Programming Based Learning Algorithms of Neural Networks with Self-Feedback Connections

Bo Zhang, Ling Zhang, and Fuchao Wu

***Abstract*— In this paper, we discuss the learning problem of neural networks with self-feedback connections and show that when the neural network is used as associative memory, the learning problem can be transformed into some sort of programming (optimization) problem. Thus, the rather mature optimization technique in programming mathematics can be used for solving the learning problem of neural networks with self-feedback connections. Two learning algorithms based on programming technique are presented. Their complexity is just polynomial. Then, the optimization of the attractive radius of the training samples is discussed using quadratic programming techniques and the corresponding algorithm is given. Finally, the comparison is made between the given learning algorithm and some other known algorithms.**

I. Introduction

The learning problem of a neural network with self-feedback connections implies that finding the weights (w_{ij}) and threshold (θ_i) of each element such that the network has some given properties. When a neural network is used for associative memory, given a set $K = \{s^\alpha, \alpha = 1, 2, \cdots, m\}$ of training samples, we may have each training sample be a stable set of a network N. Therefore, the simplest learning problem is the following.

If a network with self-feedback connections and a set $K = \{s^\alpha\}$ of training samples are given, then we determine the weights and threshold of each element in the network such that each training sample s^α is one of its stable states. And we hope that the attractive radius (see below for its definition) of each training sample is as large as possible.

Many learning rules have been presented (see [1]–[3]). The main shortcoming of these algorithms is that the performance of the networks constructed from them is not satisfactory. For example, it is difficult to have all training examples be stable states of the network or it may be that the memory capacity of the network is quite low.

In this paper, based on the optimization technique in programming mathematics, some performance of a neural network is regarded as an objective function. Then, the optimization technique is used for seeking the optimal solution. Therefore, we can create high performance networks.

We use the well-known Khachiyan's ellipsoid method [4], [5] and Potential Reduction Algorithm (PRA) [6] in quadratic programming for solving the learning problem above and have two learning algorithms whose complexity is polynomial.

Manuscript received March 15, 1993; revised December 7, 1993. This work was supported in part by National Key Project of Fundamental Research, Climbing Program.

Bo Zhang is with the Department of Computer Science, Tsinghua University, Beijing, China.

Ling Zhang is with the Departments of Computer Science at Anhui University, Hefei, Anhui; and Zhejiang University, Hangzhou, Zhejiang, both in China.

Fuchao Wu is with the Department of Computer Science, Anhui University, Hefei, Anhui, China.

IEEE Log Number 9402002.

II. Learning Problem

A. Learning Problem

A weight-sum-and-threshold neural network with self-feedback connections is given. If $x(t)$ is its input vector at moment t, the operation process of the network is described by

$$x_i(t+1) = \sigma\left(\sum_{j=1}^{n} w_{ij}x_j(t) - \theta_i\right), \qquad i = 1, 2, \cdots, n \tag{1}$$

where

$$\sigma(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

1) w_{ij} is the weight of the jth input of the ith element,
2) θ_i is the threshold of the ith element.

To have each training sample s^α be a stable state of the network, the necessary and sufficient condition is

$$\forall_i \quad \begin{cases} \sum_{j=1}^{n}(w_{ij}s_j^\alpha - \theta_i)s_i^\alpha \geq 0, & s_i^\alpha = 1 \\ \sum_{j=1}^{n}(w_{ij}s_j^\alpha - \theta_i)s_i^\alpha > 0, & s_i^\alpha = -1 \end{cases}. \tag{2}$$

Let

$$w(i) = (w_{i0}, w_{i1}, \cdots, w_{in})^T \qquad w_{i0} = \theta_i$$
$$s^\alpha = (s_0^\alpha, s_1^\alpha, \cdots, s_n^\alpha), \qquad s_0^\alpha = -1$$

$$c^\alpha(i) = s_i^\alpha s^\alpha$$
$$C(i) = \begin{pmatrix} c^1(i) \\ \vdots \\ c^m(i). \end{pmatrix}$$

Then, the learning problem of the network can be equivalently transferred into that of finding the solutions for the following inequality system

$$\forall i \qquad C(i)x \leq 0 \quad (\text{or } < 0) \tag{3}$$

where $x = (x_0, x_1, \cdots, x_n)^T$.

The coefficients are real numbers.

