Cooperative recurrent modular neural networks for constrained optimization: a survey of models and applications

Mohamed S. Kamel · Youshen Xia

Received: 5 September 2007 / Accepted: 27 November 2007 / Published online: 1 February 2008
© Springer Science+Business Media B.V. 2008

Abstract Constrained optimization problems arise in a wide variety of scientific and engineering applications. Since several single recurrent neural networks when applied to solve constrained optimization problems for real-time engineering applications have shown some limitations, cooperative recurrent neural network approaches have been developed to overcome drawbacks of these single recurrent neural networks. This paper surveys in details work on cooperative recurrent neural networks for solving constrained optimization problems and their engineering applications, and points out their standing models from viewpoint of both convergence to the optimal solution and model complexity. We provide examples and comparisons to shown advantages of these models in the given applications.

Keywords Cooperative modular neural networks · Recurrent neural networks · Constrained optimization problems · Engineering applications

Introduction

Over the past several decades, many single neural networks have made significant advancement since the pioneering work of McCulloch and Pitts (1943). They are well

established and mainly applied in classification, regression approximation, associative memory, combinatoric optimization, and component analysis fields. Typical examples of these single neural networks are the Hopfield neural network (Hopfield 1982), the multilayer perceptron (Rumerharlt et al. 1986), the radial basis function network (Broomhead and Lowe 1988), the Self-organization neural network (Kohonen 1989), the multilayer feedforward neural network (Hornik et al. 1989), and the minor component neural network (Oja 1992). However, many real-world problems are so large and complex that it is not effective for such an individual neural network to solve (Smith 1999). In order to enhance the effectiveness of the individual neural network, modular neural network approaches (Anzai and Shimada 1988) and neural network ensembles (Hansen and Salamon 1990) have been introduced. The neural network ensemble consisting of multiple neural network models attempts to improve generalization performance of these single neural networks. The modular neural network can increase the computational efficiency of solving large size/ high complex problems, by decomposing it into several subproblems with small sizes and each is handled by an efficient neural network model. But, how to divide a problem into subproblems and how to assign a neural network model to learn the subproblems are often difficult for the combination of these submodel outputs to form the global optimal solution (Jacobs et al. 1991; Anand et al. 1995; Copeland 1995; Kecman 1996; Kehagias and Petridis 1997; Kadirkamanathan and Fabri 1998; Yamaguchi and Itakura 1999; Hashem 1997; Sierra and Cruz 1998; Caelli et al. 1999). To overcome these drawbacks, cooperative modular neural network approaches were proposed (Auda and Kamel 1997a). Different from neural network ensembles and modular neural network approaches, cooperative modular neural networks can decompose automatically and combines adaptively

M. S. Kamel (✉)
Department of Electrical and Computer Engineering, University
of Waterloo, Waterloo, Canada
e-mail: mkamel@pami.uwaterloo.ca

Y. Xia (✉)
College of Mathematics and Computer Science, Fuzhou
University, Fuzhou, China
e-mail: ysxia2001@yahoo.com

individual neural network models so that a global optimal solution of the original problem can be obtained. Reported results show that the cooperative modular neural networks can be well applied to classification and pattern recognition (Auda and Kamel 1997a, b, 1998a, b, 1999; Zhang 2000; Lu and Ito 1999; Yang and Browne 2001; Oh and Suen 2002; Melin et al. 2005; Fogelman-Soulie 1993; Hodge et al. 1999; Kamel 1999; Alexandre et al. 2001; Ozawa 1998; Islam et al. 2003). Specially, in recent decade, as special one class of cooperative modular neural networks, cooperative recurrent modular neural networks for constrained optimization have been developed and well studied (Rodríguez-Vázquez et al. 1990; Glazos et al. 1998; Zhang et al. 1992; Zhang and Constantinides 1992; He and Sun 2001; Tao and Fang 2000; Xia and Wang 1995, 1998a, b, 2001, 2004a, b, 2005; Xia 1996a, b, 1997, 2003, 2004; Xia et al. 2002a, b, 2004a, b, 2005, 2006, 2007; Wang et al. 2000; Tan et al. 2000; Anguita and Boni 2002; Zhang et al. 2003; Xia and Feng 2004, 2006; Xia and Kamel 2007a, b, c, d, 2008; Tao et al. 2001; Leung et al. 2001).

Constrained optimization deals with the problem of optimizing an objective function over a set of variables. Usually, the set consists of equality and inequality constraints. If all the functions are linear, this problem is a linear program. Otherwise, it is called a nonlinear program. Many scientific and engineering problems, such as signal and image processing, regression mining, system identification, and robot control, can be solved by transforming the original problem into a constrained optimization problem. Conventional numerical methods for static constrained convex optimization, such as the interior-point method (Nesterov and Nemirovsky 1994), the projection type method (Solodov and Tseng 1996), the sequential quadratic programming method (Fukushima et al. 2003), have been studied for over a few decade. The computing time required, to obtain a solution and its accuracy, is usually dependent of the dimension and structure of the static problems to be solved. As the problem dimensions increase, the efficiency of these numerical methods will decrease, specially to these where the objective function is non-smooth. Moreover, in many practical applications, related constrained optimization problems often include a time-varying parameter (Cichocki and Unbehauen 1993). One example of such applications is the real-time signal processing in wireless communications. Another example of such applications is the real-time optimal robot control. For such applications, conventional numerical methods may not be adequate due to the problem complexity and stringent requirement on the computational time. In addition, conventional numerical methods require sometimes so strong conditions for their global convergence that practice problems does not satisfy.

Having the structures similar to their biological counterparts, artificial neural networks are representational and computational models processing information in a parallel distributed fashion (Hopfield 1982). Tank and Hopfield (1986) first developed a single recurrent neural network linear programming in real time. Their seminal work has inspired many researchers to investigate improved neural networks for on-line optimization. Kennedy and Chua (Kennedy and Chua 1988) proposed a single recurrent neural network for solving nonlinear programming problems by extending the Tank and Hopfield network model. Urahama (1996) proposed a single recurrent neural network, called the gradient projection network, for linearly constrained nonlinear programming. Although these single neural network models are of small size, they can't obtain the global optimal solution due to using penalty parameters. To overcome this drawback, cooperative recurrent neural network approaches to constrained optimization have been developed. Theoretically, it was shown that under weaker condition than the conventional numerical methods, the cooperative recurrent neural networks can converge globally to the global optimal solution (Xia et al. 2007). Moreover, they are successfully applied to classification, signal and image processing, system identification, and robot control (Wang et al. 2000; Tan et al. 2000; Anguita and Boni 2002; Xia and Wang 2001, 2004; Zhang et al. 2003; Xia and Feng 2004; Xia et al. 2002b, 2004b, 2005, 2006, 2007; Xia and Feng 2006; Xia and Kamel 2007a, b, c, d, 2008).

This paper aims to survey the standing and potential applications of current cooperative recurrent neural networks for constrained optimization in a recent decade of research. We first review briefly modular neural networks and cooperative modular neural networks, and we then survey in details the important advances in cooperative recurrent neural networks for constrained optimization and their applications, and point out their standing from computational and application viewpoint.

Modular neural networks and cooperative modular neural networks

In this section, we review in brief neural network ensembles, modular neural networks, and cooperative modular neural networks, and discuss their differences.

Single neural networks

An artificial neural network (ANN) is an parallel distributed processing system with many artificial neuron modes which attempt to simulate the structure and function of

biological neurons. The ANN is called the single neural network if there are same artificial neuron model structures, such as the perceptron network, the radial basis function network, the Hopfield neural network, etc.

Ensemble neural networks

An artificial neural network ensemble is used for enhanced performance of single neural networks for solving problems. The ensemble neural network selects several different single neural networks and the results generated by these ensemble models will be combined by some decision method. The combined result is called the ensemble output as the solution to the given problems. Figure 1(a) shows a neural network ensemble structure using three ensemble neural network candidates.

The idea of a neural network ensemble originated from using all the valuable information hidden in neural network classifiers, where each can contribute to the improvement of generalization (Hansen and Salamon 1990). An earlier neural network ensemble for classification was developed by Hansen and Salamon (1990), where the ensemble of neural network classifiers is shown to be more accurate than any of its individual neural networks if each neural network classifier is accurate. Schapire (Hansen and Salamon 1990) proved further that more accurate classifier can be obtained by combining many inaccurate classifiers. Hansen and Salamon (1990) studied a linear combination of the corresponding output of the trained ensemble candidates. Hashem (1997) extended the idea of optimal linear combinations of the trained ensemble candidates, where the combination-weights are selected to minimize the mean

squared error (MSE) with respect to the distribution of the model inputs. In the literature, neural network ensembles are also termed as multiple neural networks. The use of multiple neural systems was described in the work of Sharkey (1998). A recent review on neural network ensembles can be found in (Yang and Browne 2001).

Modular neural networks

A modular neural network (MNN) is a neural network that consists of several single neural network models, each one solves one subproblem of the original problem, and the solutions of models are integrated to form the solution of the original problem (Caelli et al. 1999). Figure 1(b) shows the general modular neural network structure using three neural network models.

The modular neural network is motivated by biological, psychology, hardware, and computational. A detailed survey of these ways can be found in (Auda and Kamel 1999). One application of the modular neural network to face recognition is to divide a human face into three different regions: the eyes, the nose, and the mouth. Each of these regions is assigned to one neural network model, and then the final face recognition is achieved by integrating the decision. Some other applications are mainly classification, clustering, and recognition. There are many modular neural network learning approaches. These methods can be roughly divided into three classes in problem decomposition (Lu and Ito 1999). The first class is that decomposition is made by the designer with prior knowledge about the problem before learning. The second class is that decomposition is based on the inherent relations among training

Fig. 1 (a) A neural network ensemble structure using three models; (b) modular neural network structure using three subproblems as decomposition

data before learning. The third type of decomposition is made during learning process.

Modular neural networks have advantages such as improved generalization, better usability, interpretability, easier hardware implementation (Auda and Kamel 1998a). On the other hand, there exist some computational problems: task-decomposition, training network modules, and combination of network outputs for the global optimal solution.

Finally, we simply compare the neural network ensemble and the modular neural network. Both the neural networks have same idea to build a bigger network by using modules as building blocks, however, there is a difference in that the modular neural network can divide the original problem into several subproblems, where each model solves a subproblem. Moreover, the output of the modular neural network integrates the solutions of single neural networks, while the output of the ensemble neural network combines the solutions of single neural networks. Furthermore, since training the modules for each subproblem is easier than training the ensemble members on the complete problem, the modular neural network is more suitable for solving larger size problems than the neural network ensemble.

Cooperative modular neural networks

A cooperative modular neural network is a neural network which combines adaptively several single neural network models so that the global optimal solution of the original problem can be obtained (Kamel 1999). More exactly, cooperative modular neural network takes into account the following features.

- Reformulating the original problem into several smaller or simpler subproblems;
- Taking into account each neural network model for the solution of each subproblem;
- Combining neural network models automatically;
- Weighting the corresponding output of each neural network model.

Figure 2 shows a general cooperative modular neural network structure using three neural network models. Compared with the neural network ensemble and the modular neural network, the cooperative modular neural network has the following advantages:

- Solving large size or high complex problems;
- Assigning each neural network model automatically to learn each of the subproblems;
- Increasing learning speed and easier hardware implementation;
- Achieving the global optimal solution of the original problem.

Thus, the cooperative modular neural networks can overcome existing drawbacks of modular neural networks. An earlier cooperative modular learning system was presented by Fogelman Soulie (1993). Two cooperative modular neural networks for classification and pattern recognition were introduced by Auda and Kamel (1997a, 1998a). A cooperative neural network, called the cross-coupled Hopfield network, for associate memory was developed by Ozawa (1998). A cooperative neural network ensemble algorithm was developed by Islam et al. (2003). These cooperative modular neural networks are mainly applied for classification, pattern recognition, and associate memory. In recent decade, cooperative recurrent neural networks were developed for constrained optimization, where each single neural network model has a feedback structure. These cooperative recurrent neural networks can be shown to be globally convergent to the global optimal solution of the original problems. Moreover, they have been effectively applied to classification, data mining, signal and image processing, system identification, and robot control.

In this papers, we survey in details cooperative recurrent neural networks for solving constrained optimization problems and their applications. When evaluating these cooperative recurrent neural networks, we discuss their algorithm performance from viewpoint of both convergence to the optimal solution and model complexity which is defined as the total number of multiplications per iteration.

Cooperative recurrent neural networks for constrained optimization

This section presents a survey of cooperative recurrent neural networks for two large classes of constrained optimization problems: linear and nonlinear programming problems and linearly constrained least absolute deviation problems.

Cooperative recurrent neural networks for linear and nonlinear convex programm

Problem formulation and single neural networks

Consider the following nonlinear programming (NP) problem.

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g(\mathbf{x}) \leq 0, \quad h(\mathbf{x}) = 0, \\ & \mathbf{x} \in X, \end{aligned} \quad (1)$$

where $\mathbf{x} \in R^n$, $f: R^n \rightarrow R$ is a real-valued continuous differentiable function of n variables, $g(\mathbf{x}) = [g_1(\mathbf{x}), \dots, g_m(\mathbf{x})]^T$ is an m -dimensional vector-valued

continuous differentiable function of n variables, $h(\mathbf{x}) = B\mathbf{x} - \mathbf{b}$, $B \in R^{r \times n}$, $\mathbf{b} \in R^r$, $X = \{\mathbf{x} \in R^n | \mathbf{l} \leq \mathbf{x} \leq \mathbf{d}\}$, $\mathbf{l} = [l_1, \dots, l_n]^T$, $\mathbf{d} = [d_1, \dots, d_n]^T$.

A vector $\mathbf{x}$ is called a feasible solution to the optimization problem (1) if $\mathbf{x}$ satisfies all constraints. The collection of all feasible points is called the feasibility set, denoted by

$$\Omega_0 = \{\mathbf{x} \in R^n | g(\mathbf{x}) \leq 0, \quad h(\mathbf{x}) = 0, \quad \mathbf{x} \in X\}.$$

A feasible point $\mathbf{x}^*$ is said to be a local optimal solution to (1) if there exists an open neighborhood $S \subset \Omega_0$ of $\mathbf{x}^*$ such that

$$f(\mathbf{x}) \geq f(\mathbf{x}^*), \quad \forall \mathbf{x} \in S$$

and $\mathbf{x}^*$ is a global optimal solution to (1) if the above inequality holds for any $\mathbf{x} \in \Omega_0$. In one special case that $f(\mathbf{x})$ and $g_i(\mathbf{x}) (i = 1, \dots, m)$ are linear functions, (1) is called linear programming (LP) problem. In another special case that $f(\mathbf{x})$ and $g_i(\mathbf{x}) (i = 1, \dots, m)$ are nonlinear convex functions, (1) is said to be a nonlinear convex programming (NCP) problem. If $\mathbf{x}^*$ is an optimal solution of the NCP problem, then there exist $\mathbf{y}^* \in R^m$, $\mathbf{z}^* \in R^r$ such that $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ satisfies the Karush-Kuhn-Tucker (KKT) condition (Bazaraa et al. 1993):

$$\begin{cases} \mathbf{y} \geq 0, & g(\mathbf{x}) \leq 0, & \mathbf{l} \leq \mathbf{x} \leq \mathbf{d} \\ B\mathbf{x} = \mathbf{b}, & \mathbf{y}^T g(\mathbf{x}) = 0 \end{cases}$$

and

$$\begin{cases} (\nabla f(\mathbf{x}) + \nabla g(\mathbf{x})\mathbf{y} - B^T \mathbf{z})_i \geq 0 & \text{if } x_i = d_i \\ (\nabla f(\mathbf{x}) + \nabla g(\mathbf{x})\mathbf{y} - B^T \mathbf{z})_i = 0 & \text{if } l_i \leq x_i \leq d_i \\ (\nabla f(\mathbf{x}) + \nabla g(\mathbf{x})\mathbf{y} - B^T \mathbf{z})_i \leq 0 & \text{if } x_i = l_i, \end{cases}$$

where $\mathbf{y} \in R^m$, $\mathbf{z} \in R^r$, $\nabla g(\mathbf{x}) = (\nabla g_1(\mathbf{x}), \dots, \nabla g_m(\mathbf{x}))$, and $\nabla g_i(\mathbf{x})$ is the gradient of g_i for $i = 1, \dots, m$.

Kennedy and Chua (1988) first proposed a single neural network for solving the NCP problem (1) as follows

$$\frac{d\mathbf{x}}{dt} = -\{\nabla f(\mathbf{x}) + s(\nabla g(\mathbf{x})(g(\mathbf{x}))^+ + \nabla h(\mathbf{x})h(\mathbf{x}))\}, \quad (2)$$

where s is a positive penalty parameter and $\mathbf{x} \in R^n$. Because the Kennedy and Chua network is essentially a gradient model based on the following penalty function

$$E_s(\mathbf{x}) = f(\mathbf{x}) + \frac{s}{2} \left\{ \sum_{j=1}^m [g_j^+(\mathbf{x})]^2 + \sum_{j=1}^r [h_j(\mathbf{x})]^2 \right\} \quad (3)$$

where s is a positive penalty parameter and $g_j^+(\mathbf{x}) = \max\{0, g_j(\mathbf{x})\}$, the equilibrium point of the Kennedy and Chua network minimizes $E_s(\mathbf{x})$, which results in difference between the true minimizer and the equilibrium point for a fixed s . Thus the Kennedy and Chua network only converges to an approximate optimal solution of the NCP problem for a finite penalty parameter.

Survey of cooperative recurrent neural network models

In order to avoid disadvantages of the penalty parameter, Rodríguez-Vázquez et al. (1990) first developed a cooperative recurrent neural network (CRNN), called switched-capacitor neural network, for solving the following convex programming problem

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g(\mathbf{x}) \leq 0. \end{aligned} \quad (4)$$

When applied to the above NCP problem (1), the CRNN model can be described by

$$\frac{d\mathbf{x}}{dt} = - \begin{cases} \sum_{i \in H(\mathbf{x})} \nabla g_i(\mathbf{x}) & \text{if } \mathbf{x} \notin \hat{S}_1 \\ \nabla f(\mathbf{x}) & \text{if } \mathbf{x} \in \hat{S}_1 \end{cases} \quad (5)$$

where $\mathbf{x} \in R^n$, $\hat{S}_1 = \{\mathbf{x} \in R^n | g(\mathbf{x}) \leq 0\}$ and $H(\mathbf{x}) = \{1 \leq i \leq m | g_i(\mathbf{x}) > 0\}$. The CRNN (5) divides automatically a whole space into the feasible region and infeasible region and takes different models in different region. It was shown in (Glazos et al. 1998) that the CRNN (5) can converge globally to an optimal solution of the NCP under the condition that the feasible region is bounded and the optimal solution lie in the interior of the feasible region. On the other hand, the CRNN (5) has to compute a time-varying index set $H(\mathbf{x})$ per iteration when the trajectory $\mathbf{x}(t) \notin \hat{S}_1$.

In 1992, Zhang and Constantinides (1992) developed a cooperative recurrent neural network, called as Lagrange programming neural network, for solving the following optimization problem with equality constraints.

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & h(\mathbf{x}) = 0. \end{aligned} \quad (6)$$

This CRNN model is defined by

$$\frac{d\mathbf{x}}{dt} = -\nabla_x L(\mathbf{x}, \mathbf{y}) \quad (7a)$$

and

$$\frac{d\mathbf{y}}{dt} = \nabla_y L(\mathbf{x}, \mathbf{y}) \quad (7b)$$

where $\mathbf{x} \in R^n$, $\mathbf{y} \in R^r$ is referred as the Lagrange multiplier vector, and $L(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) + \mathbf{y}^T h(\mathbf{x})$ as the Lagrange function for (6), and

$$\nabla_x L(\mathbf{x}, \mathbf{y}) = \frac{\partial L(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}}, \quad \nabla_y L(\mathbf{x}, \mathbf{y}) = \frac{\partial L(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}}.$$

The CRNN model (7) consists of two different gradient models of the Lagrange function. Some extensions of the Lagrange programming neural network were developed by Cichocki and Unbehauen (1993). Under the condition that $\nabla^2 f(\mathbf{x})$ is positive definite, it is shown in Xia (2003) that the CRNN model (7) can converge globally to the optimal solution to (6), where $h(\mathbf{x}) = B\mathbf{x} - \mathbf{b}$, $B \in R^{r \times n}$, and $\mathbf{b} \in R^r$.

