IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 13, 2019, accepted September 22, 2019, date of publication October 7, 2019, date of current version October 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2945807

An ℓ_0-Norm-Based Centers Selection for Failure Tolerant RBF Networks

HAO WANG, ZHANGLEI SHI, HIU TUNG WONG, CHI-SING LEUNG, (Senior Member, IEEE), HING CHEUNG SO, (Fellow, IEEE), AND RUIBIN FENG
Department of Electrical Engineering, City University of Hong Kong, Hong Kong

Corresponding author: Chi-Sing Leung (eeleungc@cityu.edu.hk)

The work was supported by the City University of Hong Kong under Grant 7005063 and Grant 9610431.

ABSTRACT There are two important issues in the construction of a radial basis function (RBF) neural network. The first one is to select suitable RBF centers. The second one is that the resultant RBF network should be with good fault tolerance. This paper proposes an algorithm that is able to select RBF centers and to train fault tolerant RBF networks simultaneously. The proposed algorithm borrows the concept from sparse approximation. In our formulation, we first define a fault tolerant objective function based on all input vectors from the training samples. We then introduce the minimax concave penalty (MCP) function, which is an approximation of ℓ_0-norm, into the objective function. The MCP term is able to force some unimportant RBF weights to zero. Hence the RBF node selection process can be achieved during training. As the MCP function is nondifferentiable and nonconvex, traditional gradient descent based algorithms are still unable to minimize the modified objective function. Based on the alternating direction method of multipliers (ADMM) framework, we develop an algorithm, called ADMM-MCP, to minimize the modified objective function. The convergent proof of the proposed ADMM-MCP algorithm is also presented. Simulation results show that the proposed ADMM-MCP algorithm is superior to many existing center selection algorithms under the concurrent fault situation.

INDEX TERMS Failure tolerant, RBF, center selection, ADMM, ℓ_0-norm, global convergence.

I. INTRODUCTION

Constructing a radial basis function (RBF) network [1]–[3] includes two key issues. The first issue is to select some suitable RBF centers. The second issue is to determine the RBF weights. There are many ways to perform RBF center selection. We can select all the input vectors from the training samples as RBF centers [4]. But this method may create a network with unnecessarily complicated structure. Another way is to randomly select a subset of the input vectors as the RBF centers, but this simple method cannot ensure that the constructed RBFs cover the input space well [5]. Other advanced methods include clustering algorithms [6], orthogonal least squares (OLS) approach [7], [8], and support vector regression (SVR) [9], [10]. However, many algorithms in this area did not consider the fault tolerant issue.

The associate editor coordinating the review of this manuscript and approving it for publication was Guiwu Wei.

Biological neural networks have the capability to tolerate fault or noise conditions [11]. For example, human can recognize an object from a noisy image/video. In addition, when a few neural cells or synapses malfunction, human brain can still works properly. Since the concept of neural networks comes from biological neural networks, researchers expected that a trained neural network has a capability to tolerate weight or neuron failure. However, many literatures reported that when fault/noise tolerant procedures are not introduced during training, the faulty version of a well trained network may have a poor performance [12]–[15].

In realizing a neural network, we face some practical issues [16]. When we use analog technology to realize a dot-product operation, the accuracy is affected by the offset voltage of operational amplifiers [17]. In addition, for analog components, we usually define their accuracy in terms of percentage of error [18]. In this case, we can use multiplicative noise to model the error. For digital technology realization, when a weight is represented in the floating

point format, the round-off errors happen and they can be described by the multiplicative noise model [19]. Apart from multiplicative noise, physical fault may happen [20]. It blocks the data/signal transmission between two connected neurons. Besides, nowadays, the very large scale integration (VLSI) implementation could be at nano-scale and transient noise/failure may happen [21].

In the last twenty years, several fault tolerant algorithms were proposed [13], [14], [22]–[24]. However, most of them assume that a trained network is affected by one kind of fault conditions only. For instance, in [13], [14], only the open node fault model was considered. Recently, [25] first described the concurrent fault situation in which a trained network is affected by multiplicative weight noise and open weight fault concurrently. However, the weight decay term in [25] is not optimal for fault tolerance situation. Later, another approach was proposed in [19]. It is based on regularization and OLS center selection. The performances of this algorithm are better than those of many existing methods. Due to the OLS approach, it cannot select centers and train an RBF network at the same time. To improve the performances of the network and complete the two steps simultaneously, an ℓ_1-norm based fault tolerant RBF center selection method [26] was proposed recently.

As the ℓ_0-norm is a much better approach in compressive sensing for retaining nonzero elements, this paper investigates to use the ℓ_0-norm to replace the ℓ_1-norm for center selection. Since the ℓ_0-norm is a noncontinuous function, it is difficult to design an algorithm to minimize an ℓ_0-norm based objective function. This paper introduces the minimax concave penalty (MCP) function [27], [28], which is an approximation of ℓ_0-norm, into the objective function. Since the MCP function is able to limit the number of RBF nodes used, minimizing the modified objective function can remove some unimportant RBF nodes during training. However, the MCP function is still nondifferentiable and nonconvex, traditional gradient descent like algorithms are unable to minimize the modified objective function. Based on the alternating direction method of multipliers (ADMM) framework [29], we develop an algorithm, called ADMM-MCP, to minimize the modified objective function. The ADMM framework breaks down the minimization problem into three parts. Each part can be solved in a much easier way, even though some of them contain nonconvex and nondifferentiable terms. A theoretical analysis of the convergence of the proposed ADMM-MCP algorithm is then provided. Simulation results show that the proposed ADMM-MCP algorithm is superior to many existing center selection algorithms under the concurrent fault situation.

The contributions of this paper are as follows.

- Based on the MCP concept (an approximation of the ℓ_0-norm), we derive a fault tolerant objective function for training RBF networks and selecting RBF nodes simultaneously.
- Based on the ADMM framework, we derive the updating equations to minimize the proposed objective function.
- We show that the training weight vector converges to a limit point. Besides, the limit point is a stationary point of the Lagrangian function of the objective function.
- Simulation shows that the performances of the proposed algorithm are better than those of many existing training algorithms. Besides, from the paired t-test result, the improvement of using the proposed algorithm over other algorithms is statistically significant.

The rest of this paper is organized as follows. The backgrounds of the ADMM framework and RBF neural networks under the concurrent fault situation are presented in Section II. In Section III, the proposed algorithm is developed. Its convergent property is presented in Section IV. Simulation results are provided in the Section V. Finally, the concluding remark is drawn in Section VI.

II. BACKGROUND

A. NOTATION

We use a lower-case or upper-case letter to represent a scalar while vectors and matrices are denoted by bold lower-case and upper-case letters, respectively. The transpose operator is denoted as $(\cdot)^{\mathrm{T}}$, and $\boldsymbol{I}$ represents the identity matrix with appropriate dimensions. Other mathematical symbols are defined in their first appearance.

B. RBF NETWORKS UNDER CONCURRENT FAULT SITUATION

In this paper, the training set is expressed as

$$\mathcal{D} = \left\{ (\boldsymbol{x}_i, y_i) : \boldsymbol{x}_i \in \mathbb{R}^K, y_i \in \mathbb{R}, i = 1, \cdots, N \right\}, \quad (1)$$

where $\boldsymbol{x}_i$ is the input vector of the i-th sample with dimension K, and y_i is the corresponding output. Similarly, the test set is expressed as

$$\mathcal{D}' = \left\{ (\boldsymbol{x}'_{i'}, y'_{i'}) : \boldsymbol{x}'_{i'} \in \mathbb{R}^K, y'_{i'} \in \mathbb{R}, i' = 1, \cdots, N' \right\}. \quad (2)$$

In the RBF approach, the input-output relationship of the data set is approximated by a weighted sum of the outputs of M RBFs, given by

$$f(\boldsymbol{x}) = \sum_{j=1}^{M} w_j a_j(\boldsymbol{x}) = \sum_{j=1}^{M} w_j \exp\left(\frac{-\left\| \boldsymbol{x} - \boldsymbol{c}_j \right\|_2^2}{s} \right), \quad (3)$$

where $a_j(\boldsymbol{x}) = \exp\left(-\left\| \boldsymbol{x} - \boldsymbol{c}_j \right\|_2^2 / s \right)$ is the output of the j-th RBF node, w_j is the corresponding weight, $\boldsymbol{c}_j$ is the center of the j-th RBF node, and s is a parameter which controls the RBF width. Usually, the RBF centers are selected from the training input vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. **If we use all training input vectors as centers, the resultant network will have serious overfitting problem for faultless situation. In addition, using all the training input vectors is waste of resources. Therefore, center selection is a key step in the training of an RBF network.**

For a fault-free network, the training set mean square error (MSE) is given by

$$\mathcal{E}_{train} = \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{w}\|_2^2, \tag{4}$$

where $\boldsymbol{w} = [w_1, \cdots, w_M]^{\mathrm{T}}$, $\boldsymbol{y} = [y_1, \cdots, y_N]^{\mathrm{T}}$, $\boldsymbol{A}$ is an $N \times M$ matrix, and the (i, j) entry of $\boldsymbol{A}$ is given by

$$[\boldsymbol{A}]_{i,j} = a_j(\boldsymbol{x}_i) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{c}_j\|_2^2}{s}\right). \tag{5}$$

In the implementation of an RBF network, weight failure may happen. Multiplicative weight noise and open weight fault are two common fault in the RBF network [18], [22]–[24], [30]–[33]. When they occur concurrently [19], [25], the implemented weights can be described by

$$\tilde{w}_j = \left(w_j + b_j w_j\right)\beta_j, \quad \forall j = 1, \cdots, M. \tag{6}$$

In (6), β_j's are variables that describe whether the weights are opened or not. When a weight β_j is opened, β_j is equal to 0. Otherwise, β_j is equal to 1. This paper assumes that β_j's are independently identically distributed (i.i.d.) binary random variables. The probability mass function of β_j's is given by

$$\mathrm{Prob}(\beta_j) = \begin{cases} P_\beta, & \text{for } \beta_j = 0, \quad (7) \\ 1 - P_\beta, & \text{for } \beta_j = 1. \quad (8) \end{cases}$$

The statistics of β_j's are then given by

$$\langle \beta_j \rangle = \langle \beta_j^2 \rangle = 1 - P_\beta, \tag{9a}$$
$$\langle \beta_j \beta_{j'} \rangle = (1 - P_\beta)^2, \quad \forall j \neq j'. \tag{9b}$$

The term $b_j w_j$ in (6) is the multiplicative noise. It can be seen that the magnitude of the noise is proportional to that of the weight. This paper assumes that b_j's are i.i.d. zero-mean random variables with variance σ_b^2. With this assumption, the statistics of b_j's are summarized as

$$\langle b_j \rangle = 0, \quad \langle b_j^2 \rangle = \sigma_b^2, \tag{10a}$$
$$\langle b_j b_{j'} \rangle = 0, \quad \forall j \neq j', \tag{10b}$$

where $\langle \cdot \rangle$ is the expectation operator.

