An analog neural network approach for the least absolute shrinkage and selection operator problem

Article *in* Neural Computing and Applications · February 2017

DOI: 10.1007/s00521-017-2863-5

4 authors, including:

Wang Hao
City University of Hong Kong
10 PUBLICATIONS **4** CITATIONS

Neural Comput & Applic
DOI 10.1007/s00521-017-2863-5

ICONIP 2015

An analog neural network approach for the least absolute shrinkage and selection operator problem

Hao Wang[1] · Ching Man Lee[1] · Ruibin Feng[1] · Chi Sing Leung[1]

Received: 30 June 2016 / Accepted: 25 January 2017

Abstract This paper addresses the analog optimization for non-differential functions. The Lagrange programming neural network (LPNN) approach provides us a systematic way to build analog neural networks for handling constrained optimization problems. However, its drawback is that it cannot handle non-differentiable functions. In compressive sampling, one of the optimization problems is least absolute shrinkage and selection operator (LASSO), where the constraint is non-differentiable. This paper considers the hidden state concept from the local competition algorithm to formulate an analog model for the LASSO problem. Hence, the non-differentiable limitation of LPNN can be overcome. Under some conditions, at equilibrium, the network leads to the optimal solution of the LASSO. Also, we prove that these equilibrium points are stable. Simulation study illustrates that the proposed analog model and the traditional digital method have the similar mean squared performance.

Keywords Analog neural network · Neural dynamics · LPNN · Local competition algorithm

✉ Chi Sing Leung
eeleungc@cityu.edu.hk

Hao Wang
hwang96-c@my.cityu.edu.hk

Ching Man Lee
cmlee34-c@my.cityu.edu.hk

Ruibin Feng
rfeng4-c@my.cityu.edu.hk

[1] Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong

1 Introduction

In the last three decades, many results about the analog neural network approach for nonlinear constrained optimization [1–3, 6, 6–8] were reported. The analog neural network approach is more preferable because it is able to provide the real-time solutions more effectively [1–3, 6–8]. In an analog neural network, the decision variables are stored in a number of neurons. Based on the given optimization problem, we define some differential equations, called neural dynamics, to manage the neurons' state. When the state of the network converges, we can obtain the solution by measuring the neuron outputs.

Since 1980's many analog neural network models were proposed for solving different kinds of engineering problems. The Hopfield model [4, 5] was demonstrated to have several applications, including analog-to-digital conversion, linear programming, and digit recognition. Also, the analog neural network approach is able to design microcode [9]. Another popular model is the cellular neural network [10–13] which has many applications in image processing. Besides, there are many analog neural network models for the winner-take-all process [8, 14–16].

Although many analog neural network models [6, 17, 18] were proposed to various nonlinear constrained optimization problems, many of them are designed for a particular optimization problem. For example, the model in [7] is designed for solving quadratic optimization. Their construction method is not applicable to general constrained optimization.

The Lagrange programming neural network (LPNN) framework [19–22] has the ability for handling general constrained optimization. Furthermore, with the augmented Lagrangian concept, the LPNN framework has the ability to handle non-convex objective function and non-convex

Published online: 20 February 2017

constraints. Recently, some new signal processing applications of the LPNN approach [20–22] were developed, including source localization and waveform design in radar systems. In these signal processing applications, the optimization problems are non-convex, and the LPNN approach is superior to the traditional numerical approaches. However, the original LPNN approach cannot handle non-differentiable objective functions and constraints.

Compressive sampling [23] is an important area in signal processing. It can resume an unknown sparse vector. It can be used in many areas, such as noise removal [25, 26] and identification of channel parameters [24]. One of the characteristics in compressive sampling is that the objective functions or constraints are usually non-differentiable. One of the representative problems in compressive sampling is the least absolute shrinkage and selection operator (LASSO) problem. Mathematically, the LASSO problem concerns to minimize the squared error in a regression problem, subject to an l_1-norm constraint on the recovered coefficients.

There are some digital methods (numerical algorithms) for solving the LASSO problem [27–32]. In the L1-regularized least squares (L1LS) method [27] and the primal-dual interior method for convex objectives (PDCO) method [28], the core part is the interior-point method. They introduce a trade-off parameter to convert the LASSO problem into the unconstrained basis pursit denoising (BPDN) problem [34]. Finally, they solve the BPDN based on the interior-point concept.

In the gradient projection for sparse reconstruction (GPSR) [29] method and the spectral projected gradient (SPG) method [30–32], the trade-off parameter of the BPDN problem is tuned to an appropriate value, such that the objective value is minimized and the constraint is satisfied. At each iteration, they perform the gradient projection to minimize the unconstrained objective function.

The local competition algorithm (LCA) [33] is an analog neural approach for compressive sampling. It introduces the internal state concept to avoid the computation of the derivative of the objective function at non-differentiable points. However, it is able to solve the unconstrained BPDN only. Hence, it would be significance to develop a LPNN model for non-differentiable constraints and non-differentiable functions.

We propose a neural model for the LASSO problem on the basis of the LPNN framework. For the proposed LASSO–LPNN model, we show that at equilibrium, the state of the LASSO–LPNN corresponds to the optimal solution for the LASSO problem. Furthermore, we show that the equilibrium points are stable. The preliminary result was reported in a conference [35].

The organization of the paper is as follows. The background on analog optimization is reviewed in Sect. 2. The proposed LASSO–LPNN model is then presented in Sect. 3. Section 4 presents the properties of the proposed model. Section 5 presents the simulation result. Section 6 concludes the paper.

2 Background on analog optimization

2.1 LPNN

One of the neural approaches for optimization is LPNN [19–22]. This approach gives us a standard procedure for building an analog neural network to handle nonlinear constrained optimization. Let $\boldsymbol{x} \in \mathbb{R}^n$ be the decision variable vector containing n elements. A general optimization with m equality constraints is expressed as

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \quad \text{s.t. } \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{0}. \tag{1}$$

where $\boldsymbol{0}$ is a zero vector, $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, and $\boldsymbol{\phi} : \mathbb{R}^n \to \mathbb{R}^m$ $(m<n)$ is a vector valued function that defines m equality constraints. In the original LPNN framework, both the constraints and the objective function should be twice differentiable. To construct the analog network, we first define the Lagrangian function, given by

$$\mathcal{L}_{lp} = f(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}), \tag{2}$$

where $\boldsymbol{\lambda}$ is an m-dimensional vector containing m Lagrange multipliers. The network has two types of neurons. One type is called variable neuron. Another one is called Lagrange neuron. We use the variable neurons to store the decision variables, $\{x_1, \ldots, x_n\}$, and use the Lagrange neurons to store the Lagrange multipliers $\{\lambda_1, \ldots, \lambda_m\}$. Their dynamics are developed from the Lagrangian function, given by

$$\frac{1}{\epsilon}\frac{d\boldsymbol{x}}{dt} = -\frac{\partial \mathcal{L}_{lp}}{\partial \boldsymbol{x}}, \tag{3}$$

$$\frac{1}{\epsilon}\frac{d\boldsymbol{\lambda}}{dt} = \frac{\partial \mathcal{L}_{lp}}{\partial \boldsymbol{\lambda}}, \tag{4}$$

where ϵ is the characteristics time. It depends on the impedance of the neural circuit. In order to simplify the notation, we set ϵ equal to 1. Equation (3) aims at minimizing the objective value, while Eq. (4) aims at constraining the variable state vector in the feasible region. With (3) and (4), the state of a LASSO–LPNN converges to an equilibrium point [19–22] if the network fulfills some mild conditions. In the above formulation, the objective function and the constraints should be differentiable.

2.2 Compressive sampling

In compressive sampling, we have an unknown vector $\boldsymbol{x} \in \mathbb{R}^n$. Furthermore, the vector is sparse. We cannot

directly measure $\boldsymbol{x}$. Instead, we have an observation vector $\boldsymbol{r} \in \mathbb{R}^m$ only, given by

$$\boldsymbol{r} = \boldsymbol{\Phi x} + \boldsymbol{\xi}, \tag{5}$$

where $\boldsymbol{\Phi}$ is an $m \times n$ matrix and $m < n$. In compressive sampling, we call it measurement matrix. The vector $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_m]^{\mathrm{T}}$ is the measurement noise vector.

To recover $\boldsymbol{x}$, we consider the LASSO problem:

$$\min_{\boldsymbol{x}} \|\boldsymbol{r} - \boldsymbol{\Phi x}\|_2^2, \quad \text{s.t. } \|\boldsymbol{x}\|_1 \le \eta, \tag{6}$$

where $\eta > 0$. The LASSO problem aims at minimizing the residue error, with the constraint that the l_1-norm of the estimated signal is less than a given value. In the LASSO problem, the constrain "$\|\boldsymbol{x}\|_1 - \eta$" is not differentiable. We cannot use the conventional LPNN approach to solve it directly.

2.3 Subdifferential and LCA

Since this paper involves non-differentiable convex functions, we first introduce the subdifferential notation. The subdifferential is the generalization version of the derivative for non-differentiable convex functions [37, 38]. It can be summarized by the following two definitions.