Xia and Wang (1995) and Xia (1996a) proposed two cooperative neural networks for solving the following linear programming (LP) problem

$$\begin{aligned} \min \quad & f(\mathbf{x}) = \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & B\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0, \\ & \mathbf{x} \in X, \end{aligned} \quad (8)$$

where $\mathbf{x} \in R^n$ and $\mathbf{c} \in R^n$. The first CRNN approach divides the LP problem into one system of two equations below

$$\begin{cases} (\mathbf{x} + B^T \mathbf{y} - \mathbf{c})^+ = \mathbf{x} \\ B\mathbf{x} = \mathbf{b}. \end{cases}$$

Its CRNN model is then defined as

$$\frac{d\mathbf{x}}{dt} = B^T(B\mathbf{x} - \mathbf{b}) - \beta(B^T \mathbf{y} - \mathbf{c}), \quad (9a)$$

and

$$\frac{d\mathbf{y}}{dt} = \beta B[(\mathbf{x} + B^T \mathbf{y} - \mathbf{c})^+ - \mathbf{b}], \quad (9b)$$

where $\mathbf{x} \in R^n$, $\mathbf{y} \in R^r$, $\beta = \|(\mathbf{x} + B^T \mathbf{y} - \mathbf{c})^+ - \mathbf{x}\|_2^2$, $(\mathbf{x})^+ = [(x_1)^+, \dots, (x_n)^+]^T$, and $(x_i)^+ = \max\{x_i, 0\}$. The second CRNN approach transforms the LP problem into another system of linear equalities and inequalities as follows

$$\begin{cases} B\mathbf{x} = \mathbf{b}, \quad B^T \mathbf{y} \leq \mathbf{c} \\ \mathbf{c}^T \mathbf{x} - \mathbf{b}^T \mathbf{y} = 0, \quad \mathbf{x} \geq 0 \end{cases} \quad (10)$$

The corresponding CRNN model is given by

$$\frac{d\mathbf{x}}{dt} = (\mathbf{c}^T \mathbf{x} - \mathbf{b}^T \mathbf{y})\mathbf{c} + B^T(B\mathbf{x} - \mathbf{b}) - (-\mathbf{x})^+, \quad (11a)$$

and

$$\frac{d\mathbf{y}}{dt} = -(\mathbf{c}^T \mathbf{x} - \mathbf{b}^T \mathbf{y})\mathbf{b} + B(B^T \mathbf{y} - \mathbf{c})^+, \quad (11b)$$

where $\mathbf{x} \in R^n$ and $\mathbf{y} \in R^r$. The two CRNNs (9) and (11) can converge globally to an optimal solution of the LP problem.

Xia (1996b) further proposed a cooperative recurrent neural network, called as a primal-dual neural network, for solving the following linear and quadratic programming (QP) problem

$$\begin{aligned} \min \quad & f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T Q \mathbf{x} + \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & B\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq 0, \end{aligned} \quad (12)$$

where Q is an $n \times n$ symmetric and positive semidefinite matrix and $\mathbf{c} \in R^n$. It is defined by

$$\frac{d\mathbf{x}}{dt} = (I + Q)[\mathbf{x} - (\mathbf{x} + B^T \mathbf{y} - Q\mathbf{x} - \mathbf{c})^+] + B^T(B\mathbf{x} - \mathbf{b}), \quad (13a)$$

and

$$\frac{d\mathbf{y}}{dt} = B(\mathbf{x} + B^T \mathbf{y} - Q\mathbf{x} - \mathbf{c})^+ - \mathbf{b}, \quad (13b)$$

where $\mathbf{x} \in R^n$ and $\mathbf{y} \in R^r$. The CRNN is based on the following equivalent system of two equations

$$\begin{cases} (\mathbf{x} + B^T \mathbf{y} - Q\mathbf{x} - \mathbf{c})^+ = \mathbf{x} \\ B\mathbf{x} = \mathbf{b}. \end{cases} \quad (14)$$

One direct extension of the CRNN (13) were developed by Xia and Wang (1998a). Another direct extension of the CRNN defined in (11) were studied by Leung et al. (2001) for solving the QP problem as follows

$$\begin{aligned} \frac{d\mathbf{x}}{dt} = & -\beta_1(\mathbf{x}, \mathbf{y}) + (-\mathbf{x})^+ + Q(B^T \mathbf{y} - Q\mathbf{x} - \mathbf{c})^+ \\ & + B^T(-B\mathbf{x} + \mathbf{b})^+, \end{aligned} \quad (15a)$$

and

$$\frac{d\mathbf{y}}{dt} = -\beta_2(\mathbf{x}, \mathbf{y}) + B(B^T \mathbf{y} - Q\mathbf{x} - \mathbf{c})^+ + (-\mathbf{y})^+, \quad (15b)$$

where $\mathbf{x} \in R^n$, $\mathbf{y} \in R^r$, $v(\mathbf{x}, \mathbf{y}) = \mathbf{x}Q\mathbf{x} + \mathbf{c}^T \mathbf{x} - \mathbf{b}^T \mathbf{y}$, and

$$\begin{aligned} \beta_1(\mathbf{x}, \mathbf{y}) &= \begin{cases} 2v(\mathbf{x}, \mathbf{y})(2Q\mathbf{x} + \mathbf{c}) & v \geq 0 \\ 0 & v < 0 \end{cases}, \\ \beta_2(\mathbf{x}, \mathbf{y}) &= \begin{cases} -2v(\mathbf{x}, \mathbf{y})\mathbf{b} & v \geq 0 \\ 0 & v < 0 \end{cases} \end{aligned} \quad (16)$$

It is based on an equivalent system of the following equations

$$\begin{cases} B\mathbf{x} = \mathbf{b}, \quad Q\mathbf{x} + \mathbf{c} + B^T \mathbf{y} \geq 0, \quad \mathbf{x} \geq 0 \\ \mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T \mathbf{x} - \mathbf{b}^T \mathbf{y} = 0, \end{cases} \quad (17)$$

The CRNN (13) and the CRNN (15) were shown to be globally convergent to an optimal solution of the QP but the CRNN (13) has a lower computational complexity than the CRNN (15).

Tao et al. (2001) developed a cooperative recurrent neural network for solving the following CP problem

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & B\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq 0. \end{aligned} \quad (18)$$

This CRNN can be described by

$$\frac{d\mathbf{x}}{dt} = (\mathbf{x} - \nabla f(\mathbf{x}) + B^T \mathbf{y})^+ - \mathbf{x} \quad (19a)$$

and

$$\frac{d\mathbf{y}}{dt} = -B(\mathbf{x} - \nabla f(\mathbf{x}) + B^T \mathbf{y})^+ + \mathbf{b}, \quad (19b)$$

where $\mathbf{x} \in R^n$ and $\mathbf{y} \in R^r$. Xia and Wang (1996) presented an improved cooperative recurrent neural network for solving the same problem. This CRNN is defined by

$$\frac{d\mathbf{x}}{dt} = (\mathbf{x} - \nabla f(\mathbf{x}) + B^T \mathbf{y})^+ - \mathbf{x} \quad (20a)$$

and

$$\frac{dy}{dt} = -Bx + b. \quad (20b)$$

It is easy to see that the proposed neural network in (20) has a simpler structure than (19) because of no additional nonlinear term: $(x - \alpha \nabla f(x) + \alpha A^T y)^+$. Under the condition that $\nabla^2 f(x)$ is positive definite, both the CRNN model (19) and the CRNN model (20) can converge globally to the optimal solution. Furthermore, the improved cooperative recurrent neural network in (20) will become the Lagrange programming network in (7) if there is no constraint $x \geq 0$ in (18).

Xia (2004) developed a cooperative neural network, called an extended projection neural network, for solving the NCP problem. According to the KKT condition and well known projection techniques (Kinderlehrer and Stampacchia 1980), the NCP problem is first decomposed into a system of three equations as follows

$$\begin{cases} P_X(x - (\nabla f(x) + \nabla g(x)y - B^T z)) = x \\ (y + g(x))^+ = y \\ Bx = b \end{cases} \quad (21)$$

where $(y)^+ = [(y_1)^+, \dots, (y_m)^+]^T$ with $(y_i)^+ = \max\{0, y_i\}$, $X = \{x \in R^n | l \leq x \leq d\}$, $P_X: R^n \rightarrow X$ is a projection function, and $P_X(x) = [P_X(x_1), \dots, P_X(x_n)]^T$, and for $i = 1, \dots, n$

$$P_X(x_i) = \begin{cases} l_i & x_i < l_i \\ x_i & l_i \leq x_i \leq d_i \\ d_i & x_i > d_i. \end{cases}$$

Based on (21), the CRNN model is described as

$$\frac{dx}{dt} = \{P_X(x - (\nabla f(x) + \nabla g(x)y - B^T z)) - x\}, \quad (22a)$$

$$\frac{dy}{dt} = \{(y + g(x))^+ - y\}, \quad (22b)$$

and

$$\frac{dz}{dt} = \{-Bx + b\}, \quad (22c)$$

where $x \in R^n$, $y \in R^m$, and $z \in R^r$. As one special case, the CRNN model (22) will become the CRNN model (20) when $X = R_+^n$ and $g(x) = 0$. As another special case, the CRNN model (22) can be used to solve the following CP problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0, \quad Bx = b, \end{aligned} \quad (23)$$

The corresponding CRNN model can be described as

$$\frac{dx}{dt} = \{-(\nabla f(x) + \nabla g(x)y - B^T z)\}, \quad (24a)$$

$$\frac{dy}{dt} = \{(y + g(x))^+ - y\}, \quad (24b)$$

and

$$\frac{dz}{dt} = \{-Bx + b\}, \quad (24c)$$

where $x \in R^n$, $y \in R^m$, and $z \in R^r$. Gao et al. (2005) presented an modified CRNN for solving the NCP problem, whose model is defined by

$$\frac{dx}{dt} = \{-(\nabla f(x) + \nabla g(x)(y + g(x))^+ - B^T(z - Bx + b))\}, \quad (25a)$$

$$\frac{dy}{dt} = \{(y + g(x))^+ - y\}, \quad (25b)$$

and

$$\frac{dz}{dt} = \{-Bx + b\}, \quad (25c)$$

where $x \in R^n$, $y \in R^m$, and $z \in R^r$. It is easy to see that the only difference between the CRNN models (24) and (25) is that the variable y of the CRNN model (24) is replaced by $(y + g(x))^+$ in the CRNN (25). As a result, the CRNN model (25) has an additional hidden layer structure that makes it more complex. Moreover, theoretically, the CRNN (25) was only shown to solve the NCP problem, where $f(x)$ is strictly convex. By comparison, the CRNN model (22) and (24) were guaranteed to solve both the NCP problem and a class of non-convex programming problems (Xia et al. 2007).

Finally, we review the model complexity of standing neural networks in (2), (20), and (25) for solving the NCP (1). Kennedy and Chua network in (2) has a single-layer structure and requires n integrators and piecewise activation functions, m^2 connection weights for $B^T B$, n processors for $\nabla f(x)$, mn processors for $\nabla g(x)$, m processors for $g(x)$, and some adders. The modified projection network in (25) has a two-layer structure and requires $n + m + r$ integrators and $n + 2m$ piecewise activation functions, $2rn$ connection weights for B^T and B , r^2 connection weights for $B^T B$, n processors for $\nabla f(x)$, mn processors for $\nabla g(x)$, m processors for $g(x)$, and some adders. The extended projection network in (20) has a two-layer structure and requires $n + m + r$ integrators and $n + 2m$ piecewise activation functions, $2rn$ connection weights for B^T and B , n processors for $\nabla f(x)$, mn processors for $\nabla g(x)$, m processors for $g(x)$, and some adders.

Cooperative recurrent neural networks for the LAD estimation

Least absolute deviation problems and formulations

Consider the following least absolute deviation (LAD) problem with general linear constraints.

$$\begin{aligned} \min \quad & f(\mathbf{x}) = \|\mathbf{D}\mathbf{x} - \mathbf{c}\|_1 \\ \text{s.t.} \quad & \mathbf{B}\mathbf{x} = \mathbf{b}, \\ & \mathbf{A}\mathbf{x} \leq \mathbf{p}. \end{aligned} \quad (26)$$

where $\mathbf{D} \in \mathbb{R}^{l \times n}$, $\mathbf{B} \in \mathbb{R}^{m \times n}$, $\mathbf{A} \in \mathbb{R}^{r \times n}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{c} \in \mathbb{R}^l$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{p} \in \mathbb{R}^r$, $\|\mathbf{c}\|_1 = \sum_{i=1}^l |c_i|$. The considered LAD problem is of major importance in the fields of linear signal and image processing, regression estimation, and system identification. One reason for solving the LAD problem is that the solutions of (26) are more robust than least square solutions in the presence of outliers, instead of solving the corresponding least square problems. Another reason is that the LAD estimation may be efficient under non-Gaussian environments such as the Laplacian.

There are several useful special cases. They are:

(i) linearly constrained L_1 norm problems

$$\begin{aligned} \min \quad & f(\mathbf{x}) = \|\mathbf{x}\|_1 \\ \text{s.t.} \quad & \mathbf{B}\mathbf{x} = \mathbf{b}, \\ & \mathbf{A}\mathbf{x} \leq \mathbf{p} \end{aligned} \quad (27)$$

(ii) the unconstrained LAD problem

$$\min \quad f(\mathbf{x}) = \|\mathbf{D}\mathbf{x} - \mathbf{c}\|_1, \quad (28)$$

(iii) the LAD problem with limit constraints

$$\begin{aligned} \min \quad & f(\mathbf{x}) = \|\mathbf{D}\mathbf{x} - \mathbf{c}\|_1 \\ \text{s.t.} \quad & \mathbf{x} \in X, \end{aligned} \quad (29)$$

where $X = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{l} \leq \mathbf{x} \leq \mathbf{d}\}$, $\mathbf{l} = [l_1, \dots, l_n]^T$, $\mathbf{d} = [d_1, \dots, d_n]^T$, and (vi) the generalized LAD problem with a linear constraint

$$\begin{aligned} \min \quad & f(\mathbf{x}, \mathbf{z}) = \|\mathbf{D}\mathbf{x} - \mathbf{z} - \mathbf{c}\|_1 \\ \text{s.t.} \quad & \mathbf{a}^T \mathbf{x} = 1 \quad \mathbf{z} \in X. \end{aligned} \quad (30)$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^l$.

The linearly constrained L_1 norm problem occurs in field of signal reforestation due to its sparse solution property. The unconstrained LAD problem and the LAD problem with limit constraints are solved for linear signal and image processing, and regression estimation. Solving the generalized LAD problem is to estimate the parameter of autoregressive signals.

Different from previous smooth NP problems, the objective functions of the LAD problem is nonsmooth and thus the LAD problem is difficult to solve in analysis and computation. There are mainly two types of numerical methods for solving the unconstrained LAD problem. One type makes use of the Linear programming (LP) procedure such as the Karmarkar's algorithm (Ruzinsky and Olsen 1989). Another type is the Huber-approximate method (Li and Swetits 1998). However, these approaches could not obtain the global optimal solution. Using switch control techniques, Cichocki and Unbehauen (1993) proposed one continuous-time recurrent neural network for solving the

constrained L_1 linear estimation problem. Based on the penalty method, He and Sun (2001) presented another continuous-time recurrent neural network for solving the L_1 nonlinear estimation problem with linear equality constraints. In addition to higher complexity in implementation, they converge to an approximate solution under the additional condition.

Cooperative recurrent neural network models

In the recent decade, cooperative recurrent neural networks for solving the LAD problems have been developed and successfully applied to the LAD problems. The basic idea is first to convert the LAD problems into several subproblems and then combine neural network models which solve these subproblems. In Xia (1996a), based on standard optimization techniques, the unconstrained LAD problem (28) was equivalently converted into one system of two equations

$$\begin{cases} \mathbf{D}^T \mathbf{y} = 0 \\ P_{X_1}(\mathbf{y} + \mathbf{D}\mathbf{x} - \mathbf{c}) = \mathbf{y}, \end{cases} \quad (31)$$

where $\mathbf{y} \in \mathbb{R}^l$ is a Lagrange multiplier vector, $X_1 = \{\mathbf{y} \in \mathbb{R}^l | \|\mathbf{y}\|_\infty \leq 1\}$, $P_{X_1}(\mathbf{y}) = [P_{X_1}(y_1), \dots, P_{X_1}(y_l)]^T$, and

$$P_{X_1}(y_i) = \begin{cases} -1 & y_i < -1 \\ y_i & -1 \leq y_i \leq 1 \\ 1 & y_i > 1, \end{cases} \quad (32)$$

The LAD problem (29) with limit constraints was equivalently reformulated as another system of two piecewise equations (Xia 1997)

$$\begin{cases} \mathbf{x} = P_X(\mathbf{x} - \mathbf{D}^T \mathbf{y}) \\ \mathbf{y} = P_{X_1}(\mathbf{y} - \mathbf{c} + \mathbf{D}\mathbf{x}), \end{cases} \quad (33)$$

where $\mathbf{y} \in \mathbb{R}^l$ is a Lagrange multiplier vector, $P_X(\mathbf{x}) = [P_X(x_1), \dots, P_X(x_n)]^T$, and

$$P_X(x_i) = \begin{cases} -l_i & x_i < -l_i \\ x_i & -l_i \leq x_i \leq d_i \\ d_i & x_i > d_i, \end{cases}$$

The generalized LAD problem (30) can be equivalently reformulated as the following system of three piecewise equations

$$\begin{cases} \mathbf{x} = P_{X_0}(\mathbf{x} - \mathbf{D}^T \mathbf{y}), \\ \mathbf{y} = P_{X_1}(\mathbf{y} - \mathbf{z} + \mathbf{D}\mathbf{x} - \mathbf{c}), \\ \mathbf{z} = P_X(\mathbf{z} + \mathbf{y}), \end{cases} \quad (34)$$

where $\mathbf{y} \in \mathbb{R}^l$ is a Lagrange multiplier vector, $P_{X_0}(\mathbf{x}) = \arg \min_{\mathbf{x}' \in \mathbb{R}^n} \|\mathbf{x} - \mathbf{x}'\|_2$, $P_{X_1}(\mathbf{y})$ is defined in (31), $P_X(\mathbf{x})$ is defined in (33). The linearly constrained L_1 norm problem (27) can be transformed into a system of three linear piecewise equations:

$$\begin{cases} B\mathbf{x} = \mathbf{b} \\ A\mathbf{x} = P_{X_2}(A\mathbf{x} - \mathbf{y}_{II}) \\ B^T \mathbf{y}_I + A^T \mathbf{y}_{II} = P_{X_1}(\mathbf{x} + B^T \mathbf{y}_I + A^T \mathbf{y}_{II}). \end{cases} \quad (35)$$

where $\mathbf{y} \in R^m$, $\mathbf{y}_I \in R^m$, $\mathbf{y}_{II} \in R^r$, $X_2 = \{\mathbf{u} \in R^r | \mathbf{u} \leq \mathbf{d}\}$ and $P_{X_2}(\mathbf{u}) = [P_{X_2}(u_1), \dots, P_{X_2}(u_r)]^T$, and

$$P_{X_2}(u_i) = \begin{cases} u_i & u_i \leq d_i \\ d_i & u_i > d_i. \end{cases} \quad (36)$$

Xia (1997) first proposed a cooperative neural network (CRNN) for solving the following L_1 norm problem

$$\begin{aligned} \min \quad & f(\mathbf{x}) = \|\mathbf{x}\|_1 \\ \text{s.t.} \quad & \mathbf{p}_1 \leq C\mathbf{x} \leq \mathbf{p}_2. \end{aligned} \quad (37)$$

$C \in R^{l \times n}$, $\mathbf{p}_1 \in R^l$, $\mathbf{p}_2 \in R^l$. The CRNN model is described by

$$\frac{d\mathbf{x}}{dt} = \{C^T[C\mathbf{x} - \mathbf{y} - P_{X_3}(C\mathbf{x} - \mathbf{y})] + P_{X_1}(\mathbf{x} + C^T \mathbf{y})\}, \quad (38a)$$

and

$$\frac{d\mathbf{y}}{dt} = C[\mathbf{x} + C^T \mathbf{y} - P_{X_1}(C\mathbf{y} + \mathbf{x})] - P_{X_3}(C\mathbf{x} - \mathbf{y}). \quad (38b)$$

It is based on the following equivalent system of two piecewise equations

$$\begin{cases} P_{X_1}(\mathbf{x} + C^T \mathbf{y}) = C^T \mathbf{y} \\ P_{X_3}(C\mathbf{x} - \mathbf{y}) = C\mathbf{x} \end{cases} \quad (39)$$

where $\mathbf{y} \in R^l$ is a Lagrange multiplier vector, $X_3 = \{\mathbf{y} \in R^l | \mathbf{p}_1 \leq \mathbf{y} \leq \mathbf{p}_2\}$, $P_{X_3}(\mathbf{y}) = [P_{X_3}(y_1), \dots, P_{X_3}(y_l)]^T$, $P_{X_3}(y_i)$ is similar to ones defined in (33), and $P_{X_1}(\mathbf{x})$ is defined in (31). The CRNN (38) is shown to be globally convergent to the optimal solution of the L_1 norm problem (36) without any additional condition.