Given a particular fault pattern, the training set MSE is

$$\begin{aligned} \tilde{\mathcal{E}}_{train} &= \frac{1}{N} \left\|\boldsymbol{y} - \boldsymbol{A}\tilde{\boldsymbol{w}}\right\|_2^2 \\ &= \frac{1}{N} \sum_{i=1}^{N} \Bigg[y_i^2 - 2y_i \sum_{j=1}^{M} \beta_j w_j a_j(\boldsymbol{x}_i) \\ &\quad + \sum_{j=1}^{M} \sum_{j'=j}^{M} \beta_j \beta_{j'} w_j w_{j'} (1 + b_j b_{j'}) a_j(\boldsymbol{x}_i) a_{j'}(\boldsymbol{x}_i) \\ &\quad + \sum_{j=1}^{M} \sum_{j'=1}^{M} (b_j + b_{j'}) \beta_j \beta_{j'} w_j w_{j'} a_j(\boldsymbol{x}_i) a_{j'}(\boldsymbol{x}_i) \\ &\quad - 2y_i \sum_{j=1}^{M} b_j \beta_j w_j a_j(\boldsymbol{x}_i) \Bigg]. \end{aligned} \tag{11}$$

From (10) and (9), the average training set MSE [19] over all possible fault patterns is given by

$$\begin{aligned} \overline{\mathcal{E}}_{train} = \left\langle \tilde{\mathcal{E}}_{train} \right\rangle &= \frac{P_\beta}{N} \sum_{i=1}^{N} y_i^2 + \frac{1 - P_\beta}{N} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{w}\|_2^2 \\ &\quad + \frac{1 - P_\beta}{N} \boldsymbol{w}^{\mathrm{T}} \left[(P_\beta + \sigma_b^2) \mathrm{diag}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right) - P_\beta \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} \right] \boldsymbol{w}. \end{aligned} \tag{12}$$

In (12), the term $\frac{P_\beta}{N} \sum_{i=1}^{N} y_i^2$ is independent of the weight vector $\boldsymbol{w}$. Hence the training objective can be defined as

$$\psi(\boldsymbol{w}) = \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{w}\|_2^2 + \boldsymbol{w}^{\mathrm{T}} \boldsymbol{R} \boldsymbol{w}, \tag{13}$$

where $\boldsymbol{R} = (P_\beta + \sigma_b^2)\mathrm{diag}\left(\frac{1}{N}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right) - \frac{P_\beta}{N}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$.

C. ADMM

The ADMM framework solves optimization problems by breaking them into smaller pieces [29]. Suppose we have the following objective function:

$$Q(\boldsymbol{z}) = \psi(\boldsymbol{z}) + g(\boldsymbol{z}) \tag{14}$$

where $\boldsymbol{z} \in \mathbb{R}^n$. The objective function can be separated into two terms: $\psi(\cdot)$ and $g(\cdot)$. If the term $g(\boldsymbol{z})$ is nonconvex and nondifferentiable, then it is difficult to minimize $Q(\boldsymbol{z})$ directly. The ADMM framework introduces a dummy vector $\boldsymbol{y} \in \mathbb{R}^n$ and reformulates the minimization problem as a constrained optimization problem, given by

$$\min_{\boldsymbol{z},\boldsymbol{y}} : \ \psi(\boldsymbol{z}) + g(\boldsymbol{y}) \tag{15a}$$
$$s.t. \ \boldsymbol{z} = \boldsymbol{y}. \tag{15b}$$

We then construct an augmented Lagrangian function:

$$\begin{aligned} L(\boldsymbol{z}, \boldsymbol{y}, \boldsymbol{\alpha}) = \psi(\boldsymbol{z}) + g(\boldsymbol{y}) + \boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{z} - \boldsymbol{y}) \\ + \frac{\rho}{2} \|\boldsymbol{z} - \boldsymbol{y}\|_2^2, \end{aligned} \tag{16}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^p$ is the Lagrange multiplier vector, and $\rho > 0$ is a parameter that affects the convergent speed. The algorithm consists of three steps, given by

$$\boldsymbol{y}^{k+1} = \arg\min_{\boldsymbol{y}} L(\boldsymbol{z}^k, \boldsymbol{y}, \boldsymbol{\alpha}^k), \tag{17a}$$
$$\boldsymbol{z}^{k+1} = \arg\min_{\boldsymbol{z}} L(\boldsymbol{z}, \boldsymbol{y}^{k+1}, \boldsymbol{\alpha}^k), \tag{17b}$$
$$\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}^k + \rho\left(\boldsymbol{z}^{k+1} - \boldsymbol{y}^{k+1}\right). \tag{17c}$$

It should be noticed that for many forms of $g(\boldsymbol{y})$, we have closed form solutions for (17a), even though $g(\boldsymbol{y})$ is nonconvex and nondifferentiable.

III. DEVELOPMENT OF ADMM-MCP

In (13), we discuss to use M RBF centers, $\{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_M\}$, to construct a network. However, we do not discuss the way to create them yet. Suppose that we use all the training input

vectors, $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$, as the RBF centers. The expression of the training objective does not change and is still given by

$$\psi(\boldsymbol{w}) = \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{w}\|_2^2 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{w}. \tag{18}$$

However, the definitions of $\boldsymbol{w}$ and $\boldsymbol{R}$ are changed to

$$\boldsymbol{w} = [w_1, \cdots, w_N]^{\mathrm{T}}, \tag{19a}$$

$$[\boldsymbol{A}]_{i,j} = a_j(\boldsymbol{x}_i) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{s}\right), \tag{19b}$$

where $i = 1, \cdots, N$, $j = 1, \cdots, N$, and $\boldsymbol{A}$ is an $N \times N$ matrix.

In the rest of this section, we will develop the ADMM-MCP algorithm, in which we pack an approximated ℓ_0 term, namely MCP term, into the objective function stated in (18). The packed MCP term has an ability to force some RBF weights to zero. Hence during training, the center selection process is achieved automatically.

A. OBJECTIVE FUNCTION AND ADMM FORMULATION

We introduce an additional ℓ_0 penalty term into (13), given by

$$Q_{\ell_0}(\boldsymbol{w}, \boldsymbol{\lambda}) = \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{w}\|_2^2 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{w} + \lambda\|\boldsymbol{w}\|_0, \tag{20}$$

where $\|\boldsymbol{w}\|_0$ is the ℓ_0-norm of the weight vector and it represents the number of nonzero entries in the vector $\boldsymbol{w}$. Parameter λ is a regularization parameter that controls the number of RBF nodes in the resultant network. Strictly speaking, the ℓ_0-norm is not a norm. Due to the nature of the ℓ_0-norm term, the problem stated in (20) is NP hard [34].

Inspired by [27], [28], the MCP function is a very attractive approximation of the ℓ_0-norm. Hence, we modify the objective function stated in (20) as

$$Q_{mcp}(\boldsymbol{w}, \boldsymbol{\lambda}) = \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{w}\|_2^2 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{w} + P_{\lambda,\gamma}(\boldsymbol{w}), \tag{21}$$

where $P_{\lambda,\gamma}(\boldsymbol{w}) = \sum_{i=1}^{M} P_{\lambda,\gamma}(w_i)$ ($\lambda > 0, \gamma > 0$) denotes the MCP function, given by

$$P_{\lambda,\gamma}(w_i) = \begin{cases} \lambda|w_i| - \dfrac{w_i^2}{2\gamma}, & \text{if } |w_i| \le \gamma\lambda, \\ \dfrac{1}{2}\gamma\lambda^2, & \text{if } |w_i| > \gamma\lambda. \end{cases} \tag{22}$$

The shape of the MCP penalty function with various settings is shown in Figure 1. Although the form of Q_{mcp} has better property, we do not have a closed form solution for minimizing $Q_{mcp}(\boldsymbol{w}, \boldsymbol{\lambda})$ directly.

We use the ADMM framework to minimize the objective function $Q_{mcp}(\boldsymbol{w}, \boldsymbol{\lambda})$. Firstly, we introduce a dummy variable vector $\boldsymbol{u} = [u_1, \ldots, u_N]^{\mathrm{T}}$ and transform the unconstrained problem, stated in (21), into the standard ADMM form, given by

$$\min_{\boldsymbol{w},\boldsymbol{u}} \ \psi(\boldsymbol{w}) + P_{\lambda,\gamma}(\boldsymbol{u}), \tag{23a}$$

$$s.t.\ \boldsymbol{u} = \boldsymbol{w}, \tag{23b}$$

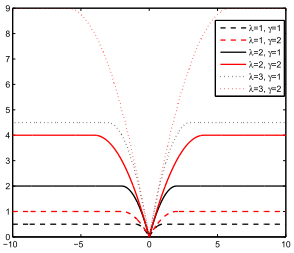


FIGURE 1. The shapes of MCP penalty function under various parameter settings.

where

$$\psi(\boldsymbol{w}) = \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{w}\|_2^2 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{w}. \tag{24}$$

We then construct the augmented Lagrangian as

$$\begin{aligned} L(\boldsymbol{w}, \boldsymbol{u}, \boldsymbol{v}) &= Q_{mcp}(\boldsymbol{w}, \boldsymbol{\lambda}) + \boldsymbol{v}^{\mathrm{T}}(\boldsymbol{u} - \boldsymbol{w}) + \frac{\rho}{2}\|\boldsymbol{w} - \boldsymbol{u}\|_2^2, \\ &= \psi(\boldsymbol{w}) + P_{\lambda,\gamma}(\boldsymbol{u}) + \boldsymbol{v}^{\mathrm{T}}(\boldsymbol{u} - \boldsymbol{w}) \\ &\quad + \frac{\rho}{2}\|\boldsymbol{w} - \boldsymbol{u}\|_2^2. \end{aligned} \tag{25}$$

B. UPDATING EQUATIONS

- According to (17a), the ADMM iteration of $\boldsymbol{u}^{k+1}$ is

$$\begin{aligned} \boldsymbol{u}^{k+1} &= \arg\min_{\boldsymbol{u}} L(\boldsymbol{w}^k, \boldsymbol{u}, \boldsymbol{v}^k), \\ &= \arg\min_{\boldsymbol{u}} P_{\lambda,\gamma}(\boldsymbol{u}) + \boldsymbol{v}^{k\,\mathrm{T}}(\boldsymbol{u} - \boldsymbol{w}^k) + \frac{\rho}{2}\left\|\boldsymbol{w}^k - \boldsymbol{u}\right\|_2^2 \\ &= \arg\min_{\boldsymbol{u}} P_{\lambda,\gamma}(\boldsymbol{u}) + \frac{\rho}{2}\left\|\boldsymbol{w}^k - \boldsymbol{u} - \frac{1}{\rho}\boldsymbol{v}^k\right\|_2^2 \end{aligned} \tag{26}$$

  where $\boldsymbol{u}^{k+1} = [u_1^{k+1}, \ldots, u_N^{k+1}]^{\mathrm{T}}$.

