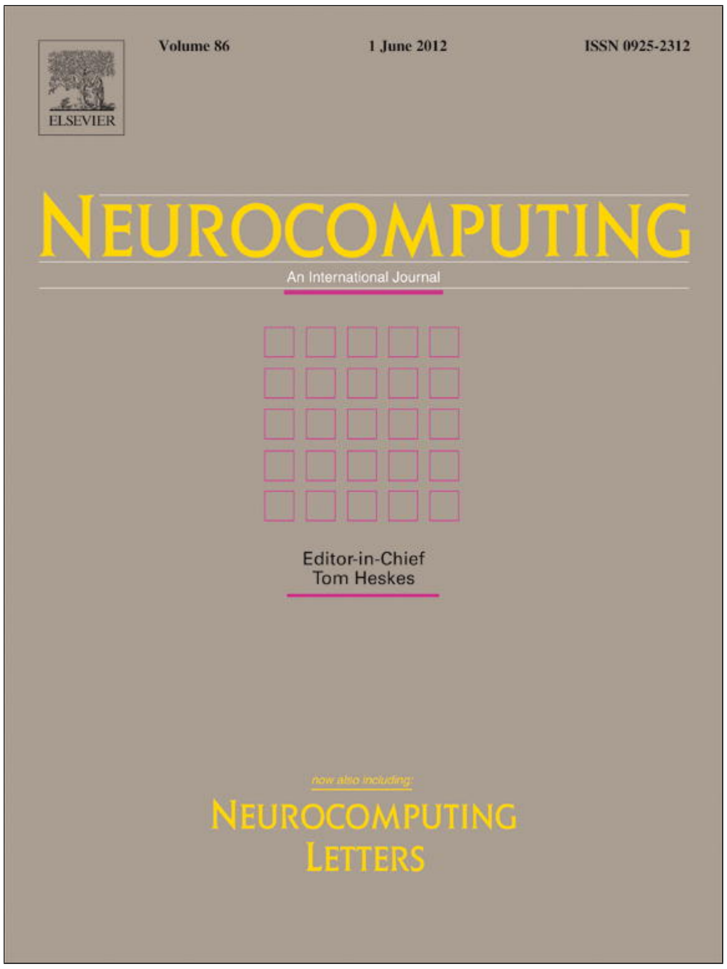


Neurocomputing 86 (2012) 193–198

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Letters

A compact neural network for training support vector machines[☆]

Yun Yang[a], Qiaochu He[b], Xiaolin Hu[c,*]

[a] Department of Mathematical Science, Tsinghua University, Beijing 100084, China
[b] Department of Automotive Engineering, Tsinghua University, Beijing 100084, China
[c] State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 30 April 2011
Received in revised form
22 November 2011
Accepted 22 January 2012
Communicated by H. Jiang
Available online 24 February 2012

Keywords:
Neural network
Support vector machine
Quadratic programming
Analog circuits

ABSTRACT

An analog neural network architecture for support vector machine (SVM) learning is presented in this letter, which is an improved version of a model proposed recently in the literature with additional parameters. Compared with other models, this model has several merits. First, it can solve SVMs (in the dual form) which may have multiple solutions. Second, the structure of the model enables a simple circuit implementation. Third, the model converges faster than its predecessor as indicated by empirical results.

1. Introduction

Engineering applications of artificial intelligence often require real-time solutions [1,2]. By employing artificial neural networks based on analog circuits implementation, the computing procedures are physically parallelled and distributed. After the pioneering work in this field by Hopfield and Tank [3,4], tremendous interests have been aroused for designing neural networks with analog circuits implementation in a variety of engineering applications (see [5–14] and references therein).

Support vector machines (SVMs) are widely used tools for classification [15] and regression [16]. It can be modeled as a quadratic programming (QP) problem, and therefore can be solved by some recurrent neural networks capable of solving this type of optimization problems, e.g., [17–20]. Specifically, [21] presents a one-layer recurrent neural network and [22] presents a simpler model in terms of circuits implementation. Both of the two networks converge to steady-states corresponding to the solutions of the SVMs under some conditions including:

- the Hessian of the objective function is positive definite; or
- the Hessian is positive semidefinite but the solution is unique.

In other words, the QP formulation has to be strictly convex, which may not be the case in many applications (see Remark 1). There exist some other networks which can potentially solve SVMs without strict convexity assumption (e.g., [18–20]), but none of them is as simple as the model in [22]. Then, is it possible to design a neural network that has this nice property but is very simple in structure?

In this letter, we present such a neural network for SVM learning. It is actually a model in [20] with additional parameters. Interestingly, this modification enables a very much simpler circuits implementation, which is comparable to the model in [22].

2. Architecture of the model

2.1. Preliminaries

Suppose that there are N training points for classification, where each input $\mathbf{z}_i \in \mathcal{R}^M$ is in one of the two classes $y_i=+1$ and $y_i=-1$, i.e., the training data is $(\mathbf{z}_i; y_i)$ for $i=1,\ldots,N$. It is well-known that the support vector machine (SVM) training problem for classification can be formulated as a convex quadratic programming (QP) problem [15]

$$\begin{aligned}\min\quad & -\sum_{i=1}^{N}\alpha_i+\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j q_{ij}\\ \text{s.t.}\quad & \sum_{i=1}^{N}\alpha_i y_i=0,\quad 0\le\alpha_i\le h,\ \forall i=1,\ldots,N,\end{aligned}\tag{1}$$

[☆] The work was supported by the National Basic Research Program (973 Program) of China under Grant 2012CB316301, National Natural Science Foundation of China under Grants 60805023 and 61134012, and Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList).