Following the CRNN (38), a cooperative neural network for solving the linearly constrained L_1 norm problem (27) can be derived directly. Let

$$C = \begin{pmatrix} B \\ A \end{pmatrix}, \quad \mathbf{p}_1 = \begin{pmatrix} \mathbf{b} \\ \boldsymbol{\xi} \end{pmatrix}, \quad \mathbf{p}_2 = \begin{pmatrix} \mathbf{b} \\ \mathbf{p} \end{pmatrix},$$

and $l = m + r$, where $\boldsymbol{\xi} = [-\infty, \dots, -\infty] \in R^r$. Then the CRNN model for solving the linearly constrained L_1 norm problem can be given by

$$\frac{d\mathbf{x}}{dt} = -\{(B^T B + A^T A)\mathbf{x} - B^T \mathbf{b} - A^T P_{X_2}(A\mathbf{x} - \mathbf{y}_{II}) + F_1(\mathbf{x}, \mathbf{y})\} \quad (40a)$$

$$\frac{d\mathbf{y}_I}{dt} = -\{B F_1(\mathbf{x}, \mathbf{y}) + B\mathbf{x} - \mathbf{b}\} \quad (40b)$$

$$\frac{d\mathbf{y}_{II}}{dt} = -\{A F_1(\mathbf{x}, \mathbf{y}) + A\mathbf{x} - P_{X_2}(A\mathbf{x} - \mathbf{y}_{II})\} \quad (40c)$$

where $\mathbf{y}_I \in R^m$, $\mathbf{y}_{II} \in R^r$, and

$$F_1(\mathbf{x}, \mathbf{y}) = P_{X_1}(\mathbf{x} + B^T \mathbf{y}_I + A^T \mathbf{y}_{II}) - B^T \mathbf{y}_I - A^T \mathbf{y}_{II}.$$

In term of the system of two Eq. 31, Xia and Wang (1998b) presented a cooperative neural network model for solving the unconstrained LAD problem below

$$\frac{d\mathbf{x}}{dt} = -D^T P_{X_1}(\mathbf{y} - \mathbf{c} + D\mathbf{x}) \quad (41a)$$

and

$$\frac{d\mathbf{y}}{dt} = -(I + DD^T)\mathbf{y} + P_{X_1}(\mathbf{y} - \mathbf{c} + D\mathbf{x}). \quad (41b)$$

Wang et al. (2000) studied the robustness performance and application of the CRNN (41), Tao and Fang (2000), and Xia and Wang (2004a) studied an extension of (41) for solving a maximum optimization problem, respectively. Compared with the individual neural network model by Cichocki and Unbehauen, the CRNN (41) can converge globally to the optimal solution of the unconstrained LAD problem without any condition.

Xia and Kamel (2007a) developed a cooperative neural network model for solving the LAD problem with limit constraints

$$\frac{d\mathbf{x}}{dt} = \{-\mathbf{x} + P_X(\mathbf{x} - D^T \mathbf{y}) - D^T(P_{X_1}(\mathbf{y} - \mathbf{c} + D\mathbf{x}) - \mathbf{y})\}, \quad (42a)$$

$$\frac{d\mathbf{y}}{dt} = \{-\mathbf{y} + P_{X_1}(\mathbf{y} - \mathbf{c} + D\mathbf{x}) + D(P_X(\mathbf{x} - D^T \mathbf{y}) - \mathbf{x})\} \quad (42b)$$

The CRNN model (42) is based on the system of two piecewise Eqs. 33. Compared with the CRNN model (41), the CRNN model (42) can converge globally to an optimal solution of the LAD problem with limit constraints and contain the CRNN model (41) as its special cases (2000).

Recently, Xia and Kamel (2007b) developed a CRNN method for solving the generalized LAD problem. Its CRNN model is defined by

$$\frac{d\mathbf{x}}{dt} = -\{\mathbf{x} - P_{X_0}(\mathbf{x} - D^T \mathbf{y}) - D^T(\mathbf{y} - P_{X_1}(\mathbf{y} + D\mathbf{x} - \mathbf{c} - \mathbf{z}))\} \quad (43a)$$

$$\frac{d\mathbf{y}}{dt} = -\{\mathbf{y} - P_{X_1}(\mathbf{y} + D\mathbf{x} - \mathbf{c} + \mathbf{z}) + D(\mathbf{x} - P_{X_0}(\mathbf{x} - D^T \mathbf{y}) - (\mathbf{z} - P_X(\mathbf{z} + \mathbf{y})))\} \quad (43b)$$

$$\frac{d\mathbf{z}}{dt} = -\{\mathbf{z} - P_X(\mathbf{z} + \mathbf{y}) + (\mathbf{y} - P_{X_1}(\mathbf{y} + D\mathbf{x} - \mathbf{c} + \mathbf{z}))\} \quad (43c)$$

Compared with the CRNN (41), the CRNN (42) can converge globally to an optimal solution of the GLAD

problem and can generalize significantly the CRNN (41) in estimate accuracy.

Finally, we review the model complexity of the CRNNs in (38), (41), (42), and (43). The CRNN in (38) has a two-layer structure and requires $n + l$ integrators and $n + l$ piecewise activation functions, n^2 connection weights for $C^T C$, $2nl$ connection weights for C , $2nl$ connection weights for C^T , and some adders. The CRNN in (41) has a two-layer structure and requires $n + l$ integrators and l piecewise activation functions, l^2 connection weights for $I + DD^T$, nl connection weights for D , nl connection weights for D^T , and some adders. The CRNN in (42) has a two-layer structure and requires $n + l$ integrators and $n + l$ piecewise activation functions, $2nl$ connection weights for D , $2nl$ connection weights for D^T , and some adders. The CRNN in (43) has a two-layer structure and requires $2n + l$ integrators and $2n + l$ piecewise activation functions, $2nl$ connection weights for D , $2nl$ connection weights for D^T , and some adders.

Survey of applications of CRNNs

In recent years cooperative recurrent neural networks have become more popular for solving problems in many applications areas. In section, we survey some of the applications of CRNNs and their current status.

CRNNs for classification

A classification problem occurs when an object needs to be assigned into a predefined class based on a group of observed data related to that object (Zhang 2000). Many problems in business, science, industry, and medicine can be treated as classification problems. The neural network approach to classification is one of the most competent in terms of performance due to its adaptive learning, high non-linearity, robustness and plausibility for hardware implementation. However, traditional neural network classification methods suffer from the inefficiency and the inability of the learning algorithm to converge when the classification problems are large in size and of high complexity (Auda and Kamel 1998a). To avoid these drawbacks, a cooperative modular neural network (CMNN) approach (Auda and Kamel 1999) was developed and successfully applied to multi-class problems. The CMNN approach first decomposes the classification problem into several simpler sub-problems and then solves each sub-problem using a simple model. A cooperative scheme based on the voting technique is used to integrate the information available at its models. The overall classification performance is dependent on the success of

integrating the model-decisions into a global one representing the whole network. Recently, cooperative recurrent neural network approaches are studied to be effectively applied to linearly inseparable problems by using support vector machine (SVM) techniques (Tan et al. 2000; Anguita and Boni 2002; Xia and Wang 2004b). Because of the main feature of SVM which transforms the classification problem into a linearly constrained convex quadratic programming problem (Cortes and Vapnik 1995), the CRNN approach based on SVM can obtain global optimal solution of the classification problem.

Classification and reformulation problems

Consider a set of training examples

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$$

where the i -th example $\mathbf{x}_i \in R^n$ (n is the dimension of the input space) belongs to two separate classes labeled by $y_i \in \{-1, 1\}$. The classification problem is to find a hyperplane in a high dimensional feature space Z , which divides the set of examples in the feature space such that all the points with the same label are on the same side of the hyperplane (Burges 1998). The SVM technique is to find an optimal decision function of the classifier given by

$$f(\mathbf{x}) = \text{sgn} \left[\sum_{i=1}^N \alpha_i y_i \cdot K(\mathbf{x}_i, \mathbf{x}_j) + b \right], \quad (44)$$

where $K(\mathbf{x}, \mathbf{y})$ is a given kernel function and $\{\alpha_j\}$ is an optimal solution of the following quadratic programming

$$\begin{aligned} \min \quad & \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j (K(\mathbf{x}_i, \mathbf{x}_j)) \alpha_i \alpha_j + \sum_{i=1}^N \alpha_i \\ \text{s.t.} \quad & \begin{cases} \sum_{i=1}^N \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq c, \quad i = 1, \dots, N \end{cases} \end{aligned} \quad (45)$$

Let $a_i = \alpha_i y_i$. Then (45) can be rewritten as

$$\begin{aligned} \min \quad & \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N a_i a_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^N a_i \\ \text{s.t.} \quad & \begin{cases} \sum_{i=1}^N a_i y_i = 0 \\ -c_i^1 \leq a_i \leq c_i^2, \quad i = 1, \dots, n \end{cases} \end{aligned} \quad (46)$$

where $c_i^1 = -c \cdot \text{sign}(1 - y_i)$ and $c_i^2 = c \cdot \text{sign}(1 + y_i)$ for $i = 1, \dots, l$. Furthermore, in term of (21), (46) can be converted into the following system of two equations

$$\begin{cases} P_X(\mathbf{a} - (Q\mathbf{a} - \mathbf{y} + \mathbf{e}\mu) - \mathbf{a}) = 0 \\ \mathbf{e}^T \mathbf{a} = 0, \end{cases} \quad (47)$$

where $\mathbf{a} = (a_1, a_2, \dots, a_N)^T$, $\mathbf{y} = (y_1, y_2, \dots, y_N)^T$, $\mathbf{e} = (1, \dots, 1)^T \in R^N$, $P_X(\mathbf{a})$ is defined in (21), and the ij -th entry of matrix Q equals to $K(\mathbf{x}_i, \mathbf{x}_j)$. Using the fact that $\mu^* = b$ shown in Anguita and Boni (2002), (47) can be rewritten as

$$\begin{cases} P_X(\mathbf{a} - (Q\mathbf{a} - \mathbf{y} + \mathbf{eb}) - \mathbf{a}) = 0 \\ \mathbf{e}^T \mathbf{a} = 0, \end{cases} \quad (48)$$

CRNNs for real-time classification

By extending the CRNN in (13), one cooperative recurrent neural network for classification was studied in Tan et al. (2000). Its dynamical equation is described by

$$\frac{d\mathbf{a}}{dt} = \lambda \{ (I + Q)(P_X(\mathbf{a} - (Q\mathbf{a} + \mathbf{eb} - \mathbf{y})) - \mathbf{a}) - \mathbf{e}\mathbf{e}^T \mathbf{a} \} \quad (51a)$$

and

$$\frac{db}{dt} = -\lambda \mathbf{e}^T P_X(\mathbf{a} - (Q\mathbf{a} + \mathbf{eb} - \mathbf{y})), \quad (51b)$$

where $\lambda > 0$ is a design constant. Another cooperative recurrent neural network was presented in Xia and Wang (2004b).

$$\frac{d\mathbf{a}}{dt} = \lambda \{ P_X(\mathbf{a} - (Q\mathbf{a} + \mathbf{eb} - \mathbf{y})) - \mathbf{a} \} \quad (52a)$$

and

$$\frac{db}{dt} = -\lambda \mathbf{e}^T \mathbf{a} \quad (52b)$$

The above CRNN is obtained directly from the CRNN (22) since (48) is a special case of (21).

The two CRNNs (51) and (52) can converge globally to the optimal solution of (45). On the other side, the CRNN (52) requires $N(N+2)$ multiplications and $N(N+3)$ additions/subtractions per iteration. While the CRNN (51) requires $N(N+2) + 2N^2$ multiplications and $N(N+3) + N(2N+1)$ additions/subtractions per iteration. Therefore, CRNN (52) has a lower model complexity for implementation than CRNN (51). Figure 3 displays a block diagram of the CRNN (52).

Experimental results

Example 1. Consider the iris benchmark data problem. The data of the iris problem are characterized with four attributes (i.e., the petal length and width, setal length and width). The goal is to classify the data based on these four attributes. The dataset consist of 150 samples belonging to three classes (i.e., virginica, versicolor, setosa), each class has 50 samples. We choose 120 samples for the training set and the remaining is used as testing set. We use the CRNN (52) with the polynomial kernel to train a SVM for the classification problem. For illustration of the CRNN (52), we train the SVM to separate class I from class II with a penalty factor $c = 0.125$ and to separate class II from class III with a penalty factor $c = 0.25$. We choose the following polynomial function

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^4$$

as the kernel of the SVM. Let the parameter $\lambda = 100$. Figure 4 plots the number of support vector points obtained by CRNN (52) for the class I and class II and Fig. 5 plots

Fig. 2 A cooperative neural network structure with three sub models

Fig. 3 A block diagram of the CRNN (52)

Fig. 4 Support vector points of the SVM using the CRNN algorithm (52) with a polynomial kernel for class I and II

Fig. 5 Support vector points of the SVM using the CRNN algorithm (52) with a polynomial kernel for class II and III

Fig. 6 Convergence of trajectory of the the CRNN algorithm (52) for class I and II

Fig. 7 Convergence of trajectory of the the CRNN algorithm (52) for class II and III

the number of the support vectors for the class II and class III. Figures 6 and 7 display the convergence of the trajectory of the CRNN (52) for the class I and class II and for the class II and class III, respectively.

Example 2. Consider the adult benchmark problem which is taken on the set of training data of the UCI “adult” benchmark. The benchmark problem is to predict whether a household has an income greater than \$50,000 based on 14 other fields in a census form. Eight of the fields are categorical, while six are continuous. The six fields are quantized into quintile, which yields a total of 123 sparse binary features. There are 9 training files in the original data of the adult benchmark. For simplicity, we adopted the first part of the training set to train an SVM. There are 1605 samples in the training set and 2000 samples in the testing set in our experiment. We use the CRNN algorithm (52) to train the SVM of Gaussian RBF kernel with width of 10 and $c = 0.5$. Let $\lambda = 10$ and $t = 2$, and let the initial point $\mathbf{z}_0 \in R^{1606}$ with the element being 1. The CRNN algorithm (52) gives 633 support vectors. We compare the result computed by the CRNN (52) with the results given by existing SOR, SMO, and SVM-light algorithms from Mangasarian and Musicant (1999). The comparison is listed in Table 1. From the listed results we see that the standard results are basically same as present results.

CRNNs for data fusion

Data fusion estimation

With the availability of multiple sensory devices, multisensor fusion or data fusion has received considerable attention in recent decades (Varshney 1997). The main potential advantage of data fusion is that by integrating

Table 1 Comparisons of results of the SOR, SMO, SVM-light, and the CRNN algorithm (52)

Algorithms	Training size	No. of iteration	No. of SV	Test set accuracy
SOR	1605	924	635	84.06
SMO	1605	3230	633	84.06
SVM-light	1605	294	634	84.25
CRNN	1605	567	633	84.1

complementary information from different sources, the uncertainty of fused information is minimized and the fused performance can be improved. As a result, data fusion techniques have been applied to many fields such as machine vision, remote sensing, signal and image progressing, and system identification (Xia et al. 2002b, 2006; Xia and Kamel 2007b, c, 2008; Agganwal 1993; Durrant-Whyte 1985; Smith and Cheesman 1986; Avitzour 1992). Consider a multi-sensor system with K ($K \geq 2$) sensors. Let the k th sensor measurement be expressed as

$$x_k(t) = a_k s(t) + n_k(t) \quad (k = 1, \dots, K; t = 1, \dots, N), \quad (53)$$

where a_k is a positive scaling coefficient, N is the number of sensor measurements, $s(t)$ denotes the unknown signal, and $n_k(t)$ represents the additive noise process at the k th sensor with zero mean. Using matrix and vector notations, the above representation can be written as

$$\mathbf{x}(t) = \mathbf{a}s(t) + \mathbf{n}(t),$$

where $\mathbf{a} = [a_1, \dots, a_K]^T$, $\mathbf{x}(t) = [x_1(t), \dots, x_K(t)]^T$, and $\mathbf{n}(t) = [n_1(t), \dots, n_K(t)]^T$. The main goal of data fusion is to combine the sensory information in a systematic manner to get a good fusion estimation of $s(t)$.

Consider the fusion estimation in a linear combination of sensor measurements

$$\beta(t) = \sum_{k=1}^K w_k x_k(t) = \mathbf{w}^T \mathbf{x}(t), \quad (54)$$

where $\mathbf{w} = [w_1, w_2, \dots, w_K]^T$ is a weight vector.

CRNNs for data fusion

There are mainly three cooperative recurrent neural network approaches to data fusion. The first one is based on a linearly constrained least squares (LCLS) method for solving the following optimization problem

$$\begin{aligned} \min \quad & \mathbf{w}^T R \mathbf{w} \\ \text{s.t.} \quad & \mathbf{a}^T \mathbf{w} = 1, \end{aligned} \quad (55)$$

where $R = \sum_{t=1}^N \mathbf{x}(t) \mathbf{x}(t)^T / N$. Its CRNN model is defined as follows (Xia et al. 2002b).

State equation

$$\begin{pmatrix} \frac{d\mathbf{w}}{dt} \\ \frac{dy}{dt} \end{pmatrix} = -\lambda \begin{pmatrix} W_1 \mathbf{w} + W_2 y - \mathbf{a} \\ W_3 \mathbf{w} + K_a y \end{pmatrix}, \quad (56)$$

where $\lambda > 0$ is a design constant.

Output equation:

$$z(t) = \mathbf{w}(t)^T \mathbf{x}(t),$$

where $y \in R$, $\mathbf{w} \in R^K$, $K_a = \mathbf{a}^T \mathbf{a}$, and the weight parameters are given by

$$\begin{cases} W_1 = 4R^T R + \mathbf{a}\mathbf{a}^T \\ W_2 = 2R\mathbf{a} \\ W_3 = 2\mathbf{a}^T R. \end{cases}$$

The second one is established by using the high order statistic method for solving

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{w}) = \frac{\sum_{t=1}^T \beta^4(t)}{(\sum_{t=1}^T \beta^2(t))^2} \\ \text{subject to} \quad & \mathbf{a}^T \mathbf{w} = 1, \end{aligned} \quad (57)$$

where $\beta(t)$ is defined in (54). Its CRNN model is described as (Xia et al. 2006).

