2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

A One-Layer Discrete-Time Projection Neural Network for Support Vector Classification

Wei Zhang and Qingshan Liu

Abstract—**This paper presents a one-layer discrete-time projection neural network described by difference equations for real-time support vector classification (SVC). The SVC is first formulated as a convex quadratic programming problem, and then a recurrent neural network with one-layer structure is designed for training the support vector machine. Furthermore, simulation results on two illustrative examples are given to demonstrate the effectiveness and performance of the proposed neural network.**

I. Introduction

Support vector machines (SVMs) are powerful tools for data classification and regression. Their foundation has been developed by Vapnik and obtained popularity in the literature [1][2]. The main feature of SVMs is that they use the structural risk minimization rather than the empirical risk minimization. Recently, many fast algorithms for SVMs have been developed [3]. Among them, Mangasarian [4] proposed the finite Newton algorithm for SVMs learning. Keerthi and DeCoste [5] introduced the modified finite Newton algorithm to speed up the finite Newton algorithm for fast solution of large scale linear SVMs.

As a software and hardware implementable approach for real-time optimization, recurrent neural networks for solving linear and nonlinear optimization problems with their engineering applications have been widely investigated [6][7][8][9][10]. Compared with traditional numerical optimization algorithms, the recurrent neural networks have fast convergence rate for real-time solutions. In 1986, Tank and Hopfield [9] proposed a recurrent neural network for solving linear programming problems which inspired the research on neurodynamic optimization and its applications. Kennedy and Chua [6] proposed the dynamical canonical nonlinear programming circuit (NPC) for nonlinear programming to generate the approximate optimal solutions based on penalty function method. From then on, the NPC for constrained optimization were widely developed and a variety of optimization neural network models were designed and investigated (e.g., see [11][12][13][14][15], and references therein). Later on, the projection method was introduced to construct recurrent neural networks for constrained optimization, such as the models in [16][17][18][19], which are globally convergent to exact optimal solutions of convex optimization and related problems. Forti *et al.* [20] proposed and investigated the generalized NPC (G-NPC) for getting the optimal solutions of non-smooth optimization problems. Recently, a one-layer recurrent neural network with a discontinuous hard-limiting activation function is proposed for quadratic programming [21], in which results on numerical examples and support vector machine (SVM) learning show the effectiveness and performance of the neural network. Another one-layer projection neural network has been proposed for solving non-smooth optimization problems subject to linear equalities and bound constraints [22]. It is proved that the output variables of the proposed neural network are globally convergent to the optimal solutions provided that the objective function is convex on a defined set. In this paper, a discrete-time neural network is presented based on the models in [22] and applied to SVM learning for classification.

The remainder of this paper is organized as follows. In Section II, the problem formulation of support vector classification and the recurrent neural network model are described. The theoretical analysis of the proposed neural network is presented in Section III. Next, in Section IV, simulation results on two benchmark problems are presented to illustrate the performance of the proposed neural network. Finally, Section V concludes this paper.

Wei Zhang is with the School of Automation, Southeast University, Nanjing 210096, China (email: azhangwei2020@126.com).

Qingshan Liu is with the School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China (email: qsliu@hust.edu.cn).

This work was supported by the National Natural Science Foundation of China under Grant 61105060, the Program for New Century Excellent Talents in University (NCET-12-0114), the Natural Science Foundation of Jiangsu Province of China under Grant BK2011594, and the Fundamental Research Funds for the Central Universities.

II. Support Vector Classification and Model Description

In this section, the SVC is first converted into a constrained quadratic programming problem, then a corresponding discrete-time recurrent neural network is proposed.

Consider a set of training examples

$$D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\},$$

where the ith sample $x_i \in \mathbb{R}^n$ (n is the dimension of the input space) belongs to two separate classes labeled by $y_i \in \{-1, 1\}$. The classification problem is to find a hyperplane in a high dimensional feature space, denoted as Z, which divides the set of examples in the feature space such that all the points with the same label are on the same side of the hyperplane [1][2]. SVC is to construct a map $\mathbf{z} = \Phi(x)$ from the input space $\mathbb{R}^n$ to a high m-dimensional feature space Z and to find an optimal hyperplane $w^T z + b = 0$ in Z to maximize the separation margin between the positive and negative examples, where $w \in \mathbb{R}^m$ and $b \in \mathbb{R}$ are decision parameters.