[*] Corresponding author.
E-mail address: xlhu@tsinghua.edu.cn (X. Hu).

where $q_{ij}=y_iy_jK_{ij}(\mathbf{z}_i,\mathbf{z}_j)$ and $K_{ij}(\mathbf{z}_i,\mathbf{z}_j)$ is the so-called kernel function which can take various forms, e.g., $K_{ij}(\mathbf{z}_i,\mathbf{z}_j)=(\mathbf{z}_i^T\mathbf{z}_j+1)^p$ with p an integer. The kernel function must satisfy Mercer's condition and the $N\times N$ dimensional matrix $\mathbf{Q}=\{q_{ij}\}$ must be positive semidefinite, which implies that the problem is convex. The parameter $h>0$ is a user-defined constant to control the tradeoff between the maximization of the margin and the minimization of errors. The above problem is referred to as the dual formulation of the SVM.

Remark 1. In many applications, $\mathbf{Q}$ may not be positive definite. For instance, in linear SVM, $K_{ij}=\mathbf{z}_i^T\mathbf{z}_j$ and $\mathbf{Q}$ can be written as AA^T where $A=(y_1\mathbf{z}_1,y_2\mathbf{z}_2,\dots,y_N\mathbf{z}_N)^T\in\mathcal{R}^{N\times M}$. When $N>M$, which is often the case in practice, rank$(\mathbf{Q})<N$ and $\mathbf{Q}$ is positive semidefinite only. For another instance, it is easy to show that when there are repeated samples in the training set, $\mathbf{Q}$ is singular and thus positive semidefinite only.

2.2. Existing models

In this subsection, we briefly review a few state-of-the-art models for solving the SVM problem. A typical one-layer recurrent neural network is presented in [21] with the following dynamic equations:

$$\tau\dot{x}_i=-x_i+P\left(\sum_{j=1}^N(1-q_{ij})x_j-y_i\mu+1\right),\quad\forall i=1,\dots,N,$$

$$\tau\dot{\mu}=\sum_{i=1}^N y_ix_i,\tag{2}$$

where $\tau>0$ is a time constant, P is a projection operator (activation function) defined as follows

$$P(x)=\begin{cases}0, & x\le 0,\\ x, & 0<x<h,\\ h, & x\ge h,\end{cases}$$

and x_i and μ denote the states of the network, which are time-varying. In fact, x_i corresponds to the variable α_i in (1). Circuit implementation of this network follows the idea that each operator in this model can be implemented by a circuit module, e.g., an op-amp.

Another SVM network is presented in [22]. The dynamic equations are as follows:

$$\tau\dot{x}_i=1-\sum_{j=1}^N q_{ij}\alpha_j-y_i\mu+d_i(\alpha_i-x_i),\quad\forall i=1,\dots,N,$$

$$\tau\dot{\mu}=\sum_{i=1}^N y_i\alpha_i,\tag{3}$$

where $d_i>0$ for all $i=1,\dots,N$, $\alpha_i=P(x_i)$ and other notations are the same as in (2). Note that, here it is α_i instead of x_i that corresponds to the variables of (1). Similar to the network (2), this network can be implemented by circuit modules, too. Its major advantage over the network (2) is that if $d_i=2+\sum_{j=1}^N|q_{ij}|$, it can be implemented by a very simple circuit, which takes advantage of the nonlinear properties of op-amps.

In [20], a neural network was proposed for solving variational inequalities and convex optimization problems. When tailored for solving (1), it is governed by the following dynamic equations:

$$\tau\dot{x}_i=1-\sum_{j=1}^N(q_{ij}+y_iy_j)\alpha_j-y_i\mu+\alpha_i-x_i,\quad\forall i=1,\dots,N,$$

$$\tau\dot{\mu}=\sum_{i=1}^N y_i\alpha_i,\tag{4}$$

where $\alpha_i=P(x_i)$.

In terms of performance, the network (4) is superior to (2) and (3) because when the Hessian matrix $\mathbf{Q}$ is positive semidefinite, only (4) can guarantee the global convergence to solutions of the SVM (1). The other two require positive definiteness of $\mathbf{Q}$, or positive semidefiniteness of $\mathbf{Q}$ plus uniqueness of the solution. Therefore, the neural network (4) has broader applications than the other two. However, its structure is not as simple as (3). Note that there exist other neural networks capable of solving positive semidefinite SVMs (e.g., [18]), but their structures are not as simple as (3) either.

2.3. A revised model

Let us revise the model (4) by adding a constant $d_i>0$ for $i=1,\dots,N$, then the dynamic equations for the revised model are

$$\tau\dot{x}_i=1-\sum_{j=1}^N(q_{ij}+y_iy_j)\alpha_j-y_i\mu+d_i(\alpha_i-x_i),\quad\forall i=1,\dots,N,$$

$$\tau\dot{\mu}=\sum_{i=1}^N y_i\alpha_i,\tag{5}$$

where $\alpha_i=P(x_i)$.