State equation

$$\begin{pmatrix} \frac{d\mathbf{w}}{dt} \\ \frac{dy}{dt} \end{pmatrix} = -\lambda \begin{pmatrix} \nabla^2 f(\mathbf{w})(\nabla f(\mathbf{w}) + \mathbf{b}y) + D\mathbf{w} - \mathbf{b} \\ \mathbf{b}^T \nabla f(\mathbf{w}) + K_b y \end{pmatrix}, \quad (58)$$

where $D = \mathbf{b}\mathbf{b}^T$, $K_b = \mathbf{b}^T \mathbf{b}$, $\nabla f(\mathbf{w})$ is the gradient of f , and $\nabla^2 f(\mathbf{w})$ is Hessian matrix of f . The third one is based on a linearly constrained least absolute deviation (LCLAD) method for finding an optimal weight vector $\mathbf{w}^*$ of the following optimization problem

$$\begin{aligned} \text{minimize} \quad & \Phi(\mathbf{w}, \mathbf{z}) = \|\mathbf{A}\mathbf{w} - \mathbf{b} + \mathbf{z}\|_1 \\ \text{subject to} \quad & \mathbf{a}^T \mathbf{w} = 1, \mathbf{z} \in X, \end{aligned} \quad (59)$$

where $\|\mathbf{z}\|_1 = \sum_{i=1}^N |z_i|$, $X = \{\mathbf{z} \in R^N | \gamma_1 \mathbf{1} \leq \mathbf{z} \leq \gamma_2 \mathbf{h}\}$ is called a set of the noise error, γ_1 and γ_2 are designing constants, and

$$\mathbf{l} = \begin{pmatrix} \mathbf{a}^T \mathbf{n}^-(t_1) / \|\mathbf{a}\|_2^2 \\ \vdots \\ \mathbf{a}^T \mathbf{n}^-(t_N) / \|\mathbf{a}\|_2^2 \end{pmatrix}, \quad \mathbf{h} = \begin{pmatrix} \mathbf{a}^T \mathbf{n}^+(t_1) / \|\mathbf{a}\|_2^2 \\ \vdots \\ \mathbf{a}^T \mathbf{n}^+(t_N) / \|\mathbf{a}\|_2^2 \end{pmatrix},$$

$$\mathbf{A} = \begin{pmatrix} \mathbf{x}(t_1)^T \\ \vdots \\ \mathbf{x}(t_N)^T \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} \mathbf{a}^T \mathbf{x}(t_1) / K \\ \vdots \\ \mathbf{a}^T \mathbf{x}(t_N) / K \end{pmatrix}.$$

Its CRNN is given below (Xia and Kamel 2008)

State equation

$$\begin{cases} \frac{d\mathbf{w}}{dt} = -\lambda\{F_1 - A^T F_2\}, \\ \frac{d\mathbf{v}}{dt} = -\lambda\{A F_1 + F_2 - F_3\}, \\ \frac{d\mathbf{z}}{dt} = -\lambda\{F_3 + F_2\}, \end{cases} \quad (60)$$

Output equation

$$\beta(t) = \mathbf{w}(t)^T \mathbf{x}(t),$$

where $\lambda > 0$ is a design constant, $F_1 = \mathbf{w} - f^1(\mathbf{w} - A^T \mathbf{v})$, $F_2 = \mathbf{v} - f^2(\mathbf{v} + \mathbf{z} + A\mathbf{w} - \mathbf{b})$, $F_3 = \mathbf{z} - f^3(\mathbf{v} + \mathbf{z})$ are three modular functions, respectively, $f^1(\mathbf{w}) = \arg \min_{\mathbf{w}^T \mathbf{a} = 1} \|\mathbf{w} - \mathbf{w}'\|_2$, and $f^2(\mathbf{z})$ and $f^3(\mathbf{z})$ are defined as

$$f^2(z_i) = \begin{cases} -1 & z_i < -1 \\ z_i & -1 \leq z_i \leq 1 \\ 1 & z_i > 1 \end{cases}, \quad f^3(z_i) = \begin{cases} l_i & z_i < l_i \\ z_i & l_i \leq z_i \leq h_i \\ h_i & z_i > h_i \end{cases}$$

respectively. Because the objective function defined in (57) is non-convex, the CRNN (58) based on the high-order statistical method suffers from the local minima problem in finding the optimal fusion solution. The CRNN (56) requires $K^2 + 3K + 1$ multiplications per iteration but obtains an optimal weight vector only under a Gaussian noise environment. By comparison, the CRNN (60) requires $4KN$ multiplications per iteration but can achieve an optimal weight vector under a non-Gaussian noise environment. Figure 8 displays a block diagram of the CRNN (60).

Examples

Example 1. Consider the nonlinear signal/sensor system

$$\begin{cases} s(t+1) = 1.7\exp(-2s^2(t)) + 0.5e(t) \\ x_i(t) = s(t) + n_i(t), \quad i = 1, \dots, K \end{cases} \quad (61)$$

where K is the number of sensors, $s(0)$ is Gaussianly distributed with zero mean and variance of unity, $n_i(t)$ are

Fig. 8 A block diagram of the CRNN (60)

white Gaussian noise, and $e(t)$ is a white noise process uniformly distributed in the interval $[-1, 1]$, where the SNR is equal to 12 dB. Figure 9(a) displays the true signal versus the number of measurements with $N = 100$. In order to recover the true signal from different observed signals contaminated by noise, we consider the data fusion problem (59). Note that $0 \leq s(t) \leq 1.7$. We take the set

$$X = \{\mathbf{z} \in R^N | \mathbf{l} \leq \mathbf{z} \leq \mathbf{h}\},$$

where

$$\mathbf{h} = \begin{pmatrix} \mathbf{c}^T \mathbf{x}(t_1)/K \\ \vdots \\ \mathbf{c}^T \mathbf{x}(t_N)/K \end{pmatrix}, \quad \mathbf{l} = \begin{pmatrix} \mathbf{c}^T \mathbf{x}(t_1)/K - 1.7 \\ \vdots \\ \mathbf{c}^T \mathbf{x}(t_N)/K - 1.7 \end{pmatrix},$$

where $\mathbf{c} = [1, \dots, 1]^T$. We perform the CRNN (60), where $K = 4$, $\mu = 30$, and initial point is zero. Figure 9(c) plots the fused signal by the CRNN (60). For a comparison, we also perform the CRNN (56). Figure 9(b) displays the fused signal employed by the CRNN (56) under same design parameters. Figure 10 lists the relative residual error between the true signal and two predicted signals, where the solid line corresponds to the CRNN (60) and the dashed line to the CRNN (56). Computed results show that the CRNN (60) gives a smaller relative error than the CRNN (56).

Example 2. This example investigates image fusion using the CRNN approach. Image fusion can be used to enhance the quality of an image so that more reliable segmentation can be performed and more discriminating features can be extracted for image processing. Consider noisy versions of Lena image. It is an eight-bit grey level image with 256 by 256 pixels. We assume that two original Lena images were scaled by two unknown coefficients 0.8 and 1.5, respectively. Let K different sensors be used for image collection, where zero mean white contaminated Gaussian noise with distribution of the form

$$f = \frac{1}{2}N(0, 2) + \frac{1}{2}N(0, 100)$$

was added to each scaled image. Figure 10(a) is one noisy Lena image measured by the sensor with the scaling coefficient being 0.8, and Fig. 10(b) is another noisy Lena image measured by one sensor with the scaling coefficient being 1.5.

Since the scaling coefficients are unknown, we first reformulate this image recovery problem as the following data fusion problem

$$\begin{aligned} \min \quad & \Phi_1(\mathbf{w}, \mathbf{z}) = \|A\mathbf{w} - \mathbf{b} + \mathbf{z}\|_1 \\ \text{s.t.} \quad & \mathbf{c}^T \mathbf{w} = 1, \quad \mathbf{z} \in X_\gamma, \end{aligned} \quad (62)$$

where A is defined in (59), $\mathbf{b} = [\mathbf{c}^T \mathbf{x}(t_1)/K, \dots, \mathbf{c}^T \mathbf{x}(t_N)/K]^T$, $\mathbf{c} = [1, \dots, 1]^T \in R^N$, $X_\gamma = \{\mathbf{z} \in R^N | \gamma_1 \mathbf{l} \leq \mathbf{z} \leq \gamma_2 \mathbf{h}\}$, and

Fig. 9 (a) True signal; (b) the fused signal by the CRNN (56); (c) the fused signal by the CRNN (60)

Fig. 10 Predictive relative residual error: The solid line corresponds to the CRNN (60) and the dashed line to the CRNN (56)

$$\mathbf{l}^1 = \begin{pmatrix} \mathbf{c}^T \mathbf{n}^-(t_1)/K \\ \vdots \\ \mathbf{c}^T \mathbf{n}^-(t_N)/K \end{pmatrix}, \quad \mathbf{h}^1 = \begin{pmatrix} \mathbf{c}^T \mathbf{n}^+(t_1)/K \\ \vdots \\ \mathbf{c}^T \mathbf{n}^+(t_N)/K \end{pmatrix}.$$

We use the CRNN (60) to compute the optimal fusion solution. Because $0 \leq s(t) \leq 255$, we take

$$\mathbf{l}^1 = \begin{pmatrix} \mathbf{c}^T \mathbf{x}(t_1)/K - 255 \\ \vdots \\ \mathbf{c}^T \mathbf{x}(t_N)/K - 255 \end{pmatrix}, \quad \mathbf{h}^1 = \begin{pmatrix} \mathbf{c}^T \mathbf{x}^+(t_1)/K \\ \vdots \\ \mathbf{c}^T \mathbf{x}^+(t_N)/K \end{pmatrix}.$$

Let $\gamma_1 = \gamma_2 = 1$, $N = 256$, $K = 20$, and let the initial point be zero. We perform the CRNN (60), where $\mu = 30$ and processing time is set as 10 s. Figure 11(c, d) are fused images by (60) for the cases with two different scaling coefficients, respectively. From Fig. 11(c, d) we can see that the fused image 11(c) is light and the fused image (d) is dark. So, we simply make the dark image scaled by 1.2 and the light image scaled by 0.7. Figure 11(e, f) lists two scaled-fused images, respectively. Apparently, each fused-scaled image has a much better quality than the noisy image and close to the original image. By comparison, both the CRNN (56) and the CRNN (58) don't work well because of the unknown scaling coefficients.

CRNN for image restoration

The aim of image restoration is to obtain a good estimate of the original image from a noise-degraded image. The image restoration problem has been studied extensively due to its many engineering applications. Here we are concerned with the linear discrete image restoration problem (Andrews and Hunt 1977).

Image restoration and estimation

Consider the following image restoration model with additive noise process.

$$\mathbf{g} = H\mathbf{f} + \mathbf{n}, \quad (63)$$

where $\mathbf{f}$ and $\mathbf{g}$ are column vectors containing the samples from the original and the degraded images, respectively. H is the degradation matrix of size $M \times N$. $\mathbf{n}$ is a column vector containing noise samples. We assume that the gray value of an image always satisfies $0 \leq (\mathbf{f})_i \leq L$. The purpose of image restoration is to remove those degradations so that the restored images become as close as possible to the original. The most simple approaches for restoring such degraded images are the mean filters. The mean filter is known to be the best for removing Gaussian white noise from noisy images. The Kalman filter is

Fig. 11 Multisensor image fusion at the pixel level using the CRNN (60): **(a)** the noise image with unknown scaling coefficient 1.5; **(b)** the noise image with unknown scaling coefficient 0.8; **(c)** the fused image

another popular approach. However, H is usually ill-conditioned and can cause a large error to occur in image estimate (Kuo and Mammone 1992). In order to overcome

for the case **(a)**; **(d)** the fused image for the case **(b)**; **(e)** the scaled-fused image for the case **(a)**; **(f)** the scaled-fused image for the case **(b)**

such difficulty, regularization methods have been proposed and widely studied (Mammone and Rothacker 1987; Katsaggelos et al. 1991). The regularization approach is

to find an optimal solution of the following optimization problem

$$\text{minimize } \eta \|Q\mathbf{f}\|_2^2 + \|\mathbf{g} - H\mathbf{f}\|_2^2, \quad (64)$$

where Q is a full rank matrix and $\eta > 0$ is a regularization parameter. The regularization solution can be directly given by

$$\mathbf{f}_\eta = (H^T H + \eta Q^T Q)^{-1} H^T \mathbf{g}. \quad (65)$$

The regularization method can be implemented by individual neural network, called the Hopfield network, for real-time image processing (Paik and Katsaggelos 1992; Perry and Guan 1996; Guan et al. 1997; Sun 2000). However, choosing a proper regularization parameter is a key factor of the regularization method to obtain a good image estimate. Some popular methods for choosing the optimal regularization parameter have been developed in Thompson et al. (1991) and Galatsanos and Katsaggelos (1992). Although the optimal regularization solution can lead to a perfect image estimate, only approximate optimal regularization parameters are available in practice since the noise variance needs to be estimated. In addition, the contaminating noise is usually non-Gaussian (e.g the uniform, Laplacian, or a combination of them). For such consideration, a robust entropic method and iterative method based on high order statistics were developed (Zervakis et al. 1995; Hong et al. 2002). In these two methods, the optimal regularization parameter needs to be estimated still.

Recently, a cooperative recurrent neural network approach was developed to relax the optimal regularization parameter of image restoration under blind Gaussian noise environments (Xia and Kamel 2007b). Let $\{\eta_k\}_1^K$ be a group of regularization parameters with positive values which are not optimal, and let the corresponding regularization

solution be $\mathbf{f}_{\eta_k}$. The CRNN approach is to find a new image estimate

$$\mathbf{f}^* = \sum_{k=1}^K w_k^* \mathbf{f}_{\eta_k}$$

such that $\mathbf{w}^* = [w_1^*, \dots, w_K^*]^T$ is an optimal fusion solution satisfying

$$\begin{aligned} \text{minimize } f_1(\mathbf{w}, \mathbf{z}) &= \|A\mathbf{w} - \mathbf{b} - \mathbf{z}\|_1 \\ \text{subject to } \sum_{k=1}^K w_k &= 1, \mathbf{z} \in X_\mu, \end{aligned} \quad (66)$$

where $X_\mu = \{\mathbf{z} \in R^N | \mu_1 \mathbf{I}^{(1)} \leq \mathbf{z} \leq \mu_2 \mathbf{h}^{(1)}\}$, μ_1 and μ_2 are positive scaling constants,

$$\mathbf{I}^{(1)} = -\frac{1}{K} \sum_{k=1}^K \mathbf{f}_{\eta_k}, \quad \mathbf{h}^{(1)} = -\frac{1}{K} \sum_{k=1}^K \mathbf{f}_{\eta_k} + L\mathbf{e}.$$

$\mathbf{e} = [1, \dots, 1]^T$, and

$$A = [\mathbf{f}_{\eta_1}, \dots, \mathbf{f}_{\eta_K}], \quad \mathbf{w} = [w_1, \dots, w_K], \quad \mathbf{b} = \frac{1}{K} \sum_{k=1}^K \mathbf{f}_{\eta_k}.$$

CRNN image restoration algorithm

The CRNN image restoration algorithm is defined as follows.

Given a group of regularization parameters $\{\eta_k\}_1^K$.

Step 1. Compute the corresponding regularization solution given by

$$\mathbf{f}_{\eta_k} = (H^T H + \eta_k I)^{-1} H^T \mathbf{g}, \quad (k = 1, \dots, K) \quad (67)$$

Step 2. Compute the optimal fusion solution of (66) by the following cooperative recurrent neural network

Fig. 12 A block diagram of the CRNN image restoration algorithm

$$\frac{d\mathbf{w}}{dt} = -\mu\{d_{\mathbf{w}} - g^1(-A^T\mathbf{v}) + A^T(\mathbf{v} - g^2(\mathbf{v} + A\mathbf{w} - \mathbf{b} - \mathbf{z}))\} \quad (67a)$$

$$\frac{d\mathbf{v}}{dt} = -\mu\{\mathbf{v} - Ag^1(-A^T\mathbf{v}) + Ad_{\mathbf{w}} - g^2(\mathbf{v} + A\mathbf{w} - \mathbf{b} - \mathbf{z}) - (\mathbf{z} - g^3(\mathbf{v} + \mathbf{z}))\} \quad (67b)$$

$$\frac{d\mathbf{z}}{dt} = -\mu\{\mathbf{z} - g^3(\mathbf{v} + \mathbf{z}) - (\mathbf{v} - g^2(\mathbf{v} + A\mathbf{w} - \mathbf{b} - \mathbf{z}))\} \quad (67c)$$

Step 3. Compute an image estimate given by

$$\mathbf{f}(t) = \sum_{k=1}^K w_k(t) \mathbf{f}_{\eta_k},$$

where $\mu > 0$ is a design constant, and $g^1(\mathbf{w})$, $g^2(\mathbf{v})$, $g^3(\mathbf{z})$ are similar to $f^1(\mathbf{w})$, $f^2(\mathbf{v})$, and $f^3(\mathbf{z})$, respectively defined in (60), and $\mathbf{w}(t) = [w_1(t), \dots, w_K(t)]^T$ is the state vector and $\mathbf{f}(t)$ is the output vector. This CRNN (67) consists of three neural network models (67a–67c), which are suitable for parallel implementation. The second and third terms in each model are referred to as the cooperative terms. The second term describes a cooperation relationship between state vectors of three systems. The third term measures a cooperation with other systems. Figure 12 displays a block diagram of the CRNN image restoration algorithm.

The CRNN image restoration algorithm has a computational complexity $O(N^2)$. In practices, however, we can reduce the computational complexity by finding the sub-optimal fusion solution. For example, using the proposed CNFR algorithm, we may find an optimal fusion solution of the following subproblem

$$\begin{aligned} &\text{minimize} \quad \|A^{(N_0)}\mathbf{w} - \mathbf{z} - \mathbf{b}^{(N_0)}\|_1 \\ &\text{subject to} \quad \sum_{k=1}^K w_k = 1, \quad \mathbf{z} \in X_1^{(N_0)}, \end{aligned}$$

where $A^{(N_0)} = [\mathbf{f}_{\lambda_1}^{(N_0)}, \dots, \mathbf{f}_{\lambda_K}^{(N_0)}]$, $\mathbf{b}^{(N_0)} = \frac{1}{K} \sum_{k=1}^K \mathbf{f}_{\lambda_k}^{(N_0)}$, $X_1^{(N_0)} = \{\mathbf{z} \in R^{N_0} | \mathbf{l}_{(N_0)}^{(1)} \leq \mathbf{z} \leq \mathbf{h}_{(N_0)}^{(1)}\}$, and $\mathbf{f}_{\lambda_i}^{(N_0)} \in R^{N_0}$ is a sub-vector of $\mathbf{f}_{\lambda_i} \in R^N$ with $N_0 \ll N$. As a result, the computational complexity of the proposed CNFR algorithm will be $O(N_0^2)$, instead of $O(N^2)$.

Examples

Example 1. Consider a 227 by 339 pixels image acquired by a camera in Fig. 13(a). The original image is blurred by 1×60 uniform motion blur and the contaminated Gaussian noise with distribution of the form

$$f = \frac{1}{2}N(0, \sigma^2) + \frac{1}{2}N(0, 2\sigma^2)$$

was added to the blurred image, where $\sigma^2 = 10$. The noisy blurred image is shown in Fig. 13(b). We first choose a group of parameters in the following sequence

$$\eta_k = (0.15 + 0.0165 * k)/100, \quad (k = 1, \dots, 10).$$

Using them we compute the regularization solutions $\mathbf{f}_{\eta_k}$ by (67). Figure 13(c, d) displays two restored images at corresponding two parameters $\eta_9 = 0.002985$ and $\eta_{10} = 0.00315$, respectively. From Fig. 13(c, d), we can see that η_9 and η_{10} are not optimal. We first take average of these restored images. Figure 13(e) plots the averaged-restored image. It has a small improvement only. We now use the CRNN image restoration algorithm further to improve the quality of the restored images. Note that $0 \leq s(t) \leq 255$. For simplicity, we let the parameters, defined in the set X_{μ} , be $L = 255$, $N = 339$, $\mu_1 = 1$, and $\mu_2 = 1$, respectively. Let the initial point of (67) be zero, processing time be $T = 5$, and $\mu = 50$. Then the fused-restored image is shown in Fig. 13(f). Obviously, the fused-restored image has a much better improvement than the previous restored images including the averaged-restored image.