The decision function of the classifier is described as

$$f_{w,b} = \text{sgn}[w^T z + b],$$

where w is a weight vector and b is a threshold. In this paper, we consider the case when the samples are not linearly separable. Then, the SVC is to find w and b satisfying

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}w^T w + c\sum_{i=1}^{N}\xi_i, \\ \text{subject to} \quad & y_i[w^T\Phi(x_i)+b] \geq 1-\xi_i,\ i=1,2,\ldots,N, \\ & \xi_i \geq 0,\ i=1,2,\ldots,N, \end{aligned} \tag{1}$$

where $c > 0$ is a regularization parameter for the tradeoff between model complexity and training error, and ξ_i measures the (absolute) difference between $w^T z + b$ and y_i. Due to a number of variables and the unknown $\Phi(x)$, it is difficult to solve problem (1) directly. Generally, solving (1) is converted into solving a dual problem

$$\begin{aligned} \text{maximize} \quad & -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j (\Phi(x_i)^T\Phi(y_j))\alpha_i\alpha_j + \sum_{i=1}^{N}\alpha_i, \\ \text{subject to} \quad & \sum_{i=1}^{N}\alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq c, \quad i=1,2,\ldots,N. \end{aligned} \tag{2}$$

We define a kernel function $K(x,y)$ satisfying $K(x_i,x_j) = \Phi(x_i)^T\Phi(x_j)$. The above dual problem is equivalently written as

$$\begin{aligned} \text{maximize} \quad & -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j K(x_i,x_j)\alpha_i\alpha_j + \sum_{i=1}^{N}\alpha_i, \\ \text{subject to} \quad & \sum_{i=1}^{N}\alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq c, \quad i=1,2,\ldots,N. \end{aligned} \tag{3}$$

Moreover, the decision function of the classifier can be represented as

$$f(x) = \operatorname{sgn}\left[\sum_{i=1}^{N}\alpha_i y_i K(x_i,x) + b\right].$$

For convenient computation here, let $a_i = \alpha_i y_i$, then (3) is rewritten as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j K(x_i,x_j) - \sum_{i=1}^{N} a_i y_i, \\ \text{subject to} \quad & \sum_{i=1}^{N} a_i = 0, \\ & -c_i^1 \leq a_i \leq c_i^2, \quad i=1,2,\ldots,N, \end{aligned} \tag{4}$$

where $c_i^1 = -c\cdot\max\{0, 1-y_i\}/2$ and $c_i^2 = c\cdot\max\{0,1+y_i\}/2$, for $i=1,2,\ldots,N$.

Furthermore, problem (4) can be written as the following compact form

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}a^T Q a - y^T a, \\ \text{subjuct to} \quad & e^T a = 0, \\ & d \leq a \leq h, \end{aligned} \tag{5}$$

where $a = (a_1, a_2, ..., a_N)^T$, $y = (y_1, y_2, ...y_N)^T$, $e = (1,1,...1)^T \in \mathbb{R}^N$, the ij-th entry of matrix Q equals to $K(x_i,x_j)$, $d = -c^1, h = c^2$, with $c^1 = (c_1^1, c_2^1, ..., c_N^1)^T$ and $c^2 = (c_1^2, c_2^2, ..., c_N^2)^T$.

To solve problem (5), the discrete-time neural network proposed in this paper is described as the following difference equations

$$\begin{aligned} u_{k+1} \;=\; & P(u_k - g(u_k)) \\ & -(I-P)(-g(u_k) + \beta(Qg(u_k)-y)), \end{aligned} \tag{6}$$

and its output equations are

$$a_k = g(u_k), \tag{7}$$

where β is a positive constant, I is the identity matrix, $P = ee^T/N$, $g: \mathbb{R}^n \to \Omega$ is a projection operator defined by

$$g(u) = \arg\min_{v\in\Omega}\|u-v\|,$$

in which $\|\cdot\|$ is the Euclidean norm.

Remark 1: Compared with the recurrent neural network proposed in [23] for SVM learning which has $N+1$ neorons, the proposed neural network in (6) and (7) has lower model complexity with N neurons. Moreover, compared with the one-layer recurrent neural network with a discontinuous hard-limiting activation function for SVM learning proposed in [21], here the activation function described by a piecewise-linear function is more convenient for implementation by the discrete-time model.

