on-line BP, particularly when the number of training signals is very large. Further, while batch BP can be improved by applying various optimization methods such as the conjugate gradient method and quasi-Newton methods [1, 4, 12], speed-up methods for on-line BP usually require *ad hoc* selection of parameters.

In the next section, we shall modify the on-line BP by utilizing the idea of successive projection method (SPM) for solving a system of nonlinear inequalities. This method has the attractive feature that we can quantitatively determine the learning rate based on the output for each input pattern.

3 Successive projection method

The learning of NN can be regarded as solving the following system of equalities for $\boldsymbol{w}$:

$$z_s^p(\boldsymbol{w}) = t_s^p, \quad s \in \{1, \cdots, N\}, \quad p \in \{1, \cdots, P\}. \tag{9}$$

If t_s^p are real numbers, this problem may be considered an approximate realization of a mapping, while if t_s^p are binary numbers, it represents the classification of patterns. Although the method to be presented in this section can be applied to both problems (cf. Section 6), we shall focus our attention mainly to the latter problem.

For a pattern classification problem, it would be appropriate to use proper thresholds T_u, T_l such that $0 < T_l < T_u < 1$ (for example $T_l = 0.1$, $T_u = 0.9$), for which the outputs of NN are required to satisfy the following system of inequalities, rather than adjusting the output of NN to 0 or 1 exactly:

$$\begin{cases} z_s^p(\boldsymbol{w}) & \geq & T_u, \quad s \in I^p, \\ z_s^p(\boldsymbol{w}) & \leq & T_l, \quad s \in J^p, \end{cases} \qquad p \in \{1, \cdots, P\}, \tag{10}$$

where

$$I^p = \{s : t_s^p = 1\}, \quad J^p = \{s : t_s^p = 0\}.$$

For convenience, we define

$$\begin{aligned} \bar{z}_s^p(\boldsymbol{w}) &= \begin{cases} z_s^p(\boldsymbol{w}), & s \in I^p, \\ -z_s^p(\boldsymbol{w}), & s \in J^p, \end{cases} \\ \bar{t}_s^p &= \begin{cases} T_u, & s \in I^p, \\ -T_l, & s \in J^p. \end{cases} \end{aligned}$$

Then (10) can be rewritten as

$$\bar{z}_s^p(\boldsymbol{w}) \ \geq \ \bar{t}_s^p, \quad s \in \{1, \cdots, N\}, \quad p \in \{1, \cdots, P\}. \tag{11}$$

To solve (11), the successive projection method (SPM), which is an iterative method for solving a system of nonlinear inequalities, can be applied.

To begin with, we consider the case $\bar{t}^p \in R$, that is $N = 1$. First we choose a pattern p and linearize the inequality (11) for pattern p at the current point $\boldsymbol{w}^l$,

$$\bar{z}^p(\boldsymbol{w}^l) + \nabla \bar{z}^p(\boldsymbol{w}^l)^T(\boldsymbol{w} - \boldsymbol{w}^l) \geq \bar{t}^p, \tag{12}$$

where the subscript s for $\bar{z}^p$ and $\bar{t}^p$ has been omitted since $N = 1$. We then determine the next point $\boldsymbol{w}^{l+1}$ using the orthogonal projection (cf. Appendix) of the current point $\boldsymbol{w}^l$ onto the half space formed by the linearized inequality (12):

$$\boldsymbol{w}^{l+1} = \boldsymbol{w}^l - \omega \, \frac{\min\,[\bar{z}^p(\boldsymbol{w}^l) - \bar{t}^p,\ 0\,]}{\|\nabla \bar{z}^p(\boldsymbol{w}^l)\|^2} \nabla \bar{z}^p(\boldsymbol{w}^l),$$

where ω is a positive constant called the relaxation parameter. Next we choose another pattern and repeat the above process until the generated sequence $\{\boldsymbol{w}^l\}$ converges to a limit [13].