Example 2. Consider restoration of multichannel imagery. In blind image restoration, in general the single-channel restoration method is fast than the multichannel restoration method, but the latter can provide a better image estimate under a Gaussian assumption.

Let a 256 by 256 pixels Cameraman image be blurred by 9×9 uniform motion blur $\mathbf{h}_r$ and both Laplace noise and contaminated Gaussian noise with the form

$$f = \frac{1}{\sqrt{2\sigma^2}} e^{-\sqrt{2/\sigma^2}|\epsilon|} + \frac{1}{2}N(0, 2) + \frac{1}{2}N(0, 20),$$

are added to the blurred image, where ϵ is a white noise uniformly distributed in the interval $[-0.5, 0.5]$ and $\sigma^2 = 0.25$. The noisy blurred image is shown in Fig. 14(a). Assume that a set of observed images $\{\mathbf{g}^{(i)}\}_{i=1}^K$ are generated by a multichannel image system, where K is the channel number and the corresponding blur has a small change $\Delta\mathbf{h}^{(i)}$ so that

$$\mathbf{h}_r^{(i)} = \mathbf{h}_r + \Delta\mathbf{h}_r^{(i)}, \quad (i = 1, \dots, K).$$

For our simulation, we let $\Delta\mathbf{h}^{(i)}$ be an 9×9 normally distributed random matrix multiplied by 0.05. Also, by the the single-channel restoration method we get an approximate blur $\mathbf{h}_r^0$, which is assumed to be an average of previous K blurs. We now use the CRNN (67) to fuse observed images $\{\mathbf{g}^{(i)}\}_{i=1}^K$, where $K = 30$ and the initial point be zero, processing time $T = 5$, and $\mu = 50$, and the set X given in example 1. When the optimal observed image $\mathbf{g}^*$ is obtained, the regularization method is then performed by using the regularization parameter being 0.001 and the blur $\mathbf{h}_r^0$. The restored-fused image is shown in Fig. 14(d). By comparison, the data fusion algorithm

Fig. 13 Simulated results in example 1: (a) original image; (b) noise blurred image; (c) image restored by η_9 ; (d) image restored by η_{10} ; (e) fused image by average; (f) fused image by the CRNN image restoration algorithm

based on the CRNN (56) obtained another fused–restored image shown in Fig. 14(c) under the same simulation parameters. Figure 14(b) displays a restored image by the regularization method, where the regularization parameter be 0.001. Obviously, the restored-fused image quality obtained by the CRNN (67) has a better improvement.

CRNNs for real-time beamforming

Problems and formulation

Adaptive antenna techniques offer the possibility of increasing the performance of wireless communication systems by maximizing directional gain and enhancing the protection towards multipath fading conditions. The beamforming processor is used to generate an optimal set of beams to track the mobiles within the coverage area of the base-station. The implemented algorithms have to rapidly

enhance the desired signal and suppressing noise and interference at the output of an array of sensors (Reed et al. 1974). So, real-time solution algorithms are very desirable.

Consider a linear array of n sensors. Let $y_j(t)$ denote the received signal on the j th sensor at instant $t = kT_c$, where T_c is a sampling interval. Then the n -dimensional vector of signals received from n sensors is given by

$$\mathbf{y}(t) = \sum_{j=1}^M \mathbf{a}(\theta_j) s_j(t) + \mathbf{v}(t), \quad (68)$$

where $\mathbf{y}(t) = [y_1(t), \dots, y_n(t)]^T \in C^n$ is a complex-valued vector, $s_j(t)$ is the signal sample transmitted by the j th user, $\mathbf{a}(\theta_j)$ is the n -dimensional array response vector of the j th user at the direction θ_j , and $\mathbf{v}(t)$ is the n -dimensional vector of white noise samples. The received signal vector $\mathbf{y}(t)$ consists of the desired signal, interferences and measurement noise. Let $\mathbf{w} = [w_1, \dots, w_n]^T$ be the complex-valued weight vector of the array. Then the

Fig. 14 Multichannel image fusion at the pixel level using the CRNN algorithm in example 2: (a) the noisy-degraded image; (b) the restored image by the regularization method; (c) the fused image by the data fusion algorithm based on (56); (d) the restored-fused image by the CRNN image restoration algorithm

received signals are weighted to produce the beamformer output given by

$$u(t) = \mathbf{w}^H \mathbf{y}(t),$$

where H denotes the Hermitian transpose. The weight vector of the LCMP beamformer is chosen to minimize the output power of the beamformer subject to linear equality constraints (Van Trees 1998). That is, the weight vector is an optimal solution of the LCMP optimization problem

$$\begin{aligned} &\text{minimize} \quad \mathbf{w}^H \mathbf{r}_y \mathbf{w} \\ &\text{subject to} \quad \mathbf{c}^H \mathbf{w} = \mathbf{f}, \end{aligned} \quad (69)$$

where $\mathbf{R}_y = E[\mathbf{y}(t)\mathbf{y}(t)^H]$ is the data covariance matrix, $\mathbf{C}$ is the $n \times m$ constraint matrix, and $\mathbf{f}$ is an m -dimensional response vector (Frost 1972). When $\mathbf{f} = 1$, $\mathbf{C}$ is the steering vector associated with the look direction. Because a quadratic inequality constraint on the weight vector of the LCMP beamformer can improve robustness to pointing errors and to random perturbations in sensor parameters, A variable loading recursive least square (RLS-VL)

algorithm (Tian et al. 2001) was developed to find the weight $\mathbf{w}^*$ of the robust LCMP beamformer satisfying the following optimization problem

$$\begin{aligned} &\text{minimize} \quad \mathbf{w}^H \mathbf{r}_y \mathbf{w} \\ &\text{subject to} \quad \mathbf{c}^H \mathbf{w} = \mathbf{f} \\ &\quad \quad \quad \mathbf{w}^H \mathbf{w} \leq \beta, \end{aligned} \quad (70)$$

where $\beta > 0$ is called tolerance factor. The robust LCMP beamformer output is then given by $u^*(t) = (\mathbf{w}^*)^H \mathbf{y}(t)$. However, in practice the RLS-VL algorithm obtains an approximate weight of the robust LCMP beamformer due to estimating a loading term (Lagrange multiplier).

Neural network approaches

Neural network approaches have been successfully applied to real-time beamforming (Du et al. 2002). Since the robust LCMP problem (70) is a complex valued one, it is first necessary to convert it into a real valued optimization

problem. We substitute $\mathbf{w} = \mathbf{w}_I + i\mathbf{w}_{II}$, $\mathbf{R}_y = \mathbf{R}_I + i\mathbf{R}_{II}$, $\mathbf{f} = \mathbf{f}_I + i\mathbf{f}_{II}$, and $\mathbf{C} = \mathbf{C}_I + i\mathbf{C}_{II}$ into (70). Then (70) can be equivalent to the following optimization problem in terms of real valued vectors

$$\begin{aligned} & \text{minimize} \quad \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} \\ & \text{subject to} \quad \mathbf{A} \mathbf{x} = \mathbf{b}, \quad \mathbf{x}^T \mathbf{x} \leq \beta, \end{aligned} \quad (71)$$

where T denotes transpose, $\mathbf{x} = [\mathbf{w}_I^T, \mathbf{w}_{II}^T]^T \in \mathbb{R}^{2n}$, and

$$\mathbf{Q} = \begin{pmatrix} \mathbf{R}_I & \mathbf{R}_{II} \\ -\mathbf{R}_{II} & \mathbf{R}_I \end{pmatrix} \in \mathbb{R}^{2n \times 2n},$$

$$\mathbf{A} = \begin{pmatrix} \mathbf{C}_I^T & \mathbf{C}_{II}^T \\ -\mathbf{C}_{II}^T & \mathbf{C}_I^T \end{pmatrix} \in \mathbb{R}^{2m \times 2n}, \quad \mathbf{b} = \begin{pmatrix} \mathbf{f}_I \\ \mathbf{f}_{II} \end{pmatrix} \in \mathbb{R}^{2m}.$$

There exist mainly three types of neural networks for the LCMP beamforming: the Hopfield neural network (Chang et al. 1992), the RBF neural network (Zooghy et al. 1998), and the minor component (MC) neural network (Fiori 2003). While the Hopfield neural network may be regarded as a special case of the Kennedy–Chua neural network (Sharkey 1998). When applied to the robust LCMP beamforming problem (1), the Kennedy–Chua neural network can be represented as

$$\frac{d\mathbf{x}(t)}{dt} = -\mathbf{Q}\mathbf{x}(t) - \lambda_s \{ \mathbf{A}^T (\mathbf{A}\mathbf{x}(t) - \mathbf{b}) + 2\mathbf{x}(t)(\mathbf{x}(t)^T \mathbf{x}(t) - \beta)^+ \}, \quad (72)$$

where $\lambda_s > 0$ is a penalty parameter. The Kennedy–Chua network has a smaller network size, but the Kennedy–Chua network converges to an approximate optimal solution of (3) due to the penalty parameter. The MC neural network can implement the beamformer with a quadratic constraint, but it has a slower convergence rate. Moreover, when implementing linear constraints, the computational complexity of the MC neural network will be added greatly.

Fig. 15 The block diagram of the CRNN algorithm (73) for beamformer

Recently, a cooperative recurrent neural network algorithm for real-time robust beamforming is developed in Xia and Feng (2006). It is defined as follows.

State equation

$$\begin{pmatrix} \frac{d\mathbf{x}(t)}{dt} \\ \frac{d\eta(t)}{dt} \\ \frac{d\mathbf{q}(t)}{dt} \end{pmatrix} = \alpha \begin{pmatrix} -\mathbf{Q}\mathbf{x}(t) - 2\eta(t)\mathbf{x}(t) + \mathbf{A}^T \mathbf{q}(t) \\ (\eta(t) + \mathbf{x}(t)^T \mathbf{x}(t) - \beta)^+ - \eta(t) \\ -\mathbf{A}\mathbf{x}(t) + \mathbf{b} \end{pmatrix} \quad (73)$$

Output equation

$$u(t) = [\mathbf{x}_I(t) + i\mathbf{x}_{II}(t)]^H \mathbf{y}(t),$$

where $\alpha > 0$ is a design parameter, $\mathbf{x}_I(t), \mathbf{x}_{II}(t) \in \mathbb{R}^n$, $\mathbf{x}(t) = [\mathbf{x}_I(t), \mathbf{x}_{II}(t)]^T \in \mathbb{R}^{2n}$, $\eta(t) \in \mathbb{R}$, $\mathbf{q}(t) \in \mathbb{R}^{2m}$, $\mathbf{z}(t) = [\mathbf{x}(t), \eta(t), \mathbf{q}(t)]^T \in \mathbb{R}^{2(n+m)+1}$ is a state vector, and $u(t)$ is the output of the beamformer. We compare the CRNN (73) with the RLS-VL algorithm. It is easy to see that the CRNN algorithm requires $4n^2 + 8mn + 5n + m + 1$ multiplications per iteration. For the RLS-VL algorithm, the RLS updating step requires $8n^2 + 8n + 1$ multiplications per iteration and the variable loading step requires $4n^2 + 8n + 5$ multiplications with one square operation per iteration. In addition, the weight $\mathbf{w}$ need to be decomposed into two components as follows

$$\mathbf{w} = \mathbf{w}_q - \mathbf{B}\mathbf{w}_a,$$

where $\mathbf{B}$ is an $n \times (n-m)$ matrix which is obtained by solving the matrix equation $\mathbf{C}^H \mathbf{B} = 0$. Therefore, the CRNN algorithm (73) has a lower computational complexity than the RLS-VL algorithm. Moreover, it was shown in Xia and Feng (2006) that the CRNN algorithm (73) can converge exponentially to the optimal beamforming weight. Figure 15 shows the block diagram of the CRNN algorithm (73) for the robust LCMP beamforming.

Example

Consider a ten-sensor uniform linear array to receive three uncorrelated, narrow-band signals corrupted by an additive white noise in a stationary environment. A desired signal with signal to noise ratio (SNR) 10 dB impinges on the array from $\phi_0 = -0.03\pi$. Two interferences with interference to noise ratio (INR) 20 dB are located at $\phi_1 = 0.29\pi$ and $\phi_2 = 0.43\pi$, respectively. Our simulation results are averaged over 200 independent trials. To compute the covariance matrix of $\mathbf{y}(t)$ we use the following average estimate

$$R_y \approx \frac{1}{N} \sum_{t=1}^N \mathbf{y}(t)\mathbf{y}(t)^H,$$

where $N = 1500$ is a sample number. In addition, $\mathbf{C}$ is taken as the steering vector associated with the look direction.

We compared the CRNN algorithm with the Kennedy–Chua neural network (KCNN) algorithm, the minor component neural network (MCNN) algorithm, and the RLS-VL algorithm. First, let the penalty parameter λ_s be 20 in the KCNN algorithm, and let the initial diagonal loading and the forgetting factor be 0.1 and 0.9999, respectively in the RLS-VL algorithm. Figure 16(a) displays the performance comparisons of array beam patterns using the CRNN, the MCNN, and the KCNN algorithm. Figure 16(b) displays the performance comparisons of array beam patterns using the CRNN and the RLS-VL algorithm, where the initial estimation of the weigh vector is zero. It can be seen that the CRNN approach is able to maintain the mainlobe and null the interferers correctly, in contrast to the other three methods.

Next, in order to illustrate the convergence rate, we compared the weight norm of the beamformer among four algorithms. The simulation results listed in Table 2 show that the CRNN algorithm has a faster convergence and a smaller weight norm than the other four algorithms.

CRNN for parameter estimation of AR signals

The autoregressive (AR) modeling approach has been widely used in speech processing, spectral estimation, noise cancellation, and forecasting (Kay 1993). The main task of this approach is to find optimal AR coefficients from a given random signal in the presence of background noise.

AR signal system and parameter estimation

Consider the parameter estimation problem of the noisy AR signal system

Fig. 16 Comparisons of beampatterns of four different beamformers in Example. (a) the solid line corresponds to the CRNN, network and the dot line to the KCNN, and the dash-dot line to the MCNN; (b) the solid line corresponds to CRNN and the dot line to the RLS-VL

$$\begin{cases} x(t) = \sum_{i=1}^p a_i x(t-i) + w(t) \\ y(t) = x(t) + u(t), \end{cases} \quad (74)$$

where $\mathbf{a} = [a_1, \dots, a_p]^T$ is the unknown AR parameter vector; $\mathbf{x}_t = [x(t-1), \dots, x(t-p)]^T$ is the regression vector of the AR process $x(t)$ of the known order p ; $w(t)$ is driving noise; and $u(t)$ is measurement noise which is assumed to be uncorrelated with $w(t)$. Let the noisy signal vector be denoted by $\mathbf{y}_t = [y(t-1), \dots, y(t-p)]^T$, and let

Table 2 Results of computational time for four algorithms, where $\beta = 0.25$ and $N = 1500$

Algorithm	RLS-VL	MC NN	K-C NN	CRNN
weight norm	0.5877	0.5218	0.50215	0.4560
CPU time (sec.)	0.77	1.025	0.8	0.6

$\mathbf{u}_t = [u(t-1), \dots, u(t-p)]^T$ be the measurement noise vector. The AR signal model can be rewritten as

$$y(t) = \mathbf{y}_t^T \mathbf{a} - n(t), \quad (75)$$

where $n(t) = \mathbf{u}_t^T \mathbf{a} - u(t) - w(t)$. Our objective is to get an optimal estimate of the AR parameters by using noisy observations $\{y(t)\}_1^N$ under a wider range of noise distributions, where N is the number of data points. A basic approach to estimate the AR coefficient vector $\mathbf{a}$ is the least square (LS) method. The LS estimate is given by

$$\mathbf{a}_{LS} = \left(\frac{1}{N} \sum_{t=1}^N \mathbf{y}_t \mathbf{y}_t^T \right)^{-1} \left(\frac{1}{N} \sum_{t=1}^N \mathbf{y}_t y(t) \right).$$

However, there is an error between the true parameter $\mathbf{a}^*$ and the LS solution $\mathbf{a}_{LS}$ with

$$\lim_{N \rightarrow \infty} \mathbf{a}_{LS} = \mathbf{a}^* - \sigma^2 R^{-1} \mathbf{a}^*,$$

where $R = E[\mathbf{y}_t \mathbf{y}_t^T]$ and σ^2 is the variance of the measurement noise. Many significant methods for improving the LS estimation based on the LS solution were presented. For example, Young, Soderstrom and Stoica developed the instrumental variable method to improve the LS solution (Young 1970; Soderstrom and Stoica 1981; Nehorai and Stoica 1988). Zheng further presented several improving techniques (Zheng 1999, 2000). We also presented using the LCLS data fusion technique to enhance the quality of the LS solution by doing K statistically independent trials of finding the LS solution (Xia 1996a). The above mentioned methods are based on second-order statistics. In order to deal with non-Gaussian environments, high-order statistic methods have been developed in Giannakis and Mendel (1990), where measurement noise is Gaussian distributed.

CRNNs for parameter estimation of AR signals

There are two cooperative neural network approaches for the optimal AR parameter estimation. One is the CRNN-LAD algorithm defined in (41), which is based on a least absolute deviation (LAD) method (Alliney and Ruzinsky 1994). Another is the CRNN-GLAD algorithm by using a generalized least absolute deviation (GLAD) method (Xia and Kamel 2008). Different from the LAD method, the GLAD method is to find the optimal solution, $(\mathbf{a}^*, \mathbf{z}^*)$, of the following GLAD problem

$$\begin{aligned} \min \quad & \varphi(\mathbf{a}, \mathbf{z}) = \|\mathbf{Y}\mathbf{a} - \mathbf{y} - \mathbf{z}\|_1 \\ \text{s.t.} \quad & \mathbf{a} \in R^N, \mathbf{z} \in \Omega_\gamma. \end{aligned} \quad (76)$$

where $\Omega_\gamma = \{\mathbf{z} \in R^N | \gamma_1 \mathbf{1} \leq \mathbf{z} \leq \gamma_2 \mathbf{h}\}$, $\gamma_1 > 0$ and $\gamma_2 > 0$ are scaling constants, $\mathbf{1} = [n^-(1), \dots, n^-(N)]^T$ and $\mathbf{h} = [n^+(1), \dots, n^+(N)]^T$, $n^-(t)$ and $n^+(t)$ are respectively

denoted as the upper and lower limits of $n(t)$, $\mathbf{y}(t) = [y(1), \dots, y(N)]$, $\mathbf{n}(t) = [n(1), \dots, n(N)]$, and

$$Y = \begin{pmatrix} y(0) & y(-1) & \dots & y(1-p) \\ y(1) & y(0) & \dots & y(2-p) \\ \vdots & \vdots & \ddots & \vdots \\ y(N-1) & y(N-2) & \dots & y(N-p) \end{pmatrix}.$$

The CRNN-GLAD approach first converts the GLAD problem into a system of piecewise equations

$$\begin{cases} Y^T \mathbf{e} = 0, \\ \mathbf{e} = g^1(\mathbf{e} + Y\mathbf{a} - \mathbf{z} - \mathbf{y}), \\ \mathbf{z} = g^2(\mathbf{z} + \mathbf{e}), \end{cases} \quad (77)$$

where

$$g^1(z_i) = \begin{cases} -1 & z_i < -1 \\ z_i & -1 \leq z_i \leq 1 \\ 1 & z_i > 1 \end{cases}, \quad g^2(z_i) = \begin{cases} l_i & z_i < l_i \\ z_i & l_i \leq z_i \leq h_i \\ h_i & z_i > h_i. \end{cases}$$

Based on (77), the CRNN-GLAD model is defined as follows

$$\frac{d\mathbf{a}}{dt} = -\lambda \{2Y^T \mathbf{e} - Y^T g^1(\mathbf{e} + Y\mathbf{a} - \mathbf{z} - \mathbf{y})\} \quad (78a)$$

$$\begin{aligned} \frac{d\mathbf{e}}{dt} = & -\lambda \{\mathbf{e} + YY^T \mathbf{e} - g^1(\mathbf{e} + Y\mathbf{a} - \mathbf{z} - \mathbf{y}) \\ & - (\mathbf{z} - g^2(\mathbf{z} + \mathbf{e}))\} \end{aligned} \quad (78b)$$

$$\frac{d\mathbf{z}}{dt} = -\lambda \{\mathbf{z} - g^2(\mathbf{z} + \mathbf{e}) - \mathbf{e} + g^1(\mathbf{e} + Y\mathbf{a} - \mathbf{z} - \mathbf{y})\} \quad (78c)$$

where $\lambda > 0$ is a design constant, and $g^1(\mathbf{z})$, and $g^2(\mathbf{z})$ are defined in (77), respectively. Figure 17 displays a block diagram of the CRNN-GLAD algorithm which requires $2N^2 + N(3p + 11) + p^2$ multiplications per iteration, where CRNN combines three recurrent neural network submodels: NN1, NN2, and NN3 and each submodel

Fig. 17 A block diagram of the CRNN-GLAD algorithm

includes cooperative terms and cooperation error term with other two submodels. The CRNN-GLAD algorithm and the CRNN-LAD algorithm can converge globally to an optimal solution but the CRNN-GLAD algorithm improves the CRNN-LAD algorithm in estimation accuracy.