Since $e_i = (1, 1, \cdots, 1)^T$ is the ith column vector of $C(i)$, $x(i) = (0, \cdots, 0, 1, 0, \cdots, 0)$, where only the value of its ith component is one, all others are zero, must be the solution of formula (3). That is, when $W = I$, an identity matrix, all states in the network are stable. This is a useless, trivial solution. To avoid the solution, the ith column vector of $C(i)$ is deleted and the ith component of x is omitted accordingly. For simplicity, the same notion $C(i)$ is used for representing the revised matrix of $C(i)$ in the following discussion.

There is no loss of generality in assuming that $x = (x_1, \cdots, x_n)^T$ in the following discussion.

B. Ellipsoid Learning Algorithm

From the specific structure of $C(i)$, when applying the ellipsoid method to formula (3), the complexity $O(n^4)$ of the method can be reduced to $O(n^2 \ln n)$.

Ellipsoid Method: Now we find the solutions of an inequality system, form (4), by the ellipsoid method [1]. The procedure is stated as follows

$$Cx \le 0 \quad (\text{or } < 0) \tag{4}$$

1) Initially, set $k = 0$, $N = 2n^2 \ln 2n$, $A_0 = nI$, where I is a $n \times n$ identity matrix and $a_0 = 0$. Let $E(A, a) = \{x \in R^n | (x-a)^T A^{-1}(x-a) \le 1\}$, where A is a positive definite $n \times n$-matrix. $E(A, a)$ is an ellipsoid with vector a as its center in the vector space R^n.
2) In the kth step,
2.1) If $k = N$, stop with no solution.
2.2) If a_k, a column vector, is a solution of form (4), stop. A feasible solution is found.
2.3) If a_k does not satisfy form (4), then choose an inequality, say $c^T x \le 0$, of the system $C\,x \le 0$ that is violated by a_k. That is $c^T a_k \ge 0$ (or > 0).
3) Set

$$b = \frac{1}{\sqrt{c^T A_k c}} A_k c$$

where c^T is the transition matrix c, $a_{k+1} = a_k - (b/n+1)$, $A_{k+1} = (n^2/n^2 - 1)(A_k - (2/n+1)bb^T)$ and go to (2).

For each i, if the corresponding formula (3) has no solution or zero solution, then

$$w_{ij} = 0, \qquad i \ne j \quad \text{and} \quad w_{ii} = 1.$$

If the corresponding formula (3) has solution $x(i)$ and $|x(i)| \ne 0$, then

$$w_{ij} = -x_j(i), \qquad i \ne j \quad \text{and} \quad w_{ii} = a_i \ge 0$$

where a_i is an arbitrary number, $0 \le a_i \le 1$.

Finally, we have a matrix $W = (w(1), w(2), \cdots, w(n))$, where $w(i) = (w_{i1}, w_{i2}, \cdots w_{in})$.

C. The Complexity of the Ellipsoid Learning Algorithm

Since form (3) is homogeneous, the solution of x may be restricted within certain range, say $|x_j| \le 1$, $j = 1, 2, \cdots, n$.

Let $B = \{x | |x_j| \le 1,\ j = 1, 2, \cdots, n\}$.

Then, formula (3) becomes

$$C(i)x \le 0 \quad (\text{or } < 0)$$
$$|x_j| \le 1, \qquad j = 1, 2, \cdots, n. \tag{5}$$

Lemma 1: The solution set of formula (5) is convex and every component of its vertices must be 1, -1 or 0.

Proof: Obviously, the solution set of formula (5) is convex.

Assume that $c^\alpha(i)$ is a row vector of $C(i)$. $c^\alpha(i)x = 0$ defines a hyperplane P_α. The vertex of the solution set must be the intersection between several hyperplanes P'_αs and several planes of B. Thus, it is only necessary to show that every component of the coordinates of the intersecting point between $k\,P'_\alpha$s and B must be 1, -1, or 0, where $0 \le k \le n$.

When $k = 0$, the intersecting point is just the vertex of B. Obviously, the requirement is satisfied.

When $k = 1$, assume that the equation of P_α is $c^1(i)x = 0$ and x^0 is the vertex of $P_\alpha \cap B$. If one of the components of x^0, say x_1^0, is not 1, -1, or 0, by assuming $c_1^1(i)x_1^0 = a > 0$, there must exist another component of x^0, say x_2^0, such that $c_2^1(i)x_2^0 = b$, $b \ne 1, -1$, or 0.