  – For (26), when $\rho\gamma > 1$, the closed form solution [27], [28] is given by

$$u_i^{k+1} = \begin{cases} \dfrac{\mathcal{S}\left(w_i^k - v_i^k/\rho, \lambda\right)}{1 - 1/\gamma}, & \text{if } |w_i^k - v_i^k/\rho| \le \gamma\lambda, \\ w_i^k - v_i^k/\rho, & \text{if } |w_i^k - v_i^k/\rho| > \gamma\lambda, \end{cases} \tag{27}$$

  for $i = 1, \cdots, N$, where $\mathcal{S}$ denotes the soft-threshold operator [35],

$$\mathcal{S}(z, \lambda) = \mathrm{sign}(z)\max\{|z| - \lambda, 0\}. \tag{28}$$

  It is worth noting that when $\gamma \to \infty$, $\mathcal{S}(\cdot, \cdot)$ is a soft-threshold function. When $\gamma \to 1$, $\mathcal{S}(\cdot, \cdot)$ is a hard-threshold function.

  – When $\rho\gamma = 1$, the closed form solution of (26) is given by

$$u_i^{k+1} = \begin{cases} 0, & \text{if } |w_i^k - v_i^k/\rho| \le \gamma\lambda, \\ w_i^k - v_i^k/\rho, & \text{if } |w_i^k - v_i^k/\rho| > \gamma\lambda. \end{cases} \tag{29}$$

– When $\rho\gamma < 1$, the closed form solution of (26) is given by

$$u_i^{k+1} = \begin{cases} 0, & \text{if } |w_i^k - \upsilon_i^k/\rho| \le \sqrt{\gamma/\rho}\lambda, \\ w_i^k - \upsilon_i^k/\rho, & \text{if } |w_i^k - \upsilon_i^k/\rho| > \sqrt{\gamma/\rho}\lambda. \end{cases} \tag{30}$$

- According to (17b), the ADMM iteration of $\boldsymbol{w}^{k+1}$ is given by

$$\begin{aligned} \boldsymbol{w}^{k+1} &= \arg\min_{\boldsymbol{w}} L(\boldsymbol{w}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^k) \\ &= \arg\min_{\boldsymbol{w}} \psi(\boldsymbol{w}) + P_{\lambda,\gamma}(\boldsymbol{u}^{k+1}) + \boldsymbol{\upsilon}^{k\mathrm{T}}(\boldsymbol{u}^{k+1} - \boldsymbol{w}) \\ &\quad + \frac{\rho}{2}\left\|\boldsymbol{w} - \boldsymbol{u}^{k+1}\right\|_2^2 \\ &= \arg\min_{\boldsymbol{w}} \psi(\boldsymbol{w}) + \frac{\rho}{2}\left\|\boldsymbol{w} - \boldsymbol{u}^{k+1} - \frac{1}{\rho}\boldsymbol{\upsilon}^k\right\|_2^2 \\ &= \left[\frac{2}{N}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + 2\boldsymbol{R} + \rho\mathbf{I}\right]^{-1}\left[\frac{2}{N}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} + \rho\boldsymbol{u}^{k+1} + \boldsymbol{\upsilon}^k\right]. \end{aligned} \tag{31}$$

- From (17c), $\boldsymbol{\upsilon}^{k+1}$ is updated as

$$\boldsymbol{\upsilon}^{k+1} = \boldsymbol{\upsilon}^k + \rho\left(\boldsymbol{u}^{k+1} - \boldsymbol{w}^{k+1}\right). \tag{32}$$

IV. ANALYSIS OF CONVERGENCE

This section presents the convergent properties of the ADMM-MCP algorithm. We use the general convergence result of nonconvex ADMM given by [36]. When we use the general convergence proof of nonconvex ADMM, we need to show that our algorithm satisfies three conditions. The general convergence result is given by Theorem 1 [36].

Theorem 1: If an ADMM based algorithm satisfies the following three conditions stated below, then the sequence $\{\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{\upsilon}^k\}$ *has at least a limit point* $\{\boldsymbol{w}^*, \boldsymbol{u}^*, \boldsymbol{\upsilon}^*\}$ *and any limit point* $\{\boldsymbol{w}^*, \boldsymbol{u}^*, \boldsymbol{\upsilon}^*\}$ *is a stationary point of the Lagrangian function.*

C1 *(Sufficient decrease condition) For each k, there exists a* $\tau_1 > 0$ *such that*

$$L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^{k+1}) - L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{\upsilon}^k) \le -\tau_1\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^k\|_2^2. \tag{33}$$

C2 *(Boundness condition) The sequence* $\{\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{\upsilon}^k\}$ *is bounded and its Lagrangian function is lower bounded.*

C3 *(Subgradient bound condition) For each* $k \in \mathbb{N}$*, there exists a* $\boldsymbol{d}^{k+1} \in \partial L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^{k+1})$*, and* $\tau_2 > 0$ *such that*

$$\|\boldsymbol{d}^{k+1}\|_2^2 \le \tau_2\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^k\|_2^2. \tag{34}$$

Proof: The proof of the theorem is in Proposition 2 in [36] and Theorem 2.9 in [37]. ■

In the rest of this section, we will show that the proposed algorithm satisfies the three conditions of Theorem 1.

Proposition 1: If ρ *is greater than a certain value, the ADMM-MCP algorithm satisfies the sufficient decrease condition in* ***C1****.*

Proof: The Lagrangian function can be rewritten as

$$L(\boldsymbol{w}, \boldsymbol{u}, \boldsymbol{\upsilon}) = \psi(\boldsymbol{w}) + \frac{\rho}{2}\left\|\boldsymbol{w} - \boldsymbol{u} - \frac{1}{\rho}\boldsymbol{\upsilon}\right\|_2^2 + P_{\lambda,\gamma}(\boldsymbol{u}) - \frac{1}{2\rho}\|\boldsymbol{\upsilon}\|_2^2. \tag{35}$$

Note that $\psi(\boldsymbol{w}) = \frac{1}{N}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{w}\|_2^2 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{w}$.

Since $\psi(\boldsymbol{w})$ is strongly convex, we can deduce that (35) is also strongly convex with respect to $\boldsymbol{w}$. Hence, based on the definition of strongly convex function, we have the relationship between $L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^k)$ and $L(\boldsymbol{w}^k, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^k)$, given by

$$L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^k) - L(\boldsymbol{w}^k, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^k) \le -\frac{a}{2}\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^k\|_2^2, \tag{36}$$

where $a > 0$.

Since $L(\boldsymbol{w}, \boldsymbol{u}, \boldsymbol{\upsilon})$ is a strictly convex function with respect to $\boldsymbol{w}$, we have

$$\nabla_{\boldsymbol{w}}\psi(\boldsymbol{w}^{k+1}) - \boldsymbol{\upsilon}^k + \rho(\boldsymbol{w}^{k+1} - \boldsymbol{u}^{k+1}) = 0. \tag{37}$$

From (32) and (37), we can deduce that

$$\nabla\psi(\boldsymbol{w}^{k+1}) = \boldsymbol{\upsilon}^{k+1} \tag{38}$$

$$\boldsymbol{u}^{k+1} - \boldsymbol{w}^{k+1} = 1/\rho\left(\boldsymbol{\upsilon}^{k+1} - \boldsymbol{\upsilon}^k\right). \tag{39}$$

Thus we have the relationship between $L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^{k+1})$ and $L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, , \boldsymbol{\upsilon}^k))$, given by

$$\begin{aligned} &L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^{k+1}) - L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^k) \\ &= \left(\boldsymbol{\upsilon}^{k+1} - \boldsymbol{\upsilon}^k\right)^{\mathrm{T}}\left(\boldsymbol{u}^{k+1} - \boldsymbol{w}^{k+1}\right) \\ &= \frac{1}{\rho}\|\boldsymbol{\upsilon}^{k+1} - \boldsymbol{\upsilon}^k\|_2^2 \\ &= \frac{1}{\rho}\|\nabla\psi(\boldsymbol{w}^{k+1}) - \nabla\psi(\boldsymbol{w}^k)\|_2^2 \\ &\le \frac{l_\psi^2}{\rho}\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^k\|_2^2, \end{aligned} \tag{40}$$

where l_ψ is a Lipschitz constant of function $\psi(\boldsymbol{w})$, and the last inequality is from the fact that $\psi(\boldsymbol{w})$ has Lipschitz continuous gradient.

Since $\boldsymbol{u}^{k+1}$ (as stated in (27), (29) and (30)) is the optimal solution of (26), we have

$$L(\boldsymbol{w}^k, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^k) - L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{\upsilon}^k) \le 0. \tag{41}$$

Combining (36), (40) and (41), we have

$$\begin{aligned} &L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^{k+1}) - L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{\upsilon}^k) \\ &= L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^{k+1}) - L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^k) \\ &\quad + L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^k) - L(\boldsymbol{w}^k, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^k) \\ &\quad + L(\boldsymbol{w}^k, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^k) - L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{\upsilon}^k) \\ &\le \left(\frac{l_\psi^2}{\rho} - \frac{a}{2}\right)\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^k\|_2^2. \end{aligned} \tag{42}$$

To ensure $l_\psi^2/\rho - a/2 < 0$, we need $\rho > 2l_\psi^2/a$. Hence the $\tau_1 = a/2 - l_\psi^2/\rho$ in **C1**. The proof for **C1** is completed. ■

Now, we show that the proposed algorithm satisfies **C2** in Theorem 1.

Proposition 2: If $\rho \geq l_\psi$, then $L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k)$ is bounded for all k, and $L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k)$ converges, as $k \to \infty$. In addition, the sequence $\{\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k\}$ is bounded.

Proof: The proof consists of two parts. The first one is that $L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k)$ is bounded. The second one is that the sequence $\{\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k\}$ is bounded.

The proof for $L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k)$ being bounded:
First, we prove that $L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k)$ is lower bounded for all k. From (38), $\nabla\psi(\boldsymbol{w}^k) = \boldsymbol{v}^k$. Thus

$$\begin{aligned} L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k) &= \psi(\boldsymbol{w}^k) + P_{\lambda,\gamma}(\boldsymbol{u}^k) + {\boldsymbol{v}^k}^{\mathrm{T}}(\boldsymbol{u}^k - \boldsymbol{w}^k) \\ &\quad + \frac{\rho}{2}\left\|\boldsymbol{w}^k - \boldsymbol{u}^k\right\|_2^2, \\ &= \psi(\boldsymbol{w}^k) + P_{\lambda,\gamma}(\boldsymbol{u}^k) + \nabla_{\boldsymbol{w}}\psi(\boldsymbol{w}^k)^{\mathrm{T}}(\boldsymbol{u}^k - \boldsymbol{w}^k) \\ &\quad + \frac{\rho}{2}\left\|\boldsymbol{w}^k - \boldsymbol{u}^k\right\|_2^2. \end{aligned} \tag{43}$$

From Lemma 3.1 in [37] and the Lipschitz continuous gradient of $\psi(\boldsymbol{w})$,

$$\psi(\boldsymbol{w}^k) + \nabla_{\boldsymbol{w}}\psi(\boldsymbol{w}^k)^{\mathrm{T}}(\boldsymbol{u}^k - \boldsymbol{w}^k) \geq \psi(\boldsymbol{u}^k) - \frac{l_\psi}{2}\|\boldsymbol{u}^k - \boldsymbol{w}^k\|_2^2. \tag{44}$$

Hence, we have

$$\begin{aligned} L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k) &\geq \psi(\boldsymbol{u}^k) + P_{\lambda,\gamma}(\boldsymbol{u}^k) \\ &\quad + \left(\frac{\rho}{2} - \frac{l_\psi}{2}\right)\|\boldsymbol{u}^k - \boldsymbol{w}^k\|_2^2. \end{aligned} \tag{45}$$

Obviously, if $\rho \geq l_\psi$, then the right hand side of (45) is greater than $-\infty$. Hence $L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k)$ is lower bounded. According to the proof of Proposition 1, we know that $L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k)$ is sufficient descent. Hence $L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k)$ is upper bounded by $L(\boldsymbol{w}^0, \boldsymbol{u}^0, \boldsymbol{v}^0)$.