If we set $d_i=2+\sum_{j=1}^N|q_{ij}+y_iy_j|$, there exists a very simple circuit for realizing this model as shown in Fig. 1, where (a) shows the block diagram and (b) shows the circuit realization scheme. Note that only the $\dot{x}_i$ equation is shown in Fig. 1(b). Here we set V_s slightly lower than V_{CC}, then the equation governing this circuit is

$$R_0C_0\dot{u}_i=\frac{V_S}{h}-\sum_{j=1}^N(q_{ij}+y_iy_j)v_j-y_iv_\mu+\left(2+\sum_{j=1}^N|q_{ij}+y_iy_j|\right)(v_i-u_i),$$

where

$$v_i=\begin{cases}0, & u_i\le 0,\\ u_i, & 0<u_i<V_s,\\ V_s, & u_i\ge V_s,\end{cases}$$

which is determined by the saturation property of the op-amp. Then v_i, u_i, v_μ and R_0C_0 correspond to α_i, x_i, μ and τ in (5), respectively. For ensuring that the resistance $R_0/(q_{ij}+y_iy_j)$ shown in Fig. 1(b) is always positive, the absolute value of $q_{ij}+y_iy_j$ is used. So, if for some j, $q_{ij}+y_iy_j$ is positive (negative), then the corresponding input voltage to the op-amp should be v_j $(-v_j)$.

In what follows, we will show that the additional constants d_i does not affect the stability property of the network. The analysis is followed by revising the proof for the network (4) presented in [20].

3. Stability analysis

Let $\mathcal{S}^*=\{\boldsymbol{\alpha}^*\in\mathcal{R}^N|\boldsymbol{\alpha}^*$ solves (1)$\}$. First, we rewrite (5) in the vector form

$$\begin{cases}\tau\dot{\mathbf{x}}=\mathbf{D}(\boldsymbol{\alpha}-\mathbf{x})-\mathbf{Q}\boldsymbol{\alpha}+\mathbf{e}-\mathbf{y}(\mu+\mathbf{y}^T\boldsymbol{\alpha}),\\ \tau\dot{\mu}=\mathbf{y}^T\boldsymbol{\alpha},\end{cases}\tag{6}$$

where $\mathbf{x}=(x_1,x_2,\dots,x_N)^T$, $\boldsymbol{\alpha}=(\alpha_1,\dots,\alpha_N)^T$, $\mathbf{e}=(1,\dots,1)^T$, $\mathbf{y}=(y_1,\dots,y_N)^T$, $\mathbf{Q}=[q_{ij}]_{N\times N}$, $\mathbf{D}=\mathrm{diag}(d_1,\dots,d_N)$, $P(\mathbf{x})=(P(x_1),\dots,P(x_N))^T$ and the other notations are the same as before.

Let $((\mathbf{x}^*)^T,\mu^*)^T$ denote an equilibrium point of (6). We have the following results.

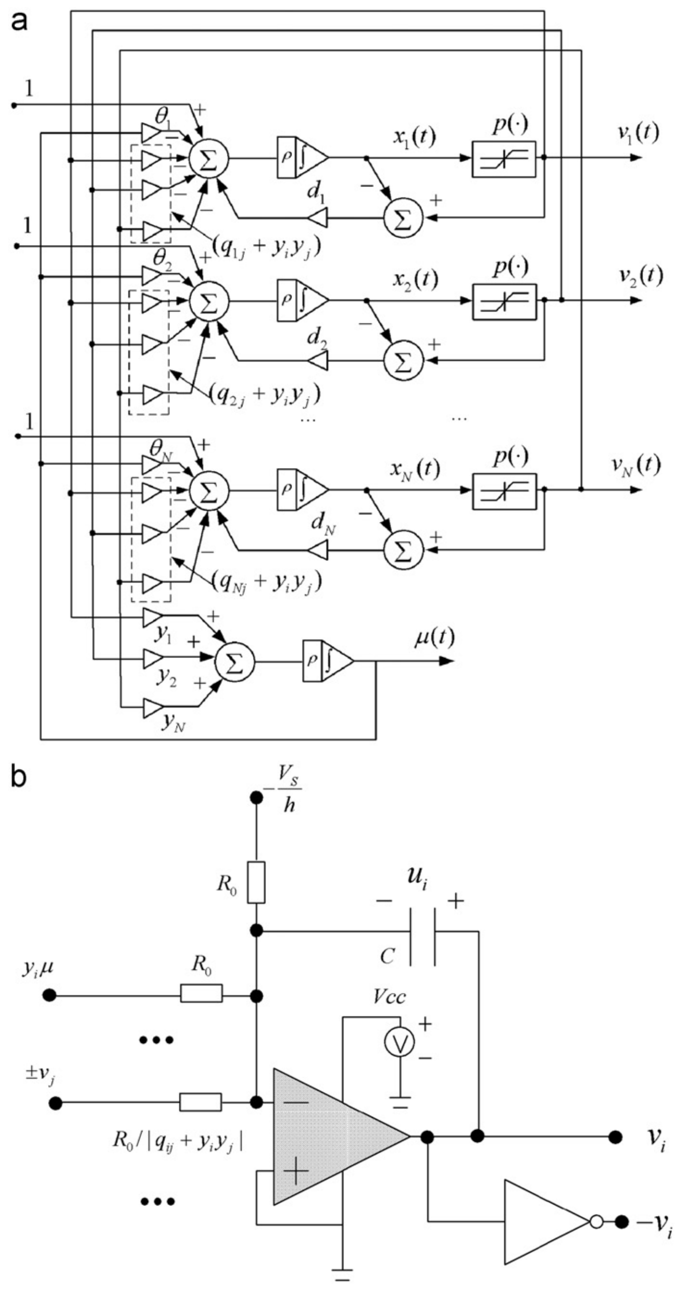


Fig. 1. (a) Block diagram of the architecture of the neural network (5). (b) Circuit realization of α_i.