If $c_2^1(i)$ and $c_1^1(i)$ have contrary sign, (or the same sign), set

$$x_1^1 = x_1^0 + \epsilon, \quad x_1^2 = x_1^0 - \epsilon; \quad x_2^1 = x_2^0 + \epsilon, \quad x_2^2 = x_2^0 - \epsilon$$

$$(x_2^1 = x_2^0 - \epsilon, \quad x_2^2 = x_2^0 + \epsilon) \quad \text{and} \quad |a \pm \epsilon| \le 1, \quad |b \pm \epsilon| \le 1.$$

For $i > 2$, letting $x_i^1 = x_i^2 = x_i^0$, we have

$$x^0 = \frac{x^1 + x^2}{2}.$$

Obviously, x^1 and $x^2 \in B$. We show that x^1 and $x^2 \in P_\alpha$. From

$$\begin{aligned} c^1 x^1 &= \sum_{j=1}^n c_j^1 x_j^1 = c_1^1 x_1^1 + c_2^1 x_2^1 + \sum_{j=3}^n c_j^1 x_j^1 \\ &= c_1^1(x_1^0 + \epsilon) + c_2^1(x_2^0 + \epsilon) + \sum_{j=3}^n c_j^1 x_j^0 \\ &= c_1^1 \epsilon + c_2^1 \epsilon + \sum_{j=1}^n c_j^1 x_j^0 = 0, \qquad x^1 \in P_\alpha. \end{aligned}$$

Similarly, $x^2 \in P_\alpha$.

Therefore, x^1 and $x^2 \in B \cap P_\alpha$. x^0 is not a vertex of $B \cap P_\alpha$. This is a contradiction. For $k = 1$, the lemma holds.

When $k = 2$, assume that the equations of P_1 and P_2 are $c^1(i)x = 0$ and $c^2(i)x = 0$, respectively.

Assume $x \in P_1 \cap P_2$. A set consisting of the components that are the same in both vectors $c^1(1)$ and $c^2(1)$ is denoted by I, otherwise denoted by J. Therefore, vectors $c^1(i)$ and $c^2(i)$ can be written as

$$c^1(i) = (c_I^1(i), c_J^1(i)), \quad c^2(i) = (c_I^2(i), c_J^2(i))$$

where $c_I^1(i)$ is a subvector consisting of the components of $c^1(i)$, which are the same as that of $c^2(i)$. $c_J^1(i)$ is a subvector consisting of the components of $c^1(i)$, which are different from that of $c^2(i)$.

Similarly, $x = (x_I, x_J)^T$

$$\begin{cases} c^1(i)x = (c_I^1(i)x_I) + (c_J^1(i)x_J) = 0 \\ c^2(i)x = (c_I^2(i)x_I) + (c_J^2(i)x_J) = 0. \end{cases} \tag{6}$$

Then, we have

$$c_I^1(i)x_I = 0, \quad c_J^1(i)x_J = 0. \tag{7}$$

In a similar way to $k = 1$, we have the vertices of $P_1 \cap P_2 \cap B$ and the components of their $x_I(x_J)$ must be 1, -1 or 0.

By induction, Lemma 1 is proved.

Lemma 2: If Q is the solution set of formula (5) and is n-dimensional, then $|Q| \ge 1/n!$, where $|Q|$ is the volume of Q.

Proof: If Q if n-dimensional, then Q is nonempty. From Lemma 1, the components of its vertices must be 1, -1, or 0. Q contains an n-dimensional simplex S at least. Assume that the vertices of S are $v_0, v_1, \cdots, v_n$. Thus

$$|S| = \frac{1}{n!}\left| \det \begin{pmatrix} 1 & 1 & \cdots & 1 \\ v_0 & v_1 & \cdots & v_n \end{pmatrix} \right|.$$

From Lemma 1, the components of v_i are 1, -1, or 0. The value of the above determinant is integral and ≥ 1, since S is n-dimensional.

Finally, $|S| \ge 1/n!$, $|Q| \ge |S| \ge 1/n!$.