The proof for $\{\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k\}$ being bounded:
Next, we prove that the sequence $\{\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k\}$ is bounded. From (42), we have

$$L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{v}^{k+1}) - L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k) \leq -\tau_1\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^k\|_2^2, \tag{46}$$

where $\tau_1 > 0$. Hence, we have

$$\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^k\|_2^2 \leq \frac{1}{\tau_1}\left(L(\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k) - L(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{v}^{k+1})\right). \tag{47}$$

Then we can deduce that

$$\begin{aligned} &\sum_{k=1}^{l}\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^k\|_2^2 \\ &\quad \leq \frac{1}{\tau_1}\left(L(\boldsymbol{w}^0, \boldsymbol{u}^0, \boldsymbol{v}^0) - L(\boldsymbol{w}^{l+1}, \boldsymbol{u}^{l+1}, \boldsymbol{v}^{l+1})\right) \\ &\quad < \infty. \end{aligned} \tag{48}$$

Even $l \to \infty$, we still have $\sum_{k=1}^{\infty}\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^k\|_2^2 < \infty$. Thus $\{\boldsymbol{w}^k\}$ is bounded.

From (40), we know

$$\|\boldsymbol{v}^{k+1} - \boldsymbol{v}^k\|_2^2 \leq l_\psi^2\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^k\|_2^2.$$

Therefore, we can deduce that

$$\sum_{i=1}^{\infty}\|\boldsymbol{v}^{k+1} - \boldsymbol{v}^k\|_2^2 < \infty. \tag{49}$$

That means, $\{\boldsymbol{v}^k\}$ is bounded.

In addition, according to (32), we have

$$\begin{aligned} &\|\boldsymbol{u}^{k+1} - \boldsymbol{u}^k\|_2^2 \\ &= \|\boldsymbol{w}^{k+1} - \boldsymbol{w}^k + \frac{1}{\rho}\left(\boldsymbol{v}^{k+1} - \boldsymbol{v}^k\right) + \frac{1}{\rho}\left(\boldsymbol{v}^{k-1} - \boldsymbol{v}^k\right)\|_2^2 \\ &\leq 2\|\boldsymbol{w}^{k+1} - \boldsymbol{w}^k\|_2^2 + \frac{2}{\rho^2}\|\boldsymbol{v}^{k+1} - \boldsymbol{v}^k\|_2^2 \\ &\quad + \frac{2}{\rho^2}\|\boldsymbol{v}^{k-1} - \boldsymbol{v}^k\|_2^2. \end{aligned} \tag{50}$$

Thus, we have

$$\sum_{i=1}^{\infty}\|\boldsymbol{u}^{k+1} - \boldsymbol{u}^k\|_2^2 < \infty. \tag{51}$$

That means, $\{\boldsymbol{u}^k\}$ is also bounded. To sum up, the sequence $\{\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{v}^k\}$ is bounded. The proof is completed. ■

*Proposition 3: The proposed ADMM-MCP algorithm satisfies the subgradient bound condition given by **C3**.*

Proof: The proof involves the derivations of three gradients. They are the gradient $\partial L/\partial \boldsymbol{w}$ of L with respect to $\boldsymbol{w}$, the limiting subgradient $\partial_{\boldsymbol{u}} L$ of L with respect to $\boldsymbol{u}$, and the subgradient $\partial_{\boldsymbol{v}} L$ of L with respect to $\boldsymbol{v}$.

For $\partial L/\partial \boldsymbol{w}$, we have

$$\begin{aligned} &\left.\frac{\partial L}{\partial \boldsymbol{w}}\right|_{(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{v}^{k+1})} \\ &\quad = \nabla_{\boldsymbol{w}}\psi(\boldsymbol{w}^{k+1}) + \rho\left(\boldsymbol{w}^{k+1} - \boldsymbol{u}^{k+1}\right) - \boldsymbol{v}^{k+1}. \end{aligned} \tag{52}$$

Since $\boldsymbol{w}^{k+1}$ is the optimal solution of $L(\boldsymbol{w}, \boldsymbol{u}^{k+1}, \boldsymbol{v}^k)$, we have

$$\nabla_{\boldsymbol{w}}\psi(\boldsymbol{w}^{k+1}) + \rho\left(\boldsymbol{w}^{k+1} - \boldsymbol{u}^{k+1}\right) - \boldsymbol{v}^k = 0. \tag{53}$$

From (52) and (53), we have

$$\left.\frac{\partial L}{\partial \boldsymbol{w}}\right|_{(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{v}^{k+1})} = \boldsymbol{v}^k - \boldsymbol{v}^{k+1}. \tag{54}$$

For the limiting subgradient $\partial_{\boldsymbol{u}} L$, we have

$$\begin{aligned} &\partial_{\boldsymbol{u}} L|_{(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{v}^{k+1})} \\ &\quad = \partial_{\boldsymbol{u}} P_{\lambda,\gamma}(\boldsymbol{u}^{k+1}) + \boldsymbol{v}^{k+1} - \rho\left(\boldsymbol{w}^{k+1} - \boldsymbol{u}^{k+1}\right), \end{aligned} \tag{55}$$

Since $\boldsymbol{u}^{k+1}$ is the optimal solution of $L(\boldsymbol{w}^k, \boldsymbol{u}, \boldsymbol{v}^k)$, we have

$$0 \in \partial_{\boldsymbol{u}} P_{\lambda,\gamma}(\boldsymbol{u}^{k+1}) + \boldsymbol{v}^k - \rho\left(\boldsymbol{w}^k - \boldsymbol{u}^{k+1}\right). \tag{56}$$

From (55) and (56), we have

$$\rho\left(\boldsymbol{w}^k - \boldsymbol{w}^{k+1}\right) + \boldsymbol{v}^{k+1} - \boldsymbol{v}^k \in \partial_{\boldsymbol{u}} L|_{(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{v}^{k+1})} \tag{57}$$

TABLE 1. Properties of the eight data sets.

Dataset	Number of features	Size of training set	Size of test set	RBF width
Abalone	7	2000	2177	0.1
ASN	5	751	752	0.5
Housing	13	400	106	2
Concrete	9	500	530	0.5
Energy	7	600	168	0.5
WQW	12	2000	2898	1
MG	6	700	685	1
Space-ga	6	1600	1507	0.1

For $\partial_{\boldsymbol{\upsilon}} L$, from (25) and (39), we have

$$\partial_{\boldsymbol{\upsilon}} L|_{(\boldsymbol{w}^{k+1},\boldsymbol{u}^{k+1},\boldsymbol{\upsilon}^{k+1})} = \boldsymbol{u}^{k+1} - \boldsymbol{w}^{k+1} = \frac{1}{\rho}\left(\boldsymbol{\upsilon}^{k+1} - \boldsymbol{\upsilon}^{k}\right). \quad (58)$$

Define

$$\boldsymbol{d}^{k+1} := \begin{bmatrix} \boldsymbol{\upsilon}^{k} - \boldsymbol{\upsilon}^{k+1} \\ \rho\left(\boldsymbol{w}^{k} - \boldsymbol{w}^{k+1}\right) + \boldsymbol{\upsilon}^{k+1} - \boldsymbol{\upsilon}^{k} \\ \frac{1}{\rho}\left(\boldsymbol{\upsilon}^{k+1} - \boldsymbol{\upsilon}^{k}\right) \end{bmatrix}. \quad (59)$$

Hence, from (54), (57), and (58),

$$\boldsymbol{d}^{k+1} \in \partial L\left(\boldsymbol{w}^{k+1}, \boldsymbol{u}^{k+1}, \boldsymbol{\upsilon}^{k+1}\right). \quad (60)$$

From the inequality stated in (40), we can deduce that

$$\|\boldsymbol{d}^{k+1}\|_2^2 \le \tau_2 \|\boldsymbol{w}^{k+1} - \boldsymbol{w}^{k}\|_2^2, \quad (61)$$

where $\tau_2 > 0$. The proof is completed. ■

Since the ADMM-MCP algorithm satisfies the three conditions of Theorem 1, the sequence $\{\boldsymbol{w}^k, \boldsymbol{u}^k, \boldsymbol{\upsilon}^k\}$ has at least one limit point $\{\boldsymbol{w}^*, \boldsymbol{u}^*, \boldsymbol{\upsilon}^*\}$ and any limit point $\{\boldsymbol{w}^*, \boldsymbol{u}^*, \boldsymbol{\upsilon}^*\}$ is a stationary point. In other words, at least, the ADMM-MCP algorithm has the local convergent property.

V. SIMULATION RESULT

A. SETTINGS

This paper considers eight datasets. Six of them are from the University of California Irvine (UCI) machine learning repository [38]. They are respectively Abalone [39], [40], Airfoil Self-Noise (ASN) [41], Boston Housing (Housing) [39], [41], Concrete [42], Energy Efficiency (Energy) [41], and Wine Quality White (WQW) [43], [44]. The other two datasets are the Space-ga [45] and Mackey-Glass (MG) system [46] datasets. For each dataset, its RBF width is selected between 0.1 to 10. Table 1 summarizes the properties of these datasets.

The performances of the resultant networks are evaluated by the average test set MSE, given by

$$\overline{\mathcal{E}}_{test} = \frac{P_\beta}{N'}\sum_{i'=1}^{N'} y_{i'}'^2 + \frac{1-P_\beta}{N'}\left\|\boldsymbol{y}' - \boldsymbol{A}'\boldsymbol{w}\right\|_2^2 + \frac{1-P_\beta}{N'}\boldsymbol{w}^{\mathrm{T}}\left[(P_\beta + \sigma_b^2)\mathrm{diag}\left(\boldsymbol{A}'^{\mathrm{T}}\boldsymbol{A}'\right) - P_\beta \boldsymbol{A}'^{\mathrm{T}}\boldsymbol{A}'\right]\boldsymbol{w}, \quad (62)$$

where $\{(\boldsymbol{x}'_{i'}, y'_{i'}), i' = 1, \cdots, N'\}$ is the test set, N' is the number of samples in the test set, $\boldsymbol{y}' = [y'_1, \ldots, y'_{N'}]$, $\mathbf{A}'$ is an $N' \times M$ matrix, and its element in i'th row and jth column is given by

$$[\mathbf{A}']_{i',j} = \exp\left(-\frac{\left\|\boldsymbol{x}'_{i'} - \boldsymbol{c}_j\right\|_2^2}{s}\right). \quad (63)$$

The two parameters, P_β and σ_b^2, describe the failure levels for open weight fault and multiplicative weight noise, respectively. We consider three fault scenarios: $\{P_\beta = \sigma_b^2 = 0.005\}$, $\{P_\beta = \sigma_b^2 = 0.01\}$, and $\{P_\beta = \sigma_b^2 = 0.05\}$. For the ADMM-MCP algorithm, we set $\gamma = 1.001$ and $\rho = 0.1$. The parameter λ is used to control the number of nodes.