Definition 1 The subgradient $\boldsymbol{\rho}$ of a convex function $\hbar$ at $\boldsymbol{x}$ in the domain of $\hbar$ is defined by the following inequality:

$$\hbar(\boldsymbol{y}) \ge \boldsymbol{\rho}^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{x}) + \hbar(\boldsymbol{x}), \ \forall \boldsymbol{y} \in \mathbf{dom} \quad \hbar. \tag{7}$$

Definition 2 The subdifferential is the collection of all subgradients of $\hbar(\boldsymbol{x})$ at $\boldsymbol{x}$, given by

$$\partial\hbar(\boldsymbol{x}) = \{\boldsymbol{\rho} \,|\, \boldsymbol{\rho}^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{x}) \le f(\boldsymbol{y}) - \hbar(\boldsymbol{x}), \ \forall \boldsymbol{y} \in \mathbf{dom} \quad \hbar\}. \tag{8}$$

Here, we use a simple function $\hbar(x) = |x|$ to explain the subdifferential concept. For x not equal to 0, the subgradient of $\hbar(x)$ equals $\mathrm{sign}(x)$ and the subdifferential equals $\mathrm{sign}(x)$ too. For x equal to 0, the subgradient of $\hbar(x)$ is any value between -1 and 1, and the subdifferential of $|x|$ is equal to the set of points between -1 and 1, denoted as $[-1, 1]$.

Similarly, given the l_1-norm $\hbar(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ of $\boldsymbol{x} \in \mathbb{R}^n$, the subdifferential is

$$\partial\|\boldsymbol{x}\|_1 = \mathcal{J}_1 \times \cdots \times \mathcal{J}_n, \tag{9}$$

$$\mathcal{J}_i = \begin{cases} [-1, 1] & x_i = 0, \\ \mathrm{sign}(x_i) & x_i \ne 0. \end{cases} \tag{10}$$

For instance, for $\boldsymbol{x} \in \mathbb{R}^3$ and $\boldsymbol{\hbar}(\boldsymbol{x}) = |x_1| + |x_2| + |x_3|$, the subdifferential at $\boldsymbol{x} = [2, -3, 0]^{\mathrm{T}}$ is equal to

$$\partial\boldsymbol{\hbar}(\boldsymbol{x})|_{[2,-3,0]} = \begin{pmatrix} 1 \\ -1 \\ [1, -1] \end{pmatrix}. \tag{11}$$

The objective function of the LCA is

$$\mathcal{L}_{lca} = \frac{1}{2}\|\boldsymbol{r} - \boldsymbol{\Phi x}\|_2^2 + \kappa\|\boldsymbol{x}\|_1. \tag{12}$$

In (12) κ is a weighting factor between the two cost terms: $\|\boldsymbol{r} - \boldsymbol{\Phi x}\|_2^2$ and $\|\boldsymbol{x}\|_1$. For a large κ, we would like to have a solution with more zero elements. For a small κ, we would like to have a solution with small approximation error.

In the LCA, we use n neurons to store $\boldsymbol{x}$. The neuron outputs are denoted as $\boldsymbol{x}$. Since the objective function is non-differentiable, we cannot set the dynamics as

$$\frac{d\boldsymbol{x}}{dt} = -\frac{\partial\mathcal{L}_{lca}}{\partial\boldsymbol{x}} \tag{13}$$

to minimize the objective function.

Instead, the LCA introduces the internal state vector $\boldsymbol{u}$ for the neurons. The mapping from the internal state to the neuron output is defined by an activation function $T_\kappa(\cdot)$:

$$x_i = T_\kappa(u_i) = \begin{cases} 0, & \text{for} \quad |u_i| \le \kappa, \\ u_i - \kappa\mathrm{sign}(u_i), & \text{for} \quad |u_i| > \kappa. \end{cases} \tag{14}$$

Some examples of the threshold are shown in Fig. 1. Clearly, the mapping $T_\kappa(\cdot)$ is well defined. When the internal state does not equal zero, the mapping is one-to-one. When the internal state equals zero, the mapping is many-to-one.

However, given the output state only, it is difficult to determine the hidden state. It is because when $x_i = 0$, the inverse mapping $T_\kappa^{-1}(0)$ is one-to-many. In other words, $T_\kappa^{-1}(0)$ is equal to a set, given by $[-\kappa, \kappa]$.

In the LCA, instead of using the output to define the dynamics, we use the internal state to define the dynamics, given by

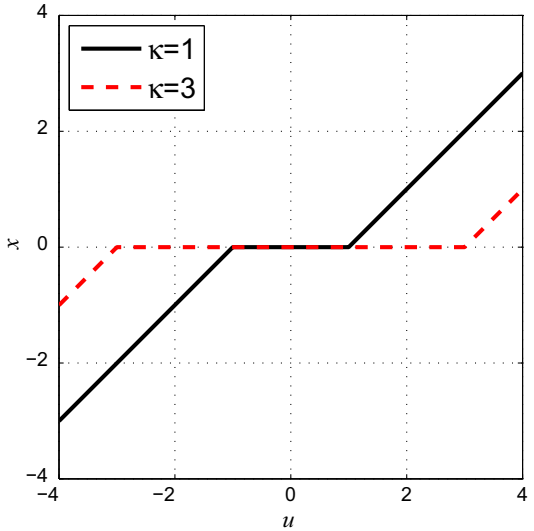

Fig. 1 Some examples of the threshold function $T_\kappa(u_i)$

$$\frac{d\boldsymbol{u}}{dt} = -\partial\mathcal{L}_{lca} = -\kappa\partial\|\boldsymbol{x}\|_1 + \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}). \tag{15}$$

From the subdifferential concept [33] and the definition of the activation function $T_\kappa(\cdot)$, one can easily show that given a known $\boldsymbol{x}$ without any *a prior* information on $\boldsymbol{u}$, we obtain

$$\boldsymbol{u} - \boldsymbol{x} = \kappa\partial\|\boldsymbol{x}\|_1. \tag{16}$$

On the other hand, given a known $\boldsymbol{u}$, we have

$$\boldsymbol{u} - \boldsymbol{x} \in \kappa\partial\|\boldsymbol{x}\|_1. \tag{17}$$

With (15) and (16), the dynamics of the LCA are given by

$$\frac{d\boldsymbol{u}}{dt} = -\boldsymbol{u} + \boldsymbol{x} + \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}). \tag{18}$$

It should be noticed that without introducing the internal state $\boldsymbol{u}$, we cannot realize $\partial\|\boldsymbol{x}\|_1$ (it equals a set when $\boldsymbol{x}$ contains zeros elements). The LCA has the ability for handling non-differentiable functions, but it cannot solve constrained problems.

3 LASSO–LPNN

3.1 Properties of LASSO

Before we develop our LASSO–LPNN model, we first study the properties of the LASSO problem. Recall that the LASSO problem is given by

$$\min_{\boldsymbol{x}} \|\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2, \ \text{ s.t. } \|\boldsymbol{x}\|_1 \le \eta. \tag{19}$$

Since the constraint and the objective function are convex, we have the following well-known convex optimization theorem [38, 39].

Theorem 1 *For the problem, stated in* (19), $\boldsymbol{x}^\star$ *is an optimal solution, if and only if, there is a* $\lambda^\star$ *and*

$$\boldsymbol{0} \in -2\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}^\star) + \lambda^\star\partial\|\boldsymbol{x}^\star\|_1, \tag{20a}$$

$$\lambda^\star \ge 0, \tag{20b}$$

$$\|\boldsymbol{x}^\star\|_1 - \eta \le 0, \tag{20c}$$

$$\lambda^\star(\|\boldsymbol{x}^\star\|_1 - \eta) = 0. \tag{20d}$$

Note that (20) *summarizes the Karush–Kuhn–Tucker (KKT) conditions of* (19). *Since the problem is convex, the KKT conditions are sufficient and necessary*.

The constraint in the LASSO problem (19) is an inequality; the LPNN approach cannot be used for solving it directly. However, when η^2 is less than a certain value, given by

$$\eta^2 < \frac{\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{r}\|_2^2}{\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\|_2^2}, \tag{21}$$

the constraint is reduced to an equality constraint. This property is summarized in Theorem 2.

Theorem 2 *Given that* $\eta^2 < \frac{\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{r}\|_2^2}{\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\|_2^2}$, *the optimization problem* (19) *becomes*

$$\min_{\boldsymbol{x}} \|\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2, \ \text{ s.t. } \|\boldsymbol{x}\|_1 = \eta. \tag{22}$$

Furthermore, a vector $\boldsymbol{x}^\star$ *is the minimum solution of* (22), *if and only if, there is a* $\lambda^\star$ *and*

$$\boldsymbol{0} \in -2\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}x^\star) + \lambda^\star\partial|\boldsymbol{x}^\star|_1, \tag{23a}$$

$$\lambda^\star > 0, \tag{23b}$$

$$|\boldsymbol{x}^\star|_1 - \eta = 0. \tag{23c}$$

It should be noticed that (23a) *states the KKT conditions, which are sufficient and necessary*.

Proof Let $(\boldsymbol{x}^\star, \lambda^\star)$ be the optimal solution. From (20b), we obtain $\lambda^\star$ is greater than or equal to zero. We first show that if $\eta^2 < \frac{\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{r}\|_2^2}{\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\|_2^2}$, then $\lambda^\star$ cannot equal zero. The contradiction method is used in our proof.

