Another Simple Recurrent Neural Network for Quadratic and Linear Programming

Xiaolin Hu and Bo Zhang

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

Abstract. A new recurrent neural network is proposed for solving quadratic and linear programming problems, which is derived from two salient existing neural networks. One of the predecessors has lower structural complexity but were not shown to be capable of solving degenerate QP problems including LP problems while the other does not have this limitation but has higer structural complexity. The proposed model inherits the merits of both models and thus serves as a competitive alternative for solving QP and LP problems. Numerical simulations are provided to demonstrate the performance of the model and validate the theoretical results.

Keywords: Recurrent neural network, Optimization, Linear programming, Quadratic programming, Stability analysis.

1 Introduction

Consider solving the following convex quadratic programming (QP) problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^TQx + p^Tx \\ \text{subject to} & Ax = b, Cx \le d, x \in X \end{array} \tag{1}$$

where $x = (x_1, \ldots, x_n)^T \in \Re^n$ is the unknown variable, $Q \in \Re^{n\times n}, p \in \Re^n, A \in \Re^{r\times n}, b \in \Re^r, C \in \Re^{m\times n}, d \in \Re^m$ are constants, and X is a nonempty box set defined as $X = \{x \in \Re^n | l_i \le x \le h_i, i = 1, \ldots, n\}$ (note that some l_i's can be $-\infty$ and some h_i's can be $+\infty$). In addition, Q is symmetric and positive semidefinite. In particular, if $Q = 0$, the problem degenerates to a linear programming (LP) problem.

During the past two decades, many recurrent neural networks have been proposed for solving optimization-related problems because of their analog circuits implementability and intrinsic parallelism which are advantageous for fast computing. In particular, for solving the QP problem (1), there exist many salient models but their real-time computational abilities are often constrained by various deficiencies such as demand for slack variables (for converting inequality constraints into equality constraints) [1, 2, 3, 4, 5, 6], calculation of matrix inverses [7, 8, 9, 10, 11], inconvenience in determining the convergence conditions [10, 5, 12], and so on. Actually, these networks are suitable for specific applications. Among those capable of solving the QP problem (1) but free of such deficiencies, two simple models refer to

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = -\lambda \begin{pmatrix} x - \mathcal{P}_X((I-Q)x - C^T y + A^T z - p) \\ y - \tilde{y} \\ Ax - b \end{pmatrix} \tag{2}$$

and

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = -\lambda \begin{pmatrix} 2(x - \mathcal{P}_X((I-Q)x - C^T \tilde{y} + A^T(z - Ax + b) - p) \\ y - \tilde{y} \\ Ax - b \end{pmatrix} \tag{3}$$

where $\tilde{y} = (y + Cx - d)^+$ and $\lambda > 0$, which were proposed in [13] and [14], respectively. The block diagram of the former is depicted in Fig. 1 and that of the latter can be depicted similarly.

The major difference between the performances of (2) and (3) is that the former can solve (1) with positive definite Q, while the latter can solve (1) with positive semi-definite Q though its structure is slightly more complicated (a detailed comparison is made in Section 2).

The major difference between the dynamic equations of the two neural networks is that the latter substitutes $(y + Cx - d)^+$ and $z - Ax + b$ for y and z respectively in the first equation of the former. Clearly, such *substitutions* do not affect the equilibrium set of (2) and consequently the two networks have the same equilibrium set. This observation raises an interesting question: will other combinations of variable substitutions also result in feasible neural networks? Note that multiple variable substitutions are available if we let the right-hand-sides of (2) and (3) be equal to zeros. The answer to this question is positive and a novel model has been figured out in this way very recently [15], which shares the same performance with (3) and the same structural complexity with (2). In this paper we present another model, also resulted from this idea. It will be shown to possess the merits of (2) and (3), and thus can compete with the model in [15].

Throughout the paper, if a is a vector, then $\|a\| = \sqrt{\sum a_i^2}$. $\Re_+^n$ denotes the nonnegative quadrant of $\Re^n$. Moreover, it is assumed that there exists at least one finite solution to problem (1).