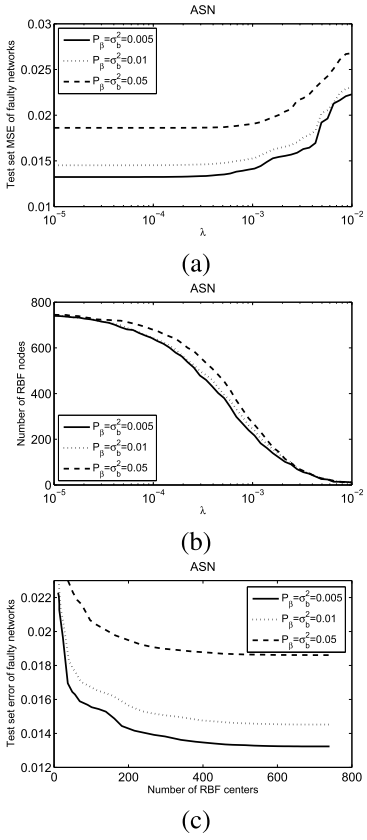


FIGURE 2. Illustration of using λ to control the number of RBF nodes in the resultant network. (a) MSE versus λ. (b) number of nodes versus λ. (c) Combining (a) and (b), we obtain MSE versus the number of RBF nodes.

B. MSE VERSUS THE NUMBER OF HIDDEN NODES

We use the ASN data set to illustrate the way to control the number of nodes. We can vary the value of λ to control the number of RBF nodes. Figure 2(a)-(b) show test set MSE versus λ and number of nodes versus λ. Combining Figure 2(a) and (b), we obtain MSE versus the number of hidden nodes, as shown in Figure 2(c). Unlike the common algorithms that provide U-shaped test set MSE curves, the proposed algorithm provides MSE curves with nearly monotonic decreasing behaviour respect to the test set MSE of faulty networks. We can observe that increasing the number of RBF nodes leads to the decrease of test set MSE of

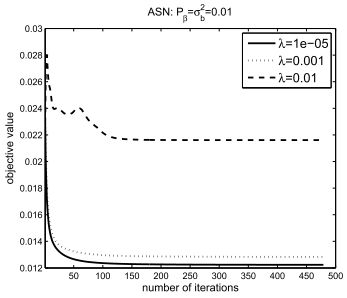


FIGURE 3. Convergent behaviors of the proposed method.

faulty networks. This is because the additional term, $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{w}$ where $\boldsymbol{R} = (P_\beta+\sigma_b^2)\mathrm{diag}\left(\frac{1}{N}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right) - \frac{P_\beta}{N}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$, not only makes the trained network to tolerate the multiplicative noise and node failure, but also has the ability to avoid overfitting. For other datasets and settings, their MSE curves have the similar behaviour.

C. CONVERGENCE

Here we use the ASN dataset with $\{P_\beta = \sigma_b^2 = 0.01\}$ as an example to intuitively demonstrate the convergence. The result is shown in Figure 3 which shows the objective value $\psi(\boldsymbol{w}) = \frac{1}{N}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{w}\|_2^2 + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{w}$ versus the number of iterations. We can see that within 100 to 200 iterations, the training objective value nearly settles down. If we increase the value of λ, then the algorithm converges to a larger objective value. It is because increasing λ leads to a restriction on the approximation ability of the resultant networks and the resultant networks have larger objective value $\psi(\boldsymbol{w})$. For all other datasets and other settings, they have similar properties of convergence.

D. COMPARISON ALGORITHMS

We compare our proposed algorithm with six other algorithms. They are, respectively, the fault tolerant OLS algorithm (OLS) [19], the fault tolerant l_1-norm approach (ADMM-l_1) [26], the l_1-norm regularization approach (l_1-reg.) [39], the support vector regression algorithm SVR [39], the orthogonal forward regression algorithm (OFR) [47] and the Homotopy method (HOM) [48]. **Our aim is to show that our proposed algorithm has better RBF center selection capability. We will show that when we do not use all the training input vectors as the RBF centers, the performances of our proposed algorithm are better than those of the six comparison algorithms.**

The fault tolerant OLS algorithm and the fault tolerant ADMM-l_1-norm algorithm have fault tolerant ability. The fault tolerant OLS algorithm includes two stages. In the first one, it uses the OLS method to generate a sorted list of RBF nodes. In the second stage, it constructs a fault tolerant RBF network with desired number of nodes. The fault tolerant ADMM-l_1 approach is our previous work based on an l_1-norm regularizer.

TABLE 2. Settings of the tuning parameters in the SVR algorithm.

Dataset	Parameters
Abalone	$C = \{0.01, 0.03, 0.06, 0.1, 0.3, 0.6, 1\}$, $\epsilon = \{1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6\}$
ASN	$C = \{0.005, 0.01, 0.03, 0.05, 0.1, 0.3, 0.5\}$, $\epsilon = \{0.01, 0.05, 0.1, 0.15, 0.175, 0.2, 0.25, 0.3, 0.35, 0.4\}$
Housing	$C = \{0.01, 0.02, 0.04, 0.08, 0.2, 0.4, 0.8\}$, $\epsilon = \{0.01, 0.02, 0.04, 0.08, 0.2, 0.4, 0.8\}$
Concrete	$C = \{0.01, 0.03, 0.06, 0.1, 0.3, 0.6, 1\}$, $\epsilon = \{0.01, 0.03, 0.06, 0.1, 0.3, 0.6, 1\}$
Energy	$C = \{0.005, 0.01, 0.05, 0.1, 0.3, 0.5\}$, $\epsilon = \{0.01, 0.05, 0.1, 0.125, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$
WQW	$C = \{0.001, 0.005, 0.01, 0.05, 0.1, 0.2\}$, $\epsilon = \{0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.2\}$
MG	$C = \{0.002, 0.003, 0.02, 0.03, 0.04, 0.05, 0.06\}$, $\epsilon = \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$
Space-ga	$C = \{0.01, 0.02, 0.03, 0.04, 0.05, 0.3, 0.4\}$, $\epsilon = \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$

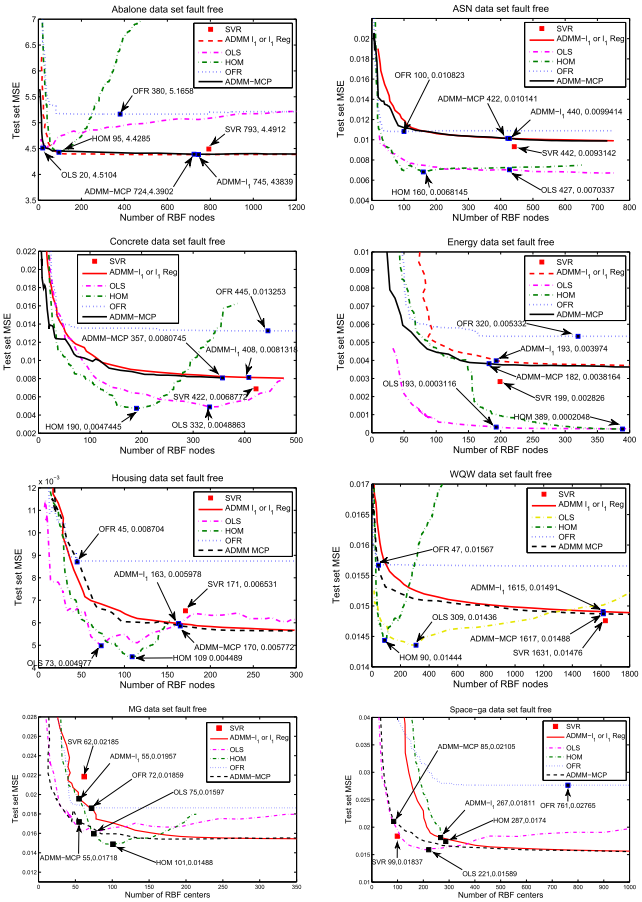


FIGURE 4. Performances of various algorithms under the fault-free situation.

The l_1-norm regularization approach [39] considers the original MSE training objective and uses an l_1-norm regularizer to control the number of RBF nodes. Its fault tolerant ability is inadequate. Especially, when the fault level is high. The SVR algorithm [39] is able to train the RBF network and to select the centers simultaneously. It uses two parameters C and ϵ to control the training process. Table 2 shows the parameter settings for different datasets. The SVR algorithm has certain fault tolerant ability. It is because the parameter C

TABLE 3. Average test MSE over 20 trials under the fault-free situation.

Dataset	ADMM-MCP		ADMM-l_1		OLS		l_1-reg.		SVR		HOM		OFR	
	AVG MSE	AVG no. of nodes	AVG MSE	AVG no. of nodes	AVG MSE	AVG no. of nodes	AVG MSE	AVG no. of nodes	AVG MSE	AVG no. of nodes	AVG MSE	AVG no. of nodes	AVG MSE	AVG no. of nodes
Abalone	4.5789	730.2	**4.5697**	726.4	4.6405	40.8	**4.5697**	726.4	4.7486	777.4	4.5920	95.6	5.2282	673.7
ASN	0.01100	409.4	0.01096	401.0	**0.00736**	401.0	0.01096	401.0	0.01020	418.1	**0.00667**	200.2	0.01275	319.1
Housing	0.00745	135.3	0.00746	134.2	**0.00688**	129.7	0.00746	134.2	0.00782	141.4	**0.00567**	104.9	0.01248	55.1
Concrete	0.00848	327.0	0.00860	351.6	**0.00660**	203.4	0.00860	351.6	0.00719	364.5	**0.00632**	179.6	0.01329	250.7
Energy	0.00453	328.5	0.00459	324.5	0.00356	324.5	0.00459	324.5	**0.00340**	339.7	**0.00293**	380.3	0.00549	190.2
WQW	0.01473	1490.0	0.01476	1460.0	**0.01424**	338.0	0.01480	1468.0	0.01471	1514.3	**0.01417**	146.8	0.01505	563.5
MG	0.01762	54.9	0.01933	57.2	**0.01576**	78.2	0.01933	57.2	0.02138	62.0	**0.01523**	110.5	0.01838	345.2
Space-ga	0.02103	103.5	0.01947	249.6	**0.01593**	308.1	0.01947	249.6	0.01975	110.6	**0.01713**	307.1	0.02831	946.4