D. The Complexity and Completeness of the Algorithms

At the beginning of the above procedure, assume that the first ellipsoid V_n has center $a_0 = (0, 0, \cdots, 0)$ and radius $\sqrt{n}$. Obviously, V_n contains B

$$|V_n| = \frac{1}{\Gamma\left(\frac{n}{2}+1\right)}(\pi n)^{n/2} \sim \frac{1}{\sqrt{\pi n}}\left(2e\frac{\pi}{n}\right)^{n/2} \cdot n^{n/2} = \frac{1}{\sqrt{\pi n}}(2e\pi)^{n/2}.$$

Since

$$|Q| \geq \frac{1}{n!} \sim \frac{1}{\sqrt{2\pi n}}\left(\frac{e}{n}\right)^n$$

$$\frac{|Q|}{|V_n|} \geq \frac{1}{\sqrt{2\pi n}}\left(\frac{e}{n}\right)^n \sqrt{\pi n}\left(\frac{1}{2e\pi}\right)^{n/2} = \frac{1}{\sqrt{2}}\left(\sqrt{\frac{e}{2\pi}}\cdot\frac{1}{n}\right)^n. \quad (8)$$

On the other hand, for each step of iterations in the ellipsoid method, the volume of the next ellipsoid shrinks at least with the factor $e^{-1/2n}$ [1]. After N steps of iterations, the total shrinking rate of the volumes is $< e^{-N/2n}$. When N is large enough such that $e^{-N/2n}$ is less than $|Q|/|V_n|$, then there is no solution if after N steps of iterations of feasible solution can not be found by the method.

From formula (8), to have N large enough, we need only choose $N/2n > \ln(\sqrt{2}(an)^n)$, where $a = \sqrt{2\pi/e}$. Then, we have $N > 2n(n \ln an + \ln\sqrt{2})$. In the ellipsoid method above, we choose $N = 2n^2 \ln 2n$. Since $2n^2 \ln 2n > 2n(n \ln an + \ln\sqrt{2})$ when $n \geq 2$, thus the method is complete.

For each step of iterations in the ellipsoid method, it is necessary to check at most mn-variable linear equations for each element of a_k. Its complexity is $O(mn)$.

For each element i, in N steps of iterations, the complexity of the ellipsoid learning algorithm is

$$O(mn(n^2 \ln n)) = O(mn^3 \ln n).$$

Therefore, we have the following theorem.

Theorem 1: The ellipsoid learning algorithm is complete and its complexity is $O(mn^4 \ln n)$, where n is the number of elements and m is the number of training samples.

Proof: Since for each element i, $i = 1, 2, \cdots, n$, the complexity of solving formula (3) is $O(mn^3 \ln n)$, the complexity of the ellipsoid learning algorithm is $O(mn^4 \ln n)$.

For each i, if formula (3) has a solution denoted by x^i, by letting $w(i) = -x^i/|x^i|$, we have normalized weights and thresholds of neural networks. In the following discussion, we only use the normalized weights and thresholds.

III. Estimating the Attractive Radii of Training Samples

Assume that $H(x, y)$ is the Hamming distance between x and y. s^α is a stable state. Let $V(s^\alpha, r) = \{x | H(s^\alpha, x) \leq r\}$.

Definition: $R(s^\alpha) = \max_r\{r | \forall x \in V(s^\alpha, r), x$ must converge to $s^\alpha\}$, $R(s^\alpha)$ is said to be the attractive radius of s^α.

In the following discussion, $\langle x, y\rangle$ denotes the inner product of x and y.

Given α and i, let $|\langle w(i), s^\alpha\rangle| = F(\alpha, i)$.

Let $I = \{j | w_j(i)s_j^\alpha > 0\}$ and $J = \{j | w_j(i)s_j^\alpha < 0\}$.

When $s_i^\alpha > 0$ ($s_i^\alpha < 0$), we arrange $\{|w_j(i)|, j \in I\}$ ($\{|w_j(i)|, j \in J\}$) as a sequence from big to small denoted by

$$u(\alpha, i, 1), u(\alpha, i, 2), \cdots.$$

Let $d(\alpha, i)$ be integer p that satisfies the following condition and $u(\alpha, i, 0) = 0$

$$2\sum_{j=0}^{p} u(\alpha, i, j) \leq F(\alpha, i) \quad \text{and} \quad 2\sum_{j=0}^{p+1} u(\alpha, i, j) > F(\alpha, i)$$

$$\left(2\sum_{j=0}^{p} u(\alpha, i, j) < F(\alpha, i) \quad \text{and} \quad 2\sum_{j=0}^{p+1} u(\alpha, i, j) \geq F(\alpha, i)\right).$$

The meaning of $d(\alpha, i)$ is that for the ith element when the Hamming distance $H(x, s^\alpha)$ between input x and s^α is less than and equal to $d(\alpha, i)$, $H(x, s^\alpha) \leq d(\alpha, i)$, the output of ith element is still equal to s_i^α. Since the difference of $\langle w(i), x\rangle$ from $\langle w(i), s^\alpha\rangle$ is $2\sum_{j=0}^{p} u(\alpha, i, j)$ at most, $\langle w(i), x\rangle \geq 0$, when $s_i^\alpha > 0$.