In problem (5), the Ω is a box set, then the calculation of the projection operator is straightforward and

$$g(u_i) = \begin{cases} h_i, & u_i > h_i, \\ u_i, & d_i \leq u_i \leq h_i, \\ d_i, & u_i < d_i. \end{cases}$$

Lemma 1: [24] For the projection operator $g(x)$, the following inequality holds

$$(u-g(u))^T(g(u)-v) \geq 0, \forall u \in \mathbb{R}^n, v \in \Omega.$$

Furthermore, according to Lemma 1, we have the following result.

Lemma 2: For the projection operator $g(x)$, the following inequality holds

$$(u-v)^T(g(u)-g(v)) \geq \| g(u)-g(v) \|^2, \forall u,v \in \mathbb{R}^n.$$

III. Theoretical Analysis

In this section, to show the performance of the proposed neural network for SVC, its optimality and convergence are investigated in detail.

Definition 1: $\bar{u} \in \mathbb{R}^n$ is said to be an equilibrium point of system (6) if

$$Pg(\bar{u}) + (I-P)(\bar{u} - g(\bar{u}) + \beta(Qg(\bar{u})-y)) = 0.$$

From the definitions of P, it is easily to obtain the following lemma.

Lemma 3: For any $a \in \mathbb{R}^N$, $e^T a = 0$ if and only if $Pa = 0$, where P is defined in (6).

Now, the optimality of the proposed neural network is described as the following theorem.

Theorem 1: $a^* \in \mathbb{R}^N$ is an optimal solution of problem (5) if and only if there exists an equilibrium point $u^* \in \mathbb{R}^N$ for system (6) such that $a^* = g(u^*)$.

Proof: Assume $\bar{u}$ to be an equilibrium point of system (6), we have

$$P\bar{a} + (I - P)(\bar{u} - \bar{a} + \beta(Q\bar{a} - y)) = 0, \tag{8}$$

where $\bar{a} = g(\bar{u})$.

Multiplying the both sides of (8) with P, we have $P\bar{a} = 0$ and

$$(I - P)(\bar{u} - \bar{a} + \beta(Q\bar{a} - y)) = 0.$$

From Lemma 3, $\bar{a} = g(\bar{u})$ is a feasible solution of problem (5).

For any feasible point a of problem (5), we have

$$(a - \bar{a})^T (I - P)(\bar{u} - \bar{a} + \beta(Q\bar{a} - y)) = 0.$$

Due to $P\bar{a} = Pa = 0$, it follows that

$$(a - \bar{a})^T (\bar{u} - \bar{a} + \beta(Q\bar{a} - y)) = 0.$$

According to Lemma 1, $(a - \bar{a})^T(\bar{a} - \bar{u}) = (a - g(\bar{u}))^T(g(\bar{u}) - \bar{u}) \geq 0$. Then, it follows that $\beta(a - \bar{a})^T(Q\bar{a} - y) = (a - \bar{a})^T(\bar{a} - \bar{u}) \geq 0$. Let $f(a) = a^T Qa/2 - y^T a$, which is convex. Then $f(a) - f(\bar{a}) \geq (a - \bar{a})^T \nabla f(\bar{a}) = (a - \bar{a})^T(Q\bar{a} - y) \geq 0$. Thus $\bar{a}$ is an optimal solution of problem (5).

Next, we prove that the opposite side of the theorem is true. Assume a^* to be an optimal solution of problem (5). According to Karush-Kuhn-Tucker conditions [25] for problem (5), there exist $v^* \in \mathbb{R}$ and $w^* \in \mathbb{R}^N$ such that

$$Qa^* - y + ev^* + w^* = 0, \tag{9}$$

$$e^T a^* = 0, \tag{10}$$

$$a^* = g(a^* + \beta w^*), \tag{11}$$

where β is a positive constant.