Now, we extend this method to the inequalities (11), where the output $\bar{z}^p_s(\boldsymbol{w})$ for each pattern p is an N-vector. Then the linearization of (11) for pattern p yields

$$\left\{ \begin{array}{c} \bar{z}^p_1(\boldsymbol{w}^l) + \nabla \bar{z}^p_1(\boldsymbol{w}^l)^T(\boldsymbol{w} - \boldsymbol{w}^l) \geq \bar{t}^p_1, \\ \bar{z}^p_2(\boldsymbol{w}^l) + \nabla \bar{z}^p_2(\boldsymbol{w}^l)^T(\boldsymbol{w} - \boldsymbol{w}^l) \geq \bar{t}^p_2, \\ \vdots \\ \bar{z}^p_N(\boldsymbol{w}^l) + \nabla \bar{z}^p_N(\boldsymbol{w}^l)^T(\boldsymbol{w} - \boldsymbol{w}^l) \geq \bar{t}^p_N. \end{array} \right. \tag{13}$$

Note that there are N half spaces corresponding to the inequalities in (13). We determine the next point as a linear combination of the projections of $\boldsymbol{w}^l$ onto those half spaces. Specifically, we put

$$\boldsymbol{w}^{l+1} = \boldsymbol{w}^l - \frac{\omega}{\nu^l} \sum_{s=1}^{N} \frac{\min\,[\bar{z}^p_s(\boldsymbol{w}^l) - \bar{t}^p_s,\ 0\,]}{\|\nabla \bar{z}^p_s(\boldsymbol{w}^l)\|^2} \nabla \bar{z}^p_s(\boldsymbol{w}^l), \tag{14}$$

where ν^l is the number of those inequalities in (11) which are violated at the current point $\boldsymbol{w}^l$. A single iteration of this method is illustrated in Fig.6 for the case where $N = 2$ and $\omega = 1$. In Fig.6, the two curves correspond to the nonlinear inequalities (11) for pattern p, while the two straight lines correspond to the linearized inequalities (13). Further the two arrows with broken line stand for the projections of $\boldsymbol{w}^l$ onto the half spaces determined by (13). The next point $\boldsymbol{w}^{l+1}$ is determined as a linear combination of those projections.

The algorithm of SPM is stated as follows.

Successive Projection Method (SPM)

Step 1 Select an initial point $\boldsymbol{w}^0$ and relaxation parameter $\omega > 0$.

Set $l := 0$ and $p := 1$.

Step 2 Determine the next point $\boldsymbol{w}^{l+1}$ by

$$\boldsymbol{w}^{l+1} = \boldsymbol{w}^l - \frac{\omega}{\nu^l} \sum_{s=1}^{N} \frac{\min\,[\bar{z}_s^p(\boldsymbol{w}^l) - \bar{t}_s^p,\ 0\,]}{\|\nabla \bar{z}_s^p(\boldsymbol{w}^l)\|^2} \nabla \bar{z}_s^p(\boldsymbol{w}^l),$$

where ν^l is the number of violated inequalities in (11) at $\boldsymbol{w}^l$.

Step 3 If a termination criterion is not satisfied, set $l := l + 1$,

$p := l(\mathrm{mod} P) + 1$ and go to **Step 2**.

4 Relation between SPM and BP

In this section, we point out that SPM can be regarded as a modification of on-line BP. Since we deal with the system (11) of inequalities rather than the system (9) of equalities, we need to slightly modify the definition of the error functions as follows:

$$\bar{e}_s^p(\boldsymbol{w}) = \frac{1}{2}\{\min\,[\bar{z}_s^p(\boldsymbol{w}) - \bar{t}_s^p,\ 0\,]\}^2 \tag{15}$$

and

$$\bar{E}\,(\boldsymbol{w}) = \sum_{p=1}^{P} \sum_{s=1}^{N} \bar{e}_s^p(\boldsymbol{w}). \tag{16}$$

It is rather straightforward to extend the error back-propagation (BP) method described in Section 3 to the modified error function $\bar{E}(\boldsymbol{w})$ defined by (16). In fact, batch BP and on-line BP can be written in a manner similar to (7) and (8), respectively, as follows:

$$\boldsymbol{w}^{l+1} = \boldsymbol{w}^l - \eta \sum_{p=1}^{P} \sum_{s=1}^{N} \nabla \bar{e}_s^p(\boldsymbol{w}^l), \tag{17}$$

$$\boldsymbol{w}^{l+1} = \boldsymbol{w}^l - \eta \sum_{s=1}^{N} \nabla \bar{e}_s^p(\boldsymbol{w}^l). \tag{18}$$