As $(\boldsymbol{x}^\star, \lambda^\star)$ is the optimal solution, the KKT conditions (20) of Theorem 1 are fulfilled. If $\lambda^\star = 0$, then from (20a) we have

$$\begin{aligned} \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{x}^\star &= \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{r} \\ \|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{x}^\star\|_2^2 &= \|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{r}\|_2^2. \end{aligned} \tag{24}$$

The above implies that

$$\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\|_2^2\|\boldsymbol{x}^\star\|_2^2 \ge \|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{r}\|_2^2. \tag{25}$$

To sum up, "$\lambda^\star = 0$" implies

$$\|\boldsymbol{x}^\star\|_2^2 \ge \frac{\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{r}\|_2^2}{\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\|_2^2}. \tag{26}$$

On the other hand, from (20c), we have

$$\eta^2 \ge \|\boldsymbol{x}^\star\|_1^2 = \left(\sum_{i=1}^{n} |x_i^\star|\right)^2. \tag{27}$$

Furthermore,

$$\begin{aligned} \left(\sum_{i=1}^{n} |x_i^\star|\right)^2 &= \sum_{i=1}^{n} {x_i^\star}^2 + \sum_{i=1}^{n}\sum_{i\neq j}^{n} |x_i^\star||x_j^\star| \\ &\ge \sum_{i=1}^{n} {x_i^\star}^2 \\ &\ge \|\boldsymbol{x}^\star\|_2^2. \end{aligned} \tag{28}$$

From (26) and (28),

$$\eta^2 \geq \|\boldsymbol{x}^\star\|_2^2 \Rightarrow \eta^2 \geq \frac{\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{r}\|_2^2}{\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\|_2^2}. \tag{29}$$

Inequality (29) contradicts with the early assumption of $\eta^2 < \frac{\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{r}\|_2^2}{\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\|_2^2}$. As a result, $\lambda^\star$ is not equal to zero. That means, $\lambda^\star$ must be greater than zero. Hence, we can simplify the KKT conditions of Theorem 1 to

$$\boldsymbol{0} \in -2\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}x^\star) + \lambda^\star \partial\|\boldsymbol{x}^\star\|_1, \tag{30a}$$

$$\lambda^\star > 0, \tag{30b}$$

$$\|\boldsymbol{x}^\star\|_1 - \eta = 0. \tag{30c}$$

Besides, from (30c), we can remove the inequality of (19). Therefore, we can express the original problem (19) as

$$\min_{\boldsymbol{x}} \|\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2, \quad \text{s.t. } \|\boldsymbol{x}\|_1 = \eta. \tag{31}$$

The proof is complete. □

3.2 Dynamics of LASSO–LPNN

From (22) in Theorem 2, the Lagrangian function is given by

$$\mathcal{L} = \|\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \lambda(\|\boldsymbol{x}\|_1 - \eta), \tag{32}$$

Since there is one constrain only, the Lagrange multiplier of the LASSO–LPNN is a scalar.

The gradients of $\mathcal{L}$ are given by

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{x}} = -2\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}) + \lambda\partial\|\boldsymbol{x}\|_1, \tag{33}$$

$$\frac{\partial\mathcal{L}}{\partial\lambda} = \|\boldsymbol{x}\|_1 - \eta. \tag{34}$$

Note that $\lambda\partial\|\boldsymbol{x}\|_1$ cannot be realized directly. Hence, we introduce the hidden state concept, by defining an internal state vector $\boldsymbol{u}$. We use n variable neurons to store $\boldsymbol{x}$ and $\boldsymbol{u}$, where $\boldsymbol{x}$ is the neuron output and $\boldsymbol{u}$ is the neuron internal state. From (14), we have the following relationship between $\boldsymbol{u}$ and $\boldsymbol{x}$, given by

$$x_i = T_1(u_i) = \begin{cases} 0, & \text{for} \quad |u_i| \leq 1, \\ u_i - \text{sign}(u_i), & \text{for} \quad |u_i| > 1. \end{cases} \tag{35}$$

Besides, there is one Lagrange neuron to hold the Lagrange multiplier λ.

In the LASSO–LPNN, we define the neuron dynamics as

$$\frac{d\boldsymbol{u}}{dt} = -\frac{\partial\mathcal{L}}{\partial\boldsymbol{x}} = 2\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}) - \lambda\partial\|\boldsymbol{x}\|_1, \tag{36a}$$

$$\frac{d\lambda}{dt} = \frac{\partial\mathcal{L}}{\partial\lambda} = \|\boldsymbol{x}\|_1 - \eta. \tag{36b}$$

From (16), we can replace $\partial\|\boldsymbol{x}\|_1$ with $\boldsymbol{u} - \boldsymbol{x}$. Finally, we contain the dynamics:

$$\frac{d\boldsymbol{u}}{dt} = 2\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}) - \lambda(\boldsymbol{u} - \boldsymbol{x}) \tag{37a}$$

$$\frac{d\lambda}{dt} = \|\boldsymbol{x}\|_1 - \eta. \tag{37b}$$

Figure 2 shows the realization of a simple LASSO–LPNN. In this simple network, there are one Lagrange neuron and four variable neurons. From the current states of the Lagrange and variable neurons, $\{\boldsymbol{x}(t), \boldsymbol{u}(t), \lambda(t)\}$, each variable neuron realizes the time differentiation du_i/dt first. The signal is then fed into an integrator. At the output of the integrator, we can obtain the internal state $u_i(t)$. To obtain the output $x_i(t)$, we apply the activation function on $u_i(t)$. There is one Lagrange neuron in a LASSO–LPNN. It realizes the time differentiation $d\lambda/dt$ first. Similar to the variable neuron, the signal is then fed into an integrator. At the output of the integrator, we obtain $\lambda(t)$. For the computer simulation, we can use the following two different equations to update the neurons, given by

$$\boldsymbol{u}(t + \Delta_t) = \boldsymbol{u}(t) + \Delta_t \frac{d\boldsymbol{u}}{dt}, \tag{38a}$$

$$\lambda(t + \Delta_t) = \lambda(t) + \Delta_t \frac{d\lambda}{dt}, \tag{38b}$$

where Δ_t is a small positive real number. it can be fixed or adaptive during the simulation.

We use Fig. 3 to show the simple result of the LASSO–LPNN. Figure 3a show an 1D signal with length equal to 128. This signal has five nonzero components only. Suppose that we use 30 measurements. In this example, we use an ± 1 random matrix as the measurement matrix. Figure 3b shows the measurement signal. When we use the pseudoinverse approach, we cannot obtain our expected signal, as shown in Fig. 3c. From Fig. 3d, when the