Examples

Example 1. Consider the AR signal system given in Zheng (1999)

$$\begin{cases} x(t) = 1.352x(t-1) - 1.338x(t-2) \\ \quad + 0.662x(t-3) - 0.240x(t-4) + 10e(t) \\ y(t) = x(t) + u(t), \end{cases} \quad (79)$$

where $u(t)$ is a white noise uniformly distributed in $[-1, 1]$ multiplied by variance 0.38, and $e(t)$ is a chaotic processes generated by a logistic mapping equation

$$e(t) = 4e(t-1)(1 - e(t-1)),$$

with the initial value $e(0) = 0.2$. The above AR signal model can be rewritten as

$$\begin{aligned} y(t) = & 1.352y(t-1) - 1.338y(t-2) \\ & + 0.662y(t-3) - 0.240y(t-4) - n(t) \end{aligned} \quad (80)$$

where

$$\begin{aligned} n(t) = & 1.352u(t-1) - 1.338u(t-2) + 0.662u(t-3) \\ & - 0.240u(t-4) - 10e(t) - u(t) \end{aligned}$$

is non-Gaussian colored noise. We collect 1000 data samples by the input $e(t)$. We first computed the LS solution

$$\mathbf{a}_{LS} = [1.595, -1.418, 0.8253, -0.0844]^T,$$

with a 2-norm error of $\|\mathbf{a}_{LS} - \mathbf{a}^*\|_2 = 0.3415$, where $\mathbf{a}^* = [1.352, -1.338, 0.662, -0.240]$. To perform the CRNN-GLAD algorithm for the optimal AR parameter estimation, we let $N = 400$, $\gamma_1 = \gamma_2 = 1$, $\mathbf{l} = [0, \dots, 0]^T$, and $\mathbf{h} = -[\hat{y}, \dots, \hat{y}]^T$, where $\hat{y}$ is the arithmetic mean of $y(t)$. The state trajectory of the CRNN-GLAD algorithm is always convergent to an optimal solution. For example, let the initial point be zero and $\lambda = 450$. Figure 18 displays the convergence behavior of the state trajectory of the CRNN-GLAD algorithm. We compare the CRNN-GLAD algorithm with the CRNN-LAD algorithm, the ILS algorithm (Zheng 1999), and the SVM algorithm (Rojo-Alvarez et al. 2004). The computed results were obtained by averaging 100 independent Monte Carlo simulations. The results are summarized in Table 1. Similar to the paper (Zheng 1999), in the table, $\hat{\mathbf{a}}$ denotes the arithmetic mean of the estimated $\mathbf{a}^*$ based on 100 independent test, l_2 norm error is defined by

$$NE = \|\hat{\mathbf{a}} - \mathbf{a}^*\|_2,$$

the relative error is defined by

$$RE = \|\hat{\mathbf{a}} - \mathbf{a}^*\|_2 / \|\mathbf{a}^*\|_2,$$

and the normalized root mean-square error is defined by

$$RMSE = \sqrt{\frac{1}{100} \sum_{k=1}^{100} \frac{\|\mathbf{a}^* - \mathbf{a}_i\|_2^2}{\|\mathbf{a}^*\|_2^2}},$$

where $\mathbf{a}_i$ stands for the estimate of $\mathbf{a}^*$ in the i -th trial. From Table 3 we can see that the CRNN-GLAD algorithm is more computationally efficient and can obtain an improved

Fig. 18 Convergence behavior of the state trajectory of the CRNN-GLAD algorithm in Example 1

Table 3 Computed results of estimated parameters of four algorithms for Example 1

solution	True	SVM algorithm	ILS algorithm
$a(1)$	1.352	1.3713 ± 0.0674	1.3407 ± 0.0108
$a(2)$	-1.338	-1.4013 ± 0.0932	-1.2612 ± 0.034
$a(3)$	0.662	0.7750 ± 0.0419	0.6033 ± 0.026
$a(4)$	-0.24	-0.3109 ± 0.0629	-0.1975 ± 0.0143
NE		0.1344	0.1062
RE		6.3%	5.4%
RMSE		7.3%	5.9%
Solution	True	CRNN-LAD algorithm	CRNN-GLAD algorithm
$a(1)$	1.352	1.5783 ± 0.0104	1.3667 ± 0.0079
$a(2)$	-1.338	-1.3756 ± 0.0162	-1.3256 ± 0.0031
$a(3)$	0.662	0.7923 ± 0.0129	0.6729 ± 0.0103
$a(4)$	-0.24	-0.0773 ± 0.0119	-0.2152 ± 0.0088
NE		0.3102	0.0332
RE		15.2%	1.6%
RMSE		17.3%	2.1%

solution with the smaller norm error than other three algorithms.

Example 2. Consider the AR signal system given in Al-Smadi and Alshamali (2002)

$$\begin{cases} x(t) = -0.7907x(t-1) - 0.042x(t-2) \\ \quad + 0.556x(t-3) - 0.0247x(t-4) \\ \quad - 0.385x(t-5) - 0.303x(t-6) + e(t) \\ y(t) = x(t) + u(t), \end{cases} \quad (81)$$

where $\{u(t)\}$ is additive Gaussian noise at SNR of 10 dB, the drive noise $e(t)$ is given by

$$e(t) = w(t) + 0.345w(t-1) + 0.53w(t-2) \\ + 0.399w(t-3) + 0.814w(t-4),$$

and $\{w(t)\}$ is exponential random process with variance of 0.0625. We collect 1000 data samples by the input $e(t)$. It can be seen that the least square (LS) algorithm gives the following estimate

$$\mathbf{a}_{LS} = [-0.7093, 0.1254, 0.7520, 0.2356, -0.1950, -0.2054]$$

with the norm error 0.334. To perform the CRNN-GLAD algorithm for the optimal AR parameter estimation, we let $N = 600$, $\gamma_1 = \gamma_2 = 1$, $\mathbf{l} = [0, \dots, 0]^T$, and $\mathbf{h} = -[\hat{y}, \dots, \hat{y}]^T$, where $\hat{y}$ is the arithmetic mean of $y(t)$. The state trajectory of the CRNN-GLAD algorithm is always convergent to an optimal solution. For example, let the initial point be zero and $\mu = 450$. Figure 19 displays the convergence behavior of the state trajectory of the CRNN-GLAD algorithm. We compare the CRNN-GLAD algorithm with the CRNN-LAD algorithm, Giannakis and Mendel (GM) algorithm (Giannakis and Mendel 1990), and Al-Smadi and Alshamali (AA) algorithm (2002). The computed results were obtained by averaging 100 independent Monte

Fig. 19 Convergence behavior of the state trajectory of the CRNN-GLAD algorithm in Example 2

Carlo simulations. The results are summarized in Table 4. From Table 2 we can see that the CRNN-GLAD algorithm is more computationally efficient and can obtain an improved solution with the smaller norm error than both the three algorithms.

CRNNs for nonlinear system identification

Identification of nonlinear systems has received considerable attentions and has been widely applied in engineering applications, including signal and image processing, communications, and control (Ljung 1999; Tekalp et al. 1985).

Nonlinear system identification

Consider the following discrete-time noisy system

$$\begin{cases} x(t) = g_0(\varphi_x(t)) + w(t) \\ y(t) = x(t) + n(t), \end{cases} \quad (82)$$

where $g_0(\cdot)$ is the unknown function, $\varphi_x(t)$ is the regression vector defined by

$$\varphi_x(t) = [x(t-1), \dots, x(t-p), u(t-1), \dots, u(t-q)]^T,$$

$u(t)$ and $x(t)$ are the signal input and output of the system, p and q are the orders of the system, $y(t)$ is the observed data,

Table 4 Computed results of estimated parameters of four algorithms for Example 2

Solution	True	GM algorithm	AA algorithm
$a(1)$	-0.7907	-0.6516 ± 0.0259	-0.7742 ± 0.0167
$a(2)$	-0.0420	0.0881 ± 0.0194	-0.0364 ± 0.0142
$a(3)$	0.5560	0.5283 ± 0.0187	0.5572 ± 0.0097
$a(4)$	0.0247	0.0211 ± 0.0126	-0.0230 ± 0.0063
$a(5)$	-0.3850	-0.3942 ± 0.0122	-0.3980 ± 0.0089
$a(6)$	-0.3030	-0.3060 ± 0.0186	-0.3105 ± 0.0083
NE		0.1627	0.0530
RE		16.7%	4.8%
RMSE		17.8%	5.4%
Solution	True	CRNN-LAD algorithm	CRNN-GLAD algorithm
$a(1)$	-0.7907	-0.7271 ± 0.0059	-0.7817 ± 0.0105
$a(2)$	-0.0420	-0.0038 ± 0.0114	-0.0353 ± 0.0039
$a(3)$	0.5560	0.5465 ± 0.0077	0.5637 ± 0.0085
$a(4)$	0.0247	0.0416 ± 0.0046	0.0272 ± 0.0033
$a(5)$	-0.3850	-0.3387 ± 0.0102	-0.3638 ± 0.0092
$a(6)$	-0.3030	-0.2599 ± 0.0096	-0.2903 ± 0.0063
NE		0.0993	0.0283
RE		9.2%	2.61%
RMSE		11.8%	4.3%

$w(t)$ is driving white Gaussian noise with zero mean, and $n(t)$ is measurement white Gaussian noise with zero mean.

The problem of system identification is as follows: Based on received data from the noisy system (82), find a smooth function estimate g of g_0 such that the mean square error (MSE) criterion

$$E[(g(\varphi_y(t)) - g_0(\varphi_x(t)))^2] \quad (83)$$

is minimized, where $\varphi_y(t) = [y(t-1), \dots, y(t-p), u(t-1), \dots, u(t-q)]^T$. Many of the identification methods make efforts in minimizing the approximation error. For example, theoretically, the local polynomial approaches can have the approximation error reduced to zero as the number of measurements (Koukoulas and Kalouptsidis 2000). Neural networks with universal approximation capability can provide parallel approximation techniques (Hornik et al. 1989; Chen and Billings 1992; Kadiramanathan and Niranjana 1993). The support vector machine (SVM) method can minimize one regularized risk function (Muller et al. 1997)

$$E_1(\alpha) = \frac{1}{N} \sum_{t=1}^N L(g(\varphi_y(t)) - y(t)) + \gamma \|\alpha\|^2, \quad (84)$$

where

$$g(\varphi_y(t)) = \alpha_0 + \sum_{k=1}^N \alpha_k K(\varphi_y(k), \varphi_y(t)),$$

N is a number of measurements, $\gamma > 0$ a regularization parameter, $L(g(\varphi_y(t)) - y(t))$ is an ϵ -insensitive cost function, $K(\mathbf{u}, \mathbf{v})$ is a given kernel function, and $\varphi_y(t)$ is a regression vector including noise. Weight coefficients $\{\alpha_k\}$ and α_0 are obtained by solving a quadratic convex programming. The least square support vector machine (LS-SVM) method can minimize another regularized quadratic function (Suykens et al. 2002)

$$E_2(\alpha) = \frac{1}{N} \sum_{t=1}^N (g(\varphi_y(t)) - y(t))^2 + \gamma \|\alpha\|^2. \quad (85)$$

where the weight coefficients are obtained by solving linear equations. The direct weight optimization method can minimize an upper bound of the MSE (Roll et al. 2005).

Cooperative recurrent neural network

Recently, a cooperative neural network algorithm, called a cooperative learning algorithm, for nonlinear system modeling was developed (Xia and Kamel 2007c). The CRNN algorithm is based on a measurement fusion method which minimizes both the noise error $E[(g(\varphi_y(t)) - y(t))^2]$ and the approximation error $E[(g(\varphi_y(t)) - g(\varphi_x(t)))^2]$. The measurement fusion method is described as follows.

Step 1. Construct optimal fused data

$$z^*(t) = \sum_{i=1}^{\tau} w_k^* y_i(t) \quad (86)$$

by finding the optimal solution, $\{w_k^*\}$, of the following

$$\begin{aligned} \text{minimize} \quad & \frac{1}{N} \sum_{t=1}^N \left(\sum_{k=-\tau}^{\tau} w_i y(t+k) \right)^2 \\ \text{subject to} \quad & \sum_{i=1}^{\tau} w_i = 1. \end{aligned} \quad (87)$$

where $\tau > 0$ is a sensor number and $\{w_k\}$ are weight coefficients.

Step 2. Compute optimal weight coefficients, (α^*, b_0^*) , of the robust LS-SVM function estimate by solving the following piecewise equations

$$\begin{cases} f(\alpha - (Q + \frac{1}{\gamma} I)\alpha - b_0 \mathbf{e}_r + \mathbf{y}) - \alpha = 0 \\ \sum_{k=1}^r \alpha_k = 0, \end{cases} \quad (88)$$

where r is the number of training data points, $\mathbf{e}_r = [1, \dots, 1]^T \in R^r$, $\mathbf{y} = [y_1, \dots, y_r]^T$, $I \in R^{r \times r}$, and $Q = (K(\mathbf{x}_k, \mathbf{x}_k))$, $f(\mathbf{u}) = [f(u_1), \dots, f(u_N)]^T$, for $i = 1, \dots, r$

$$f(u_i) = \begin{cases} -\gamma\epsilon & u_i < -\gamma\epsilon \\ u_i & -\gamma\epsilon \leq u_i \leq \gamma\epsilon \\ \gamma\epsilon & u_i > \gamma\epsilon. \end{cases}$$

and $\epsilon > 0$ is a bounded limit the outlier errors on the estimates.

Step 3. Compute the optimal measurement fusion function estimate by incorporating the optimal fused data and optimal weight coefficients into the LS-SVM function estimate.

$$g(\varphi_y(t)) = b_0 + \sum_{k=1}^r \alpha_k K(\varphi_y(t), \varphi_{z^*}(t))$$

The CRNN algorithm is described as follows.

Given a kernel function $K(\cdot, \cdot)$, the initial weight vectors $\mathbf{w}^{(0)}$ and $(\alpha^{(0)}, b_0^{(0)})$, and simulation parameters τ , γ , ϵ , and r .

Step I. Learning algorithm (A)

(i) Compute

$$\mathbf{w}^{(k)} = (1 - \beta) P_{\tau} \mathbf{e} \mathbf{e}^T B \mathbf{w}^{(k-1)} + \mathbf{Q}_{\tau}, \quad (89)$$

where $0 < \beta < 1$, $\mathbf{q}_{\tau} = P_{\tau} \mathbf{e}$, and

$$B = \frac{1}{N} \sum_{t=1}^N \mathbf{y}_t \mathbf{y}_t^T, \quad P_{\tau} = (\tau B + \mathbf{e} \mathbf{e}^T (I - \beta B))^{-1},$$

$$\mathbf{y}_t = [y_1(t), \dots, y_{\tau}(t)]^T.$$

(ii) If $\|\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}\|^2 \leq \delta_1$, then goto (iii). Otherwise, let $k = k + 1$ and goto (i).

(iii) Compute optimal fused data

$$z^{(k)}(t) = \sum_{i=1}^{\tau} w_i^{(k)} y_i(t), \quad (t = 1, \dots, N)$$

Step II. Learning algorithm (B)

- (i) Compute $\varphi_y(t) = [y(t-1), y(t-2), \dots, y(t-p), u(t-1), \dots, u(t-q)]^T$ and compute $\hat{Q} = (Q + I/\gamma)/s$, $\hat{\mathbf{y}} = \mathbf{y}/s$, and $\mathbf{e}_1 = \mathbf{e}_r/s$, where Q , $\mathbf{y}$, and $\mathbf{e}_r$ are defined in (88) and

$$s = \left\| \begin{pmatrix} Q + I/\gamma & \mathbf{e}_r \\ -\mathbf{e}_r^T & 0 \end{pmatrix} \right\|_2.$$

- (ii) Compute $\mathbf{u}^{(k)} = ((I - \hat{Q})\alpha^{(k)} - \hat{\mathbf{y}} + b^{(k)})\mathbf{e}_1$, and

$$f(\mathbf{u}^{(k)}) = [f(u_1^{(k)}), \dots, f(u_r^{(k)})]^T,$$

where $f(u_i)$ is defined in (88).

- (iii) Compute

$$\begin{pmatrix} \alpha^{(k+1)} \\ b^{(k+1)} \end{pmatrix} = \begin{pmatrix} \alpha^{(k)} \\ b^{(k)} \end{pmatrix} - h \begin{pmatrix} (\hat{Q} + I_1)(f(\mathbf{u}^{(k)}) - \alpha^{(k)}) - \mathbf{e}_1 \mathbf{e}_1^T \alpha^{(k)} \\ \mathbf{e}_1^T f(\mathbf{u}^{(k)}) - 2\mathbf{e}_1^T \alpha^{(k)} \end{pmatrix}, \quad (90)$$

where $h > 0$ is the learning step length.

- (vi) Compute

$$r(\alpha^{(k)}, b_0^{(k)}) = \|f(\mathbf{u}^{(k)}) - \alpha^{(k)}\|^2 + (\mathbf{e}_1^T \alpha^{(k)})^2.$$

If $r(\alpha^{(k)}, b_0^{(k)}) \leq \delta_2$, then output $(\alpha^{(k)}, b_0^{(k)})$. Otherwise, let $k = k + 1$ and goto Step 1.

Step III. Compute the optimal measurement fusion function estimate given by

$$g(\varphi_{z^*}(t)) = \sum_{i=1}^r \alpha_i^* K(\varphi_y(i), \varphi_{z^*}(t)) + b_0^*, \quad (t = 1, \dots, N) \quad (91)$$

where

$$\varphi_{z^*}(t) = [z^*(t-1), z^*(t-2), \dots, z^*(t-p), u(t-1), \dots, u(t-q)]^T.$$

It easy to see that the CRNN algorithm can perform the two learning algorithms in parallel and combine automatically their output results with a feedforward network, which implements (91) by performing $3\tau^2 + 2N^2 + 6N$ multiplication per iteration. Figure 20 displays a block diagram of the CRNN algorithm. It was shown that unlike conventional individual learning algorithms for system identification, the CRNN algorithm has an improved performance in the MSE criterion.