Let $u^* = a^* + \beta w^*$, then $a^* = g(u^*)$. It follows that

$$u^* = g(u^*) + \beta w^*. \tag{12}$$

From (9), $w^* = -(Qa^* - y) - ev^*$ and substituting it into (12), we have

$$u^* = g(u^*) - \beta(Qa^* - y + ev^*). \tag{13}$$

Multiplying the both sides of (13) with e^T follows that

$$\beta v^* = e^T(g(u^*) - u^* - \beta(Qa^* - y))/n. \tag{14}$$

Substituting (14) into (13) results that

$$(I - P)(u^* - g(u^*) + \beta(Qa^* - y)) = 0. \tag{15}$$

According to (10) and Lemma 3, we have $Pa^* = 0$; $i.e., Pg(u^*) = 0$. Combining with (15), one gets that

$$Pg(u^*) + (I - P)(u^* - g(u^*) + \beta(Qa^* - y) = 0.$$

Thus u^* is an equilibrium point of system (6). ■

Definition 2: The output vector of the neural network in (6) and (7) is said to be globally convergent to the unique optimal solution a^* of problem (5) if, for any initial value $u_0 \in \mathbb{R}^N$,

$$\lim_{k \to \infty} a_k = a^*.$$

Now, we present the convergence of the proposed neural network.

Theorem 2: For any initial value $u_0 \in \mathbb{R}^N$, the output vector a_k of the neural network in (6) and (7) is globally convergent to the unique optimal solution if $\beta < 2/\|Q\|$.

Proof: According to Theorem 1, $\bar{a} = g(\bar{u})$ is an optimal solution of problem (5), where $\bar{u}$ is an equilibrium point of system (6) which satisfies

$$\bar{u} = P(\bar{u} - \bar{a}) - (I - P)(-\bar{a} + \beta(Q\bar{a} - y)). \tag{16}$$

By subtracting from the both sides of (6) and (16), it follows that

$$\begin{aligned} u_{k+1} - \bar{u} \quad = \quad & P(u_k - a_k - \bar{u} + \bar{a}) \\ & -(I - P)(-a_k + \beta Q a_k + \bar{a} - \beta Q\bar{a}), \end{aligned}$$

where $a_k = g(u_k)$.

Then, since $P(I - P) = 0$, we have

$$\begin{aligned} & \|u_{k+1} - \bar{u}\|^2 \\ = \quad & \|P(u_k - a_k - \bar{u} + \bar{a})\|^2 \\ & + \|(I - P)(-a_k + \beta Q a_k + \bar{a} - \beta Q\bar{a})\|^2. \end{aligned}$$

Assume

$$J_1 = \|P(u_k - a_k - \bar{u} + \bar{a})\|^2$$

and

$$\begin{aligned} J_2 \quad &= \quad \|(I - P)(-a_k + \beta Q a_k + \bar{a} - \beta Q\bar{a})\|^2 \\ &\leq \quad \| - a_k + \beta Q a_k + \bar{a} - \beta Q\bar{a}\|^2. \end{aligned}$$

On one hand, we have

$$\begin{aligned} J_1 \quad = \quad & \|P(u_k - \bar{u})\|^2 + \|P(a_k - \bar{a})\|^2 \\ & -2(u_k - \bar{u})^T P(a_k - \bar{a}) \\ = \quad & (u_k - \bar{u})^T(P - I)(u_k - \bar{u}) \\ & + \|u_k - \bar{u}\|^2 + \|P(a_k - \bar{a})\|^2 \\ & -2(u_k - \bar{u})^T(P - I)(a_k - \bar{a}) \\ & -2(u_k - \bar{u})^T(a_k - \bar{a}) \end{aligned}$$

According to Lemma 2, $(u_k - \bar{u})^T(a_k - \bar{a}) \geq \|a_k - \bar{a}\|^2$, then

$$\begin{aligned} J_1 \quad \leq \quad & (u_k - \bar{u})^T(P - I)(u_k - \bar{u}) + \|u_k - \bar{u}\|^2 \\ & + \|P(a_k - \bar{a})\|^2 - 2(u_k - \bar{u})^T(P - I)(a_k - \bar{a}) \\ & -2\|a_k - \bar{a}\|^2 \\ = \quad & (u_k - \bar{u})^T(P - I)(u_k - \bar{u}) + \|u_k - \bar{u}\|^2 \\ & + (a_k - \bar{a})^T(P - I)(a_k - \bar{a}) \\ & -2(u_k - \bar{u})^T(P - I)(a_k - \bar{a}) - \|a_k - \bar{a}\|^2. \end{aligned}$$

On the other hand, we have

$$\begin{aligned}
& \|(I-P)(u_k-\bar{u}-a_k+\bar{a})\|^2 \\
= & \|(I-P)(u_k-\bar{u})\|^2+\|(I-P)(a_k-\bar{a})\|^2 \\
& -2(u_k-\bar{u})^T(I-P)(a_k-\bar{a}) \\
= & -(u_k-\bar{u})^T(P-I)(u_k-\bar{u}) \\
& -(a_k-\bar{a})^T(P-I)(a_k-\bar{a}) \\
& +2(u_k-\bar{u})^T(P-I)(a_k-\bar{a}),
\end{aligned}$$

where the last equality holds due to $(I-P)^2=I-P$.