2 Formulation of the Model

As for problem (1), the Karush-Kuhn-Tucker (KKT) conditions can be expressed in the following way [13, 14]: x is an optimum of (1) if and only if there exist $y \in \Re^m$ and $z \in \Re^r$ so that

$$\begin{cases} x = \mathcal{P}_X((I-Q)x - C^T y + A^T z - p) \\ y = (y + Cx - d)^+ \\ Ax - b = 0 \end{cases} \tag{4}$$

where $\mathcal{P}_X(\cdot)$ and $(\cdot)^+$ are two activation functions, whose definitions can be found in any of the references [2, 14, 13, 5, 11, 16, 15, 17]. Obviously, the identical equilibrium set of (2) and (3) coincides with the KKT point set of problem (1).

If we substitute y in the first equation of (2) with $(y + Cx - d)^+$, which is available from the second equation, to constitute $\tilde{x}$, and then substitute x in the third equation of (2) with $\tilde{x}$, we arrive at the following dynamic equations with only the scaling factors different:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = -\lambda \begin{pmatrix} 2(x - \tilde{x}) \\ y - \tilde{y} \\ 2(A\tilde{x} - b) \end{pmatrix} \tag{5}$$

where $\tilde{y} = (y + Cx - d)^+$, $\tilde{x} = \mathcal{P}_X((I - Q)x - C^T\tilde{y} + A^Tz - p)$ and $\lambda > 0$. These equations represent a new network for solving problem (1). The output can be regarded as either x or $\tilde{x}$ because in Section 3 it will be shown that the state equations will always evolve to one of the equilibrium points and at any equilibrium point x is equal to $\tilde{x}$. It is easy to verify the following results.

Theorem 1. *A point x^* is a solution of* (1) *if and only if there exist y^* and z^* such that $((x^*)^T, (y^*)^T, (z^*)^T)^T$ is an equilibrium point of* (5)*.*

Then, to verify the viability of the new model, we need to show whether its stability conditions and structural complexity are comparable with existing models especially (2) and (3). In Section 3 it will be shown that the stability results of the proposed model requires only the positive semidefiniteness of Q for solving (1), the same as the neural network (3). In the rest of this section, we compare its structural complexity with those of (2) and (3).

First, we show that the structural complexity of the proposed neural network is the same as (2). One would argue that if the detailed expressions of $\tilde{y}$ and $\tilde{x}$ are substituted into (5), the equations in (5) will get much longer than the corresponding ones in (2); therefore, the structure of the network (5) is much more complicated than that of (2). However, nobody would do that in hardware implementation. Actually, though $\tilde{y}$ appears three times in (5), it does not mean that this term is needed to be implemented three times. Only one block is enough and its output can be connected to three different spots. Likewise, though $\tilde{x}$ appears twice in (5), it requires just one implementation. Based on this idea, on one hand, if we count the numbers of multiplications and additions/substractions to be performed on the right-hand-sides of (2) and (5), respectively, it will be found that the numbers are the same for the two equations. On the other hand, if we count the numbers of integrators (for realizing integrations), amplifiers (for realizing activation functions $\mathcal{P}_X(\cdot)$ and $(\cdot)^+$), multipliers, summators and interconnections required by hardware implementation of the two networks, respectively, it will be found that all of these numbers are identical for two networks. This result is made clearer in Fig. 1. In the figure, if we change the starting positions of two connections in the neural network (2) to other two positions, and meanwhile change two scaling factors, the figure becomes exactly the diagram of the neural network (5). In this sense, the proposed neural network possesses the same structural complexity as the neural network (2).