Since the gradient of the error function $\bar{e}_s^p(\boldsymbol{w})$ is given by

$$\nabla \bar{e}_s^p(\boldsymbol{w}) = \min\, [\bar{z}_s^p(\boldsymbol{w}) - \bar{t}_s^p,\ 0\,] \nabla \bar{z}_s^p(\boldsymbol{w}),$$

(14) particularly becomes

$$\boldsymbol{w}^{l+1} = \boldsymbol{w}^l - \sum_{s=1}^{N} \frac{\omega}{\nu^l\, \|\nabla \bar{z}_s^p(\boldsymbol{w}^l)\|^2} \nabla \bar{e}_s^p(\boldsymbol{w}^l). \tag{19}$$

Notice that (19) assigns different “step-sizes” $\omega/(\nu^l\, \|\nabla \bar{z}_s^p(\boldsymbol{w}^l)\|^2)$ to output units $s = 1, \cdots, N$, while (18) uses the common learning rate η for all output units. When the number of output units is one, SPM may be regarded as essentially the same as the modified on-line BP, because SPM uses the same search direction as the modified on-line BP. However, the learning rate η in on-line BP is usually held constant or controlled using an *ad hoc* rule. In SPM, the step-sizes $\omega/(\nu^l\, \|\nabla \bar{z}_s^p(\boldsymbol{w}^l)\|^2)$ are adaptively determined on the basis of the output of NN for each input pattern. Thus the proposed algorithm may be expected to be more effective than the on-line BP. Finally, it is worth mentioning that the step-sizes $\omega/(\nu^l \|\nabla \bar{z}_s^p(\boldsymbol{w}^l)\|^2)$ of SPM can be easily calculated because each $\nabla \bar{z}_s^p(\boldsymbol{w}^l)$ in (14), or equivalently (19), has already been obtained to calculate $\nabla e_s^p(\boldsymbol{w}^l)$ in the same way as BP.

5 Simulation results

In this section, we report numerical results with SPM and on-line BP. We used a three layer neural network having M input units, L hidden units and N output units, which will be denoted by M-L-N. We employed the following termination criterion, which is common to both SPM and on-line BP:

$$E(\boldsymbol{w}) \leq \frac{1}{2}((0.005) \times P \times N), \tag{20}$$

where P is the number of input patterns. If the termination criterion (20) is not satisfied within a certain maximum number of iterations, we judged that the learning was unsuccessful. Here, a single iteration means a single sweep of all input patterns. To measure the efficiency of the algorithms, we consider two indices; the rate of success and the average CPU time. The former is the ratio of the successful trials to the whole trials using different initial weights and the latter is the average CPU time of the successful trials.

All computer programs were coded in C and run on a DEC 3000/300. For all problems, we conducted preparatory experiments to find suitable values of the following parameters: the relaxation parameter $\omega \in (0.1, 2.0)$ in SPM, the learning rate $\eta \in (0.1, 4.0)$ in on-line BP and the parameter $\lambda \in (0.2, 20.0)$ in the sigmoid function f given by (3). The results shown below are those for the most appropriate parameter values found in the preparatory experiments. In all cases, initial weights were randomly chosen from the open interval $(-1, 1)$. The two values of training signals, T_u and T_l, were set to be 0.9 and 0.1, respectively.

5.1 Simple test problems

First, we applied both on-line BP and SPM to two test problems, which we call Problem A and Problem B. Problem A is to realize the relation given by

$$\begin{aligned} t_1 &= (x_1 + x_2 + x_3) \bmod 2, \\ t_2 &= x_1 \wedge x_3, \\ t_3 &= x_1 \vee x_3, \end{aligned}$$

where $\boldsymbol{t} = (t_1, t_2, t_3)^T \in \{0, 1\}^3$and $\boldsymbol{x} = (x_1, x_2, x_3)^T \in \{0, 1\}^3$ represent the training signals and input patterns, respectively. Thus the number of training signals is $2^3 = 8$. Problem B is to realize the relation shown in Table 1. Note that the input patterns in Problem A are binary while those in Problem B are real-valued, though the training signals are binary in both problems. We use a 3-3-3 NN for Problem A and a 4-4-2 NN for Problem B. The results for both problems are summarized in Table 2, where "CPU sec"and "# Iterations" respectively mean the average CPU time in second and the average number of iterations for 20 trials using different initial weights. For both problems, the maximum number of iterations was set to be 1000. Table 2 indicates that for both problems, SPM requires less CPU time and iterations than on-line BP.