Then,

$$\begin{aligned}
J_1 \leq & \|u_k-\bar{u}\|^2-\|(I-P)(u_k-\bar{u}-a_k+\bar{a})\|^2 \\
& -\|a_k-\bar{a}\|^2 \\
\leq & \|u_k-\bar{u}\|^2-\|a_k-\bar{a}\|^2.
\end{aligned}$$

Moreover, it is easy to get that

$$\begin{aligned}
J_2 & \leq \|\beta Q a_k-a_k-(\beta Q\bar{a}-\bar{a})\| \\
& \leq \|\beta Q-I\|\|a_k-\bar{a}\|.
\end{aligned}$$

Next, since Q is symmetric, we have

$$\|\beta Q-I\|=\max_{\lambda}|\beta\lambda-1|,$$

where λ is the eigenvalue of matrix Q. Since Q is positive definite, we get that $\lambda>0$. From $\beta<2/\|Q\|$, it follows that

$$-1<\beta\lambda-1\leq\beta\|Q\|-1<1.$$

Then

$$\max_{\lambda}|\beta\lambda-1|<1.$$

From above analysis, we have

$$\begin{aligned}
\|u_{k+1}-\bar{u}\|^2 & = J_1+J_2 \\
& \leq \|u_k-\bar{u}\|^2-\|a_k-\bar{a}\|^2 \\
& \quad +\max_{\lambda}(\beta\lambda-1)^2\|a_k-\bar{a}\|^2.
\end{aligned}$$

Then,

$$\|u_{k+1}-\bar{u}\|^2\leq\|u_k-\bar{u}\|^2-(1-\max_{\lambda}(\beta\lambda-1)^2)\|a_k-\bar{a}\|^2. \quad (17)$$

Consider the following Lyapunov function

$$V(u)=\|u-\bar{u}\|^2.$$

From (17), we get that

$$\begin{aligned}
V(u_{k+1})-V(u_k) & \leq -(1-\max_{\lambda}(\beta\lambda-1)^2)\|a_k-\bar{a}\|^2 \\
& \leq 0. \qquad (18)
\end{aligned}$$

For any initial point $u_0\in\mathbb{R}^N$, if $\beta<2/\|Q\|$, $V(u_k)$ is non-increasing as $k\to\infty$ and $\{u_k\}$ is bounded. Then there exists an increasing sequence $\{k_m\}$ with $\lim_{m\to\infty}k_m=\infty$ and a limit point $\widetilde{u}$ such that $\lim_{m\to\infty}u_{k_m}=\widetilde{u}$. Thus $\widetilde{u}$ is a ω-limit point of u_k.

According to the LaSalle invariance principle for discrete-time system [26] , u_k will converge to $\mathcal{M}$, the largest invariant subset of the following set

$$E=\{u_k\in\mathbb{R}^N: V(u_{k+1})-V(u_k)=0\}.$$

Note that, from (18), if $V(u_{k+1})-V(u_k)=0$ and $\beta<2/\|Q\|$, we have $a_k=\bar{a}$; i.e., the output vector a_k is globally convergent to the unique optimal solution $\bar{a}$ for any initial value $u_0\in\mathbb{R}^N$. ■

IV. Simulation Results

In the section, to illustrate the performance of the proposed neural network in (6) and (7) for SVC, we present the simulation results on two benchmark problems.