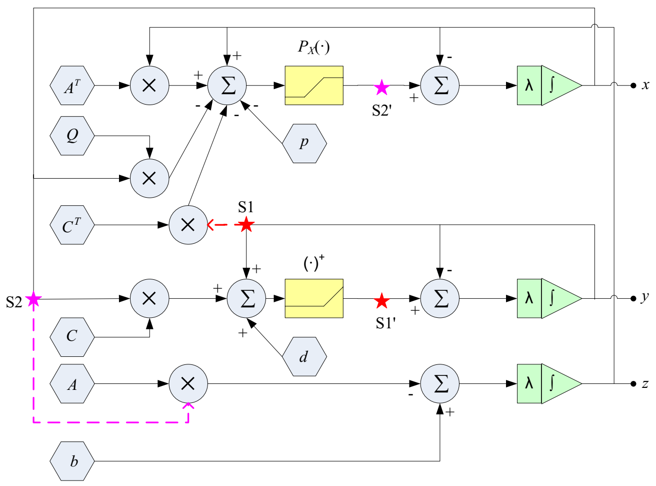

Fig. 1. Block diagrams of the neural networks (2) and (5). If the two dashed lines are connected, the figure depicts the diagram of (2). If the first dashed line (counted top-down, the same convention is adopted in what follows) is not started from point S1, but from point S1', the second line is not started from point S2, but from point S2', while the first and third integrators use 2λ as their scaling factors, the diagram then switches to the network (5).

Let us then examine the complexity of the neural network (3). From its dynamic equations, it is easy to see that no matter how the computing order of different terms is arranged, one more operation of addition is required to incorporate $-Ax+b$ to the term $A^T(z-Ax+b)$ in the top equation of (3), though the result of $Ax-b$ can be obtained directly from the bottom equation. This leads to r more connections and one more summator in its architecture than in the architectures of (2) and (5), which can be perceived in Fig. 1. When the number of equality constraints is large relative to inequality constraints, e.g., in k-WTA applications [8, 15], the inefficiency of the network (3) becomes prominent.

3 Stability Analysis

In this section, we will show that the proposed neural network (5) possesses the same convergence result as the neural network (3), that is, the global convergence property requires the positive semidefiniteness of Q only. In what follows, the equilibrium set of the neural network is denoted by Ω^*. Since it is assumed that (1) has at least one finite solution, according to Theorem 1, there exists at least one finite point in Ω^*. The following lemma follows from [14].

Lemma 1. *The function* $\|\tilde{y}\|^2$ *is convex and continuously differentiable on* $\Re^{n+m}$. *In addition,* $\nabla\|\tilde{y}\|^2 = 2\begin{pmatrix} C^T\tilde{y} \\ \tilde{y} \end{pmatrix}$.

Lemma 2. *Consider the following function*

$$V(u) = \phi(u) - \phi(u^*) - (u-u^*)^T \nabla\phi(u^*) + \frac{1}{2}\|u-u^*\|^2, \tag{6}$$

where $u = ((x)^T, (y)^T, (z)^T)^T$ *and* $u^* = ((x^*)^T, (y^*)^T, (z^*)^T)^T \in \Omega^*$ *is a finite point and* $\phi(u) = x^T Qx/2 + p^T x + \|\tilde{y}\|^2/2$. *It has the following properties.*

1. $V(u)$ *is convex and continuously differentiable on* $\Re^{n+m+r}$.
2. $V(u) \geq \|u-u^*\|^2/2$ *for all* $u \in \Re^{n+m+r}$.
3. $\nabla V(u)^T F(u) \geq 2\|x-\tilde{x}\|^2 + \|y-\tilde{y}\|^2$ *for all* $u \in \Re^{n+m+r}$, *where*

$$F(u) = \begin{pmatrix} 2(x-\tilde{x}) \\ y-\tilde{y} \\ 2(A\tilde{x}-b) \end{pmatrix}.$$

Proof. From Lemma 1, $V(u)$ is continuously differentiable on $\Re^{n+m+r}$. To prove its convexity, what we only need to show is the convexity of the function $\psi(u) = \phi(u) - \phi(u^*) - (u-u^*)^T\nabla\phi(u^*)$. In view that $\phi(u)$ is convex and $\nabla\psi(u) = \nabla\phi(u) - \nabla\phi(u^*)$, it is easy to validate this fact.

2) This result follows directly from the convexity of $\phi(u)$, which ensures $\psi(u)$ defined above is nonnegative.

