Neurocomputing 202 (2016) 98–103

Contents lists available at ScienceDirect

Neurocomputing

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

# A new neural network for $l_1$-norm programming ☆

Cuiping Li, Xingbao Gao *, Yawei Li, Rui Liu

*School of Mathematics and Information Science, Shaanxi Normal University, Xi'an 710119, People's Republic of China*

ARTICLE INFO

*Article history:*
Received 18 August 2015
Received in revised form
9 March 2016
Accepted 25 March 2016
Communicated by Qingshan Liu
Available online 6 May 2016

*Keywords:*
Stability
Neural network
$l_1$-norm programming
Convergence

ABSTRACT

This paper presents a new neural network for solving $l_1$-norm problems with equality and box constraints by introducing a new vector. The proposed model is proved to be Lyapunov stable and converges to an exact optimal solution of the original problem for every starting point. Compared with some existing continuous-time neural networks, the proposed model has the fewest neurons and a low complexity. The simulation results show the validity and transient behavior of the proposed model.



## 1. Introduction

Consider $l_1$-norm problem:

$$\begin{cases} \min & \|x\|_1 \\ \text{s.t.} & Dx = d,\ l \le x \le h, \end{cases} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n)^T \in R^n$ is the decision vector, $D \in R^{m\times n}$, $d \in R^m$, $l, h \in R^n (l \le h)$, and $\|\cdot\|_1$ denotes $l_1$-norm, i.e., $\|x\|_1 = \sum_{i=1}^{n} |x_i|$.

$l_1$-norm optimization problem has many important applications in scientific and engineering fields including image restoration, signal processing, parameter estimation, blind source separation, system identification, time-frequency, time-delay estimation and so on (see [2,24,26,40] and the references therein). Due to the sparsity properties of the solution, $l_1$-norm optimization problem has been extensively studied (see [1,2,4,22,24–27,31,33,36]). In many engineering applications, (1) is usually desired to solve in real time. But traditional methods (see [5–7,9–12] and the references therein) cannot satisfy real-time requirement because the computing time for a solution is greatly dependent on the dimension and structure of the problem, and the complexity of the algorithm used [18]. On the contrary, neural network models can be implemented by VLSI and optical technologies, where the computational procedure is truly distributed and in parallel. Thus they are more competent for real-time applications than traditional numerical algorithms.

Since the milestone articles [7,8] were published, many neural networks for solving optimization problems have been developed (see [13–36,39,41–46]). Among them, models in [39,41,45,44] for constrained optimization problems can be applied to solve (1) and require $n$ state variables, but they are discontinuous. The model in [41] includes a penalty parameter and needs to compute the converse of some matrix. By means of the optimization conditions, Wang et al. [26] proposed a neural network to the problem (1) without box constraints, i.e., $h = -l = +\infty$. Even though this model has a good stability performance, it is hard to solve the problem (1) directly since the box constraints cannot be converted into linear equality constraints. Furthermore, using the idea in [26], Wang and Peterson [27] proposed a model to solve constrained least absolute deviation problems. However, the resulting model for the problem (1) requires $2m+4n$ state variables, and it could not guarantee the convergence to an optimal solution (see [24] and Example 1 in Section 3). Because the problem (1) can be formulated as a linear variational inequality problem (see [24] and Lemma 1 in Section 2), models in [13,14,16,18,24,30] can be applied to it, but each resulting model requires $m+2n$ state variables, and the stability of models in [14,16,18,30] cannot be guaranteed (see [17,24]). Moreover, even though the problem (1) can also be transformed into a minimax problem (see [26]), the model in [17] could not be used to the problem (1) since it is only designed to solve convex quadratic minimax problems without linear equation constraints. On the other hand, models in [24,31,32] can solve least absolute deviation problems and have a good stability, yet the model in [32] can be only used to solve the problem (1) without equality constraints, and the size of each model in [24] and [31] is $m+2n$. Same as its continuous-time version in [31], a discrete-time neural network in [36] also requires $m+2n$

☆This work is supported by the National Natural Science Foundation of China (No. 61273311).

* Corresponding author.
*E-mail address:* xinbaog@snnu.edu.cn (X. Gao).

http://dx.doi.org/10.1016/j.neucom.2016.03.042
0925-2312/

state variables. Thus, it is necessary for us to construct a new model for (1) with the fewest neurons and a good stability.