Lemma 1. *A point $\gamma^* \triangleq ((\mathbf{x}^*)^T,\mu^*)^T$ is an equilibrium point of* (6) *if and only if $\boldsymbol{\beta}^* \triangleq (P(\mathbf{x}^*)^T,\mu^*)^T$ is a Karush–Kuhn–Tucker* (*KKT*) *point of the problem* (1), *where $\boldsymbol{\alpha}^* \triangleq P(\mathbf{x}^*) \in \mathcal{S}^*$.*

Proof. Since (1) is a special case of the problem studied in [23], it can be seen that a point $\boldsymbol{\alpha}^* \in \mathcal{S}^*$ if and only if there exists $\mu^* \in \mathcal{R}$ such that

$$\begin{cases} \boldsymbol{\alpha}^* = P(\boldsymbol{\alpha}^* - (\mathbf{Q}\boldsymbol{\alpha}^* - \mathbf{e} + \mathbf{y}\mu^*)), \\ \mathbf{y}^T\boldsymbol{\alpha}^* = 0. \end{cases} \tag{7}$$

The point $\boldsymbol{\beta}^* = ((\boldsymbol{\alpha}^*)^T,\mu^*)^T$ is called the KKT point. From [24], the first equation above is equivalent to

$$\begin{cases} (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T(\mathbf{Q}\boldsymbol{\alpha}^* - \mathbf{e} + \mathbf{y}\mu^*) \geq 0, \quad \forall \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{h}, \\ \mathbf{y}^T\boldsymbol{\alpha}^* = 0, \end{cases} \tag{8}$$

where $\mathbf{h} = (h,\ldots,h)^T$. It can be seen that this equation is further equivalent to

$$(\alpha_i - \alpha_i^*)(\mathbf{Q}\boldsymbol{\alpha}^* - \mathbf{e} + \mathbf{y}\mu^*)_i \geq 0, \quad \forall 0 \leq \alpha_i \leq h,\ i = 1,\ldots,N. \tag{9}$$

Otherwise, there exists j and $0 \leq \alpha_j \leq h$ such that $(\alpha_j - \alpha_j^*)(\mathbf{Q}\boldsymbol{\alpha}^* - \mathbf{e} - \mathbf{y}\mu^*)_j < 0$. Let $\overline{\boldsymbol{\alpha}} = (\alpha_1^*,\ldots,\alpha_{j-1}^*,\alpha_j,\alpha_{j+1}^*,\ldots,\alpha_N^*)^T$, then

$$(\overline{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^*)^T(\mathbf{Q}\boldsymbol{\alpha}^* - \mathbf{e} + \mathbf{y}\mu^*) = (\alpha_j - \alpha_j^*)(\mathbf{Q}\boldsymbol{\alpha}^* - \mathbf{e} + \mathbf{y}\mu^*)_j < 0,$$

which contradicts (8). Since $d_i > 0$, (9) is equivalent to

$$(\alpha_i - \alpha_i^*)d_i^{-1}(\mathbf{Q}\boldsymbol{\alpha}^* - \mathbf{e} + \mathbf{y}\mu^*)_i \geq 0, \quad \forall 0 \leq \alpha_i \leq h,\ i = 1,\ldots,N$$

and in the sequel, equivalent to

$$(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T\mathbf{D}^{-1}(\mathbf{Q}\boldsymbol{\alpha}^* - \mathbf{e} + \mathbf{y}\mu^*) \geq 0, \quad \forall \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{h}.$$

From [24], this equation is equivalent to

$$\boldsymbol{\alpha}^* = P(\boldsymbol{\alpha}^* - \mathbf{D}^{-1}(\mathbf{Q}\boldsymbol{\alpha}^* - \mathbf{e} + \mathbf{y}\mu^*)). \tag{10}$$

Therefore, (7) is equivalent to

$$\begin{cases} \boldsymbol{\alpha}^* = P(\boldsymbol{\alpha}^* - \mathbf{D}^{-1}(\mathbf{Q}\boldsymbol{\alpha}^* - \mathbf{e} + \mathbf{y}\mu^*)), \\ \mathbf{y}^T\boldsymbol{\alpha}^* = 0. \end{cases} \tag{11}$$

Now, let P_Ω be the projection operator on $[\mathbf{0},\mathbf{h}] \times \mathcal{R}$, $\boldsymbol{\beta} = (\boldsymbol{\alpha}^T,\mu)^T$, and

$$G(\boldsymbol{\beta}) = \begin{cases} \boldsymbol{\alpha} - \mathbf{D}^{-1}(\mathbf{Q}\boldsymbol{\alpha} - \mathbf{e} + \mathbf{y}(\mu + \mathbf{y}^T\boldsymbol{\alpha})), \\ \mu + \mathbf{y}^T\boldsymbol{\alpha}. \end{cases} \tag{12}$$

Then (11) becomes $\boldsymbol{\beta}^* = P_\Omega(G(\boldsymbol{\beta}^*))$. Let $\gamma^* = ((\mathbf{x}^*)^T,\mu^*)^T$ be an equilibrium point of (6). Clearly, it is a solution of $\gamma = G(P_\Omega(\gamma))$. In addition, any solution of $\gamma = G(P_\Omega(\gamma))$, say γ^*, corresponds to a solution of $\boldsymbol{\beta} = P_\Omega(G(\boldsymbol{\beta}))$, i.e., $P_\Omega(\gamma^*)$, which can be obtained by applying a projection operator to both sides of $\gamma^* = G(P_\Omega(\gamma^*))$. In other words, every equilibrium point of (6) corresponds to a KKT point of the SVM problem.