3) The gradient of $V(u)$ is given by

$$\nabla V(u) = \begin{pmatrix} (I+Q)(x-x^*) + C^T\tilde{y} - C^T y^* \\ \tilde{y} - 2y^* + y \\ z - z^* \end{pmatrix}.$$

In the following projection inequality (see [18,13,14,16])

$$(\mathcal{P}_\Omega(u) - u)^T(v - \mathcal{P}_\Omega(u)) \leq 0, \forall u \in \Re^n, v \in \Omega,$$

let $\Omega = X, u = x - Qx - p - C^T\tilde{y} + A^T z$ and $v = x^*$, then

$$(\tilde{x} - x^*)^T(x - Qx - p - C^T\tilde{y} + A^T z - \tilde{x}) \geq 0.$$

Since u^* satisfies (4), according to the equivalence of the projection equation and variational inequality [18] we know

$$(\tilde{x} - x^*)^T(Qx^* + p + C^T y^* - A^T z^*) \geq 0.$$

Adding the above two equations gives

$$(\tilde{x} - x^*)^T(x - Qx - C^T\tilde{y} + A^T z - \tilde{x} + Qx^* + C^T y^* - A^T z^*) \geq 0.$$

Similarly, we can derive $(\tilde{y}-y^*)^T(y+Cx-d-\tilde{y}) \geq 0$ and $(\tilde{y}-y^*)^T(-Cx^*+d) \geq 0$. Adding them gives

$$(\tilde{y} - y^*)^T(y + Cx - Cx^* - \tilde{y}) \geq 0.$$

Given these equations, we have

$$\begin{aligned}\nabla V(u)^T F(u) =& 2\|x-\tilde{x}\|^2 + 2(x-\tilde{x})^T(\tilde{x}-x^* + Qx - Qx^* + C^T\tilde{y} - C^T y^*)\\ &+ \|y-\tilde{y}\|^2 + 2(y-\tilde{y})^T(\tilde{y}-y^*) + 2(A\tilde{x}-b)^T(z-z^*)\\ =& 2\|x-\tilde{x}\|^2 + \|y-\tilde{y}\|^2 - 2(\tilde{x}-x^*)^T(\tilde{x}-x^* + Qx - Qx^* + C^T\tilde{y}\\ &- C^T y^*) + 2(x-x^*)^T(\tilde{x}-x^*) + 2(x-x^*)^T(Qx-Qx^*)\\ &+ 2(x-x^*)^T(C^T\tilde{y} - C^T y^*) + 2(\tilde{y}-y^*)^T(y-\tilde{y})\\ &+ 2(\tilde{x}-x^*)^T(A^T z - A^T z^*)\\ =& 2\|x-\tilde{x}\|^2 + \|y-\tilde{y}\|^2 + 2(\tilde{x}-x^*)^T(x-\tilde{x} - Qx + Qx^* - C^T\tilde{y}\\ &+ C^T y^* + A^T z - A^T z^*) + 2(x-x^*)^T(Qx-Qx^*)\\ &+ 2(\tilde{y}-y^*)^T(Cx - Cx^* + y - \tilde{y})\\ \geq& 2\|x-\tilde{x}\|^2 + \|y-\tilde{y}\|^2 + 2(x-x^*)^T Q(x-x^*).\end{aligned}$$

Since Q is positive semidefinite, the conclusion holds.

Lemma 3. *For any initial point $u(t_0) = (x(t_0)^T, y(t_0)^T, z(t_0)^T)^T \in X \times \Re^{m+r}$, the neural network* (5) *has a unique continuous solution $x(t)$ for all $t \geq t_0$ and $x(t)$ stays in X forever.*

Proof. Taking Lemma 2 into account, the results can be reasoned following similar arguments for proving Theorem 2 in [14], which are omitted here for brevity.