Fig. 20 A block diagram of the CRNN algorithm for nonlinear system identification

Examples

Example 1. Consider a nonlinear benchmark system given by Narendra and Li. The system is defined in state-space form by

$$\begin{cases} x_1(t+1) = \sin(x_2(t)) \left(\frac{x_1(t)}{1+x_1(t)^2} + 1 \right), \\ x_2(t+1) = x_2(t) \cos(x_2(t)) + x_1(t) e^{-(x_1(t)^2+x_2(t)^2)/8} \\ \quad + \frac{u(t)^3}{(1+u(t)^2+0.5\cos(x_1(t)+x_2(t)))}, \\ y(t+1) = \frac{x_1(t+1)}{1+0.5\sin(x_2(t+1))} + \frac{x_2(t+1)}{1+0.5\sin(x_1(t+1))} \\ \quad + n(t), \end{cases} \quad (92)$$

where $n(t)$ is a white Gaussian noise with variance 0.1 and the states are assumed not to be measurable. We collect 300 data samples by using a uniformly distributed random input $u(t) \in [-2.5, 2.5]$. Also, to validate the learning model, the input signal

$$u(t) = \sin \frac{2\pi t}{10} + \sin \frac{2\pi t}{25}, \quad t = 1, \dots, 200$$

was used. According to the discussion in Roll et al. (2005), we take the regression vector as follows

$$\varphi_y(t) = [y(t-1), y(t-2), y(t-3), u(t-1), u(t-2), u(t-3)]^T.$$

To perform the CRNN algorithm, we let the parameters defined in the learning algorithm (B) be $r = 200$, $\gamma = 100$, and $\epsilon = 0.1$, and we let the parameters defined in the learning algorithm (A) $\tau = 4$ and the initial weight be zero. Figure 21 shows the variation of the predicted signal using the CRNN function estimate. Moreover, Fig. 22 shows the variation of the predicted signal using the LS-SVM function estimate. Figure 23 displays that the absolute error between the original signal and the predicted signal given

Fig. 21 The variation of the signal by the CRNN algorithm vs the number of measurements in Example 3

Fig. 22 The variation of the signal by the LS-SVM predictor algorithm vs the number of measurements in Example 3

by the two function estimates. It is easy to see that the present function estimate has smaller errors than the LS-SVM function estimate. Moreover, the LS-SVM method gives the computed result of fit 45.3% ($\pm 2.3\%$), while the CRNN approach gives a better fit 51.7% ($\pm 2.1\%$). Furthermore, the computed result by the CRNN algorithm can be compared with the ones obtained by the DWO method and the individual neural network with 20 hidden sigmoidal function given in Roll et al. (2005), where the DWO method gave 49.7% fit, and the individual neural network then reaches 47.1% fit.

Example 2. Consider real-life sea clutter data (Leung et al. 2000). The radar used for data collection is a coherent dual-polarized X-band radar. The radar site was located at $44^{\circ}36.72'$ N $63^{\circ}25.41'$ W on a cliff at Osborne Head, Nova Scotia, Canada facing the Atlantic Ocean at a height of about 30 m above mean sea level and an open ocean view of

Fig. 23 The absolute error variation of two signal prediction in example 3: the solid line corresponds to the the CRNN algorithm and the dot line to the LS-SVM algorithm

about 130° . The database contains a variety of targets, including small boats and beach ball targets. A data set is used in this study to investigate the spatial temporal dynamics of sea clutter (Drosopoulos 1994). Its original radar image plotted in Fig. 24(a). Here, the y-axis denotes the range, and the x-axis represents the time. When visible, the targets appears in the staring images as a dark horizontal line. Since the target was not anchored and so it fluctuates with the waves, causing it to be more easy to detect at some times than at others. The target, which is a boat, appears as a solid dark line in the image without any breaking.

To use the CRNN algorithm, we choose the simulation parameters, $p = 10$, $r = 100$, $\gamma = 100$, and $\epsilon = 1$, and we take the Gaussian RBF function as a kernel function. For the measurement fusion, we fuse 5 neighboring cells in the spatial domain and the resulting image is plotted in Fig. 24(b). We also perform a comparison between the CRNN predictor and the LS-SVM predictor. Figures 23 and 25 display the mean square prediction errors (MSE), where the solid line corresponds to the LS-SVM predictor, and the dashed line to the CRNN predictor, respectively.

Based on the obtained results, we have two observations. The predicted radar image, looks very close to their originals, which means that the CRNN predictor provides a good model for these sea clutter data. Next, the CRNN algorithm produces a better model than the LS-SVM algorithm.

CRNNs for kinematic control of redundant robot mainpulators

Problems and resolution

Redundant robots are those having more degree (DOF) of freedom than that required for position and orientation

Fig. 24 (a) Radar image of the data set. (b) Predicted radar image

Fig. 25 Prediction error of the data set. The solid line corresponds to the LS-SVM algorithm and the dashed line to the CRNN algorithm

in a given workspace (Sciavicco and Siciliano 2000). The use of kinematically redundant manipulators is expected to increase dramatically in recent years because of their ability to avoid the internal singularity configurations and obstacles (Baker and Wampler 1988; Maciejewski and Klein 1985), and to optimize dynamic performance (Klein and Huang 1983), as well as to conduct the end-effector motion task.

The forward kinematics problem of a robot manipulator in robotics is concerned with the transformation of position and orientation information in a joint space to a Cartesian space described by a forward kinematics equation:

$$\mathbf{r}(t) = F(\theta(t)), \quad (93)$$

where $\theta(t)$ is an m -vector of joint variables, $\mathbf{r}(t)$ is an n -vector of the end-effector position and orientation, and $F(\cdot)$ is a continuous nonlinear function with structure and parameters for a given manipulator. The inverse kinematics problem is very difficult to solve because of the singularity of the underlying nonlinear mapping F and the requirement of real-time solution in sensor-based robotic operations (Guez and Ahmad 1989). Therefore, many efforts have been made in the redundancy resolution of manipulators (Cheng et al. 1994). Conventionally, differentiating (93) with respect to time yields linear relation between joint velocity $\dot{\theta}(t) \in R^m$ and Cartesian velocity $\dot{\mathbf{r}}(t)$.

$$\dot{\mathbf{r}}(t) = J(\theta(t))\dot{\theta}(t), \quad (94)$$

where $J(\theta(t)) \in R^n \times m$ is the Jacobian matrix of F and can be rank-deficient for singular configuration. Because the redundancy resolution problem is to find the solution trajectory $\dot{\theta}(t)$ of (93) together with limit constraints of redundant robots (94), we consider the following resolution problem of kinematically redundant manipulators

$$\begin{aligned} \text{Min } f_1(\dot{\theta}) &= \frac{1}{2}(\dot{\theta} + z)^T Q(\dot{\theta} + z), \\ \text{s.t. } J(\theta)\dot{\theta} &= \dot{\mathbf{r}}(t), \quad \theta^- \leq \theta \leq \theta^+, \quad \dot{\theta}^- \leq \dot{\theta} \leq \dot{\theta}^+. \end{aligned} \quad (95)$$

where $Q \in R^m \times m$ is symmetric and positive definite, the superscripts $+$ and $-$ denote the upper and lower limits, respectively, $z = \alpha(\theta - \theta(0))$ and α is a positive parameter to scale the magnitude of the manipulator response to joint displacement. For simplicity, the joint limit constraints can be replaced by joint velocity constraints in term of the first order Taylor series

$$\dot{\theta}(t) \approx \frac{\theta(t) - \theta(t - \Delta t)}{\Delta t},$$

where $\theta(t - \Delta t)$ is joint position vector at time $t - \Delta t$, and Δt is the sampling time. Thus both the joint velocity constraints and the position constraints can be represented by new joint velocity constraints

$$\dot{\theta}_{\min}(t) \leq \dot{\theta} \leq \dot{\theta}_{\max}(t)$$

where

$$\dot{\theta}_{\min}(t) = \max \left\{ \dot{\theta}^-, \frac{\theta^-(t) - \theta(t - \Delta t)}{\Delta t} \right\},$$

$$\dot{\theta}_{\max}(t) = \min \left\{ \dot{\theta}^+, \frac{\theta^+(t) - \theta(t - \Delta t)}{\Delta t} \right\}.$$

So, the above resolution problem can be rewritten as

$$\begin{aligned} \text{Min} \quad & f(\dot{\theta}) = \frac{1}{2} \dot{\theta}^T Q \dot{\theta} + z^T Q \dot{\theta}, \\ \text{s.t.} \quad & J(\theta) \dot{\theta} = \dot{\mathbf{r}}(t), \quad \dot{\theta}_{\min}^- \leq \dot{\theta} \leq \dot{\theta}_{\max}^+. \end{aligned} \quad (96)$$

Feedforward neural networks have been used for robot kinematic control, such as the multilayer perceptron trained via supervised learning, by using the backpropagation algorithm (Arteaga-Bravo 1990). Recurrent neural networks with feedback connections have also been applied for kinematic control (Bekey 1992). Unlike the feedforward neural networks, most recurrent neural networks do not need off-line supervised learning and thus are more suitable for real-time robot control in uncertain environments. For example, Ding and Tso (1999) applied the Tank–Hopfield neural network approach to the kinematic resolution problem. To overcome the disadvantage of the penalty parameter arisen in the Tank–Hopfield network, Wang et al. (1999) presented a Lagrangian network for the kinematic resolution without considering physical limit constraints. Recently, three cooperative recurrent neural networks were applied to kinematic resolution problems to effectively handle the physical limit constraints.

CRNNs for on-line resolution of constrained kinematic redundancy

The first CRNN, called as the dual neural network (Xia and Wang 2001), is defined as follows.

State equation:

$$\begin{cases} \frac{d\mathbf{u}_I}{dt} = \lambda(g((Q^{-1} - I_1)\mathbf{u}_I + Q^{-1}J^T\mathbf{u}_{II}) - Q^{-1}(\mathbf{u}_I + J^T\mathbf{u}_{II})) \\ \frac{d\mathbf{u}_{II}}{dt} = \lambda(\dot{\mathbf{r}}(t) - JQ^{-1}(\mathbf{u}_I + J^T\mathbf{u}_{II})) \end{cases} \quad (97)$$

Output equation:

$$\dot{\theta}(t) = Q^{-1}(J^T, I_1)\mathbf{u}(t) + \dot{\mathbf{r}}(t).$$

where $\lambda > 0$ is a scaling constant, $I_1 \in R^{n \times n}$ is a unit matrix, $\mathbf{u}_I \in R^n$, $\mathbf{u}_{II} \in R^m$, $\mathbf{u} = [\mathbf{u}_I^T, \mathbf{u}_{II}^T]^T$, $g(\mathbf{x}) = [g(x_1), \dots, g(x_n)]^T$,

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

and $\mathbf{l} = \dot{\theta}_{\min}^-$ and $\mathbf{h} = \dot{\theta}_{\max}^+$. Other two CRNNs are called the primal-dual neural networks (Xia 1996b; Xia and Wang 2001). One has two-layer structure given by

State equation

$$\begin{pmatrix} \frac{d\mathbf{z}_I}{dt} \\ \frac{d\mathbf{z}_{II}}{dt} \end{pmatrix} = \lambda \begin{pmatrix} I_1 + Q & -J^T \\ J & I_2 \end{pmatrix} \begin{pmatrix} -\mathbf{z}_I + g(Q\mathbf{z}_I + \dot{\mathbf{r}}(t) - J^T\mathbf{z}_{II}) \\ -J\mathbf{z}_I + \dot{\mathbf{r}}(t) \end{pmatrix}, \quad (98)$$

Output equation

$$\dot{\theta}(t) = \mathbf{z}_I(t),$$

where $I_2 \in R^m \times m$ is a unit matrix and $\mathbf{z}_I(t) \in R^n$ and $\mathbf{z}_{II}(t) \in R^m$ are state variables. The other has one-layer structure defined as

State equation

$$\begin{cases} \frac{d\mathbf{z}_I}{dt} = \lambda(-\mathbf{z}_I + g(Q\mathbf{z}_I + \dot{\mathbf{r}}(t) - J^T\mathbf{z}_{II})) \\ \frac{d\mathbf{z}_{II}}{dt} = \lambda(-J\mathbf{z}_I + \dot{\mathbf{r}}(t)) \end{cases} \quad (99)$$

Output equation

$$\dot{\theta}(t) = \mathbf{z}_I(t).$$

It is easy to see that the total number of multiplications performed per iteration for the CRNN (97) is $3n^2 + nm + n^2m + O(n^3)$, where $O(n^3)$ is total numbers of multiplications performed for computing the inverse matrix Q^{-1} , the total number of multiplications performed per iteration for the CRNN (98) is $3n^2 + nm + n^2m + O(n^3)$, and the total number of multiplications performed per iteration for the CRNN (99) is $n^2 + 2nm + n + 1$. Therefore, the CRNN (99) has a lower implementation complexity than both the CRNN (97) and the CRNN (98). Figure 26 displays a block diagram of the CRNN (99) for resolving constrained kinematic redundancy.

Examples

Consider a 7-DOF redundant manipulator (PA-10) made by Mitsubishi. The PA10 manipulator (Portable General

Fig. 26 A block diagram of the CRNN algorithm (99) for resolving constrained kinematic redundancy

Purpose Intelligent Arm) has seven degrees-of-freedom (three rotation axes and four pivot axes). The mechanical configuration and coordinate system, together with other specifications, of PA10 redundant manipulator can be found in Seraji et al. (1993). Joint angle limits $\theta^\pm$ and joint rate bounds $\dot{\theta}^\pm$ are listed in Zhang et al. (2003). Simulation has been performed for the path-following task that the end-effector of PA10 manipulator tracks a given straight-line in the 3-dimensional workspace. For simplicity, let the weighting matrix $W = I$ and $\alpha = 4$ in the simulation.

We simulate the redundancy resolution problem of the PA10 manipulator for a straight-line motion with the length of 0.2 m in the 3D workspace, based on the CRNN (99) with $\lambda = 10^8$. The task time of the motion is 10 s, and the PA10 initial state is $\theta(0) = [0, \pi/4, 0, 2\pi/3, 0, -\pi/4, 0]^T$. Figure 27 shows the 3-dimensional motion trajectories of PA10 manipulators based on the CRNN (99). Figure 28 illustrates the transient behavior of the joint variables θ ,

Fig. 27 Simulated trajectory of the seven DOF PA-10 redundant manipulators along a line based on (15)

Fig. 28 Joint variables of the PA10 manipulator along a line based on (15)

Fig. 29 illustrates the transient behavior of the joint velocity variables $\dot{\theta}$. Figure 30 illustrates the transient behavior of the Cartesian position error e , where e_x , e_y and e_z denote the components of tracking position error e respectively along the x , y and z axes of the base frame. Figure 31 illustrates the transient behavior of the velocity error $\dot{e}$, where $\dot{e}_x$, $\dot{e}_y$ and $\dot{e}_z$ denote respectively the x , y and z -axis components of tracking velocity error $\dot{e}$ at the end-effector of PA10 robot arm. It is easy to see that the maximal position and velocity absolute errors are respectively less than 10^{-7} mm/s and 2×10^{-8} mm/s. For a comparison of the convergence rate, we compare the CRNN (99) with the CRNN (97). Let the scaling constant in the two CRNNs be $\lambda = 10^7$. Tables 5 and 6 display their results under four different initial points. From these results we can see that

Fig. 29 Joint velocity variables of the PA10 manipulator along a line based on (15)

Fig. 30 Velocity errors of the PA10 manipulator along a line based on (15)

Fig. 31 Velocity errors of the PA10 manipulator along a line based on (15)

Table 5 CRNN (99) for line motion

Initial point	CPU time	l_2 -norm position error	l_2 -norm velocity error
$[0, -\pi/4, 0, \pi/2, 0, -\pi/4, 0]^T$	0.88	2×10^{-7}	2×10^{-8}
$[0, \pi/4, 0, 2\pi/3, 0, -\pi/4, 0]^T$	0.69	10^{-7}	2×10^{-8}
$[0, -\pi/4, 0, -2\pi/3, 0, \pi/4, 0]^T$	0.66	10^{-7}	2×10^{-8}
$[0, \pi/4, 0, -\pi/2, 0, \pi/4, 0]^T$	0.77	3×10^{-7}	4×10^{-8}

Table 6 CRNN (97) for line motion

Initial point	CPU time	l_2 -norm position error	l_2 -norm velocity error
$[0, -\pi/4, 0, \pi/2, 0, -\pi/4, 0]^T$	1.16	3×10^{-7}	5×10^{-8}
$[0, \pi/4, 0, 2\pi/3, 0, -\pi/4, 0]^T$	1.04	10^{-7}	4×10^{-8}
$[0, \pi/4, 0, -2\pi/3, 0, \pi/4, 0]^T$	0.9	10^{-7}	10^{-7}
$[0, \pi/4, 0, -\pi/2, 0, \pi/4, 0]^T$	0.99	3×10^{-7}	6×10^{-8}

the convergence rate of the CRNN (99) is faster than that of the CRNN (97).

Conclusion

This paper reviews briefly the advantages of the cooperative modular neural networks over single recurrent neural networks and modular neural networks, and then surveys current work on cooperative recurrent neural network for constrained optimization problems, their standing models, and their engineering applications from computation and application viewpoints. Despite the good performance of

these standing models, there are some important issues that require further investigation. The following several recommendations are given for future research.

Cooperative recurrent neural networks for constrained optimization were shown to have a fast convergence under general regular conditions. But it may have slower convergence rate under relatively weaker conditions. Moreover, cooperative recurrent neural networks for solving constrained nonsmooth optimization problems have often a slower convergence. Thus, investigating ways to speed the convergence of these recurrent neural networks would be significant.

Cooperative recurrent neural networks for constrained optimization were shown to reach a global optimal solution. Many scientific and engineering problems, such as signal and image processing, regression mining, system identification, and robot control, can be solved by transforming the original problem into a constrained optimization problem. However, some real-world problems are so complex that their models are unknown or uncertain. How to design a cooperative modular neural network which can obtain a global optimal solution of such complex problems needs to be studied.

Continuous-time cooperative recurrent neural networks for nonlinear programming can be easily performed by using well known ODE software in MATLAB. Moreover, these continuous-time cooperative recurrent neural networks can lead to discrete-time cooperative recurrent neural networks. Thus, it would be useful for digital implementation to give a fixed step size in these discrete-time cooperative recurrent neural networks.