Example 1: Consider the iris benchmark problem. The data of the iris problem are characterized with four attributes (i.e., the petal length and width, setal length and width). The goal is to classify the class of iris based on these four attributes. The data set consist of 150 samples belonging to three classes, each class has 50 samples. We choose the following polynomial function

$$K(u,v)=(u^Tv+1)^p$$

as the kernel of the SVM. We set $p=2$. Fig. 1 depicts the transient behaviors of the proposed neural network with random initial values for classes 1 and 2, from which we can observe that the state variables are convergent to an equilibrium point of system (5). The classification results are shown in Fig. 2 to illustrate the good performance of the proposed neural network with respect to the petal length and petal width. Fig. 3 depicts the transient behaviors of the proposed neural network with random initial values for classes 1 and 3, from which we can see that the state variables are convergent to an equilibrium point. The classification results are shown in Fig. 4 to illustrate the good performance of the proposed neural network for the setal length and setal width.

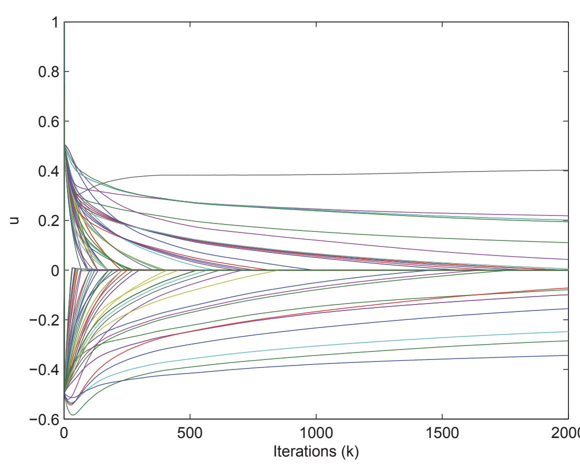

Fig. 1. Transient behaviors of the neural network for classes 1 and 2 in Example 1.

Example 2: Consider the ionosphere data set as the benchmark problem. The data is characterized with 34 attributes. The data set consists of 351 samples belonging to two classes. We choose the RBF function

$$K(u,v)=\exp(-r\|u-v\|^2),\quad r>0$$

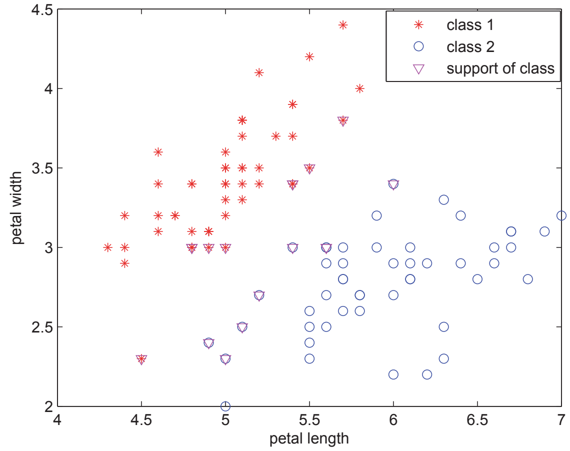

Fig. 2. Support vector classification results for classes 1 and 2 in Example 1.

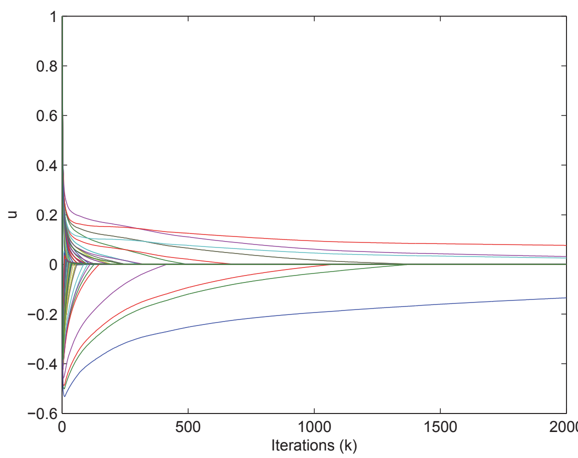

Fig. 3. Transient behaviors of the neural network for classes 1 and 3 in Example 1.