5.2 Pattern classification problems

Next we applied on-line BP and SPM to pattern classification problems. Classification means dividing a set of given sample patterns $\boldsymbol{x}^p \in R^M$, $p \in \{1, \cdots, P\}$ into D disjoint classes $\Omega_1, \cdots, \Omega_D$. We generated 5 problems of various size as shown in Table 3. In each problem, sample patterns $\boldsymbol{x}^p, p \in \{1, \cdots, P\}$ are distributed in the box with each side $(-100, 100)$. For example, sample (input) patterns for Problem No.5 are illustrated in Fig.6. The neural network used for each problem is also shown in Table 3. For example, we used a 2-20-2 NN for Problem No.1. Here the number of hidden units for each NN was set to be enough large, which were obtained in preparatory experiments. The simulation results are shown in Table 4, where the numbers are the average taken for 5 trials using different initial weights, all of which resulted in successful learning. From Table 4, we see that SPM outperformed on-line BP for all problems except Problem No.1. In particular, the results for Problem Nos.1–3 suggest that, as the dimension M of input patterns becomes large, SPM tends to be faster than on-line BP. Moreover, even if the size of the used NN is large, like the one for Problem No.5, SPM spent less CPU time than on-line BP.

6 Conclusion

In this paper, we have proposed a new learning algorithm based on the successive projection method, in which the step-size is determined adaptively based on the output for each input pattern. Through computational experiments, we compared SPM with on-line BP. The simulation results show that SPM usually requires less CPU time than the standard on-line BP, in spite of the fact that SPM spends more CPU time per iteration than on-line BP in calculating the step-size. This indicates that the adaptive choice of the step-size in SPM works quite effectively.

We have mainly focused our attention to classification-type problems, which can be formulated as a system of inequalities (cf. (11)). However, SPM can also be applied to an approximate realization of a mapping, which may be represented as a system of equalities (cf. (9)), for which the error functions are given by (4) and (5) instead of (15) and (16). In this case, the update formula (14) should be replaced by

$$\boldsymbol{w}^{l+1} = \boldsymbol{w}^l - \frac{\omega}{N} \sum_{s=1}^{N} \frac{z_s^p(\boldsymbol{w}^l) - t_s^p}{\|\nabla z_s^p(\boldsymbol{w}^l)\|^2} \nabla z_s^p(\boldsymbol{w}^l).$$

In the special case where $N = 1$, i.e., the output layer has only one unit, the equation-based SPM has been discussed in [6, pp.181-188]. Local convergence of such SPM may be guaranteed by the convergence theorem established in [9]. This suggests that local convergence of general SPM as proposed in this paper may still be ensured under mild assumptions. However, global convergence of SPM seems to have many factors which have not elucidated yet and demands further investigation. Finally it would be worth mentioning the recent work [7, 8, 11] on the theoretical convergence analysis of error back-propagation algorithms.

Appendix

A projection onto a half space

The problem of finding the orthogonal projection of $\boldsymbol{w}^l$ onto the half space determined by (12) can be formulated as the convex quadratic programming

$$\begin{array}{ll} \min & \frac{1}{2}\|\boldsymbol{w}-\boldsymbol{w}^l\|^2 \\ \text{s.t.} & \bar{z}^p(\boldsymbol{w}^l)+\nabla\bar{z}^p(\boldsymbol{w}^l)^T(\boldsymbol{w}-\boldsymbol{w}^l)\geq\bar{t}^p. \end{array} \tag{21}$$

The Kuhn-Tucker conditions [2] for this problem can be written

$$\boldsymbol{w}-\boldsymbol{w}^l-r\nabla\bar{z}^p(\boldsymbol{w}^l)=\mathbf{o}, \tag{22}$$

$$\bar{z}^p(\boldsymbol{w}^l)+\nabla\bar{z}^p(\boldsymbol{w}^l)^T(\boldsymbol{w}-\boldsymbol{w}^l)-\bar{t}^p\geq 0, \tag{23}$$

$$r\geq 0, \tag{24}$$

$$r\cdot(\bar{z}^p(\boldsymbol{w}^l)+\nabla\bar{z}^p(\boldsymbol{w}^l)^T(\boldsymbol{w}-\boldsymbol{w}^l)-\bar{t}^p)=0, \tag{25}$$

where r is a Lagrange multiplier.