Similar to Lemma 2 in [20], it can be proved that there is a bijective mapping between the solutions of $\boldsymbol{\beta} = P_\Omega(G(\boldsymbol{\beta}))$ and the solutions of $\gamma = G(P_\Omega(\gamma))$. Then, a point γ^* is an equilibrium point of (6) if and only if $P_\Omega(\gamma^*) = ((\boldsymbol{\alpha}^*)^T,\mu^*)^T$ is a KKT point of the SVM problem. Since $\mathbf{Q}$ is positive semidefinite, any KKT point corresponds to a solution of the SVM problem. Hence, $\boldsymbol{\alpha}^* \in \mathcal{S}^*$. This completes the proof. □

Theorem 1. *If $\mathcal{S}^*$ is not empty, then* (6) *always converges to a KKT point of the SVM problem, denoted by $((\mathbf{x}^*)^T,\mu^*)^T$, so that $\boldsymbol{\alpha}^* \triangleq P(\mathbf{x}^*) \in \mathcal{S}^*$.*

Proof. Define a function

$$V(\gamma;\gamma^*) = \sum_{i=1}^{N} x_i(\alpha_i - \alpha_i^*) + \frac{1}{2}\sum_{i=1}^{N}((\alpha_i^*)^2 - \alpha_i^2) + \frac{1}{2}(\mu - \mu^*)^2, \tag{13}$$

where $\gamma = (\mathbf{x}^T,\mu)^T, \boldsymbol{\alpha} = P(\mathbf{x})$ and $\gamma^* = ((\mathbf{x}^*)^T,\mu^*)^T$ is an equilibrium point of (6). According to Lemma 4 in [20], $V(\gamma(t);\gamma^*)$ is nonnegative and continuously differentiable on $\mathcal{R}^{N+1}$, and its gradient is given by

$$\nabla_\gamma V = \begin{pmatrix} \boldsymbol{\alpha} - \boldsymbol{\alpha}^* \\ \mu - \mu^* \end{pmatrix}. \tag{14}$$

Suppose that $\gamma(t)$ is the solution of (6) with initial condition $\gamma(0) = \gamma_0$. Following a similar procedure to that in [20], it is easy to calculate the time derivative of V with respect to (6),

$$\begin{aligned} \tau\dot{V}(t) &= (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T[\mathbf{D}(\boldsymbol{\alpha} - \mathbf{x}) - \mathbf{Q}\boldsymbol{\alpha} + \mathbf{e} - \mathbf{y}(\mu + \mathbf{y}^T\boldsymbol{\alpha})] + (\mu - \mu^*)\mathbf{y}^T\boldsymbol{\alpha} \\ &= -(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T\mathbf{Q}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) - (\mathbf{y}^T(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*))^2 \\ &\quad -(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T\mathbf{D}(\mathbf{x} - \boldsymbol{\alpha}) - (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T(\mathbf{Q}\boldsymbol{\alpha}^* - \mathbf{e} + \mathbf{y}\mu^*). \end{aligned}$$

Since $\mathbf{Q}$ is positive semidefinite, according to (8), if we can prove $\psi(\mathbf{x},\mathbf{x}^*) \triangleq (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T\mathbf{D}(\mathbf{x}-\boldsymbol{\alpha}) \geq 0$, then $\dot{V}(t) \leq 0$ for $t \geq 0$. This is actually true by noticing that

$$\phi_i(x_i,x_i^*) \triangleq (\alpha_i-\alpha_i^*)d_i(\alpha_i-x_i) \leq 0,$$

which can be reasoned as follows.

(1) If $0 \leq x_i \leq h$, then $\alpha_i = x_i$ and $\phi_i = 0$.
(2) If $x_i < 0$, then $\alpha_i = 0$ and $\phi_i = (0-\alpha_i^*)d_i(0-x_i) \leq 0$ because $0 \leq \alpha_i^* \leq h$.
(3) If $x_i > h$, then $\alpha_i = h$ and $\phi_i = (h-\alpha_i^*)d_i(h-x_i) \leq 0$ because $0 \leq \alpha_i^* \leq h$.

Hence, $\psi(\mathbf{x},\mathbf{x}^*) = \sum_i \phi_i(x_i,x_i^*) \leq 0$. The rest of the proof is similar to the arguments for proving Theorem 2 in [20], which is omitted here for brevity. □

Theorem 1 shows that only positive semidefiniteness of $\mathbf{Q}$ is needed for ensuring the global convergence of the trajectory of (5), in contrast to the models (2) and (3) which require stronger conditions.