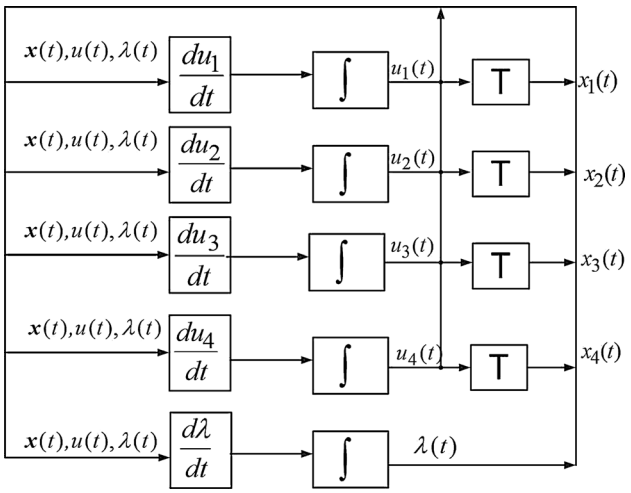

Fig. 2 Realization of the LASSO–LPNN

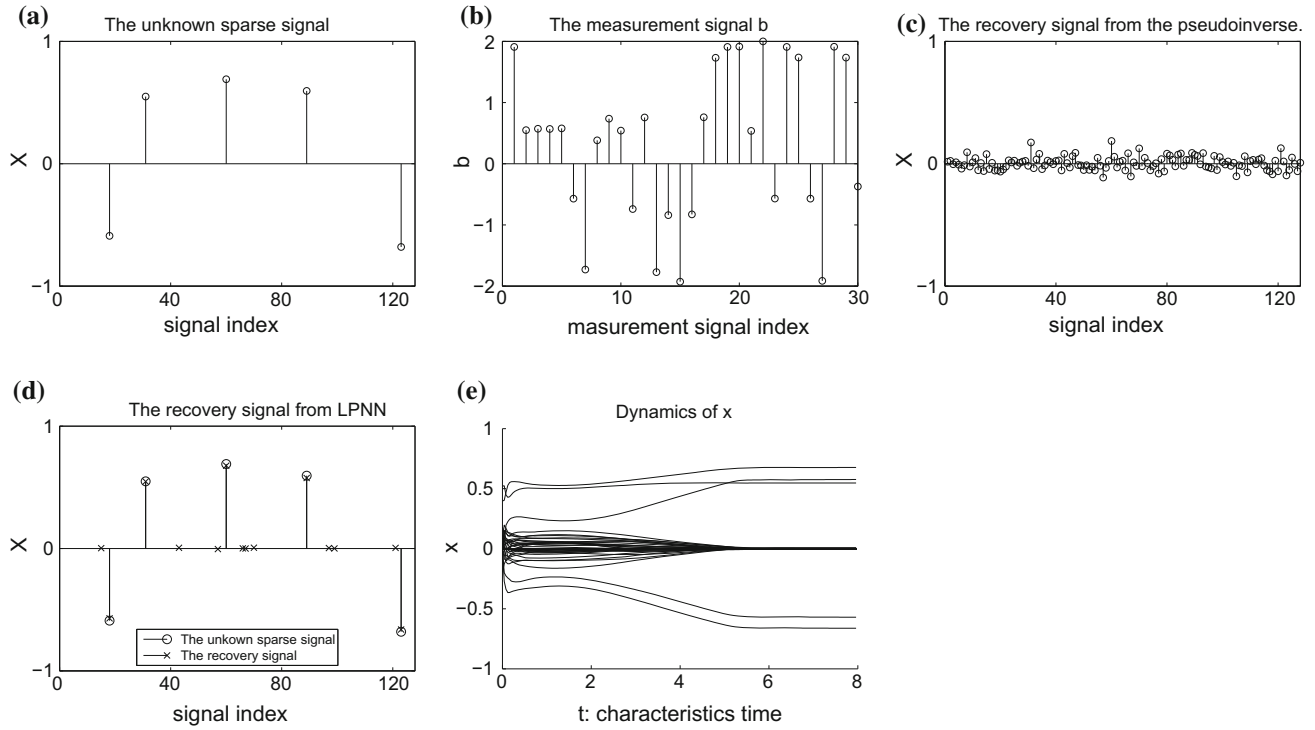

Fig. 3 Simple example. **a** The unknown sparse signal. **b** The measurement signal. **c** The recovered signal from pseudoinverse. **d** The recovered signal from the LASSO–LPNN. **e** The dynamics of LPNN. In the figures, we show nonzero elements only

LASSO–LPNN approach is used, we obtain a much better recovery signal. Figure 3e shows the dynamics of the variable neurons. From the figure, the LASSO–LPNN converges to a stable state in 5 time units.

4 Properties of LPNN

In analog optimization, we are interested in two questions. First, we are interested in whether an equilibrium point of the LASSO–LPNN corresponds to the optimal solutions of the LASSO problem or not. Second, we are interested in whether an equilibrium point is stable or not.

We first present Theorem 3 which shows that an equilibrium point of the network fulfills the KKT conditions (30) of Theorem 2. Since the KKT conditions of Theorem 2 are necessary and sufficient, the equilibrium point of the network achieves the optimal solution of the LASSO problem.

Theorem 3 *Let* $\{\boldsymbol{u}^\star, \lambda^\star\}$ *be the equilibrium state of the LASSO–LPNN. Furthermore,* $\lambda^\star > 0$. *Let* $\boldsymbol{x}^\star$ *be the output of the hidden state* $\boldsymbol{u}^\star$. *At* $\{\boldsymbol{u}^\star, \boldsymbol{x}^\star, \lambda^\star\}$, *the KKT conditions in* (23a) *are fulfilled. Therefore* $\boldsymbol{x}^*$ *is an optimal solution.*

Proof From (37), at equilibrium,

$$\left.\frac{d\boldsymbol{u}}{dt}\right|_{(\boldsymbol{u}^\star,\lambda^\star)} = 2\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}^\star) - \lambda^\star(\boldsymbol{u}^\star - \boldsymbol{x}^\star) = 0\ , \tag{39}$$

$$\left.\frac{d\lambda}{dt}\right|_{(\boldsymbol{u}^\star,\lambda^\star)} = \|\boldsymbol{x}^\star\|_1 - \eta = 0. \tag{40}$$

Now we need to prove that (39) and (40) lead to the KKT conditions (23a) of Theorem 2. From (17), we have

$$2\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}^\star) - \lambda^\star(\boldsymbol{u}^\star - \boldsymbol{x}^\star) \in 2\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}^\star) - \lambda^\star \partial\|\boldsymbol{x}^\star\|. \tag{41}$$

Furthermore, from (39), we obtain

$$0 \ \in 2\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}\boldsymbol{x}^\star) - \lambda^\star \partial\|\boldsymbol{x}^\star\|. \tag{42}$$

Hence, we obtain that (39) leads to (23aa). From the given condition that $\lambda^\star$ is greater than zero, we obtained (23ab). Besides, (40) is identical to (23ac). To sum up, the KKT conditions (23a) are satisfied. Since the problem is convex, the KKT conditions are sufficient and necessary. Therefore, an equilibrium, with $\lambda^\star > 0$, corresponds to an optimal solution. We complete the proof. □

Another issue, that we should consider, is whether the equilibrium point is asymptotically stable or not. To discuss the stability, we follow the concept of active neurons and inactive neurons in the LCA [36, 37]. Their formal definitions are given as follows.

Definition 3 For the active neurons, the magnitudes of their internal states u_i's are greater than or equal to 1 and

the outputs x_i's are strictly increasing with the internal states. The collection Γ of indices of the active neurons is denoted as $\Gamma = \{i \in [1,n], |u_i| > 1\}$. Also, we define $\boldsymbol{\Phi}_\Gamma$ as the matrix constructed from the columns of $\boldsymbol{\Phi}$ indexed by Γ. With respect to the active neurons, their outputs $\boldsymbol{x}_i$'s are strictly increasing with $\boldsymbol{u}_i$,i.e.,

$$\frac{d\boldsymbol{x}_i}{d\boldsymbol{u}_i} = 1 \ \forall i \in \Gamma. \tag{43}$$

Definition 4 For the inactive neurons, the magnitudes of their internal states u_i's are less than 1 and the corresponding outputs x_i's are equal to 0. The collection Γ^c of indices of the inactive neurons is denoted as $\Gamma^c = \{i \in [1,n], |u_i| \leq 1\}$. Also, we define $\boldsymbol{\Phi}_{\Gamma^c}$ as the matrix constructed from the columns of $\boldsymbol{\Phi}$ indexed by Γ^c. For the inactive neurons, their outputs $\boldsymbol{x}_i$'s are equal to zero,

$$\frac{d\boldsymbol{x}_i}{d\boldsymbol{u}_i} = 0 \ \forall i \in \Gamma^c. \tag{44}$$

With the notations of active neurons and inactive neurons, Theorem 4 tells us that the equilibrium point is asymptotically stable.

Theorem 4 *Given an equilibrium point,* $\{\boldsymbol{u}^\star, \lambda^\star\}$ *with* $\boldsymbol{u}^\star \neq 0$ *and* $\lambda^\star > 0$, *the point is asymptotically stable.*

Proof After we introduce Definitions 3 and 4, we can split the LASSO–LPNN dynamics (37) into three equations:

$$\frac{d\boldsymbol{u}_\Gamma}{dt} = 2\boldsymbol{\Phi}_\Gamma^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}_\Gamma \boldsymbol{x}_\Gamma) + \lambda(\boldsymbol{x}_\Gamma - \boldsymbol{u}_\Gamma), \tag{45a}$$

$$\frac{d\lambda}{dt} = \|\boldsymbol{x}\|_1 - \eta, \tag{45b}$$

$$\frac{d\boldsymbol{u}_{\Gamma^c}}{dt} = 2\boldsymbol{\Phi}_{\Gamma^c}^{\mathrm{T}}(\boldsymbol{r} - \boldsymbol{\Phi}_\Gamma \boldsymbol{x}_\Gamma) + \lambda(\boldsymbol{x}_{\Gamma^c} - \boldsymbol{u}_{\Gamma^c}), \tag{45c}$$

At the equilibrium point $(\boldsymbol{u}^\star, \boldsymbol{x}^\star)$, the linearization of (45) is expressed as

$$\left[\begin{array}{c} \frac{d\boldsymbol{u}_\Gamma}{dt} \\ \frac{d\lambda}{dt} \\ \frac{d\boldsymbol{u}_{\Gamma^c}}{dt} \end{array}\right]\Bigg|_{(\boldsymbol{u}^\star, \lambda^\star)} = -\lambda^\star \boldsymbol{A} \left[\begin{array}{c} \boldsymbol{u}_\Gamma - \boldsymbol{u}_\Gamma^\star \\ \lambda - \lambda^\star \\ \boldsymbol{u}_{\Gamma^c} - \boldsymbol{u}_{\Gamma^c}^\star \end{array}\right], \tag{46}$$

where

$$\boldsymbol{A} = \left[\begin{array}{c|c|c} \frac{2}{\lambda^\star}\boldsymbol{\Phi}_\Gamma^{\mathrm{T}}\boldsymbol{\Phi}_\Gamma & \frac{1}{\lambda^\star}(\boldsymbol{u}_\Gamma^\star - \boldsymbol{x}_\Gamma^\star) & \varnothing_{n_a \times (n-n_a)} \\ \hline \frac{1}{\lambda^\star}(\boldsymbol{x}_\Gamma^{\star\,\mathrm{T}} - \boldsymbol{u}_\Gamma^{\star\,\mathrm{T}}) & 0 & \varnothing_{1\times(n-n_a)} \\ \hline \frac{2}{\lambda^\star}\boldsymbol{\Phi}_{\Gamma^c}^{\mathrm{T}}\boldsymbol{\Phi}_\Gamma & \frac{1}{\lambda^\star}(\boldsymbol{u}_{\Gamma^c}^\star - \boldsymbol{x}_{\Gamma^c}^\star) & \boldsymbol{I}_{(n-n_a)\times(n-n_a)} \end{array}\right]. \tag{47}$$

where n_a is the number of active neurons, $\varnothing$ is a zero matrix. Its size is indicated by its subscript. Also, $\boldsymbol{I}$ is an identity matrix. Its size is indicated by its subscript. From the classical control theory, when the real parts of all the eigenvalues of $\boldsymbol{A}$ are positive, the equilibrium point is asymptotically stable.