Theorem 2. *For any $u(t_0) \in X \times \Re^{m+r}$ the neural network* (5) *is stable in the sense of Lyapunov and its trajectory $u(t)$ converges to a point in Ω^*. In particular, if there is only one point in Ω^*, the neural network is asymptotically stable.*

Proof. According to Lemma 3, for any initial point $u(t_0) \in X \times \Re^{m+r}$, the neural network has a unique continuous trajectory $u(t)$ for all $t \geq t_0$. In addition $u(t)$ is bounded for all $t \geq t_0$. According to Lemma 3, $x(t) \in X$ for all $t \geq t_0$. Consider the function defined in (6). It follows from Lemma 2 that

$$\frac{dV(u(t))}{dt} = -\lambda \nabla V(u)^T F(u) \leq -2\lambda\|x-\tilde{x}\|^2 - \lambda\|y-\tilde{y}\|^2 \quad \forall t \geq t_0.$$

Hence, the neural network is stable in the sense of Lyapunov. Clearly, if $du/dt = 0$, then $dV/dt = 0$. According to the LaSalle invariance principle, $u(t)$ converges to the largest invariant set $\mathcal{M}$ in $\{u \in \Re^{n+m+r} | dV(u)/dt = 0\}$. In what follows we show $\mathcal{M} = \Omega^*$. Clearly, any point in Ω^* also belongs to $\mathcal{M}$. Consider any point $u \in \mathcal{M}$. Since $dV/dt = 0$, then $x = \tilde{x}$ and $y = \tilde{y}$ from the above equation, which implies $dx/dt = -2\lambda(x-\tilde{x}) = 0$. It follows that x is in the steady state (a constant), so is $\tilde{x}$. Denote $A\tilde{x} - b$ by c where c is a constant. If $c \neq 0$, then $dz/dt = -2\lambda c$ and $z \to \infty$ when $t \to +\infty$, which contradicts the boundedness of $u(t)$. Consequently, $c = 0$ and $dz/dt = 0$. It follows that $u \in \Omega^*$, and hence $\mathcal{M} = \Omega^*$.

Since $u(t)$ is bounded over $[t_0, +\infty)$, there exists a convergent subsequence $t_0 < \cdots < t_n < t_{n+1} < \cdots$ such that $\lim_{k\to+\infty} u(t_k) = \hat{u}$, where$\hat{u} \in \Omega^*$. Define another Lyapunov function $\hat{V}(u)$ the same as $V(u)$ in (6) except that u^* in $V(u)$ is replaced by $\hat{u}$. It is easy to see that $\hat{V}(u)$ decreases along the trajectory of (5) and satisfies $\hat{V}(\hat{u}) = 0$. Therefore, for any $\varepsilon > 0$, there exists $q > 0$ such that, for all $t \geq t_q$,

$$\|u(t) - \hat{u}\|^2/2 \leq \hat{V}(u(t)) \leq \hat{V}(u(t_q)) < \varepsilon,$$

that is, $\lim_{t\to+\infty} u(t) = \hat{u}$.

In particular, if Ω^* contains a unique point, from the above analysis, the neural network is asymptotically stable. The proof is completed.

Therefore, the proposed model is as excellent as (3) in terms of performance.

4 Illustrative Examples

In this section, we will discuss two examples to demonstrate the performance of the proposed neural network. The simulations were conducted in MATLAB.

Example 1. Consider the following problem discussed in [14]:

$$\begin{aligned} \text{minimize} \quad & (x_1 + 3x_2 + x_3)^2 + 4(x_1 - x_2)^2 \\ \text{subject to} \quad & x_1^3 - 6x_2 - 4x_3 + 3 \leq 0, \\ & 1 - x_1 - x_2 - x_3 = 0, x \geq 0. \end{aligned}$$

This problem has a unique solution $x^* = (0, 0, 1)^T$, and the associated Lagrange multipliers are $y^* = 0, z^* = -2$. It is easy to verify that the problem is convex on $X = \Re^3_+$. According to Theorem 2, the neural network (5) should converge to u^* with any initial point $u(0) \in \Re^3_+ \times \Re^2$ and be asymptotically stable at u^*. All simulations verified this fact. In addition, it was interesting to note that even with $x(0) \notin X$, the neural network still always converged to u^*. Fig. 2 shows such an example. This phenomenon indicates that the stability result in Theorem 2 leaves space for improving.