From the dynamic equations of the neural network (5), it differs from the neural network (4) only in the additional scaling factors d_i. One may wonder the influence of these factors on the convergence rate of the network. Clearly, the function (13) can be also used as a Lyapunov function for the network (4) and its time derivative with respect to (4) is

$$\tau\dot{V}(t) = -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T\mathbf{Q}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)-(\mathbf{y}^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*))^2$$
$$-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T(\mathbf{x}-\boldsymbol{\alpha})-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T(\mathbf{Q}\boldsymbol{\alpha}^*-\mathbf{e}+\mathbf{y}\mu^*).$$

From the proof of Theorem 1 it is easy to show that

$$(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T\mathbf{D}(\mathbf{x}-\boldsymbol{\alpha}) \geq (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T(\mathbf{x}-\boldsymbol{\alpha}) \geq 0,$$

as $d_i \geq 2$ for all i. Hence, before convergence, with the same parameters τ and h, if at any time the states of the two neural networks are identical, then the Lyapunov function $V(t)$ has the same positive value while it decreases faster with respect to (5) than with respect to (4). In other words, the neural network (5) converges to the correct states faster than (4) at such time instants. However, this does not mean that (5) always converges faster than (4) even with the same parameters and initial points, because after evolving sometime the states of the two networks will become different and the above relationship no longer holds. To test the convergence rates of the networks we then resort to numerical simulations (see Example 3 in the next section)

4. Numerical simulations

Example 1. We use the proposed SVM network to classify Fisher's Iris data set (a detailed description of this data set can be found in 'fisheriris' in Matlab). There are 150 points in this data set. Three kinds of Iris are to be classified based on four attributes: petal width, petal length, sepal width and sepal length. We use the SVM to separate class 1 and class 2. Since there are 17 repeated training samples in the data set, the Hessian $\mathbf{Q}$ here is singular and positive semidefinite only. According to Theorem 1, the proposed network can guarantee the convergence of the solution in this scenario. We trained an SVM using 100 data point, with $h=1$ and a Gaussian kernel where $\sigma = 1$. The trajectories of the five non-saturated x_i are depicted in Fig. 2(a), starting from the zero initial point. Since two of the data points are identical, only four trajectories are shown. The solution is characterized by 23 support vectors and with only one misclassified points. The separating function as well as the two attributes (sepal length and sepal width) of the training data set are depicted in Fig. 2(b).

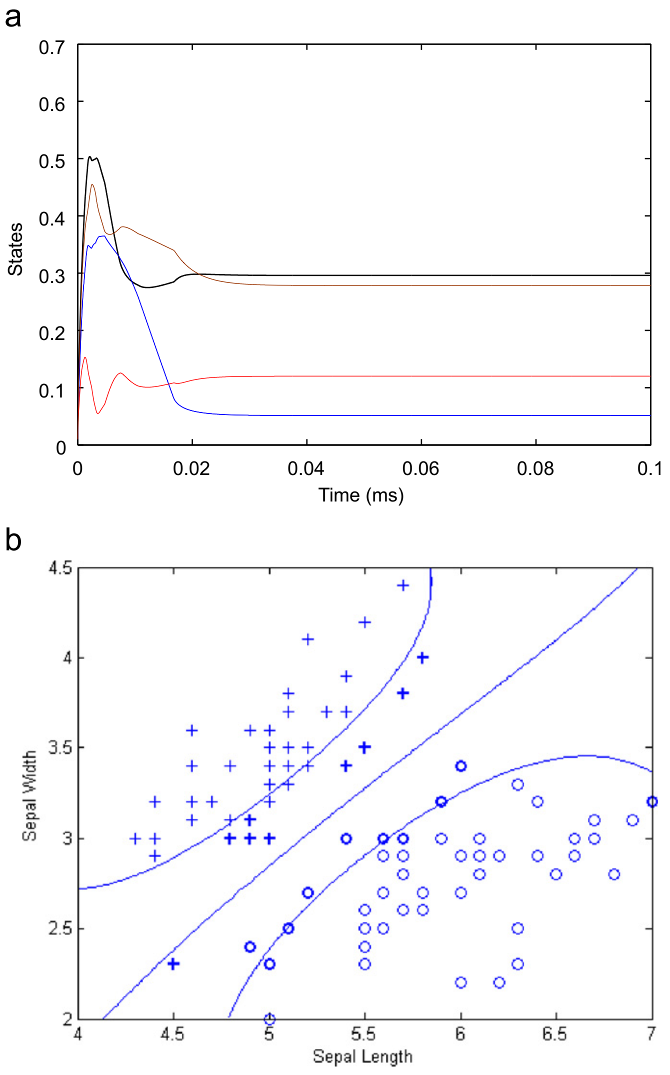
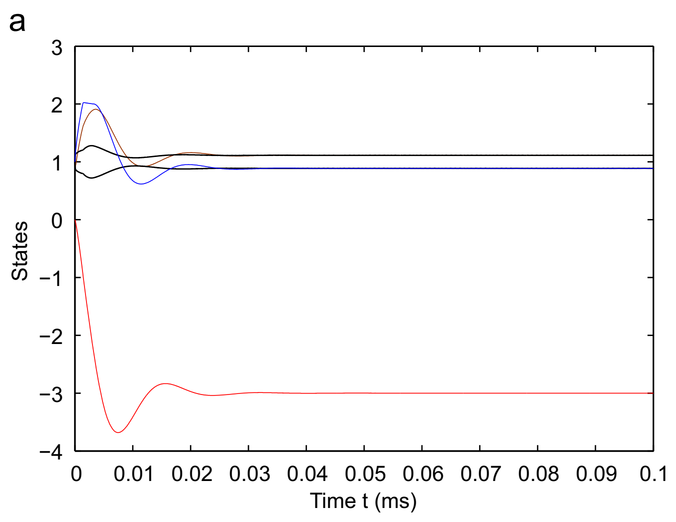


Fig. 2. (a) Time evolution of x_i of the neural network (5) with $\tau = 1$ μs. (b) Decision surface of the Iris data set using Gaussian kernel with $h=1$ and $\sigma = 1$. The margin is indicated and the support vectors are highlighted.