Let us first expresses $\boldsymbol{A}$ as

$$\boldsymbol{A} = \left[\begin{array}{c|c} \boldsymbol{B} & \varnothing_{(n_a+1)\times(n-n_a)} \\ \hline \boldsymbol{C} & \boldsymbol{I}_{(n-n_a)\times(n-n_a)} \end{array}\right] \tag{48}$$

where

$$\boldsymbol{B} = \left[\begin{array}{c|c} \frac{2}{\lambda^\star}\boldsymbol{\Phi}_\Gamma^{\mathrm{T}}\boldsymbol{\Phi}_\Gamma & \frac{1}{\lambda^\star}(\boldsymbol{u}_\Gamma^\star - \boldsymbol{x}_\Gamma^\star) \\ \hline \frac{1}{\lambda^\star}(\boldsymbol{x}_\Gamma^{\star\,\mathrm{T}} - \boldsymbol{u}_\Gamma^{\star\,\mathrm{T}}) & 0 \end{array}\right], \tag{49}$$

and

$$\boldsymbol{C} = \left[\begin{array}{c|c} \frac{2}{\lambda^\star}\boldsymbol{\Phi}_{\Gamma^c}^{\mathrm{T}}\boldsymbol{\Phi}_\Gamma & \frac{1}{\lambda^\star}(\boldsymbol{u}_{\Gamma^c}^\star - \boldsymbol{x}_{\Gamma^c}^\star) \end{array}\right], \tag{50}$$

In the following, we first prove that the real parts of all the eigenvalues of $\boldsymbol{B}$ are positive. Based on this property of $\boldsymbol{B}$, the real parts of all the eigenvalues of $\boldsymbol{A}$ are positive too. Therefore, at equilibrium, the state is asymptotically stable.

Eigenvalues of $\boldsymbol{B}$

(Rank of $\boldsymbol{B}$:)

Firstly, "$\lambda^\star > 0$" and "$\boldsymbol{u}^\star \neq 0$" imply that $\frac{1}{\lambda^\star}(\boldsymbol{u}_\Gamma^\star - \boldsymbol{x}_\Gamma^\star)$ is not a zero vector. Also, $\frac{2}{\lambda^\star}\boldsymbol{\Phi}_\Gamma^{\mathrm{T}}\boldsymbol{\Phi}_\Gamma$ is positive or semi-positive definite. Suppose that n_a is the number of active neurons, and assume that $m > n_a$. It should be noticed that in compressive sampling, the number m of measurements is usually much greater than the number n_a of nonzero elements

Clearly, if the rank of $\boldsymbol{\Phi}_\Gamma$ is equal to n_a, then $\boldsymbol{\Phi}_\Gamma^{\mathrm{T}}\boldsymbol{\Phi}_\Gamma$ is positive definite and the rank of $\boldsymbol{\Phi}_\Gamma^{\mathrm{T}}\boldsymbol{\Phi}_\Gamma$ is equal to n_a too. From the fact [40] that if the elements of $\boldsymbol{\Phi}$ are independently identical normal random variables, then the probability that the rank of $\boldsymbol{\Phi}_\Gamma$ equals to n_a is very close to 1. It should be noticed that when we use ± 1 random matrix, the probability that rank of $\boldsymbol{\Phi}_\Gamma$ equals n_a tends 1 for large m, such as $m > 40$.

Recall from (49) that

$$\boldsymbol{B} = \left[\begin{array}{c|c} \frac{2}{\lambda^\star}\boldsymbol{\Phi}_\Gamma^{\mathrm{T}}\boldsymbol{\Phi}_\Gamma & \frac{1}{\lambda^\star}(\boldsymbol{u}_\Gamma^\star - \boldsymbol{x}_\Gamma^\star) \\ \hline \frac{1}{\lambda^\star}(\boldsymbol{x}_\Gamma^{\star\,\mathrm{T}} - \boldsymbol{u}_\Gamma^{\star\,\mathrm{T}}) & 0 \end{array}\right], \tag{51}$$

Let $\boldsymbol{v}$ (a nonzero column vector) be $\frac{1}{\lambda^\star}(\boldsymbol{u}_\Gamma^\star - \boldsymbol{x}_\Gamma^\star)$. Since $-\boldsymbol{u}_\Gamma^\star + \boldsymbol{x}_\Gamma^\star \neq 0$, it implies that $\frac{1}{\lambda^\star}(\boldsymbol{u}_\Gamma^\star - \boldsymbol{x}_\Gamma^\star) \neq 0$. Hence, matrix $\boldsymbol{B}$ can be expressed as

$$\boldsymbol{B} = \left[\begin{array}{c|c} \boldsymbol{S} & \boldsymbol{v} \\ \hline -\boldsymbol{v}^{\mathrm{T}} & 0 \end{array}\right], \tag{52}$$

where $S = \frac{2}{\lambda^\star}\Phi_\Gamma^{\mathrm{T}}\Phi_\Gamma$ is an $n_a \times n_a$ symmetric positive definite matrix. As $\boldsymbol{S}$ is invertible, we have

$$\left[\begin{array}{c|c} \boldsymbol{S} & \boldsymbol{v} \\ \hline -\boldsymbol{v}^{\mathrm{T}} & 0 \end{array}\right]\left[\begin{array}{c|c} \boldsymbol{I} & -\boldsymbol{S}^{-1}\boldsymbol{v} \\ \hline \varnothing_{n_a\times n_z} & 1 \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{S} & \varnothing_{n_a\times 1} \\ \hline -\boldsymbol{v}^{\mathrm{T}} & \boldsymbol{v}^{\mathrm{T}}\boldsymbol{S}^{-1}\boldsymbol{v} \end{array}\right]. \tag{53}$$

After taking determinant on both sides,

$$\left|\left[\begin{array}{c|c} \boldsymbol{S} & \boldsymbol{v} \\ \hline -\boldsymbol{v}^{\mathrm{T}} & 0 \end{array}\right]\right| = |\boldsymbol{S}|\cdot\left|\boldsymbol{v}^{\mathrm{T}}\boldsymbol{S}^{-1}\boldsymbol{v}\right|. \tag{54}$$

On the right-hand side, $\left|\boldsymbol{v}^{\mathrm{T}}\boldsymbol{S}^{-1}\boldsymbol{v}\right|$ is nonzero as $\boldsymbol{S}^{-1}$ is positive definite. As a result, the determinant of $\boldsymbol{S}$ (i.e.,$|\boldsymbol{S}|$) is nonzero too. It can conclude that the rank of $\boldsymbol{B}$ equals to n_a+1.

(Eigenvalues of $\boldsymbol{B}$:)

Let α be an eigenvalue of $\boldsymbol{B}$. Denote $[\boldsymbol{V}^{\mathrm{T}},\omega]^{\mathrm{T}}$ as the corresponding eigenvector. It is well known that if $[\boldsymbol{V}^{\mathrm{T}},\omega]^{\mathrm{T}}$ is an eigenvector, then it cannot be a zero vector.

Now, we are going to show that $\boldsymbol{V}$ is not a zero vector by the means of contradiction. Suppose $\boldsymbol{V}$ is zero vector, i.e., $\boldsymbol{V}=\varnothing_{n_a\times 1}$. From the definition of eigenvector,

$$\boldsymbol{B}\left[\begin{array}{c}\varnothing_{n_a\times 1}\\ \hline \omega\end{array}\right] = \alpha\left[\begin{array}{c}\varnothing_{n_a\times 1}\\ \hline \omega\end{array}\right]. \tag{55}$$

On the other hand, from the definition of $\boldsymbol{B}$ (see (49)), we have

$$\boldsymbol{B}\left[\begin{array}{c}\varnothing_{n_a\times 1}\\ \hline \omega\end{array}\right] = \alpha\left[\begin{array}{c}\frac{1}{\lambda^{\star}}\cdot\omega(\boldsymbol{u}_{\Gamma}^{\star}-\boldsymbol{x}_{\Gamma}^{\star})\\ \hline 0\end{array}\right], \tag{56}$$

From (55) and (56), we have $\omega=0$, which contradicts the early assumption of eigenvectors. Therefore, we have proved that $\boldsymbol{V}\neq 0$.

Now, we are going to prove that the eigenvalue α has positive real part. Since $[\boldsymbol{V}^{\mathrm{T}},\omega]^{\mathrm{T}}$ is an eigenvector, we obtain

$$\mathrm{Real}\left\{\left[\tilde{\boldsymbol{V}}^{\mathrm{T}}\middle|\tilde{\omega}\right]\boldsymbol{B}\left[\begin{array}{c}\boldsymbol{V}^{\mathrm{T}}\\ \hline \omega\end{array}\right]\right\} = \mathrm{Real}(\alpha)(\|\boldsymbol{V}\|_2^2+\|\omega\|_2^2), \tag{57}$$

where $\tilde{\boldsymbol{V}}$ is the conjugate of $\boldsymbol{V}$, and $\tilde{\omega}$ is the conjugate of ω.