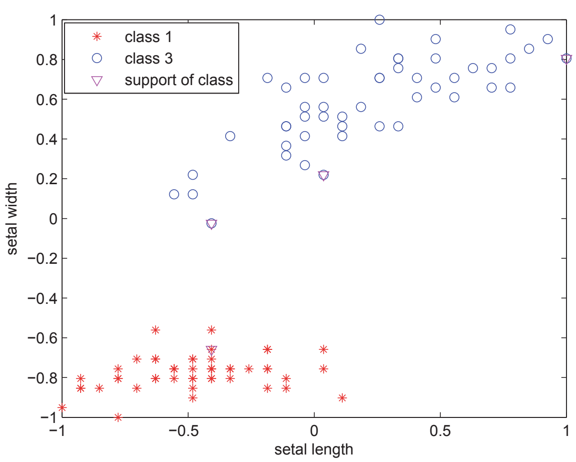

Fig. 4. Support vector classification results for classes 1 and 3 in Example 1.

as the kernel of the SVM. We set $r = 0.1$. Fig. 5 depicts the transient behaviors of the proposed neural network with random initial values. The classification results are shown in Fig. 6 for the 1st and 2nd characters and Fig. 7 for the 10th and 12th characters.

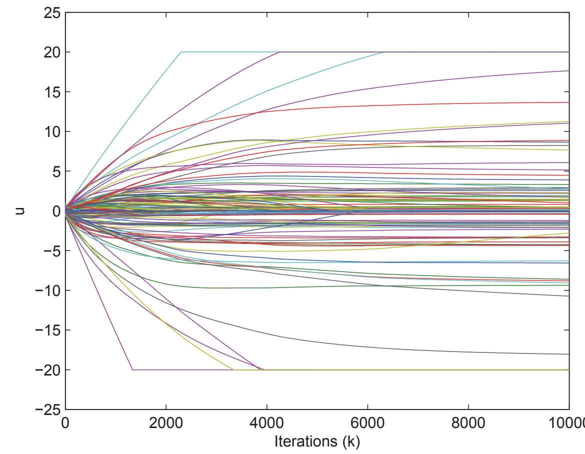

Fig. 5. Transient behaviors of the neural network in Example 2.

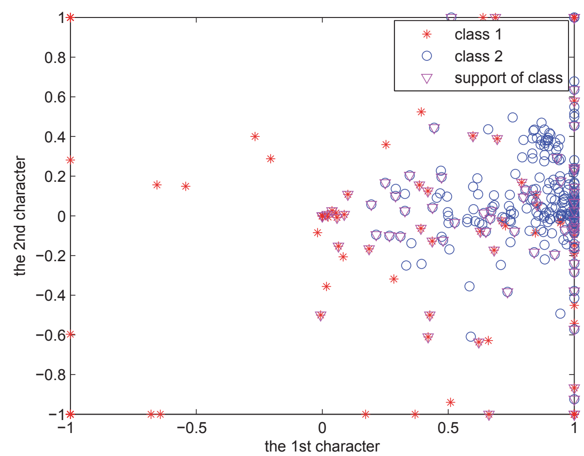

Fig. 6. Support vector classification results for the 1th and 2th characters in Example 2.

V. CONCLUSIONS

This paper presents a discrete-time one-layer recurrent neural network for support vector classification learning. Compared with the existing recurrent neural networks for SVC, the proposed neural network has lower model complexity. Moreover, the feasibility and optimality of the proposed neural network are guaranteed by theoretical results. By using the Lyapunov method, the neural network is proved to be efficient for solving the SVC problems. Furthermore, simulation results on two benchmark problems are given to illustrate the effectiveness and characteristics of the proposed neural network.

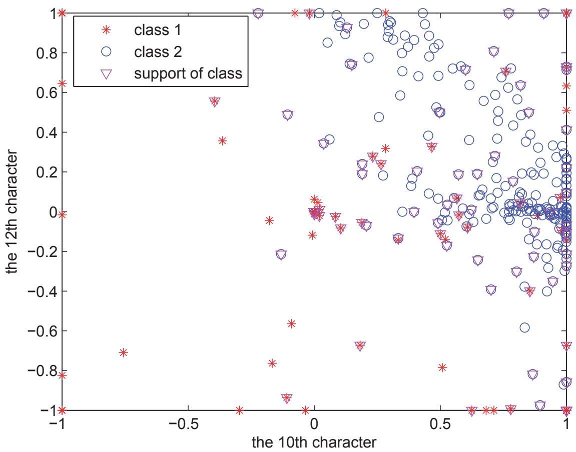

Fig. 7. Support vector classification results for the 10th and 12th characters in Example 2.

References

[1] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[2] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[3] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[4] O. Mangasarian, "A finite newton method for classification problems," *Data Mining Institute Technical Report*, pp. 01–11, 2001.