Example 2. Consider training an SVM with the data in $\mathcal{R}^2$: $\mathbf{z}_1 = (2,2)^T, \mathbf{z}_2 = (1,2)^T, \mathbf{z}_3 = (1,1)^T, \mathbf{z}_4 = (2,1)^T$, with labels $(+1,-1,-1,+1)$, respectively. For a linear SVM, the solution to the primal problem ($\mathbf{w} = (2,0)^T, \mu = -3$, which correspond to the weights and bias of the hyperplane, respectively) is unique. But the corresponding $\boldsymbol{\alpha}_i$'s are not unique, because

$$\mathbf{Q} = \begin{pmatrix} 8 & -6 & -4 & 6 \\ -6 & 5 & 3 & -4 \\ -4 & 3 & 2 & -3 \\ 6 & -4 & -3 & 5 \end{pmatrix}$$

is semidefinite. Fig. 3(a) plots the state trajectories of $(\mathbf{x}(t)^T, \mu(t))^T$ of the network (5) with the initial point $(\mathbf{x}_0^T, \mu_0)^T = (-1,-1,-1,-1,0)^T$, which reaches the solution $\boldsymbol{\alpha}^* = (1.1125, 0.8869, 1.1131, 0.8873)^T$. Fig. 3(b) plots the state trajectories with the initial point $(\mathbf{x}_0^T, \mu_0)^T = (0,0,0,0,0)^T$, which reaches $\boldsymbol{\alpha}^* = (1.0640, 0.9363, 1.0637, 0.9361)^T$. Both of them correspond to the same correct separating hyperplane.



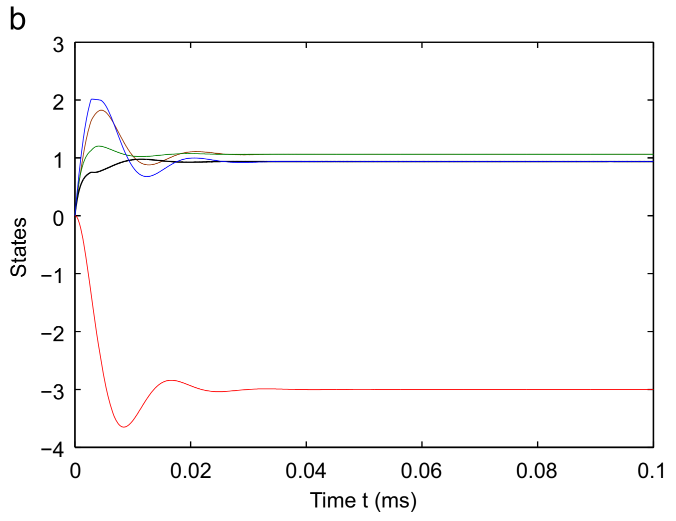


Fig. 3. Time evolution of the states of the network (5) with initial condition (a) $(\mathbf{x}_0^T,\mu_0)^T=(-1,-1,-1,-1,0)^T$ and (b) $(\mathbf{x}_0^T,\mu_0)^T=(0,0,0,0,0)^T$. $\tau=1$ μs and $h=2$.

Example 3. From the dynamic equations, it is seen that the neural network (5) differs from the neural network (4) only in the scaling factors d_i. In this example, through numerical simulations we test if the additional scaling factors introduced in the model brings any advantages or disadvantages in terms of the computing efficiency. A few sets of 2D training data with sample size N were randomly generated, in which half of samples were generated through an isotropic Gaussian distribution with mean $(-3,-3)$ and the other half were generated through an isotropic Gaussian distribution with mean $(3,3)$. The standard deviations of the two Gaussians were both 2. The task was to separate the data with a linear SVM, similar to Example 2. The two neural networks (4) and (5) were both simulated for 30 trials starting from random initial points between -50 and 50. Whenever

$$\|(\tau\dot{\mathbf{x}}(t)^T,\tau\dot{\mu}(t))^T\|^2 \le 10^{-5}$$

was met, the simulation was terminated and the corresponding time t was recorded as the time needed by the network for solving this particular problem.

The average computing time for different sample sizes N are plotted in Fig. 4. It is seen from the figure that the network (5) is faster than the network (4). This conclusion is supported by the one-tailed t-tests. In fact, the p values for $N=10,12,16,20$ are, respectively, 0.017, 0.015, 0.010, 3.9×10^{-5}.

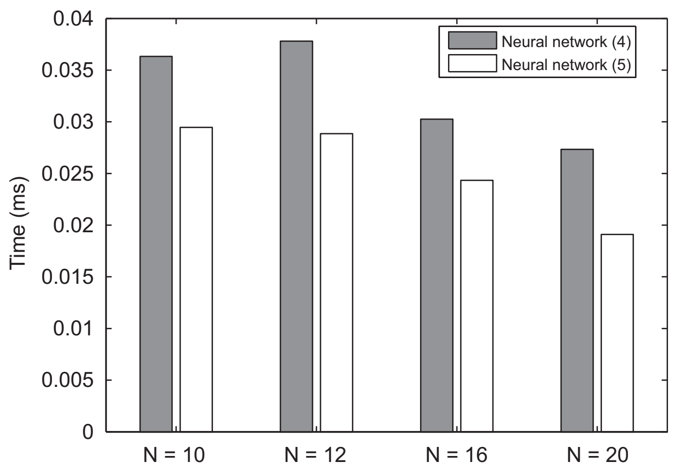


Fig. 4. Statistics of the computing time of the two neural networks (4) and (5) for different sample sizes N. $\tau=1$ μs and $h=1$.