Based on the above analysis, we present a new neural network for (1) by introducing a new vector in this paper. The new model is proved to be Lyapunov stable, and can approach an exact optimal solution of (1) for any starting point. In contrast to existing models for the problem (1), the proposed model has the fewest state variables and a low complexity. The performance of the presented model is shown by three numerical simulation examples.

Throughout this paper, we assume that (1) has an optimal solution $x^*$, and satisfies the Slater condition [1], i.e., there is a point $\overline{x} \in R^n$ such that $D\overline{x} = d$ and $l_i < \overline{x}_i < h_i$ for $i = 1, 2, \ldots, n$. The projection operator $P_U$ on a closed convex set $U \subset R^n$ is defined as

$$P_U(u) = \operatorname*{argmin}_{v \in U} \|u - v\|,$$

where $\|u\| = (\sum_{i=1}^n u_i^2)^{1/2}$, and a basic property for it [4,19] is

$$[v - P_U(v)]^T [P_U(v) - w] \geq 0, \quad \forall v \in R^n, w \in U. \tag{2}$$

A neural network is said to be Lyapunov stable and globally asymptotically stable if the corresponding dynamical system is so [34].

The rest of this paper is organized as follows. Section 2 constructs a neural network to (1) and analyzes its stability and convergence. Numerical simulations and conclusion are found in Sections 3–4, respectively.

## 2. Model and stability

This section will build a model for (1) and analyze its stability.

### 2.1. Proposed model

To design a new model, we first prove the following results.

**Lemma 1.** *$x^*$ is an optimal solution of (1) if and only if there exists a $(\mu^*, s^*) \in R^m \times R^n$ such that*

$$\begin{cases} x^* = P_{\Omega_1}(x^* + D^T\mu^* - s^*) \\ s^* = P_{\Omega_2}(s^* + x^*), \ Dx^* = d, \end{cases} \tag{3}$$

*where $\Omega_1 = [l, h]$ and $\Omega_2 = \{x \in R^n \mid -1 \leq x_i \leq 1, \ i = 1, 2, \ldots, n\}$.*

**Proof.** Obviously, (1) can be transformed into

$$\begin{cases} \min & \|y\|_1 \\ \text{s.t.} & Dx = d, \ y = x, \ x \in \Omega_1, \end{cases} \tag{4}$$

and the Lagrange function for the above problem is

$$L(x, y, \mu, s) = \|y\|_1 - \mu^T(Dx - d) - s^T(y - x),$$

which is defined on $\Gamma = \Omega_1 \times R^n \times R^m \times R^n$. From the saddle point theorem in [2], $(x^*, y^*) \in R^{2n}$ is an optimal solution of (4) if and only if there is a $(\mu^*, s^*) \in R^{m+n}$ such that $(x^*, y^*, \mu^*, s^*)$ is a saddle point of $L(x, y, \mu, s)$ on $\Gamma$, i.e.,

$$L(x^*, y^*, \mu, s) \leq L(x^*, y^*, \mu^*, s^*) \leq L(x, y, \mu^*, s^*), \quad \forall (x, y, \mu, s) \in \Gamma. \tag{5}$$

From the first inequality of (5), we have

$$(\mu - \mu^*)^T(Dx^* - d) + (s - s^*)^T(y^* - x^*) \geq 0, \quad \forall (\mu, s) \in R^{m+n}.$$

Then $y^* = x^*$ and $Dx^* = d$. From the second inequality of (5), we get

$$\|y\|_1 - \|y^*\|_1 - (s^*)^T(y - y^*) + (x - x^*)^T(s^* - D^T\mu^*) \geq 0,$$

for all $(x, y) \in \Omega_1 \times R^n$. Thus

$$\begin{cases} \|y\|_1 \geq \|y^*\|_1 + (s^*)^T(y - y^*), \ \forall y \in R^n, \\ (x - x^*)^T(s^* - D^T\mu^*) \geq 0, \ \forall x \in \Omega_1. \end{cases}$$