Now, α is fixed. For each i, we obtain $d(\alpha, i)$, $i = 1, 2, \cdots$. We also arrange $d(\alpha, i)$ as a sequence from small to big denoted by

$$d(\alpha, 1), d(\alpha, 2), \cdots.$$

Let $\rho(\alpha)$ be integer r that satisfies the following condition.

When $i \leq r$, $d(\alpha, i) \geq i$ and $d(\alpha, r+1) < r+1$.

Now, we have the following theorem.

Theorem 2: The attractive radius of training sample $s^\alpha \geq \rho(\alpha)$.

Proof: Assume $x \in V(s^\alpha, \rho(\alpha))$ and $H(x, s^\alpha) = k$, $0 < k \leq \rho(\alpha)$.

From the definition of $\rho(\alpha)$, when $i \geq k$, we have

$$d(\alpha, i) \geq d(\alpha, k) \geq k.$$

Let $M = \{j | x_j \neq s_j^\alpha\}$ and $x_0 = -1$.

If $s_i^\alpha > 0$, we have

$$\sum_{j=0}^{n} w_{ij}x_j = \sum_{j=0}^{n} w_{ij}s_j^\alpha + \sum_{j=0}^{n} w_{ij}(x_j - s_j^\alpha)$$
$$\geq F(\alpha, i) - 2\sum_{j \in M \cap I} |w_{ij}| \geq F(\alpha, i) - 2\sum_{j=1}^{k} u(\alpha, i, j) \geq 0$$

where $I = \{j | w_j(i)s_i^\alpha > 0\}$.

If $s_i^\alpha < 0$, we have

$$\sum w_{ij}x_j < 0.$$

In other words, when $i \geq k$, the ith coordinate y_i of y, the output of x, is equal to that of s_i^α. Therefore, we have

$$H(y, s^\alpha) <= n - (n - k + 1) = k - 1.$$

That is, when $x \in V(s^\alpha, \rho(\alpha))$ $(x \neq s^\alpha)$, its Hamming distance from s^α will be decreased by one after a self-feedback cycle of the network. Element x must converge to s^α at most in $\rho(\alpha)$ times of feedback cycles. This proves our claim.

We show an example as follows.

Assume

$$s^\alpha = (1, 1, 1, 1, 1, 1, 1, 1)$$
$$w(1) = (1, 1, 1, 1, 1, -1, -1, -1)$$
$$w(2) = (1, 1, 1, 1, 1, 1, -1, -1)$$
$$w(3) = \cdots = w(8) = (1, 1, 1, 1, 1, 1, 1, -1).$$

For $i = 1$, a sequence of $u(\alpha, 1, j)$ is 1, 1, 1, 1, 1, $F(\alpha, i) = 2$. Thus, $d(\alpha, 1) = 1$.

For $i = 2$, a sequence of $u(\alpha, 2, j)$ is 1, 1, 1, 1, 1, 1, $F(\alpha, i) = 4$.

Thus, $d(\alpha, 2) = 2$.

For $i = 3, \cdots, 8$, a sequence of $u(\alpha, i, j)$ is 1, 1, 1, 1, 1, 1, 1, $F(\alpha, i) = 6$.

Thus, $d(\alpha, i) = 3, i \geq 3$.

We have a sequence of $d(\alpha, i)$

1, 2, 3, 3, $\cdots$, 3.

Since $d(\alpha, 1) = 1 \geq 1$, $d(\alpha, 2) = 2 \geq 2$, $d(\alpha, 3) = 3 \geq 3$, $d(\alpha, 4) = 3 < 4$, we have $\rho(\alpha) = 3$.

The lower bound of attractive radius of s^α is 3.