Example 2. To illustrate the performance of the proposed neural network for linear programming, we now solve a classical transportation problem [19]. Suppose that M suppliers, each with a given amount of goods p_i, are required to supply N consumers, each with a given limited capacity q_j. For each supplier-consumer pair, the cost of transporting a single unit of goods is given as c_{ij}. The transportation problem is then to find a least-expensive flow of goods from the suppliers to the consumers that satisfies the consumers' demand. Formally, the problem can be described as follows

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{M}\sum_{j=1}^{N} c_{ij}x_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{N} x_{ij} \leq p_i, \quad \forall 1 \leq i \leq M \end{aligned}$$

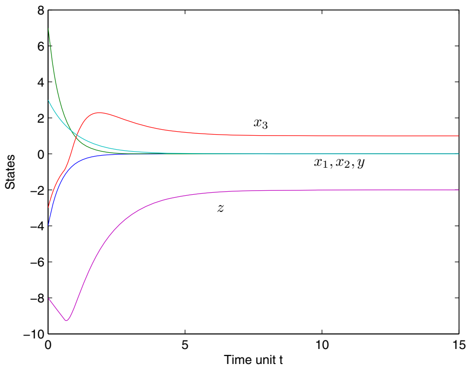

Fig. 2. State trajectory of the neural network (5) with $\lambda = 1, u(0) = (-4, 7, -3, 3, -8)^T$ in Example 1

$$\sum_{i=1}^{M} x_{ij} \le q_j, \quad \forall 1 \le j \le N$$

$$\sum_{i=1}^{M}\sum_{j=1}^{N} c_{ij}x_{ij} = \min\left(\sum_{i=1}^{M} p_i, \sum_{j=1}^{N} q_j\right)$$

$$x_{ij} \ge 0, \quad \forall 1 \le i \le M, 1 \le j \le N.$$

The problem can be rewritten in the standard form of LP problem (1) with $Q = 0$, and the neural network (5) can be used to solve the problem. For illustration purpose, let us consider a problem with $M = 3$ and $N = 4$. The parameters are $p = (10, 16, 18)^T, q = (13, 5, 15, 10)^T$ and

$$c = \begin{pmatrix} 0.1 & 0.2 & 0.1 & 0.5 \\ 0.5 & 0.1 & 1.0 & 0.8 \\ 1.0 & 0.1 & 0.4 & 0.1 \end{pmatrix}.$$

The unique solution is

$$x^* = \begin{pmatrix} 3 & 0 & 7 & 0 \\ 10 & 5 & 0 & 0 \\ 0 & 0 & 8 & 10 \end{pmatrix}.$$

The neural network (5) ($Q = 0$) was simulated to solve the problem and obtained the same solution. Fig. 3 depicts the output trajectory $x(t)$ in one simulation with $\lambda = 1$ and a random initial point, which converge to x^*.

5 Concluding Remarks

In the paper a new recurrent neural network was presented for quadratic and linear programming. This model shares much similarity with two classical models but combines their merits, that is, simple structure and good performance.

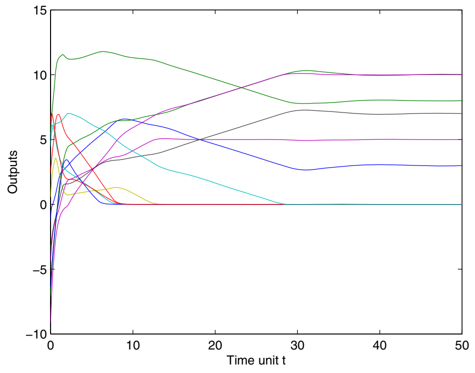

Fig. 3. Output trajectory of the neural network (5) with $\lambda = 1$ and a random initial point $u(0)$ in Example 2

The contribution of this invention is twofold. On one hand, it enriches the family of recurrent neural networks for solving optimization problems, and therefore offers flexibility for circuits practitioners in neural circuits design. On the other hand, its design idea, actually, obtaning new models from existing ones by variable substitutions, may shed light to the development of more powerful models. Up to now, two new models have been devised using this idea (the other is presented in [15]). An interesting question arises: how many models on earth are there in this category besides the known four? We believe that this open question will stimulate many further investigations.