By the definition of $\boldsymbol{B}$ (see (49)), we have

$$\mathrm{Real}\left\{\left[\tilde{\boldsymbol{V}}^{\mathrm{T}}\middle|\tilde{\omega}\right]\boldsymbol{B}\left[\begin{array}{c}\boldsymbol{V}^{\mathrm{T}}\\ \hline \boldsymbol{\omega}\end{array}\right]\right\} = \mathrm{Real}\{\frac{2}{\lambda^{\star}}\cdot\tilde{\boldsymbol{V}}^{\mathrm{T}}\boldsymbol{\Phi}_{\Gamma}^{\mathrm{T}}\boldsymbol{\Phi}_{\Gamma}\boldsymbol{V}^{\mathrm{T}}\}. \tag{58}$$

From (57) and (58),

$$\mathrm{Real}\left\{\frac{2}{\lambda^{\star}}\cdot\tilde{\boldsymbol{V}}^{\mathrm{T}}\boldsymbol{\Phi}_{\Gamma}^{\mathrm{T}}\boldsymbol{\Phi}_{\Gamma}\boldsymbol{V}^{\mathrm{T}}\right\} = \mathrm{Real}(\alpha)\left(\|\boldsymbol{V}\|_2^2+\|\omega\|_2^2\right). \tag{59}$$

Since $\frac{2}{\lambda^{\star}}\cdot\boldsymbol{\Phi}_{\Gamma}^{\mathrm{T}}\boldsymbol{\Phi}_{\Gamma}$ is positive definite, the real part of α is positive. Hence, we can conclude that the real parts of all the eigenvalues of $\boldsymbol{B}$ are positive.

Eigenvalues of $\boldsymbol{A}$

After we prove that the real parts of all the eigenvalues of $\boldsymbol{B}$ are positive, we are now going to prove that the real parts of all the eigenvalues of $\boldsymbol{A}$ are positive.

Recall that $\boldsymbol{B}$ is full rank. Hence, we can diagonalize it, given by

$$\boldsymbol{B} = \boldsymbol{M}\boldsymbol{Y}\boldsymbol{M}^{-1}, \tag{60}$$

where

$$\boldsymbol{Y} = \begin{bmatrix} y_1 & 0 & \cdots & 0 \\ 0 & \ddots & 0 & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ 0 & \cdots & \cdots & y_{n_a+1} \end{bmatrix} \tag{61}$$

is a diagonal matrix. In $\boldsymbol{Y}$, the diagonal components y_i's are the eigenvectors of $\boldsymbol{B}$. The column vectors of $\boldsymbol{M}$ are called the right eigenvectors, and the row vectors of $\boldsymbol{M}^{-1}$ are called left eigenvectors.

We consider a matrix $\boldsymbol{Z}$, given by

$$\boldsymbol{Z} = \left[\begin{array}{c|c} \boldsymbol{M} & \varnothing_{(n_a+1)\times(n-n_a)} \\ \hline \varnothing_{(n-n_a)\times(n_a+1)} & \boldsymbol{I}_{(n-n_a)\times(n-n_a)} \end{array}\right]. \tag{62}$$

Since $\boldsymbol{Z}$ is invertible, its inverse is given by

$$\boldsymbol{Z}^{-1} = \left[\begin{array}{c|c} \boldsymbol{M}^{-1} & \varnothing_{(n_a+1)\times(n-n_a)} \\ \hline \varnothing_{(n-n_a)\times(n_a+1)} & \boldsymbol{I}_{(n-n_a)\times(n-n_a)} \end{array}\right], \tag{63}$$

We define a new matrix $\tilde{\boldsymbol{A}}$, given by

$$\tilde{\boldsymbol{A}} = \boldsymbol{Z}\boldsymbol{A}\boldsymbol{Z}^{-1}. \tag{64}$$

From (48),

$$\tilde{\boldsymbol{A}} = \left[\begin{array}{c|c} \boldsymbol{Y} & \varnothing_{(n_a+1)\times(n-n_a)} \\ \hline \boldsymbol{C}\boldsymbol{M} & \boldsymbol{I}_{(n-n_a)\times(n-n_a)} \end{array}\right]. \tag{65}$$

As a result, $\tilde{\boldsymbol{A}}$ is a lower triangular matrix and its diagonal elements

$$\{y_1,\ldots,y_{n_a+1},1,\ldots,1\}$$

are the corresponding eigenvalues. Their real parts are greater than zero.

To complete the prove, we are going to prove that that $\tilde{\boldsymbol{A}}$ and $\boldsymbol{A}$ have the same set of eigenvalues. Recall that $\boldsymbol{A}$ is a full rank matrix. We can diagonalize it, given by

$$\boldsymbol{A} = \boldsymbol{U}\dot{\boldsymbol{Z}}\boldsymbol{U}^{-1}, \tag{66}$$

where $\dot{\boldsymbol{Z}}$ is a diagonal matrix. Its diagonal components are the eigenvalues of $\boldsymbol{A}$. From (66), we can express $\tilde{\boldsymbol{A}}$ as

$$\tilde{\boldsymbol{A}} = \boldsymbol{Z}\boldsymbol{A}\boldsymbol{Z}^{-1} = (\boldsymbol{Z}\boldsymbol{U})\dot{\boldsymbol{Z}}(\boldsymbol{Z}\boldsymbol{U})^{-1}. \tag{67}$$

From (66) and (67), $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}}$ have the identical set of eigenvalues. Now we can conclude that all the eigenvalues of $\boldsymbol{A}$ have the positive real parts. Therefore, from the classical control theory, the equilibrium state is asymptotically stable. We complete the proof. □

5 Simulation

This section tests the effectiveness of the proposed LASSO–LPNN model. The LASSO problem is a convex problem. There are some digital methods (numerical algorithms) for solving it [27–32]. Those methods can be roughly classified into two approaches. One approach is based on the interior-point method. Another one is based on the gradient projection. For the interior-point based approach, we select the LASSO–PDCO [28] method as the comparison method. For the interior-point based approach, we select the LASSO–SPGL1 method [30, 32] as the comparison method. This section would like to investigate whether the performances of the proposed analog method are similar to those of the two digital methods.

Two signal lengths, $\{n = 512, n = 4096\}$, are considered. Let N_z be the number of nonzero components in a sparse signal. For $n = 512$, we considers three sparsity cases: $\{N_z = 15, 20, 25\}$. While for $n = 4096$, we considers three sparsity cases: $\{N_z = 75, 100, 125\}$. For the nonzero components, their values are either equal to -2 or 2. We use ± 1 random matrix as the measurement matrix. Each column of the measurement matrix is then normalized.

In the experiments, we consider various numbers of measurement signals. We run the experiments 100 times with different sparse signals and measurement matrices for each setting. The measurement noise follows the Gaussian distribution. We consider three noise power levels. They are $\{\sigma^2 = 0.02^2, 0.01^2, \text{ and } 0.002^2\}$.

Figures 4, 5 show their MSE performances. The performances of our proposed analog LASSO–LPNN method are quite similar to those of the two comparison numerical methods. When the number of measurements reaches a certain threshold, the reconstruction error of the three methods is quite similar. For $n = 512$ and $n = 4096$, there are no significant differences in the number of required measurements and the reconstruction error among the LASSO–SPGL1 method, the LASSO–PDCO method, and

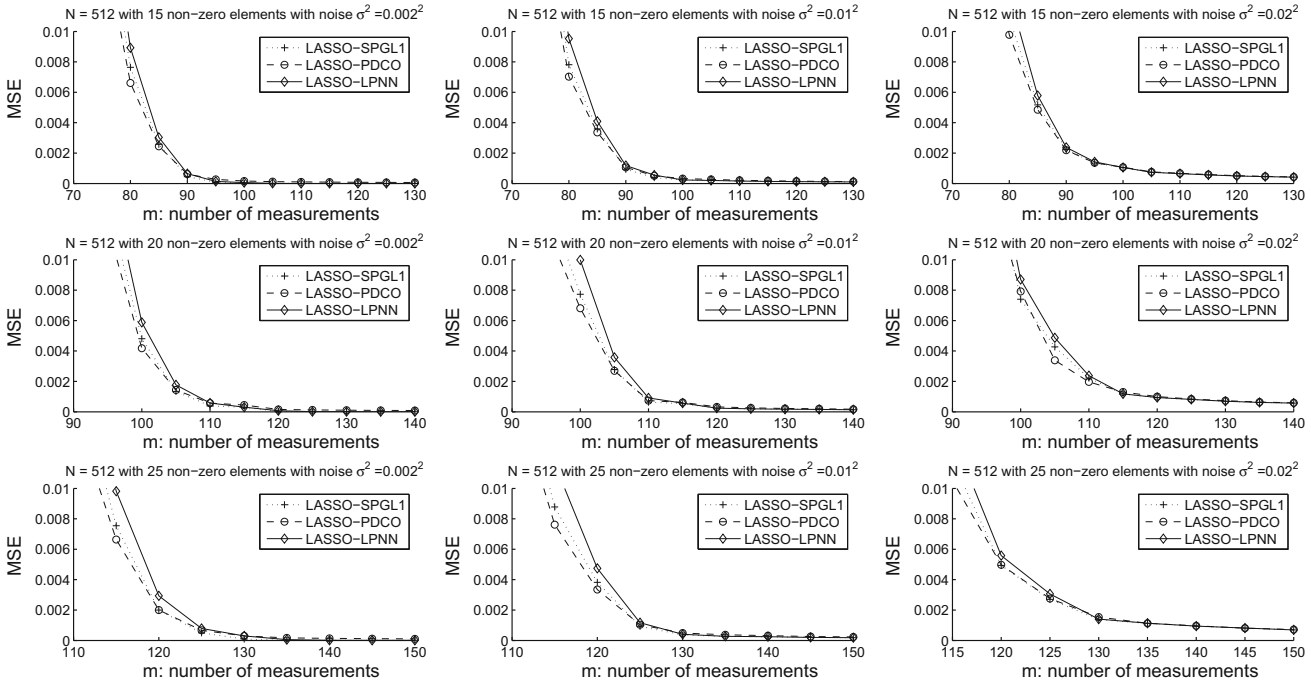

Fig. 4 MSE performances, where $n = 512$. We consider three cases of nonzero components. The number of nonzero components are $\{15, 20, 25\}$. One comparison digital method (SPGL1) is considered. The experiments are repeated 100 times using different random matrices