If $x(0) = (-1, -1, -1, 1, 1, 1, 1, 1)$ is a given initial state, it is easy to know that $\langle w(1), x(0)\rangle = -4$, $\langle w(2), x(0)\rangle = -2$, $\langle w(i), x(0)\rangle = 0$, $i \geq 3$, and $x(1) = y(1) = (-1, -1, 1, 1, 1, 1, 1, 1)$. Similarly, $x(2) = y(2) = (-1, 1, 1, 1, 1, 1, 1, 1)$ and $x(3) = y(3) = (1, 1, 1, 1, 1, 1, 1, 1) = s^\alpha$. Therefore, $x(0)$ converges to s^α in three steps. If $x(0) = (1, 1, 1, 1, 1, -1, -1, -1)$, then $x(1) = y(1) = s^\alpha$, it converges to s^α only in one step.

IV. The Optimization of Attractive Radii

From the previous section, it is known that the larger the $F(\alpha, i)$, the larger the attractive radius. Let $F(\alpha) = \max_{w(i)} F(\alpha, i)$. This means that we choose a proper weight-threshold such that $F(\alpha)$ is maximal. The problem can be reduced to the following quadratic programming problem.

Constraints: i is fixed

$$C(i)x \leq 0 (< 0)$$

$$|x| = 1. \tag{9}$$

Objective functions: $\max_x \{\min_\alpha |\langle x, c^\alpha(i)\rangle|\}$ where

$$c^\alpha(i) = (s_i^\alpha s^\alpha).$$

We now show that solving the problem (9) is equivalent to the solution of the quadratic programming (10) below.

Assume that $P(i)$ is the convex closure of $\{c^\alpha(i), \alpha = 1, 2, \cdots, m\}$. Construct a quadratic programming problem below.

$$\textit{Constraint:} \quad x \in P(i)$$

$$\textit{Objective:} \quad \min_{x \in P(i)} |x|. \tag{10}$$

Theorem 3: If $x^*(i)$ is a nonzero solution of formula (10), then $w(i) = x^*(i)/|x^*(i)|$ is the solution of formula (9).

Proof: By reduction to absurdity. Assume that x^0 is a solution of form (9) and $x^0 \neq x^*(i)/|x^*(i)|$. Let $|\alpha x^0| = |x^*(i)|$. Through αx^0 construct a hyperplane $\sum$ perpendicular to αx^0, where $\alpha > 0$.

Obviously, $x^*(i)$ and origin O are on the same side of hyperplane $\sum$ denoted by $\sum_-$. Otherwise, $x^*(i)$ and origin O are on the different sides of $\sum$, respectively. Then, the line connecting $x^*(i)$ and the origin must intersect with $\sum$. (Thus there are more than one intersecting points between $\sum$ and the sphere which center is the origin and radius is $|x^*(i)|$.) Since $\sum$ is perpendicular to αx^0, this is a contradiction.

The intersection between $\sum_-$ and $P(i)$ is nonempty. From $C(i)x \leq 0 (< 0)$, there exists a vertex c^1 of $P(i)$ such that c^1 is in $\sum_-$ and $|\langle \alpha x^0, c^1\rangle| < |\alpha x^0| = |x^*(i)|$.

Since $|x^*(i)| = \min_\alpha |\langle x^*(i), c^\alpha(i)\rangle|$, x^0 is not a solution of form (9). This proves our claim.

We use the Potential Reduction Algorithm (PRA) [6] for solving form (10).

First, we rewrite the form (10) as follows

$$\min |x| \leftrightarrow \min x^T x$$

where x is a column vector, $x^T = (x_1, \cdots, x_n)$ a row vector. $x_j = \sum_{\alpha=1}^m \lambda_\alpha c_j^\alpha$ is represented by vector form as follows

$$x = C^T(i)\lambda, \qquad \text{where } C(i) = \begin{pmatrix} c_1^1, & c_2^1, & \cdots & c_n^1 \\ \cdot & \cdot & \cdot & \cdot \\ c_1^m, & c_2^m, & \cdots & c_n^m \end{pmatrix},$$

$$\lambda = (\lambda_1, \cdots, \lambda_m)^T.$$

Thus, $x^T x = (\lambda^T C(i))^T(\lambda^T C(i)) = C^T(i)\lambda\lambda^T C(i) = \lambda^T C(i)C^T(i)\lambda = \lambda^T Q\lambda$ where $Q = C(i)C^T(i)$ is a semi-positive definite symmetric matrix.