[5] S. Keerthi and D. DeCoste, "A modified finite newton method for fast solution of large scale linear svms," *Journal of Machine Learning Research*, vol. 6, no. 1, pp. 341–361, 2006.

[6] M. Kennedy and L. Chua, "Neural networks for nonlinear programming," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 554–562, May 1988.

[7] Q. Liu, C. Dang, and J. Cao, "A novel recurrent neural network with one neuron and finite-time convergence for k-winners-take-all operation," *IEEE Transactions on Neural Networks*, vol. 21, pp. 1140–1148, July 2010.

[8] Q. Liu and J. Cao, "Global exponential stability of discrete-time recurrent neural network for solving quadratic programming problems subject to linear constraints," *Neurocomputing*, vol. 74, no. 17, pp. 3494–3501, 2011.

[9] D. Tank and J. Hopfield, "Simple neural optimization networks: An a/d converter, signal decision circuit, and a linear programming circuit," *IEEE Transactions on Circuits and Systems*, vol. 33, pp. 533–541, May 1986.

[10] J. Wang and Y. Xia, "Analysis and design of primal-dual assignment networks," *IEEE Transactions on Neural Networks*, vol. 9, pp. 183–194, Jan. 1998.

[11] A. Bouzerdoum and T. Pattison, "Neural network for quadratic optimization with bound constraints," *IEEE Transactions on Neural Networks*, vol. 4, pp. 293–304, Mar. 1993.

[12] C. Maa and M. Shanblatt, "Linear and quadratic programming neural network analysis," *IEEE Transactions on Neural Networks*, vol. 3, pp. 580–594, July 1992.

[13] J. Wang, "A deterministic annealing neural network for convex programming," *Neural Networks*, vol. 7, no. 4, pp. 629–641, 1994.

[14] Q. Liu, J. Cao, and G. Chen, "A novel recurrent neural network with finite-time convergence for linear programming," *Neural Computation*, vol. 22, no. 11, pp. 2962–2978, 2010.

[15] Z. Guo, Q. Liu, and J. Wang, "A one-layer recurrent neural network for pseudoconvex optimization subject to linear equality constraints," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1892–1900, 2011.

[16] Y. Xia, H. Leung, and J. Wang, "A projection neural network and its application to constrained optimization problems," *IEEE Transactions Circuits and Systems-I*, vol. 49, pp. 447–458, Apr. 2002.

[17] Y. Xia, G. Feng, and J. Wang, "A novel recurrent neural network for solving nonlinear optimization problems with inequality constraints," *IEEE Transactions on Neural Networks*, vol. 19, pp. 1340–1353, Aug. 2008.

[18] X. Hu and B. Zhang, "An alternative recurrent neural network for solving variational inequalities and related optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics-B*, vol. 39, no. 6, pp. 1640–1645, 2009.

[19] Q. Liu and Y. Yang, "Global exponential system of projection neural networks for system of generalized variational inequalities and related nonlinear minimax problems," *Neurocomputing*, vol. 73, no. 10–12, pp. 2069–2076, 2010.

[20] M. Forti, P. Nistri, and M. Quincampoix, "Generalized neural network for nonsmooth nonlinear programming problems," *IEEE Transactions on Circuits and Systems-I*, vol. 51, pp. 1741–1754, Sept. 2004.

[21] Q. Liu and J. Wang, "A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming," *IEEE Transactions on Neural Networks*, vol. 19, pp. 558–570, Apr. 2008.

[22] Q. Liu and J. Wang, "A one-layer projection neural network for nonsmooth optimization subject to linear equalities and bound constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, pp. 812–824, May 2013.

[23] Y. Xia and J. Wang, "A one-layer recurrent neural network for support vector machine learning," *IEEE Transactions on Systems, Man and Cybernetics-B*, vol. 34, pp. 1261–1269, Apr. 2004.

[24] D. Kinderlehrer and G. Stampacchia, *An Introduction to Variational Inequalities and Their Applications*. New York: Academic, 1982.

[25] M. Bazaraa, H. Sherali, and C. Shetty, *Nonlinear Programming: Theory and Algorithms (3rd Ed.)*. Hoboken, New Jersey: John Wiley & Sons, 2006.

[26] J. LaSalle, *The Stability of Dynamical Systems*. Society for Industrial Mathematics, 1976.