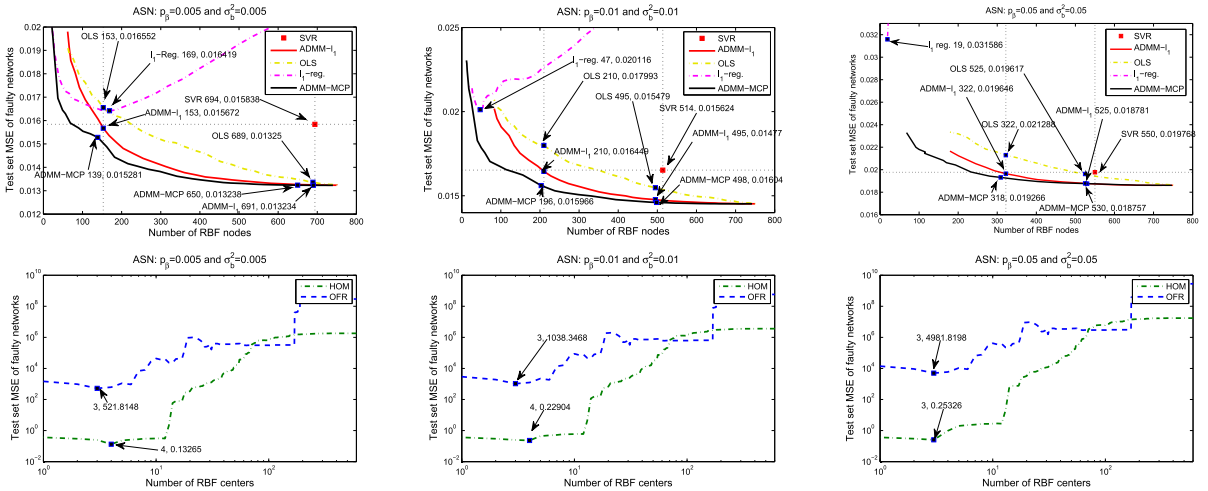


FIGURE 5. Performances of various algorithms under the faulty situation.

is capable to limit the magnitudes of the trained weights. The parameter ϵ is used to control its approximation ability. However, the main drawback of SVR algorithm is that there is no simple way to find an appropriate pair of C and ϵ. In our experiment, we use trial-and-error method to determine them.

The Homotopy method [48] is an incremental learning method. It has an l_1-norm regularization term, and it can tune its regularization parameter automatically. The OFR algorithm [47] is an incremental learning method too. It chooses one RBF center at a time with the orthogonal forward regression procedure. For OFR, an l_2-norm regularization term is used. It can also tune the regularization parameter automatically during training process.

In the following two experiments, the simulation was ran 20 times. In each trial, the samples of dataset were randomly split for training and testing set.

E. COMPARISON:FAULT FREE CASE

This subsection investigates the performances of various algorithms under **the fault-free situation.** It gives us a baseline of the performances under the fault-free situation. Some typical examples are given by Figure 4. In the fault-free case, the performances of the fault tolerant l_1-norm approach and the l_1-norm regularization approach are substantially same with each other. For OLS, HOM, OFR and SVR, we select their minimum MSE and the corresponding number of nodes to represent their performances. For other algorithms, we use the points where the number of nodes is close to the best result of SVR to represent their performances.

Table 3 shows the average test set error over the 20 trials. From the table and Figure 4, it can be observed that, under the fault-free environment, the performances of OLS and HOM are better than other algorithms.

F. COMPARISON:FAULTY CASE

This subsection compares the proposed algorithm with the aforementioned algorithms under the concurrent fault situation. We show that when we do not use all the training input vectors as the RBF centers, the performances of our proposed algorithm are better than those of the six comparison algorithms. Three fault levels are considered. They are $\{P_\beta = \sigma_b^2 = 0.005\}$, $\{P_\beta = \sigma_b^2 = 0.01\}$ and $\{P_\beta = \sigma_b^2 = 0.05\}$.

The typical result of one of the 20 trials of the ASN dataset under different fault levels is given by Figure 5. In the figure, the first, second and third columns show the test MSE results of faulty networks for $\{P_\beta = \sigma_b^2 = 0.005\}$, $\{P_\beta = \sigma_b^2 = 0.01\}$ and $\{P_\beta = \sigma_b^2 = 0.05\}$, respectively. Here we use **the first column** in Figure 5 to discuss the result.

- The performances of the HOM and OFR algorithms are very poor. Their minimum test set MSE values are equal to 0.13265 and 521.8148, respectively. They are much

TABLE 4. Average test MSE over 20 trials under the concurrent fault situation.

Dataset	Fault level	ADMM-MCP		ADMM-l_1		OLS		l_1-reg.		SVR		HOM		OFR	
		AVG MSE	AVG no. of nodes	AVG MSE	AVG no. of nodes	AVG MSE	AVG no. of nodes	AVG MSE	AVG no. of nodes	AVG MSE	AVG no. of nodes	AVG MSE	AVG no. of nodes	AVG MSE	AVG no. of nodes
Abalone	$P_\beta = \sigma_b^2 = 0.005$	**5.1617**	140.9	5.3611	156.2	5.3269	156.2	5.5938	28.3	5.388	767.0	51.3834	7.5	580584	3.1
	$P_\beta = \sigma_b^2 = 0.01$	**5.2587**	174.8	5.5254	202.2	5.4416	202.2	5.9977	25.1	5.5537	748.0	56.2897	5.4	1155290	3.1
	$P_\beta = \sigma_b^2 = 0.05$	**5.6169**	313.8	6.2372	356.8	5.8697	356.8	8.4599	19.2	6.2801	854.6	62.6329	3.9	5542897	3.1
ASN	$P_\beta = \sigma_b^2 = 0.005$	**0.01512**	142.4	0.01617	154.6	0.01810	154.6	0.01672	165.2	0.01641	467.2	0.08730	3.3	569.2	2.8
	$P_\beta = \sigma_b^2 = 0.01$	**0.01596**	193.9	0.01711	209.1	0.01928	209.1	0.02054	57.2	0.01731	609.3	0.09517	3.1	1132.7	2.8
	$P_\beta = \sigma_b^2 = 0.05$	**0.02006**	323.4	0.02086	347.2	0.02262	347.2	0.03078	20.0	0.02105	621.4	0.11130	3.0	5434.4	2.8
Housing	$P_\beta = \sigma_b^2 = 0.005$	**0.01527**	56.8	0.01583	61.9	0.01654	61.9	0.01798	12.9	0.01716	88.2	0.05489	4.7	3.721	2.4
	$P_\beta = \sigma_b^2 = 0.01$	**0.01656**	53.6	0.01743	60.9	0.01832	60.9	0.02053	10.6	0.01883	75.5	0.06362	4.0	7.362	2.4
	$P_\beta = \sigma_b^2 = 0.05$	**0.02135**	55.9	0.02343	62.0	0.02476	62.0	0.02999	6.9	0.02476	87.9	0.07864	3.4	35.160	2.4
Concrete	$P_\beta = \sigma_b^2 = 0.005$	**0.01218**	124.2	0.01334	131.5	0.01756	131.5	0.01258	138.5	0.01376	278.5	0.04479	13.1	89.61	4.1
	$P_\beta = \sigma_b^2 = 0.01$	**0.01387**	114.7	0.01489	122.6	0.01978	122.6	0.01483	94.1	0.01514	271.8	0.05112	11.3	178.25	4.1
	$P_\beta = \sigma_b^2 = 0.05$	**0.01724**	133.1	0.01910	140.0	0.02515	140.0	0.02358	32.3	0.01942	284.9	0.06605	8.9	855.10	4.1
Energy	$P_\beta = \sigma_b^2 = 0.005$	**0.00518**	150.2	0.00560	161.1	0.00568	161.1	0.00542	155.0	0.00566	293.1	0.02966	26.5	0.05854	9.3
	$P_\beta = \sigma_b^2 = 0.01$	**0.00560**	156.1	0.00610	167.0	0.00632	167.0	0.00591	159.0	0.00619	363.8	0.03859	21.8	0.06895	8.4
	$P_\beta = \sigma_b^2 = 0.05$	**0.00761**	222.2	0.00856	233.3	0.00849	233.3	0.01026	123.4	0.00870	211.1	0.06420	14.3	0.10650	4.8
WQW	$P_\beta = \sigma_b^2 = 0.005$	**0.01641**	106.6	0.01678	119.6	0.01732	119.6	0.01702	39.6	0.01685	1346.1	0.03350	5.2	535.7	2.2
	$P_\beta = \sigma_b^2 = 0.01$	**0.01660**	142.6	0.01701	169.2	0.01753	169.2	0.01788	29.6	0.01710	1461.0	0.03970	5.0	1065.9	2.2
	$P_\beta = \sigma_b^2 = 0.05$	**0.01729**	355.8	0.01763	385.6	0.01807	385.6	0.02419	22.6	0.01811	697.8	0.06621	4.4	5114.1	2.2
MG	$P_\beta = \sigma_b^2 = 0.005$	**0.01924**	151.7	0.02145	168.6	0.02041	168.6	0.02069	75.4	0.02160	62.0	0.05536	27.9	4636.1	5.4
	$P_\beta = \sigma_b^2 = 0.01$	**0.01974**	209.3	0.02168	226.8	0.02107	226.8	0.02252	60.1	0.02181	62.4	0.07124	25.9	9225.5	5.4
	$P_\beta = \sigma_b^2 = 0.05$	**0.02218**	381.9	0.02338	397.5	0.02345	397.5	0.03300	47.8	0.02345	63.3	0.13866	18.8	44262.7	5.4
Space-ga	$P_\beta = \sigma_b^2 = 0.005$	**0.01948**	396.6	0.02081	433.8	0.01999	433.8	0.02018	257.2	0.02098	110.3	0.16969	37.1	1.6789	3.8
	$P_\beta = \sigma_b^2 = 0.01$	**0.01996**	415.8	0.02158	447.5	0.02055	447.5	0.02121	233.6	0.02198	111.5	0.18764	31.1	3.1215	3.6
	$P_\beta = \sigma_b^2 = 0.05$	**0.02171**	849.6	0.02311	884.9	0.02221	884.9	0.02724	213.6	0.02334	121.8	0.23805	17.9	14.098	3.3

TABLE 5. The paired t-test between ADMM-MCP and OLS. For the one-tailed test with 95% level of confidence and 20 trials, the critical t-value is 1.729.