So, $s^* \in \partial\|y^*\|_1$, and $x^* = P_{\Omega_1}(x^* + D^T\mu^* - s^*)$ by (2). Since $\|y^*\|_1 = \sum_{i=1}^n |y_i^*|$ and

$$\partial|y_i^*| \begin{cases} = 1, & \text{if } y_i^* > 0 \\ \in [-1, 1], & \text{if } y_i^* = 0 \\ = -1, & \text{if } y_i^* < 0, \end{cases}$$

for $i = 1, \ldots, n$, $s^* = P_{\Omega_2}(s^* + y^*)$ by $s^* \in \partial\|y^*\|_1$. Therefore (3) holds true.□

From Lemma 1, we know that the optimal solution of the problem (1) can be gotten by solving (3). Let $u^* = s^* + x^*$, then $s^* = P_{\Omega_2}(u^*)$, $x^* = u^* - P_{\Omega_2}(u^*)$, and we can obtain the following systems:

$$\begin{cases} u^* - P_{\Omega_2}(u^*) = P_{\Omega_1}[u^* - 2P_{\Omega_2}(u^*) + D^T\mu^*] \\ D[u^* - P_{\Omega_2}(u^*)] = d, \end{cases} \tag{6}$$

from (3). Clearly, double projections are made on the right-hand-side of the first equality of (6), i.e., first projection onto $\Omega_2$ and then onto $\Omega_1$. Moreover, the below lemma reveals the relationship between solutions of (1) and (6).

**Lemma 2.** *Let $\mathcal{K}^* = \{z = (u^T, \mu^T)^T \in R^{m+n} \mid z \text{ solves (6)}\}$. Then two statements are true.*

(i) *If $x$ is an optimal solution of (1), then there is a $z \in R^{m+n}$ such that $z \in \mathcal{K}^*$ with $x = u - P_{\Omega_2}(u)$.*
(ii) *If $z \in \mathcal{K}^*$, then $u - P_{\Omega_2}(u)$ is an optimal solution of (1).*

**Proof.** Obviously (i) holds by the above analysis.

(ii) Let $x = u - P_{\Omega_2}(u)$, then $P_{\Omega_2}(u) = P_{\Omega_2}[x + P_{\Omega_2}(u)]$, $Dx = d$ and $x = P_{\Omega_1}[x - P_{\Omega_2}(u) + D^T\mu]$ by (6), i.e., a pair $(x, P_{\Omega_2}(u), \mu)$ is a solution of (3). Thus $x = u - P_{\Omega_2}(u)$ is an optimal solution of (1) by Lemma 1.□

From Lemma 2, each optimal solution of (1) can be obtained by solving (6), and vice versa. Let $z = (u^T, \mu^T)^T$, then a neural network to solve (1) can be defined as follows:

- *State equation*

$$\frac{dz}{dt} = \frac{d}{dt}\begin{pmatrix} u \\ \mu \end{pmatrix} = -\rho \begin{pmatrix} x - P_{\Omega_1}(2x - u + D^T\mu) \\ DP_{\Omega_1}(2x - u + D^T\mu) - d \end{pmatrix} \tag{7}$$

- *Output equation*

$$x = u - P_{\Omega_2}(u), \tag{8}$$

where $\rho > 0$ is a scaling constant.

Obviously, $\mathcal{K}^*$ defined in Lemma 2 is the equilibrium point set of the proposed neural network (7), and two operators $P_{\Omega_1}(\cdot)$ and $P_{\Omega_2}(\cdot)$ could be easily implemented by using piecewise-activation functions [15]. One can see that the circuit realizing (7)–(8) includes $m+n$ integrators, $2mn$ connection weights, $m+n$ activation functions for $P_{\Omega_1}(\cdot)$ and $P_{\Omega_2}(\cdot)$, some amplifiers and adders. Thus (7)–(8) can be implemented in simple hardware units [19].

### 2.2. Model comparisons

This subsection shows the advantages of the proposed model (7)–(8) by comparing it with four existing models.

First, let us focus on two models in [24], whose state equations for (1) can be formulated as

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = -\rho \begin{pmatrix} x - P_{\Omega_1}(x - y + D^Tz) \\ 2[y - P_{\Omega_2}(y + P_{\Omega_1}(x - y + D^Tz))] \\ 2[DP_{\Omega_1}(x - y + D^Tz) - d] \end{pmatrix} \tag{9}$$