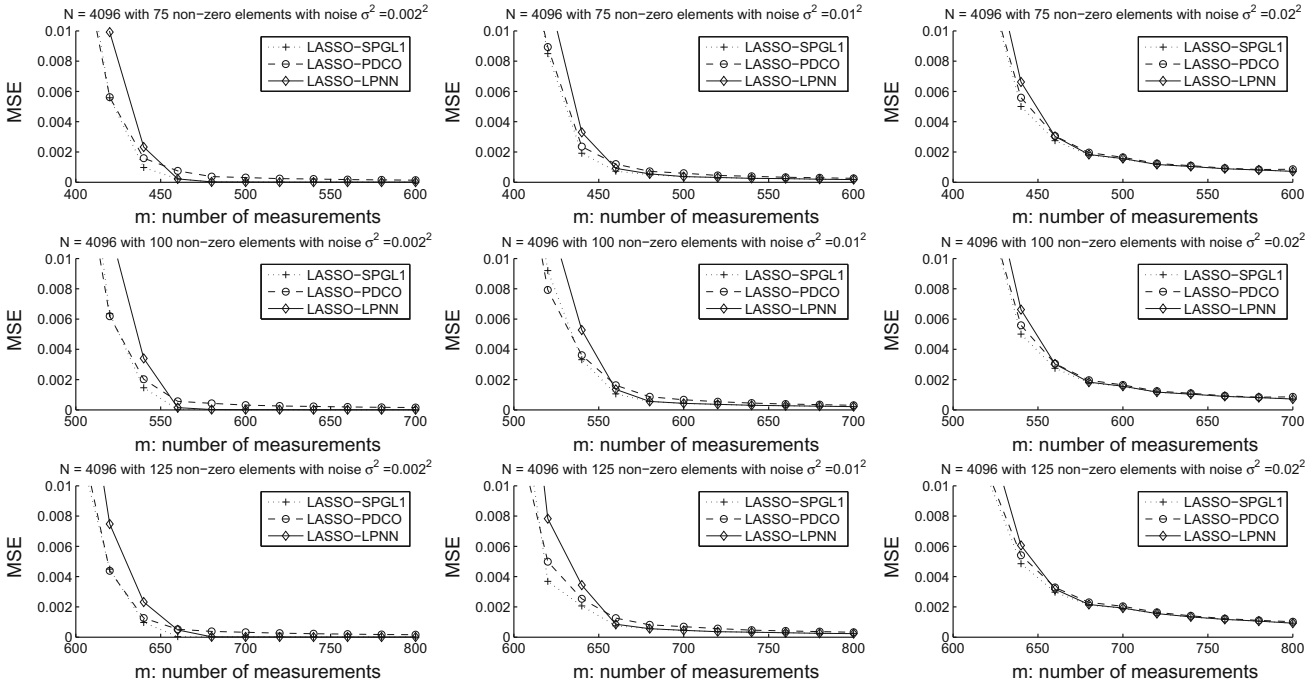

Fig. 5 MSE performances, where $n = 4096$. We consider three cases of nonzero components. The number of nonzero components are $\{75, 100, 125\}$. One comparison digital method (SPGL1) is considered. The experiments are repeated 100 times using different random matrices

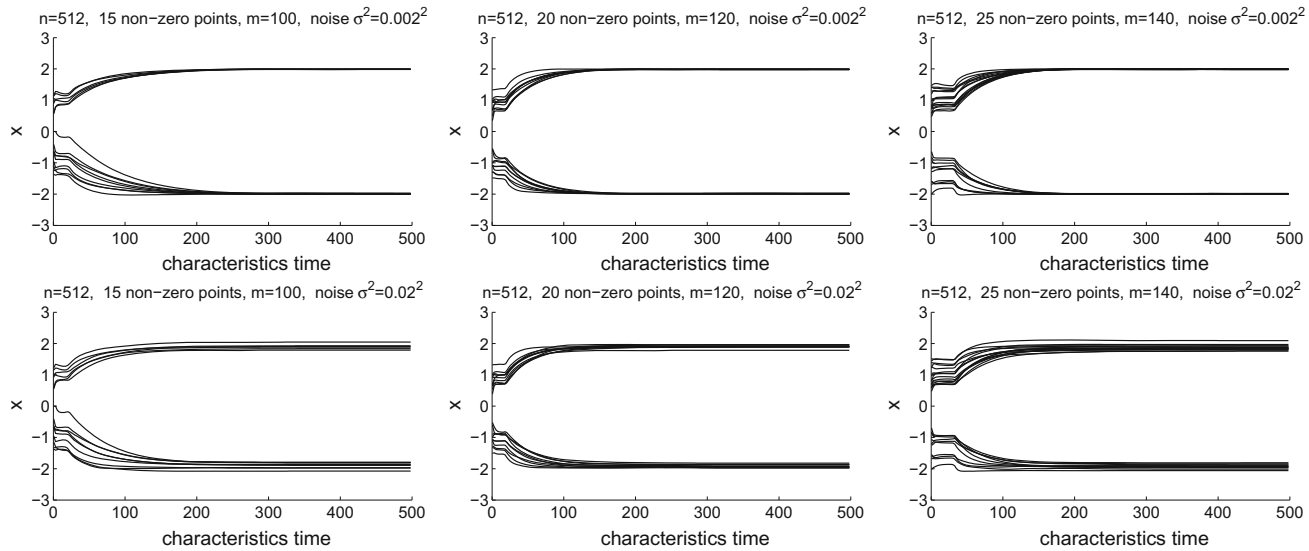

Fig. 6 Some examples of the dynamics of the LASSO–LPNN, where $n = 512$. From the figures, it can be seen that around 200–300 characteristics time units are enough for the model to settle down. In the figures, we show the dynamics of the nonzero elements only. If we show all the dynamics, there will be 512 curves in a figure

our analog LASSO–LPNN method. For $n = 4096$, the LASSO–PDCO method is slightly poorer than the LASSO–SPGL1 method and our analog LASSO–LPNN method.

It should be noticed that the LASSO problem is convex, and the three methods should have the similar MSE performance. In the LASSO–SPGL1 method and the LASSO–PDCO method, there are some tuning parameters. They may affect the accuracy of the sparse solution. Therefore, there are some small MSE differences among the LASSO–SPGL1 method, the LASSO–PDCO method, and our analog LASSO–LPNN method.

For $n = 512$, with 15 nonzero elements, we need around 95 measurement signals for recovering the sparse signal

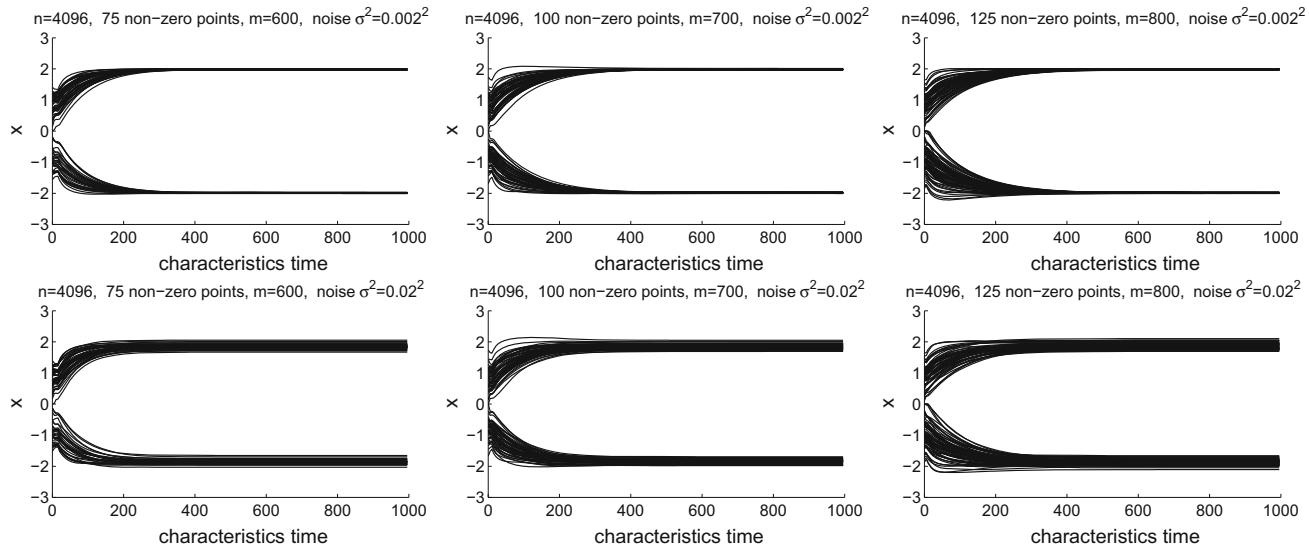

Fig. 7 Some examples of the dynamics of the LASSO–LPNN, where $n = 4096$. From the figures, it can be seen that around 200–300 characteristics time units are enough for the model to settle down. In the figures, we show the dynamics of the nonzero elements only. If we show all the dynamics, there will be 4096 curves in a figure