Dataset	Fault level	OLS AVG MSE	ADMM-MCP AVG MSE	AVG improvement	Standard error	t-value	p-value	Confidence interval of AVG improvement
Abalone	$P_\beta = \sigma_b^2 = 0.005$	5.3269	5.1617	0.1652	0.0198	8.3	4.54×10^{-08}	[0.12376,0.20676]
	$P_\beta = \sigma_b^2 = 0.01$	5.4416	5.2587	0.1829	0.0159	11.7	1.99×10^{-10}	[0.14757,0.21831]
	$P_\beta = \sigma_b^2 = 0.05$	5.8697	5.6169	0.2528	0.0109	23.2	1.04×10^{-15}	[0.22816,0.27743]
ASN	$P_\beta = \sigma_b^2 = 0.005$	0.01810	0.01512	0.00298	0.00023	12.8	4.44×10^{-11}	[0.00249,0.00347]
	$P_\beta = \sigma_b^2 = 0.01$	0.01928	0.01596	0.00332	0.00017	19.0	3.93×10^{-14}	[0.00295,0.00368]
	$P_\beta = \sigma_b^2 = 0.05$	0.02262	0.02006	0.00256	0.00015	16.7	3.97×10^{-13}	[0.00224,0.00288]
Housing	$P_\beta = \sigma_b^2 = 0.005$	0.01654	0.01527	0.00127	0.00016	7.9	1.12×10^{-07}	[0.00093,0.00161]
	$P_\beta = \sigma_b^2 = 0.01$	0.01832	0.01656	0.00176	0.00016	10.9	6.54×10^{-10}	[0.00142,0.00210]
	$P_\beta = \sigma_b^2 = 0.05$	0.02476	0.02135	0.00341	0.00024	14.2	6.82×10^{-12}	[0.00290,0.00392]
Concrete	$P_\beta = \sigma_b^2 = 0.005$	0.01756	0.01219	0.00538	0.00097	5.6	1.17×10^{-05}	[0.00335,0.00741]
	$P_\beta = \sigma_b^2 = 0.01$	0.01978	0.01387	0.00591	0.00109	5.4	1.49×10^{-05}	[0.00364,0.00818]
	$P_\beta = \sigma_b^2 = 0.05$	0.02515	0.01724	0.00791	0.00136	5.8	6.70×10^{-06}	[0.00506,0.01076]
Energy	$P_\beta = \sigma_b^2 = 0.005$	0.00568	0.00518	0.00050	0.00005	9.9	3.14×10^{-09}	[0.00040,0.00061]
	$P_\beta = \sigma_b^2 = 0.01$	0.00632	0.00560	0.00072	0.00006	12.3	8.96×10^{-11}	[0.00060,0.00085]
	$P_\beta = \sigma_b^2 = 0.05$	0.00849	0.00761	0.00088	0.00006	15.8	1.06×10^{-12}	[0.00076,0.00100]
WQW	$P_\beta = \sigma_b^2 = 0.005$	0.01732	0.01641	0.00091	0.00006	14.2	7.35×10^{-12}	[0.00077,0.00104]
	$P_\beta = \sigma_b^2 = 0.01$	0.01753	0.01660	0.00093	0.00006	15.5	1.53×10^{-12}	[0.00081,0.00106]
	$P_\beta = \sigma_b^2 = 0.05$	0.01807	0.01729	0.00078	0.00004	30.1	8.50×10^{-18}	[0.00641,0.00740]
MG	$P_\beta = \sigma_b^2 = 0.005$	0.02041	0.01924	0.00118	0.00011	10.9	6.54×10^{-10}	[0.00095,0.00141]
	$P_\beta = \sigma_b^2 = 0.01$	0.02107	0.01974	0.00133	0.00007	19.0	3.87×10^{-14}	[0.00118,0.00147]
	$P_\beta = \sigma_b^2 = 0.05$	0.02345	0.02218	0.00127	0.00010	12.3	7.99×10^{-11}	[0.00106,0.00149]
Space-ga	$P_\beta = \sigma_b^2 = 0.005$	0.01999	0.01948	0.00050	0.00006	8.5	3.21×10^{-8}	[0.00038,0.00063]
	$P_\beta = \sigma_b^2 = 0.01$	0.02055	0.01996	0.00059	0.00005	12.9	3.76×10^{-11}	[0.00049,0.00068]
	$P_\beta = \sigma_b^2 = 0.05$	0.02221	0.02171	0.00051	0.00005	10.9	6.28×10^{-10}	[0.00041,0.00060]

higher than the test set MSE values obtained from the other algorithms. In the figure, the red colored marker shows the minimum test set MSE of the SVR algorithm. The minimum value is equal to 0.0015838 and the number of used nodes is 694. When the ADMM-MCP, ADMM-l_1, and OLS algorithms uses around 694 nodes, their MSE values are lower than that of the SVR algorithms. **From Figure 5, the MSE curve of the proposed ADMM-MCP is lower than those of the other algorithms. That means, in terms of the number of nodes and the test set MSE, the performances of the ADMM-MCP are better than those of the other algorithms.**

- To further discuss the result, we can use the SVR and ADMM-ℓ_1 algorithms as anchors. As mentioned in the above, for SVR, the minimum value is equal to 0.0015838 and the number of used nodes is 694. The ADMM-ℓ_1 algorithm is able to use 157 nodes to achieve a similar test set MSE value, i.e., 0.015672. For the OLS algorithm, when around 150 nodes are used, the MSE value is 0.016552 which is higher than that of the ADMM-ℓ_1. For the ℓ_1 algorithm, the minimum test

TABLE 6. The paired t-test between ADMM-MCP and ADMM-l_1. For the one-tailed test with 95% level of confidence and 20 trials, the critical t-value is 1.729.

Dataset	Fault level	ADMM-l_1 AVG MSE	ADMM-MCP AVG MSE	AVG improvement	Standard error	t-value	p-value	Confidence interval of AVG improvement
Abalone	$P_\beta = \sigma_b^2 = 0.005$	5.3611	5.1617	0.1994	0.0146	13.7	1.33×10^{-11}	[0.16900,0.22989]
	$P_\beta = \sigma_b^2 = 0.01$	5.5254	5.2587	0.2667	0.0206	12.9	3.55×10^{-11}	[0.22012,0.31331]
	$P_\beta = \sigma_b^2 = 0.05$	6.2372	5.6169	0.6203	0.0248	25.0	2.68×10^{-16}	[0.56413,0.67641]
ASN	$P_\beta = \sigma_b^2 = 0.005$	0.01617	0.01512	0.00105	0.00007	14.3	6.51×10^{-12}	[0.00090,0.00121]
	$P_\beta = \sigma_b^2 = 0.01$	0.01711	0.01596	0.00115	0.00008	14.5	4.88×10^{-12}	[0.00099,0.00132]
	$P_\beta = \sigma_b^2 = 0.05$	0.02086	0.02006	0.00080	0.00009	8.5	3.29×10^{-08}	[0.00060,0.00100]
Housing	$P_\beta = \sigma_b^2 = 0.005$	0.01583	0.01527	0.00056	0.00010	5.7	8.50×10^{-06}	[0.00035,0.00076]
	$P_\beta = \sigma_b^2 = 0.01$	0.01743	0.01656	0.00086	0.00011	7.7	1.59×10^{-07}	[0.00063,0.00110]
	$P_\beta = \sigma_b^2 = 0.05$	0.02343	0.02135	0.00208	0.00019	11.0	5.39×10^{-10}	[0.00169,0.00248]
Concrete	$P_\beta = \sigma_b^2 = 0.005$	0.01334	0.01218	0.00116	0.00009	12.8	4.50×10^{-11}	[0.00097,0.00136]
	$P_\beta = \sigma_b^2 = 0.01$	0.01489	0.01387	0.00102	0.00009	11.0	5.21×10^{-10}	[0.00082,0.00121]
	$P_\beta = \sigma_b^2 = 0.05$	0.01910	0.01724	0.00186	0.00014	13.3	2.23×10^{-11}	[0.00156,0.00215]
Energy	$P_\beta = \sigma_b^2 = 0.005$	0.00560	0.00518	0.00042	0.00005	8.2	6.00×10^{-08}	[0.00031,0.00053]
	$P_\beta = \sigma_b^2 = 0.01$	0.00610	0.00560	0.00050	0.00006	8.7	2.39×10^{-08}	[0.00038,0.00062]
	$P_\beta = \sigma_b^2 = 0.05$	0.00856	0.00761	0.00095	0.00009	10.1	2.12×10^{-09}	[0.00076,0.00115]
WQW	$P_\beta = \sigma_b^2 = 0.005$	0.01678	0.01641	0.00037	0.00003	12.5	6.30×10^{-11}	[0.00031,0.00043]
	$P_\beta = \sigma_b^2 = 0.01$	0.01701	0.01660	0.00041	0.00003	12.0	1.21×10^{-10}	[0.00034,0.00048]
	$P_\beta = \sigma_b^2 = 0.05$	0.01763	0.01729	0.00034	0.00002	14.0	9.45×10^{-12}	[0.00029,0.00039]
MG	$P_\beta = \sigma_b^2 = 0.005$	0.02145	0.01924	0.00222	0.00021	10.8	8.14×10^{-10}	[0.00179,0.00265]
	$P_\beta = \sigma_b^2 = 0.01$	0.02168	0.01974	0.00194	0.00019	10.2	1.96×10^{-9}	[0.00154,0.00234]
	$P_\beta = \sigma_b^2 = 0.05$	0.02338	0.02218	0.00121	0.00015	8.1	7.54×10^{-8}	[0.00089,0.00152]
Space-ga	$P_\beta = \sigma_b^2 = 0.005$	0.02081	0.01948	0.00133	0.00026	5.1	3.26×10^{-5}	[0.00078,0.00187]
	$P_\beta = \sigma_b^2 = 0.01$	0.02158	0.01996	0.00162	0.00025	6.6	1.39×10^{-6}	[0.00110,0.00214]
	$P_\beta = \sigma_b^2 = 0.05$	0.02311	0.02171	0.00140	0.00022	6.5	1.52×10^{-6}	[0.00095,0.00185]

set MSE is 0.016419 and the number of the used nodes is 169. For the proposed ADMM-MCP algorithm, it is able to use 139 nodes only to lower the test set MSE to 0.015281. Clearly, the performances of the proposed ADMM-MCP are better than those of the comparison algorithms.

For each fault level and each dataset, we repeated the experiment 20 trials. In each trail, the samples of dataset were randomly split for training and testing set. The results are summarized in Table 4. From the table, it can be seen that under the concurrent fault situation, even we select the best MSE values of SVR, l_1-reg., HOM, and OFR, their performance are still unacceptable. Especially, when the fault level is high, their test MSE values are much higher than those of the remaining algorithms. The ADMM-MCP, ADMM-l_1 and OLS algorithms can effectively reduce the influence of the concurrent fault. Among them, the ADMM-MCP is the best which has smaller average MSE values and uses fewer number of nodes.

G. PAIRED T-TEST

This subsection uses the paired t-test to illustrate that the improvement of our proposed algorithm is statistically significant. From Table 4, The ADMM-ℓ_1 and OLS algorithm are the second best and the third best. Hence we perform the paired tests between ADMM-MCP and ADMM-ℓ_1, and between ADMM-MCP and OLS. The paired test results are summarized in Tables 5 and 6. For the one-tailed test with 95% level of confidence and 20 trials, the critical t-value is 1.729.

From the tables, we can see that all the test t-values are greater than 1.729 and all p-values are smaller than 0.05. In other words, we have enough confidence to say that on average the proposed ADMM-MCP is better than the ADMM-l_1 and OLS algorithm. Besides, all confidence intervals in the two tables do not include zero. Therefore, we can further confirm that the improvement of the proposed ADMM-MCP is statistically significant.