To find $\boldsymbol{w}$ and r satisfying (22)-(25), we consider two cases, $r>0$ and $r=0$. If $r>0$, it follows from (25) that

$$\bar{z}^p(\boldsymbol{w}^l)+\nabla\bar{z}^p(\boldsymbol{w}^l)^T(\boldsymbol{w}-\boldsymbol{w}^l)-\bar{t}^p=0. \tag{26}$$

From (26) and (22), we have

$$r=-\frac{\bar{z}^p(\boldsymbol{w}^l)-\bar{t}^p}{||\nabla\bar{z}^p(\boldsymbol{w}^l)||^2}. \tag{27}$$

Substituting this r in (22), we get

$$\boldsymbol{w}=\boldsymbol{w}^l-\frac{\bar{z}^p(\boldsymbol{w}^l)-\bar{t}^p}{||\nabla\bar{z}^p(\boldsymbol{w}^l)||^2}\nabla\bar{z}^p(\boldsymbol{w}^l). \tag{28}$$

Note that, by $r>0$ and (27), we must have

$$\bar{z}^p(\boldsymbol{w}^l)-\bar{t}^p<0. \tag{29}$$

On the other hand, if $r=0$, it follows from (22) that

$$\boldsymbol{w}=\boldsymbol{w}^l. \tag{30}$$

This together with (23) implies that

$$\bar{z}^p(\boldsymbol{w}^l)-\bar{t}^p\geq 0.$$

Hence, it follows from (28)-(30) that the solution of the quadratic programming problem (21) can be represented as

$$\boldsymbol{w} = \boldsymbol{w}^l - \frac{\min\,[\bar{z}^p(\boldsymbol{w}^l) - \bar{t}^p,\ 0\,]}{||\nabla\,\bar{z}^p(\boldsymbol{w}^l)||^2}\nabla\,\bar{z}^p(\boldsymbol{w}^l).$$

References

[1] R. Battiti, "First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method," *Neural Computation*, vol. 4, no. 2, pp. 141-166, 1992.

[2] M. S. Bazaraa, H. D. Sherali and C. M. Shetty, *Nonlinear Programming : Theory and Algorithms (Second Edition)*, New York : John Wiley & Sons, 1993.

[3] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, West Sussex : John Wiley & Sons, 1993.

[4] S. Ergezinger and E. Thomsen, "An Accelerated Learning Algorithm for Multilayer Perceptrons: Optimization Layer by Layer," *IEEE Trans. Neural Networks*, vol. 6, no. 1, pp. 31-42, 1995.

[5] J. Hertz, A. Krogh and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Redwood City : Addison-Wesley, 1991.

[6] T. Ibaraki and M. Fukushima, *Methods of Optimization* (in Japanese), Tokyo : Kyoritsu Shuppan, 1993.

[7] Z. Q. Luo and P. Tseng, "Analysis of an Approximate Gradient Projection Method with Applications to the Back Propagation Algorithm," Working Paper, Dept. of Electrical and Computer Engineering, McMaster University, Hamilton, August, 1993.

[8] O. L. Mangasarian and M. V. Solodov, "Serial and Parallel Backpropagation Convergence via Nonmonotone Perturbed Minimization," Technical Report #1149, Computer Sciences Dept., University of Wisconsin, Madison, April, 1993.

[9] S. F. McCormick, "The Methods of Kaczmarz and Row Orthogonalization for Solving Linear Equations and Least Squares Problems in Hilbert Space," *Indiana University Mathematical Journal*, vol. 26, no. 6, pp. 1137-1150, 1977.

[10] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing*, D. E. Rumelhart, J. L. McClelland and the PDP Research Group ed., Cambridge, MA : MIT Press, 1986, ch. 8, pp. 318-362.

[11] M. V. Solodov and S. K. Zavriev, "Stability Properties of the Gradient Projection Method with Applications to the Backpropagation Algorithm," Technical Report 94-05, Computer Sciences Dept., University of Wisconsin, Madison, June, 1994.