regardless of the measurement noise level, as shown in the first row of Fig. 4. From the figures, the noise level σ^2 affects the reconstruction quality only. It does not much affect the number of required measurements. When we increase the number of nonzero elements to 25, we require around 125 measurement signals regardless of the measurement noise level. When n is equal to 4096 and there are 75 nonzero elements, we need around 475 measurement signals regardless of the measurement noise level, as shown in the first row of Fig. 5. Again, the noise level σ^2 does not much affect the number of required measurements. When we increase the number of nonzero elements to 125, we require around 675 measurement signals.

Figures 6 and 7 show some examples of the convergent behavior of the LASSO–LPNN model under various settings. As shown in Fig. 6, with $n = 512$, the outputs of the network can settle down around 150–250 time units. For $n = 4096$, the outputs can settle down around 250–400 time units, as shown in Fig. 7.

6 Concluding remark

This paper developed an analog neural network, namely LASSO–LPNN, for the LASSO problem based on the LCA concept. Experiments are carried out to verify the ability of the proposed LASSO–LPNN. Besides, we theoretically show that an equilibrium point of the model fulfills the KKT conditions of the LASSO problem. That means, when the state of the LASSO–LPNN settles down, we can obtain the optimal solution. We also show that the equilibrium points of the network are asymptotically stable. That means, the equilibrium points, or saying the optimal solutions of the LASSO problem, are achievable by the proposed LASSO–LPNN model. Our current result focuses on the l_1 constraint which can be considered as the surrogate of the l_0 constraint. Hence, one of the important extensions is to investigate the non-convex constraints, which involves the l_p-norm with $p < 1$ in the constraints.

Acknowledgements This work is partially supported by the Research Grants Council, Hong Kong, under Grant Number, CityU 115612.

Compliance with ethical standards

Conflict of interest Authors declare that they do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

References

1. Cichocki A, Unbehauen R (1993) Neural networks for optimization and signal processing. Wiley, London
2. MacIntyre J (2013) Applications of neural computing in the twenty-first century and 21 years of Neural Computing & Applications. Neural Computing Appl 23(3):657–665
3. Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. In: Proceedings of the National Academy of Sciences, 79, 2554–2558
4. Tank D, Hopfield JJ (1986) Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit. IEEE Trans Circuits Syst 33(5):533–541
5. Duan S, Dong Z, Hu X, Wang L, Li H (2016) Small-world Hopfield neural networks with weight salience priority and memristor synapses for digit recognition. Neural Computing Appl 27(4):837–844
6. Chua LO, Lin GN (1984) Nonlinear programming without computation. IEEE Trans Circuits Syst 31:182–188

7. Liu Q, Wang J (2008) A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming. IEEE Trans Neural Netw 19(4):558–570
8. Wang J (2010) Analysis and design of a k-winners-take-all model with a single state variable and the heaviside step activation function. IEEE Trans Neural Netw 21(9):1496–1506
9. Bharitkar S, Tsuchiya K, Takefuji Y (1999) Microcode optimization with neural networks. IEEE Trans Neural Netw 10(3):698–703
10. Chua LO, Yang L (1988) Cellular neural networks: theory. IEEE Trans Circuits Syst 35(10):1257–1272
11. Ho TY, Lam PM, Leung CS (2008) Parallelization of cellular neural networks on GPU. Pattern Recognit 41(8):2684–2692
12. Lin YL, Hsieh JG, Kuo YS, Jeng JH (2016) NXOR- or XOR-based robust template decomposition for cellular neural networks implementing an arbitrary Boolean function via support vector classifiers. Neural Computing Appl (accepted)
13. Liu X (2016) Improved convergence criteria for HCNNs with delays and oscillating coefficients in leakage terms. Neural Computing Appl 27(4):917–925
14. Sum J, Leung CS, Tam P, Young G, Kan WK, Chan LW (1999) Analysis for a class of winner-take-all model. IEEE Trans Neural Netw 10(1):64–71
15. Liu S, Wang J (2006) A simplified dual neural network for quadratic programming with its KWTA application. IEEE Trans Neural Netw 17(6):1500–1510
16. Xiao Y, Liu Y, Leung CS, Sum J, Ho K (2012) Analysis on the convergence time of dual neural network-based kwta. IEEE Trans Neural Netw Learn Syst 23(4):676–682
17. Gao XB (2003) Exponential stability of globally projected dynamics systems. IEEE Trans Neural Netw 14:426–431
18. Hu X, Wang J (2007) A recurrent neural network for solving a class of general variational inequalities. IEEE Trans Syst Man Cybern B Cybern 37(3):528–539
19. Zhang S, Constantinidies AG (1992) Lagrange programming neural networks. IEEE Tran Circuits Syst II 39:441–452
20. Leung CS, Sum J, So HC, Constantinides AG, Chan FKW (2014) Lagrange programming neural networks for time-of-arrival-based source localization. Neural Computing Appl 24(1):109–116
21. Liang J, So HC, Leung CS, Li J, Farina A (2015) Waveform design with unit modulus and spectral shape constraints via Lagrange programming neural network. IEEE J Sel Top Signal Process 9(8):1377–1386
22. Liang J, Leung CS, So HC (2016) Lagrange programming neural network approach for target localization in distributed MIMO radar. IEEE Trans Signal Process 64(6):1574–1585
23. Donoho DL, Elad M (2003) Optimally sparse representation in general (nonorthogonal) dictionaries via l_1 minimization. Proc Natl Acad Sci 100(5):2197–2202
24. Gilbert AC, Tropp JA (2005) Applications of sparse approximation in communications. In: Proceedings of the international symposium on information theory ISIT 2005:1000–1004
25. Sahoo SK, Lu W(2011) Image denoising using sparse approximation with adaptive window selection. In: Proceedings of the 8th international conference on information, communications and signal processing (ICICS) 2011, 1–5
26. Rahmoune A, Vandergheynst P, Frossard P (2012) Sparse approximation using m-term pursuit and application in image and video coding. IEEE Trans Image Process 21(4):1950–1962
27. Kim SJ, Koh K, Lustig M, Boyd S, Gorinevsky D (2007) An interior-point method for large-scale ‘1-regularized least squares. IEEE J Sel Top Sig Proc 1(4):606–617
28. Saunders MA (2005) Matlab software for convex optimization. http://www.stanford.edu/group/SOL/software/pdco.html
29. Figueiredo M, Nowak R, Wright S (2007) Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems, IEEE. J Sel Top Sig Proc 1(4):586–597
30. Berg E, Friedlander MP (2008) Probing the pareto frontier for basis pursuit solutions. SIAM J Sci Computing 31(2):890912
31. Berg E, Friedlander MP (2011) Sparse optimization with least-squares constraints. SIAM J Optim 21(4):1201–1229
32. Berg E, Friedlander MP (2007) SPGL1: a solver for large-scale sparse reconstruction. http://www.cs.ubc.ca/labs/scl/spgl1
33. Rozell CJ, Johnson DH, Baraniuk RG, Olshausen BA (2008) Sparse coding via thresholding and local competition in neural circuits. Neural Comput 20(10):2526–2563
34. Chen SS, Donoho DL, Saunders MA (1998) Atomic decomposition by basis pursuit. SIAM J Sci Comput 20(1):33–61
35. Feng R, Lee CM, Leung CS (2015) Lagrange programming neural network for the L1-norm constrained quadratic minimization. In: Proceedings of the ICONIP 2015, Istanbul, Turkey, 3, pp 119–126
36. Balavoine A, Rozell CJ, Romberg J (2011) Global convergence of the locally competitive algorithm. In: Proceedings of the IEEE signal processing education workshop (DSP/SPE) (2011) Sedona. Arizona, USA, pp 431–436
37. Balavoine A, Romberg J, Rozell CJ (2012) Convergence and rate analysis of neural networks for sparse approximation. IEEE Trans Neural Netw Learn Syst 23(9):1377–1389
38. Gordon G, Tibshirani R (2012) Karush–Kuhn–Tucker conditions, Optimization Fall 2012 Lecture Notes
39. Guenin B, Konemann J, Tunel T (2014) A gentle introduction to optimization. Cambridge University Press, Cambridge
40. Feng X, Zhang Z (2007) The rank of a random matrix. Appl Math Comput 185(1):689–694