Acknowledgements

The work was supported by the National Natural Science Foundation of China Grants 60805023, 60621062 and 60605003, National Key Foundation R&D Project Grants 2003CB 317007, 2004CB318108 and 2007CB311003, China Postdoctoral Science Foundation Grants 20080430032 and 200801072, and Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList).

References

1. Xia, Y.: A New Neural Network for Solving Linear and Quadratic Programming Problems. IEEE Trans. Neural Netw. 7, 1544–1547 (1996)
2. Tao, Q., Cao, J., Sun, D.: A Simple and High Performance Neural Network for Quadratic Programming Problems. Applied Mathematics and Computation 124, 251–260 (2001)
3. Gao, X., Liao, L.: A Neural Network for Monotone Variational Inequalities with Linear Constraints. Physics Letters A 307, 118–128 (2003)

4. Ghasabi-Oskoei, H., Mahdavi-Amiri, N.: An Efficient Simplified Neural Network for Solving Linear and Quadratic Programming Problems. Applied Mathematics and Computation 175, 452–464 (2006)
5. Yang, Y., Cao, J.: Solving Quadratic Programming Problems by Delayed Projection Neural Network. IEEE Trans. Neural Netw. 17, 1630–1634 (2006)
6. Barbarosou, M.P., Maratos, N.G.: A Nonfeasible Gradient Projection Recurrent Neural Network for Equality-Constrained Optimization Problems. IEEE Trans. Neural Netw. 19, 1665–1677 (2008)
7. Zhang, Y., Wang, J.: A Dual Neural Network for Convex Quadratic Programming Subject to Linear Equality and Inequality Constraints. Physics Letters A 298, 271–278 (2002)
8. Liu, S., Wang, J.: A Simplified Dual Neural Network for Quadratic Programming with Its KWTA Application. IEEE Trans. Neural Netw. 17, 1500–1510 (2006)
9. Hu, X., Wang, J.: Solving Generally Constrained Generalized Linear Variational Inequalities Using the General Projection Neural Networks. IEEE Trans. Neural Netw. 18, 1697–1708 (2007)
10. Liu, Q., Wang, J.: A One-Layer Recurrent Neural Network with a Discountinuous Hard-Limiting Activation Function for Quadratic Programming. IEEE Trans. Neural Netw. 19, 558–570 (2008)
11. Hu, X., Wang, J.: Design of General Projection Neural Networks for Solving Monotone Linear Variational Inequalities and Linear and Quadratic Optimization Problems. IEEE Trans. Syst. Man, Cybern. B 37, 1414–1421 (2007)
12. Forti, M., Nistri, P., Quincampoix, M.: Generalized Neural Network for Nonsmooth Nonlinear Programming Problems. IEEE Trans. Circuits Syst. I 51, 1741–1754 (2004)
13. Xia, Y.: An Extended Projection Neural Network for Constrained Optimization. Neural Computation 16, 863–883 (2004)
14. Gao, X.: A Novel Neural Network for Nonlinear Convex Programming. IEEE Trans. Neural Netw. 15, 613–621 (2004)
15. Hu, X., Zhang, B.: A New Recurrent Neural Network for Solving Convex Quadratic Programming Problems with an Application to the k-Winners-Take-All Problem. IEEE Trans. Neural Netw. (accepted)
16. Hu, X., Wang, J.: An Improved Dual Neural Network for Solving a Class of Quadratic Programming Problems and Its K-Winners-Take-All Application. IEEE Trans. Neural Netw. 19, 2022–2031 (2008)
17. Wang, Z., Perterson, B.S.: Constrained Least Absolute Deviation Neural Networks. IEEE Trans. Neural Netw. 19, 273–283 (2008)
18. Kinderlehrer, D., Stampcchia, G.: An Introduction to Variational Inequalities and Their Applications. Academic, New York (1980)
19. Hitchcock, F.L.: The Distribution of a Product from Several Sources to Numerous Localities. J. Math. Phys. 20, 224–230 (1941)