[12] H. Takagi, S. Sakagami and H. Togawa, "Evaluation of Nonlinear Optimization Methods for the Learning Algorithm of Artificial Neural Networks" (in Japanese), *Trans. IEICE*, D-II, vol. J74-D-II, no.4, pp. 528-535, 1991.

[13] K. Tatsumi and M. Fukushima, "A Backpropagation Algorithm as a Successive Projection Method"(in Japanese), *Trans. ISCIE*, to appear.

Table 1: Data for test problem B

Pattern No.	Input patterns	Training signals
1	$(-0.6999, 55.5742, -0.5560, -0.4991)$	(1, 1)
2	$(8.8891, -0.0467, 0.4607, -0.4149)$	(0, 0)
3	$(9.8389, -0.9782, 4.1380, -0.7414)$	(1, 1)
4	$(9.0640, -0.2813, -0.3456, 5.1668$	(0, 1)
5	$(-0.2130, 73.2292, 3.1101, -0.7203)$	(1, 1)
6	$(7.4477, 42.2620, 0.0860, 7.9690)$	(0, 0)

Table 2: Results for test problems A and B

Problem		CPU sec	#Iterations	Rate of success
Problem A	on-line BP	0.073	188.7	20/20
	SPM	0.043	72.0	19/20
Problem B	on-line BP	0.195	537.4	20/20
	SPM	0.047	98.4	18/20

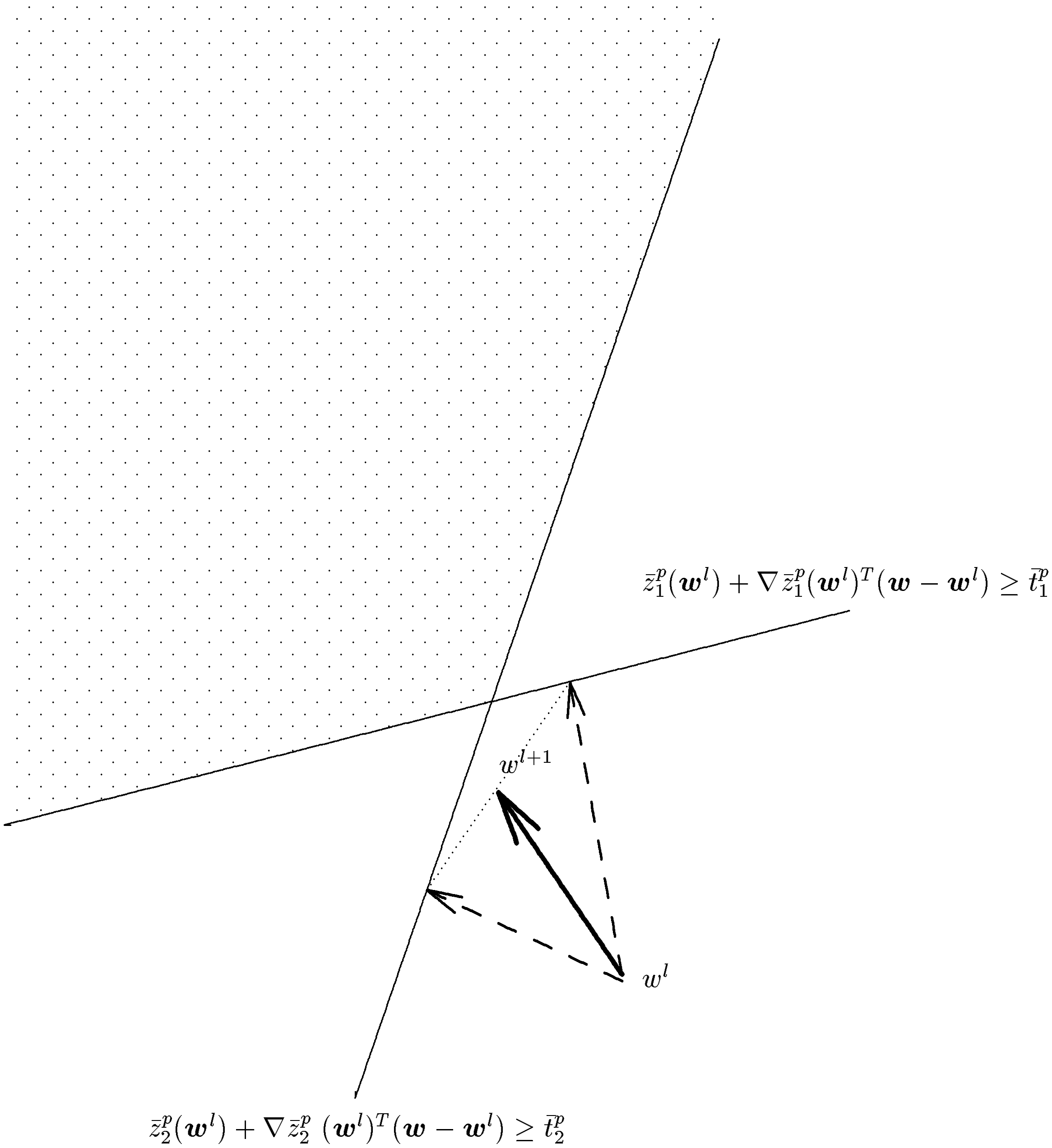
Table 3: Data of pattern classification problems