VI. CONCLUSION

In the paper, the fault tolerant RBF neural network and its center selection problem are studied. Based on ADMM framework and ℓ_0-norm, this paper proposes the ADMM-MCP algorithm. First we introduce an ℓ_0-norm term, which has an ability to remove some unimportant RBF nodes during training, into the fault tolerant objective function. Since ℓ_0-norm is noncontinuous, we cannot use traditional gradient descent like algorithms to minimize the modified objective function. We then approximate the ℓ_0-norm term with the MCP function. However, the MCP-based objective function is still nonconvex and nonsmooth, traditional gradient descent like algorithms cannot handle it. This paper then applies the ADMM framework to construct an algorithm, namely ADMM-MCP, to train an RBF network and to select RBF nodes simultaneously. The ADMM framework breaks down the update into three parts. Each part can be effectively solved, even though some parts contain nonconvex and nondifferentiable terms. In addition, we prove that the algorithm converges. From the experimental results, our ADMM-MCP algorithm is superior to many other existing algorithms.

REFERENCES

[1] H. Yu, P. D. Reiner, T. Xie, T. Bartczak, and B. M. Wilamowski, "An incremental design of radial basis function networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 10, pp. 1793–1803, Oct. 2014.

[2] F. Gianfelici, "RBF-based technique for statistical demodulation of pathological tremor," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 10, pp. 1565–1574, Oct. 2013.

[3] R. Eickhoff and U. Rückert, "Robustness of radial basis functions," *Neurocomputing*, vol. 70, nos. 16–18, pp. 2758–2767, 2007.

[4] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, Sep. 1990.

[5] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1998.

[6] S. Chen, "Nonlinear time series modelling and prediction using Gaussian RBF networks with enhanced clustering and RLS learning," *Electron. Lett.*, vol. 31, no. 2, pp. 117–118, Jan. 1995.

[7] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[8] J. B. Gomm and D. L. Yu, "Selecting radial basis function network centers with recursive orthogonal least squares training," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 306–314, Mar. 2000.

[9] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag, 1995.

[10] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 9. 1997, pp. 155–161.

[11] B. M. Yu, "Neuroscience: Fault tolerance in the brain," *Nature*, vol. 532, pp. 449–450, Apr. 2016.

[12] G. Bolt, "Fault models for artificial neural networks," in *Proc. IJCNN*, Singapore, 1991, pp. 1373–1378.

[13] Z.-H. Zhou and S.-F. Chen, "Evolving fault-tolerant neural networks," *Neural Comput. Appl.*, vol. 11, nos. 3–4, pp. 156–160, Jun. 2003.

[14] C.-S. Leung and J. P.-F. Sum, "A fault-tolerant regularizer for RBF networks," *IEEE Trans. Neural Netw.*, vol. 19, no. 3, pp. 493–507, Mar. 2008.

[15] R. Martolia, A. Jain, and L. Singla, "Analysis & survey on fault tolerance in radial basis function networks," in *Proc. Int. Conf. Comput., Commun. Automat.*, May 2015, pp. 469–473.

[16] J. A. Clemente, W. Mansour, R. A. Ayoubi, F. Serrano, H. Mecha, H. Ziade, W. El Falou, and R. Velazco, "Hardware implementation of a fault-tolerant Hopfield neural network on FPGAs," *Neurocomputing*, vol. 171, pp. 1606–1609, Jan. 2016.

[17] G. Carvajal and M. Figueroa, "Model, analysis, and evaluation of the effects of analog VLSI arithmetic on linear subspace-based image recognition," *Neural Netw.*, vol. 55, pp. 72–82, Jul. 2014.

[18] J. L. Burr, "Digital neural network implementations," in *Neural Networks, Concepts, Applications, and Implementations*, vol. 3. Englewood Cliffs, NJ, USA: Prentice-Hall, 1995, pp. 237–285.

[19] C.-S. Leung, W. Y. Wan, and R. Feng, "A regularizer approach for RBF networks under the concurrent weight failure situation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1360–1372, Jun. 2017.

[20] R. A. Nawrocki and R. M. Voyles, "Artificial neural network performance degradation under network damage: Stuck-at faults," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, San Jose, CA, USA, Jul. 2011, pp. 442–449.

[21] H. R. Mahdiani, S. M. Fakhraie, and C. Lucas, "Relaxed fault-tolerant hardware implementation of neural networks in the presence of multiple transient errors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1215–1228, Aug. 2012.

[22] J. L. Bernier, J. Ortega, E. Ros, I. Rojas, and A. Prieto, "A quantitative study of fault tolerance, noise immunity, and generalization ability of MLPs," *Neural Comput.*, vol. 12, no. 12, pp. 2941–2964, 2000.

[23] M. Conti, S. Orcioni, and C. Turchetti, "Training neural networks to be insensitive to weight random variations," *Neural Netw.*, vol. 13, no. 1, pp. 125–132, 2000.

[24] M. Lee, K. Hwang, and W. Sung, "Fault tolerance analysis of digital feed-forward deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 5031–5035.

[25] C.-S. Leung and J. P.-F. Sum, "RBF networks under the concurrent fault situation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1148–1155, Jul. 2012.

[26] H. Wang, R.-B. Feng, Z.-F. Han, and C.-S. Leung, "ADMM-based algorithm for training fault tolerant RBF networks and selecting centers," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3870–3878, Aug. 2018.

[27] C.-H. Zhang, "Nearly unbiased variable selection under minimax concave penalty," *Ann. Statist.*, vol. 38, no. 2, pp. 894–942, 2010.

[28] P. Breheny and J. Huang, "Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection," *Ann. Appl. Statist.*, vol. 5, no. 1, p. 232, 2011.

[29] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[30] M. Stevenson, R. Winter, and B. Widrow, "Sensitivity of feedforward neural networks to weight errors," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 71–80, Mar. 1990.

[31] Z. Han, R.-B. Feng, W. Y. Wan, and C.-S. Leung, "Online training and its convergence for faulty networks with multiplicative weight noise," *Neurocomputing*, vol. 155, pp. 53–61, May 2015.

[32] J. B. Burr, "Digital neural network implementations," in *Neural Networks, Concepts, Applications, and Implementations*, vol. 3. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991, pp. 237–285.

[33] J. L. Bernier, J. Ortega, I. Rojas, and A. Prieto, "Improving the tolerance of multilayer perceptrons by minimizing the statistical sensitivity to weight deviations," *Neurocomputing*, vol. 31, pp. 87–103, Mar. 2000.

[34] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[35] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[36] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," 2015, *arXiv:1511.06324*. [Online]. Available: https://arxiv.org/abs/1511.06324

[37] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods," *Math. Program.*, vol. 137, nos. 1–2, pp. 91–129, 2013.

[38] M. Lichman. (2013). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[39] Q. Zhang, X. Hu, and B. Zhang, "Comparison of ℓ_1-norm SVR and sparse coding algorithms for linear regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 8, pp. 1828–1833, Aug. 2015.

[40] M. Sugiyama and H. Ogawa, "Optimal design of regularization term and regularization parameter by subspace information criterion," *Neural Netw.*, vol. 15, no. 3, pp. 349–361, Apr. 2002.

[41] S. Li, Z.-H. You, H. Guo, X. Luo, and Z.-Q. Zhao, "Inverse-free extreme learning machine with optimal information updating," *IEEE Trans. Cybern.*, vol. 46, no. 5, pp. 1229–1241, May 2016.

[42] D. L. Ly and H. Lipson, "Optimal experiment design for coevolutionary active learning," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 394–404, Jun. 2014.

[43] R. Ekambaram, S. Fefilatyev, M. Shreve, K. Kramer, L. O. Hall, D. B. Goldgof, and R. Kasturi, "Active cleaning of label noise," *Pattern Recognit.*, vol. 51, pp. 463–480, Mar. 2016.

[44] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decis. Support Syst.*, vol. 47, no. 4, pp. 547–553, 2009.

[45] *StatLib—Datasets Archive*. [Online]. Available: http://lib.stat.cmu.edu/datasets

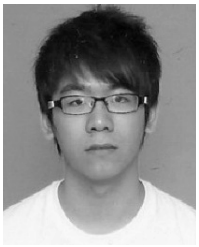
[46] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27-1–27-27, 2011.

[47] X. Hong, S. Chen, J. Gao, and C. J. Harris, "Nonlinear identification using orthogonal forward regression with nested optimal regularization," *IEEE Trans. Cybern.*, vol. 45, no. 12, pp. 2925–2936, Dec. 2015.

[48] D. M. Malioutov, M. Cetin, and A. S. Willsky, "Homotopy continuation for sparse signal representation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 5, Mar. 2005, pp. 733–736.

HAO WANG received the Ph.D. degree in electronic engineering from the City University of Hong Kong, in 2019. His current research interests include neural networks and machine learning.

ZHANGLEI SHI is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, City University of Hong Kong. His current research interests include neural networks and machine learning.



HIU TUNG WONG received the B.S. degree (Hons.) in applied physics from the City University of Hong Kong, Hong Kong, in 2014, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering. His research interests include neural networks and machine learning.

CHI-SING LEUNG received the Ph.D. degree in computer science from The Chinese University of Hong Kong, Hong Kong, in 1995.

He is currently a Professor with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong. He has authored over 120 journal articles in the areas of digital signal processing, neural networks, and computer graphics. His current research interests include neural computing and computer graphics.

Dr. Leung was a member of the Organizing Committee of ICONIP 2006. He received the 2005 IEEE TRANSACTIONS ON MULTIMEDIA Prize Paper Award for his article titled The Plenoptic Illumination Function, in 2005. He was the Program Chair of ICONIP 2009 and ICONIP 2012. He is a Governing Board Member and the Vice President of the Asian Pacific Neural Network Assembly (APNNA). He is/was the Guest Editor of several journals, including *Neural Computing and Applications*, *Neurocomputing*, and *Neural Processing Letters*.

HING CHEUNG SO was born in Hong Kong. He received the B.Eng. degree in electronic engineering from the City University of Hong Kong, in 1990, and the Ph.D. degree in electronic engineering from The Chinese University of Hong Kong, in 1995, where he was a Postdoctoral Fellow, from 1995 to 1996.

From1990 to 1991, he was an Electronic Engineer with the Research and Development Division, Everex Systems Engineering Ltd., Hong Kong. From 1996 to 1999, he was a Research Assistant Professor with the Department of Electronic Engineering, City University of Hong Kong, where he is currently an Associate Professor. His research interests include detection and estimation, fast and adaptive algorithms, multidimensional harmonic retrieval, robust signal processing, sparse approximation, and source localization.

Dr. So has been elected as a Fellow of the IEEE in recognition of his contributions to spectral analysis and source localization. In addition, he was an Elected Member of the Signal Processing Theory and Methods Technical Committee, IEEE Signal Processing Society, in 2011, where he is Chair of the awards subcommittee, in 2015. He was on the Editorial Board of the IEEE TRANSACTIONS ON SIGNAL PROCESSING, from 2010 to 2014, *Signal Processing*, in 2010, *Digital Signal Processing*, in 2011, and the *IEEE Signal Processing Magazine*, in 2014.

RUIBIN FENG received the Ph.D. degree in electronic engineering from the City University of Hong Kong, in 2017. His current research interests include neural networks and medical imaging.