5. Concluding remarks

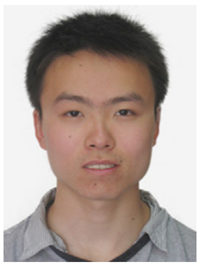
In this letter, a compact neural network architecture for solving support vector classification problems is presented. The model has several advantages compared with existing ones. First, its structure enables simple circuit realization by taking advantage of the inherent properties of op-amps. Second, it only requires convexity of the problem rather than strict convexity, which is in contrast to the models proposed in [21,22]. In terms of the dynamic equations, this model is actually an improved version of an existing neural network with additional scaling factors. Numerical simulations indicate that these scaling factors can speed up the convergence rate, which can be regarded as the third advantage. Finally, the model can be easily extended for solving convex quadratic programming problems.

References

[1] T. Yoshikawa, Foundations of Robotics: Analysis and Control, MIT Press, Cambridge, MA, 1990.
[2] A. Cichocki, R. Unbehauen, Neural Networks for Optimization and Signal Processing, Wiley, New York, 1993.
[3] J.J. Hopfield, D.W. Tank, Computing with neural circuits: a model, Science 233 (1986) 625–633.
[4] D.W. Tank, J.J. Hopfield, Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit, IEEE Trans. Circuits Syst. 33 (1986) 533–541.
[5] A. Rodríguez-Vázquez, R. Domínguez-Castro, A. Rueda, J.L. Huertas, E. Sánchez-Sinencio, Nonlinear switched-capacitor neural networks for optimization problems, IEEE Trans. Circuits Syst. 37 (1990) 384–397.
[6] Y. Xia, M.S. Kamel, A generalized least absolute deviation method for parameter estimation of autoregressive signals, IEEE Trans. Neural Networks 19 (2008) 107–118.
[7] Y. Shen, J. Wang, An improved algebraic criterion for global exponential stability of recurrent neural networks with time-varying delays, IEEE Trans. Neural Networks 19 (2008) 528–531.
[8] Q. Liu, J. Wang, A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming, IEEE Trans. Neural Networks 19 (2008) 558–570.
[9] L. Cheng, Z.-G. Hou, M. Tan, A delayed projection neural network for solving linear variational inequalities, IEEE Trans. Neural Networks 20 (2009) 915–925.
[10] Q. Liu, Y. Yang, Global exponential system of projection neural networks for system of generalized variational inequalities and related nonlinear minimax problems, Neurocomputing 73 (2010) 2069–2076.
[11] Q. Liu, J. Cao, A recurrent neural network based on projection operator for extended general variational inequalities, IEEE Trans. Syst. Man Cybern. B 40 (2010) 928–938.
[12] H. Tang, H. Li, Z. Yi, A discrete-time neural network for optimization problems with hybrid constraints, IEEE Trans. Neural Networks 21 (2010) 184–1189.

[13] Z. Zeng, T. Huang, W.X. Zheng, Multistability of recurrent neural networks with time-varying delays and the piecewise linear activation function, IEEE Trans. Neural Networks 21 (2010) 1371–1377.
[14] Z. Zeng, T. Huang, New passivity analysis of continuous-time recurrent neural networks with multiple discrete delays, J. Ind. Manage. Optim. 7 (2011) 283–289.
[15] C. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Discovery 2 (1998) 121–167.
[16] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Stat. Comput. 14 (2004) 199–222.
[17] Y. Zhang, J. Wang, A dual neural network for convex quadratic programming subject to linear equality and inequality constraints, Phys. Lett. A 298 (2002) 271–278.
[18] X. Hu, B. Zhang, An alternative recurrent neural network for solving variational inequalities and related optimization problems, IEEE Trans. Syst. Man Cybern. B 39 (2009) 1640–1645.
[19] X. Hu, C. Sun, B. Zhang, Design of recurrent neural networks for solving constrained least absolute deviation problems, IEEE Trans. Neural Networks 21 (2010) 1073–1086.
[20] X. Gao, L. Liao, A new projection-based neural network for constrained variational inequalities, IEEE Trans. Neural Networks 20 (2009) 373–388.
[21] Y.S. Xia, J. Wang, A one layer recurrent neural network for support vector machine learning, IEEE Trans. Syst. Man Cybern. B 34 (2004) 1261–1269.
[22] R. Perfetti, E. Ricci, Analog neural network for support vector machine learning, IEEE Trans. Neural Networks 17 (2006) 1085–1091.
[23] Y. Xia, An extended projection neural network for constrained optimization, Neural Comput. 16 (2004) 863–883.
[24] D. Kinderlehrer, G. Stampcchia, An Introduction to Variational Inequalities and Their Applications, Academic, New York, 1980.

Yun Yang received the B.S. degree in mathematical science from Tsinghua University, Beijing, China, in 2011. He is now a Ph.D. student at the Department of Statistical Science, Duke University, NC, USA. His current research interests include machine learning and Bayesian statistics.



Qiaochu He received the B.E. degree in automotive engineering from Tsinghua University, Beijing, China, in 2011. He is now a Ph.D. student at the Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA, USA. His current research interests include computational intelligence and operations research.

Xiaolin Hu received the B.E. and M.E. degrees in automotive engineering from Wuhan University of Technology, Wuhan, China, and the Ph.D. degree in Automation and Computer-Aided Engineering from The Chinese University of Hong Kong, Hong Kong, China, in 2001, 2004, 2007, respectively. He is now an assistant professor at the State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), and Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests include computational intelligence and computational neuroscience.