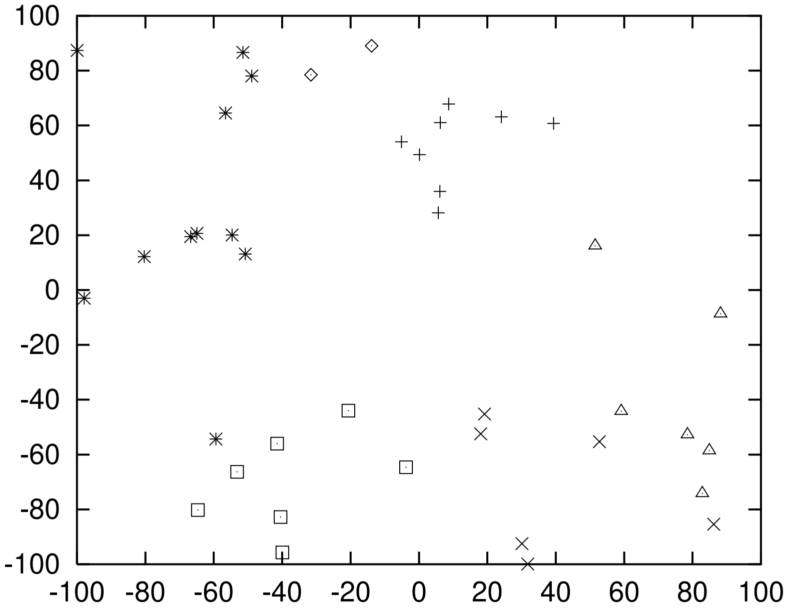
Problem No.	#Classes	NN	#Patterns
1	3	2-20-2	40
2	3	3-20-2	40
3	3	6-40-2	40
4	4	3-14-2	40
5	6	2-50-6	40

Table 4: Results for pattern classification problems

Problem No.		CPU sec	#Iterations	CPU sec per iteration
1	on-line BP	7.16	996.6	0.0072
	SPM	7.37	832.4	0.0089
2	on-line BP	2.61	323.8	0.0080
	SPM	1.82	170.8	0.0107
3	on-line BP	0.597	97.6	0.0061
	SPM	0.210	26.0	0.0080
4	on-line BP	5.56	570.4	0.0097
	SPM	1.73	145.0	0.012
5	on -line BP	1.76	51.0	0.035
	SPM	0.77	14.8	0.052

List of Figures


$\bar{z}_1^p(\boldsymbol{w}^l) + \nabla \bar{z}_1^p(\boldsymbol{w}^l)^T(\boldsymbol{w} - \boldsymbol{w}^l) \geq \bar{t}_1^p$
w^{l+1}
w^l
$\bar{z}_2^p(\boldsymbol{w}^l) + \nabla \bar{z}_2^p\ (\boldsymbol{w}^l)^T(\boldsymbol{w} - \boldsymbol{w}^l) \geq \bar{t}_2^p$


100
80
60
40
20
0
-20
-40
-60
-80
-100
-100 -80 -60 -40 -20 0 20 40 60 80 100