References

- Agganwal JK (1993) Multisensor fusion for computer vision. Springer-Verlag, Berlin
- Alexandre LA, Campilho AC, Kamel M (2001) On combining classifiers using sum and product rules. Pattern Recognit Lett 22:1283–1289
- Alliney S, Ruzinsky SA (1994) An algorithm for the minimization of mixed L_1 and L_2 norms with application to bayesian-estimation. IEEE Trans Signal Process 42:618–627
- Al-Smadi A, Alshamali A (2002) Fitting ARMA models to linear non-Gaussian processes using higher order statistics. Signal Processing 82:1789–1793
- Anand R, Mehrotra K, Mohan CK, Ranka S (1995) Efficient classification for multiclass problems using modular neural networks. IEEE Trans Neural Netw 6:117–124
- Andrews HC, Hunt BR (1977) Digital image restoration. Prentice-Hall, New York
- Anguita D, Boni A (2002) Improved neural network for SVM learning. IEEE Trans Neural Netw 13:1243–1244
- Anzai Y, Shimada T (1988) Modular neural networks for shape and/or location recognition. Int Neural Network Soc First Ann Meeting, Boston, USA, 158 pp
- Arteaga-Bravo FJ (1990) Multilayer backpropagation network for learning the forward and inverse kinematics equations. Proc Int Conf Neural Netw, vol II, Washington, DC, pp 319–322

- Auda G, Kamel MS (1997a) CMNN: cooperative modular neural network. *Int Conf Neural Netw* 1:226–231
- Auda G, Kamel MS (1997b) CMNN: cooperative modular neural networks for pattern recognition. *Pattern Recognit Lett* 18:1391–1398
- Auda G, Kamel MS (1998a) Modular neural network classifiers: a comparative study. *J Intell Robot Syst* 21:117–129
- Auda G, Kamel MS (1998b) CMNN: cooperative modular neural network. *Neurocomputing* 20:189–207
- Auda G, Kamel MS (1999) Modular neural networks: a survey. *Int J Neural Syst* 9:129–151
- Avitzour D (1992) A maximum likelihood approach to data association. *IEEE Trans AES* 28(2):560–566
- Baker DR, Wampler CW II (1988) On the inverse kinematics of redundant manipulators. *Int J Robot Res* 7:3–21
- Bazaraa MS, Sherali HD, Shetty CM (1993) *Nonlinear programming – theory and algorithms*, 2nd edn. Wiley, New York
- Bekey GA (1992) Robotics and neural networks. In: Kosko B (ed) *Neural networks for signal processing*. Prentice-Hall, Englewood, pp 161–185
- Broomhead DS, Lowe D (1988) Multivariable functional interpolation and adaptive networks. *Complex Syst* 2:321–355
- Burges CJ (1998) A tutorial on support vector machines for pattern recognition. *Data Mining Knowl Discov* 2(2):1–47
- Caelli T, Guan L, Wen W (1999) Modularity in neural computing. *Proc IEEE* 87:1497–1518
- Chang PR, Yang WH, Chan KK (1992) A neural network approach to MVDR beamforming problem. *IEEE Trans Antennas Propagation* 40:313–322
- Chen S, Billings SA (1992) Neural networks for nonlinear dynamic system modelling and identification. *Int J Control* 56:319–346
- Chen LJ, Narendra KS (2004) Identification and control of a nonlinear discrete-time system based on its linearization: a unified framework. *IEEE Trans Neural Netw* 15:663–673
- Cheng F-T, Chen T-H, Sun Y-Y (1994) Resolving manipulator redundancy under inequality constraints. *IEEE J Robot Automat* 10(1):65–71
- Cheng F-T, Sheu R-J, Chen T-H (1995) The improved compact QP method for resolving manipulator redundancy. *IEEE Trans Syst Man Cybern* 25(11):1521–1530
- Cichocki A, Unbehauen R (1993) *Neural networks for optimization and signal processing*. Wiley, England
- Copeland M (1995) Modular network control for robot manipulator. *Proc IEEE Visualize Future* 26–29:183–187
- Cortes C, Vapnik V (1995) Support-vector networks. *Mach Learn* 20:273–297
- Ding H, Tso SK (1999) A fully neural-network-based planning scheme for torque minimization of redundant manipulators. *IEEE Trans Industr Electr* 46:199–206
- Drosopoulos A (1994) Description of the OHGR database. Defence Research Establishment, Ottawa, Ont, Tech. Note 94–14
- Du KL, Lai AKY, Cheng KKM, Swamy MNS (2002) Neural methods for antenna array signal processing: a review. *Signal Processing* 82:547–561
- Durrant-Whyte HF (1985) Integrating distributed sensor information: an application to robot system coordinate. *Proc IEEE Int Conf Syst Man Cyber*, pp 415–419
- Fiori S (2003) Neural minor component analysis approach to robust constrained beamforming. *IEEE Proc Vision Image Signal Process* 150:205–218
- Fogelman-Soulie F (1993) Multi-modular neural network-hybrid architectures: a review. *Proc Int Joint Conf Neural Netw* 3:25–29
- Frost OL (1972) An algorithm for linear constrained adaptive array processing. *Proc IEEE* 60:926–935
- Fukushima M, Luo ZQ, Tseng P (2003) A sequential quadratically constrained quadratic programming method for differentiable convex minimization. *SIAM J Optim* 13:1098–1119
- Galatsanos NP, Katsaggelos AK (1992) Methods for choosing the regularization parameter and estimate the noise variance in image restoration and their relation. *IEEE Trans Image Process* 1:322–336
- Gao XB, Liao LZ, Qi LQ (2005) A novel neural network for variational inequalities with linear and nonlinear constraints. *IEEE Trans Neural Netw* 16:1305–1317
- Giannakis GB, Mendel JM (1990) Cumulant-based order determination of non-Gaussian ARMA models. *IEEE Trans Acoust* 38:1411–1423
- Glazos MP, Hui S, Zak SH (1998) Sliding modes in solving convex programming problems. *SIAM J Control Optim* 36:680–697
- Guan L, Anderson JA, Sutton JP (1997) A network of networks processing model for image regularization. *IEEE Transactions On Neural Networks* 8:169–174
- Guez A, Ahmad Z (1989) Accelerated convergence in the inverse kinematics via multilayer feedforward networks. *Proc Int Conf Neural Netw*, vol II, Washington, DC, pp 341–344
- Hansen LK, Salamon P (1990) Neural network ensembles. *IEEE Trans Pattern Anal Mach Intell* 12:993–1002
- Hashem S (1997) Optimal linear combination of neural networks. *Neural Netw* 10:599–614
- Haykin S (1994) *Neural networks: a comprehensive foundation*. Macmillan Publishing, New York
- He Y, Sun Y (2001) Neural network-based L_1 norm optimization approach for fault diagnosis of nonlinear circuits with tolerance. *IEEE Proc Circuit Devices* 148:223–229
- Hodge L, Auda G, Kamel MS (1999) Learning decision fusion in cooperative modular neural networks. *Int Joint Conf Neural Netw* 4:2777–2781
- Hong MC, Stathaki T, Katsaggelos AK (2002) Iterative regularized least-mean mixed-norm image restoration. *Opt Eng* 41:2515–2524
- Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational ability. *Proc Natl Acad Sci USA Biophys* 79:2554–2558
- Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. *Neural Netw* 2:359–366
- Islam MM, Yao X, Murase K (2003) A constructive algorithm for training cooperative neural network ensembles. *IEEE Trans Neural Netw* 14:820–834
- Jacobs RA, Jordan MI, Barto AG (1991) Task decomposition through competition in modular connectionist architecture: the what and where vision tasks. *Conit Sci* 15:219–250
- Kadirkamanathan V, Fabri SG (1998) Recursive structure estimation for nonlinear identification with modular network. *Proc IEEE*, pp 343–348
- Kadirkamanathan V, Niranjana M (1993) A function estimation approach to sequential learning with neural networks. *Neural Comput* 5:954–975
- Kamel MS (1999) Neural networks: the state of the art. *Proceeding of the eleventh international conference on microelectronics*, pp 22–24, Nov
- Katsaggelos AK, Biemond J, Schafer RW, Mersereau RM (1991) A regularized iterative image-restoration algorithm. *IEEE Trans Signal Process* 39:914–929
- Kay SM (1993) *Fundamentals of statistical signal processing: estimation theory*. Prentice-Hall, Englewood Cliffs
- Kecman V (1996) System identification using modular neural network with improved learning. *Proceedings of International Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing* 21–23:40–48

- Kehagias A, Petridis V (1997) Predictive modular neural networks for time series classification. *Neural Netw* 10:31–49
- Kennedy MP, Chua LO (1988) Neural networks for nonlinear programming. *IEEE Trans Circuits Syst* 35(5):554–562
- Kinderlehrer D, Stampacchia G (1980) An introduction to variational inequalities and their applications. Academic Press, New York
- Klein CA, Huang CH (1983) Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Trans Syst Man Cybern* 13:245–250
- Kohonen T (1989) Self-organization and associative memory, 3rd edn. Springer-verlag, Berlin
- Koukoulas P, Kalouptsidis N (2000) Second-order Volterra system identification. *IEEE Trans Signal Process* 48:3574–3577
- Kuo SS, Mammone JJ (1992) Image-restoration by convex projections using adaptive constraints and L_1 norm estimation. *IEEE Trans Signal Process* 40:159–168
- Leung H, Hennessey G, Drosopoulos A (2000) Signal detection using the radial basis function coupled map lattice. *IEEE Trans Neural Netw* 11(5):1133–1151
- Leung Y, Chen KZ, Jiao YC et al (2001) A new gradient-based neural network for solving linear and quadratic programming problems. *IEEE Trans Neural Netw* 12:1074–1083
- Li W, Swetits JJ (1998) The linear L_1 estimator and the Huber m-estimator. *SIAM J Optim* 8:457–475
- Li W, Huang Xinping, Leung H (2004) Performance evaluation of digital beamforming strategies for satellite communications. *IEEE Trans Aerosp Electron Syst* 40:12–26
- Ljung L (1999) System identification – theory for the user, 2nd edn. Prentice-Hall, Upper Saddle River
- Lu BL, Ito M (1999) Task decomposition and module combination based on class relations: a modular neural network for pattern classification. *IEEE Trans Neural Netw* 10:1244–1250
- Maa CY, Shanblatt MA (1992) Linear and quadratic programming neural network analysis. *IEEE Trans Neural Netw* 3:580–594
- Maciejewski AA, Klein CA (1985) Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *Int J Robot Res* 4(3):109–117
- Mammone RJ, Rothacker RJ (1987) General iterative method of restoring linearly degraded images. *J Opt Soc Am A Opt Image Sci Vis* 4:208–215
- Mangasarian OL, Musicant DR (1999) Successive overrelaxation for support vector machines. *IEEE Trans Neural Netw* 10:1032–1037
- McCulloch W, Pitts W (1943) A logical calculus of the ideas imminent in nervous activity. *Bull Math Biophys* 5:115–133
- Melin P, Felix C, Castillo O (2005) Face recognition using modular neural networks and the fuzzy Sugeno integral for response integration. *Int J Intell Syst* 20:275–291
- Muller KR, Smola AJ, Ratsch G, Schokopf BS, Kohlmorgen J, Vapnik V (1997) Predicting time series with support vector machines. *Proceedings of ICANN, Springer, LNCS* 1327:999–1004
- Nehorai A, Stoica P (1988) Adaptive algorithms for constrained ARMA signals in the presence of noise. *IEEE Trans Acoust* 36:1282–1291
- Nesterov Y, Nemirovsky A (1994) Interior-point polynomial methods in convex programming. *Studies in Applid Mathematics, SIMA*, 13
- Oh IS, Suen CY (2002) A class-modular feedforward neural network for handwriting recognition. *Pattern Recognit* 35:229–244
- Oja E (1992) Principal components, minor components and linear neural networks. *Neural Netw* 5:927–935
- Ozawa S (1998) An associative memory model derived from cross-coupled Hopfield nets and its role in noise-space dynamics. *Electr Eng* 117:1253–1258
- Paik JK, Katsaggelos AK (1992) Image restoration using a modified Hopfield network. *IEEE Trans Image Process* 1:49–63
- Perry SW, Guan L (1996) A partitioned modified Hopfield neural network algorithm for real-time image restoration. *Real-Time Imaging* 2:215–224
- Reed IS, Mallett JD, Brennan LE (1974) Rapid convergence rate in adaptive arrays. *IEEE Trans Aerosp Electron Syst* AES-10:853–863
- Rodríguez-Vázquez A, Domínguez-Castro R, Rueda A, Huertas JL, Sánchez-Sinencio E (1990) Nonlinear switched-capacitor ‘neural networks’ for optimization problems. *IEEE Trans Circuits Syst* 37:384–397
- Rojo-Alvarez JL, Martinez-Ramon M, de Prado-Cumplido M, Artes-Rodriguez A, Figueiras-Vidal AR (2004) Support vector method for robust ARMA system identification. *IEEE Trans Signal Process* 52:155–164
- Roll J, Nazin AA, Ljung L (2005) Nonlinear system identification via direct weight optimization. *Automatica* 41:475–490
- Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by backpropagating errors. *Nature* 323:533–536
- Ruzinsky SA, Olsen ET (1989) L_1 and L_∞ minimization via a variant of Karmarkar’s algorithm. *IEEE Trans Signal Process* 37:245–253
- Schapire RE (1990) The strength of weak learnability. *Mach Learn* 5:197–227
- Sciavicco L, Siciliano B (2000) Modelling and control of robot manipulators. Springer-Verlag London, Great Britain
- Seraji H, Long MK, Lee TS (1993) Motion control of 7-DOF arms: the configuration control approach. *IEEE Trans Rob Autom* 4:125–139
- Sharkey AJC (1998) Combining artificial neural nets: ensemble and modular multi-nets systems. Springer-Verlag, New York
- Sierra A, Cruz CS (1998) Global and local neural network ensembles. *Pattern Recogn Lett* 19:651–655
- Smith KA (1999) Neural networks for combinatorial optimization: a review of more than a decade of research. *Inform J Comput* 11:15–28
- Smith RC, Cheesman P (1986) On the representation of estimation and spatial uncertainty. *Int J Robot Res* 5(1):56–68
- Soderstrom T, Stoica P (1981) Comparison of some instrumental variable methods-consistency and accuracy aspects. *Automatica* 17:101–115
- Solodov MV, Tseng P (1996) Modified projection-type methods for monotone variational inequalities. *SIAM J Control Optim* 2:1814–1830
- Sun Y (2000) Hopfield neural network based algorithms for image restoration and reconstruction – Part I: algorithms and simulations. *IEEE Trans Signal Process* 48:2105–2118
- Suykens JAK, Gestel TV, Brabanter JD, Moor BD, Vandewalle J (2002) Least squares support vector machines. World Scientific, Singapore
- Tan Y, Xia Y, Wang J (2000) Neural network realization of support vector methods for pattern classification. *Proc IEEE Int Joint Conf Neural Netw*, pp 411–416
- Tank DW, 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
- Tao Q, Fang TJ (2000) The neural network model for solving minimax problems with constraints. *Control Theory Control Appl* 17(1):82–84 (Chinese Edition)
- Tao Q, Cao JD, Xue MS, Qiao H (2001) A high performance neural network for solving nonlinear programming problems with hybrid constraints. *Phys Lett A* 288(2):88–94
- Tekalp AM, Kaufman H, Woods JW (1985) Fast recursive estimation of the parameters of a space-varying autoregressive image model. *IEEE Trans Acoust* 33:469–472

- Thompson AM, Brown JC, Kay JW, Titterton DM (1991) A study of methods of choosing the smoothing parameter in image restoration by regularization. *IEEE Trans Pattern Anal Mach Intell* 13:326–339
- Tian Z, Bell KL, Van Trees HL (2001) A recursive least squares implementation for LCMP beamforming under quadratic constraint. *IEEE Trans Signal Process* 49:1138–1145
- Van Trees HL (1998) Optimum array processing: detection and estimation theory. *Class Notes: INFT* 836
- Varshney PK (1997) Multisensor data fusion. *J Electron Commun Eng* 9:245–253
- Wang J, Hu Q, Jiang D (1999) A Lagrangian neural network for kinematic control of redundant robot manipulators. *IEEE Transactions on Neural Networks* 10:1123–1132
- Wang ZS, Cheung JY, Xia YS, Chen JD (2000) Minimum fuel neural networks and their applications to overcomplete signal representations. *IEEE Trans Circuits Syst I* 47(8):1146–1159
- Xia YS (1996a) A new neural network for solving linear programming problems and its applications. *IEEE Trans Neural Netw* 7:525–529
- Xia YS (1996b) A new neural network for solving linear and quadratic programming problems. *IEEE Trans Neural Netw* 7:1544–1547
- Xia YS (1997) Neural network for solving extended linear programming. *IEEE Trans Neural Netw* 8:803–806
- Xia YS (2003) Global convergence analysis of Lagrangian networks. *IEEE Trans Circuits Syst I* vol 50
- Xia YS (2004) An extended projection neural network for constrained optimization. *Neural Comput* 16:863–883
- Xia YS, Feng G (2004) A modified neural network for quadratic programming with real-time applications. *Neural Inf Process* 3:69–76
- Xia YS, Feng G (2006) A neural network for robust LCMP beamforming. *Signal Processing* 86(10):2901–2912
- Xia YS, Kamel MS (2007a) Cooperative recurrent neural networks for the constrained L1 norm estimator. *IEEE Trans Signal Process* 55:3192–3205
- Xia YS, Kamel MS (2007b) Novel cooperative neural fusion algorithms for image restoration. *IEEE Trans Image Process* 16:367–381
- Xia YS, Kamel MS (2007c) A measurement fusion method for nonlinear system identification and its cooperative learning algorithm. *Neural Comput* 19:1589–1632
- Xia YS, Kamel MS (2007d) Cooperative recurrent neural networks for the constrained L1 norm estimator. *IEEE Trans Signal Process* 55:3192–3205
- Xia YS, Kamel MS (2008) A generalized least absolute deviation method for parameter estimation of autoregressive signals. *IEEE Trans Neural Netw* 19:107–118
- Xia YS, Wang JS (1995) Neural network for solving linear programming problems with bounded variables. *IEEE Trans Neural Netw* 6:515–519
- Xia YS, Wang J (1998a) A general methodology for designing globally convergent optimization neural networks. *IEEE Trans Neural Netw* 9:1331–1343
- Xia YS, Wang J (1998b) Neural network for solving the least absolute deviation fitting problem. *Neurcomputing* 19:13–21
- Xia YS, Wang J (2001) A dual neural network for kinematic control of redundant robot manipulators. *IEEE Trans Syst Man Cybern B* 31(1):147–154
- Xia YS, Wang J (2004a) A general projection neural network for solving monotone variational inequality and related optimization problems. *IEEE Trans Neural Netw* 15:318–328
- Xia Y, Wang J (2004b) A one-layer recurrent neural network for support vector machine learning. *IEEE Trans Syst Man Cybern B* 1261–1269
- Xia YS, Wang J (2005) A recurrent neural network for solving nonlinear convex programs subject to linear constraints. *IEEE Trans Neural Netw* 16:379–386
- Xia YS, Leung H, Chan H (2006) A prediction fusion method for reconstructing spatial temporal dynamics using support vector machines. *IEEE Trans Circuits Syst Ii-Express Briefs* 53:62–66
- Xia YS, Henry L, Wang J (2002a) A projection neural network and its application to constrained optimization problems. *IEEE Trans Circuits Syst I* 49(4):447–458
- Xia YS, Henry L, Bossé E (2002b) Neural data fusion algorithms based on a linearly constrained least square method. *IEEE Trans Neural Netw* 13(2):320–329
- Xia Y, Feng G, Wang J (2004a) A Recurrent neural network with exponential convergence for solving convex quadratic program and related linear piecewise equations. *Neural Netw* 17:1003–1015
- Xia YS, Wang J, Fork LM (2004b) Grasping force optimization for multifingered robotic hands using a recurrent neural network. *IEEE Trans Rob Autom* 20:549–554
- Xia YS, Feng G, Wang J (2005) A primal-dual neural network for on-line resolving constrained kinematic redundancy. *IEEE Trans Syst Man Cybern B* 35:54–64
- Xia YS, Feng G, Kamel MS (2007) Development and analysis of neural dynamical approaches to solving nonlinear programming problems with application to optimal control. *IEEE Trans Automat Contr*, December
- Yamaguchi S, Itakura H (1999) A modular neural network for control of mobile robots. *Int Conf Neural Inform Processing* 2:661–666
- Yang S, Browne A (2001) Neural network ensembles: combining multiple models for enhanced performance using a multistage approach. *Expert Syst* 21:279–288
- Young PC (1970) An instrumental variable methods for real time identification of a noise process. *Automatica* 6:2711–287
- Zervakis ME, Katsaggelos AK, Kwon TM (1995) A class of robust entropic functionals for image restoration. *IEEE Trans Image Process* 4:752–773
- Zhang GP (2000) Neural networks for classification: a survey. *IEEE Trans Syst Man Cybern B* 30:451–459
- Zhang S, Constantinides AG (1992) Lagrange programming neural networks. *IEEE Trans Circuits Syst* 39(7):441–452
- Zhang S, Zhu X, Zou L-H (1992) Second order neural networks for constrained optimization. *IEEE Trans Neural Netw* 3:1021–1024
- Zhang Y, Wang J, Xia YS (2003) A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits. *IEEE Trans Neural Netw* 14:658–667
- Zheng WX (1999) A least-squares based method for autoregressive signals in the presence of noise. *IEEE Trans Circuits Syst Ii-Analog Digital Signal Process* 46:81–85
- Zheng WX (2000) Estimation of the parameters of autoregressive signals from colored noise-corrupted measurements. *IEEE Signal Process Lett* 7:201–204
- Zooghyby AHEI, Christodoulou CG, Georgiopoulos M (1998) Neural network-based adaptive beamforming for one- and two-dimensional antenna arrays. *IEEE Trans Antennas Propagation* 46:1891–1893