Form (10) is rewritten as follows

$$\textit{Constraint:} \quad R = \left\{\lambda_\alpha \geq 0, \ \sum_{\alpha=1}^m \lambda_\alpha = 1\right\}$$

$$\textit{Objective:} \quad \min_{\lambda \in R} \lambda^T Q\lambda. \tag{11}$$

Form (11) is just the form (2.1) in [6] for solving convex quadratic knapsack problems so we can directly use the PRA algorithm.

For simplicity, we use x instead of λ in the following algorithm.

PRA Algorithm: Given $e^T x^0 = 1$, $x^0 > 0$ and $s^0 = 2Qx^0 - y^0 e > 0$, where e is a unit vector and initially $x^0 = e/n$.

1) Let $\Delta^0 = (x^0)^T s^0$
2) For $k = 0$, if $(x^k)^T s^k \geq 2^{-L}$, where L is the size of the problem, then
3) Begin
   a) Compute Δy and Δx according to forms (12) and (13), respectively. From form (14), we have $\overline{\theta}$

$$\begin{aligned} \text{Let} x^{k+1} &= x^k + \overline{\theta}\Delta x \\ y^{k+1} &= y^k + \overline{\theta}\Delta y \\ s^{k+1} &= 2Qx^{k+1} - y^{k+1}e \\ \Delta^{k+1} &= (x^{k+1})^T s^{k+1} \end{aligned}$$

$$k \leftarrow k + 1$$

4) End
5) Where

$$\Delta y = \frac{-e^T V\eta}{e^T Ve} \tag{12}$$

$$\Delta x = \Delta y Ve + V\eta \tag{13}$$

$$\overline{\theta} = \frac{\frac{1}{4}\sqrt{3}}{|(X^k S^k)^{-0.5}| \left|(X^k S^k)^{-0.5}\left(\frac{\Delta^k}{\rho}e - X^k S^k e\right)\right|} \tag{14}$$

$$\text{where} (X^k S^k)^{-0.5} = diag\left(\frac{1}{\sqrt{s_1^i s_1^k}}, \cdots, \frac{1}{\sqrt{x_n^k s_n^k}}\right),$$

$$\Delta^k = (x^k)^T s^k$$

$$\frac{1}{|(X^k S^k)^{-0.5}|} = \min_{1 \leq j \leq n} \sqrt{x_j^k s_j^k}$$

6) and

$$\eta = \left(\frac{\Delta^k}{\rho}\right)(X^k)^{-1}e - S^k e$$

$$\begin{aligned} V &= [S^k(X^k)^{-1} + 2Q]^{-1} \\ &= (S^k)^{-1}X^k - (S^k)^{-1}X^k R[I + R^T(S^k)^{-1}X^k R]^{-1}R^T(S^k)^{-1}X^k \end{aligned}$$

$$X^k = diag(x^k), \quad S^k = diag(s^k), \quad R = \sqrt{2}C(i)$$

$$\rho \geq n + \sqrt{n}.$$

Theorem 4: Using PRA for solving form (11), there are $O((p-n)L)$ iterations at most and each iteration has $O(m^2n)$ arithmetic operations at most.

Proof: The theorem can directly be obtained from theorem 2.2 in [6].

The learning algorithm of neural networks with self-feedback connections by using the PRA algorithm for solving the corresponding programming problems, i.e., form (11), is called PRA learning algorithm.

Corollary 1: Using the PRA learning algorithm, if there exists an i such that the corresponding form (11) has a nontrivial solution, i.e., its objective function > 0, then the corresponding neural network has a solution. Otherwise, there only exists a trivial solution.

PRA Learning Algorithm:

1) For each i, solving the corresponding (11), we have $\lambda(i)$. Then, $\lambda(i)$ is transformed into $x(i)$.

a) If $|x(i)| \neq 0$, then

$$w_{ij} = -x_j(i), \qquad i \neq j$$

$$w_{ii} = a_i, \qquad 0 \leq a_i \leq 1.$$

We have $w(i) = (w_{i1}, \cdots, w_{in})$. Then, $w(i)$ is normalized.

b) If $|x(i)| = 0$, then

$$w_{ij} = 0, \qquad i \neq j$$

$$w_{ii} = 1$$

2) Let $W = (w(1), w(2), \cdots, w(n))$. W is the solution.

Obviously, if $\forall i$, $|x(i)| = 0$, then $W = I$ is a trivial solution.

V. Comparison Among Different Algorithms

If a neural network is used as an associative memory, we present the following criteria for comparing different algorithms. They are

- learning complexity,
- the stable state percentage of training samples,
- the attractive radii of training samples and memory capacity, and
- the distribution of spurious stable states.

We will make the comparison among Hebb's, δ-rule, pseudo inverse (overlap matrix), ellipsoid, and PRA learning algorithms below.

Hebb's Rule [1]:

$$w_{ij} = \begin{cases} \sum_{\alpha=1}^{m} \xi_i^\alpha \xi_j^\alpha, & i \neq j \\ 0, & i = j \end{cases}$$

where $\xi^\alpha = (\xi_1^\alpha, \xi_2^\alpha, \cdots, \xi_n^\alpha)$, $\alpha = 1, 2, \cdots, m$, is a set of training samples.

- learning complexity $O(mn^2)$, where m is the size of training samples, n is the number elements.
- Beside the training samples are orthogonal, having each training sample be a stable state of the network cannot be ensured.
- Memory capacity is only $\leq 0.15n$.
- There are probably many spurious stable states.

δ-rule [2]:

$$w_{ij}(t+1) = \alpha e_i(t) w_{ij}(t), \qquad i, j = 1, 2, \cdots, n$$

where $\alpha > 0$ is the length of steps in iteration.

- There is no satisfactory theoretical upper bound of its learning complexity. From a large number of simulation results, it is shown that generally the iteration times are quite large.
- It unable to guarantee that all training samples are stable states.
- Only when α is a small enough positive number, the network is convergent but the convergence time is quite long.

Pseudo Inverse Method [3], [7]:

$$w_{ij} = \frac{1}{n}\left(\sum_{\mu,\nu}^{m} \xi_i^\mu \xi_j^\mu C_{\mu\nu}^{-1}\right)$$

where C^{-1} is the inverse matrix of C, $C = (C_{\mu\nu})$

$$C_{\mu\nu} = \frac{1}{n}\sum_{i=1}^{n} \xi_i^\mu \xi_j^\nu, \qquad \mu, \nu = 1, 2, \cdots, m$$

- learning complexity $O(m^2n^2)$
- As long as all training samples are linearly independent, we can ensure each of them to be a stable state.
- Memory capacity is less than n.
- If the evolution of networks is asynchronous, they are convergent.

Ellipsoid Method [4]:

- The upper bound of learning complexity is $O(m^3n^2 \ln n)$. The computation in each iteration is quite simple.
- Each training sample can be guaranteed to be a stable state while the training samples are not necessarily linearly independent.

PRA Method [6]:

- learning complexity $O(m^2n^2\sqrt{n}L)$
- Each training sample can be guaranteed to be a stable state while the training samples are not necessarily linearly independent.
- In some sense the attractive radii of training samples can be made to be optimal.

VI. Conclusions

Based on programming technique, two learning algorithms of neural network are presented. The networks constructed by these two algorithms have several advantages. They have larger memory capacity and larger attractive radii of training samples. The states corresponding to training samples are stable. Unlike Hebb and pseudo inverse methods, in Ellipsoid and PRA it is not necessary for the weights to be symmetric, i.e., $w_{ij} = w_{ji}$. As associative memory, the properties above are very important.

The learning complexity of the algorithms is slightly increased but still polynomial. In the specific cases, the complexity $O(n^4)$ of the ellipsoid algorithm can be reduced to $O(n^2 \ln n)$.

References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Nat. Acad. Sci.*, vol. 79, 1982, pp. 2554–2558.

[2] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing.* Cambridge, MA: MIT Press, 1986.

[3] I. Kanter and H. Sompolinsky, "Associative recall of memory without errors," Physical Review A, vol. 35, no. 1, pp. 380–392, Jan. 1987.

[4] M. Grotschel *et al. Geometric Algorithms and Combinatorial Optimization.* New York: Springer-Verlag, 1990.

[5] L. Collatz and W. Wetterling, *Optimization Problems.* Berlin-Heidelberg: Springer-Verlag, 1975.

[6] P. M. Pardalos, Y. Ye, and C.-G. Han, "Algorithms for the solution of quadratic knapsack problems," *Linear Algebra and Its Applications,* Vol. 152, pp. 69–92, July 1, 1991.

[7] B. Müller and J. Reinhardt, *Neural Networks—An Introduction.* Berlin-Heidelberg: Springer-Verlag, 1990.