Received August 7, 2019, accepted August 15, 2019, date of publication August 22, 2019, date of current version September 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2936973

Multikernel Adaptive Filters Under the Minimum Cauchy Kernel Loss Criterion

WEI SHI, KUI XIONG, AND SHIYUAN WANG, (Senior Member, IEEE)
Chongqing Key Laboratory of Nonlinear Circuits and Intelligent Information Processing, College of Electronic and Information Engineering, Southwest University, Chongqing 400715, China

Corresponding author: Shiyuan Wang (wsy@swu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61671389, and in part by the Fundamental Research Funds for the Central Universities under Grant XDJK2019B011.

ABSTRACT The Cauchy loss has been successfully applied in robust learning algorithms in the presence of large outliers, but it may suffer from performance degradation in complex nonlinear tasks. To address this issue, by transforming the original data into the reproducing kernel Hilbert spaces (RKHS) with the kernel trick, a novel Cauchy kernel loss is developed in such a kernel space. Based on the minimum Cauchy kernel loss criterion, the *multikernel minimum Cauchy kernel loss* (MKMCKL) algorithm is proposed by mapping the input data into the multiple RKHS. The proposed MKMCKL algorithm can provide the performance improvement of the kernel adaptive filter (KAF) based on a single kernel, and also improve the stability of the multikernel adaptive filter based on the quadratic loss in impulsive noises, efficiently. To further curb the growth of network of MKMCKL, a novel sparsification method is presented to prune redundant data, thus reducing its computational and storage burdens. Simulations on different nonlinear applications illustrate the performance superiorities of the proposed algorithms in impulsive noises.

INDEX TERMS Multikernel algorithm, robust adaptive filter, minimum Cauchy kernel loss criterion, kernel selection, sparsification.

I. INTRODUCTION

Kernel adaptive filters (KAFs), developed by transforming the original input data into a high-dimensional reproducing kernel Hilbert spaces (RKHS), have been widely applied in chaotic time-series prediction, nonlinear system identification, and nonlinear channel equalization [1]. In the study of KAFs, the important issues are the design of filtering algorithms based on specific loss functions, the kernel selection, and the sparsification method for curbing the network size growth of KAFs.

For the loss function, the quadratic loss functions, such as mean square error (MSE) and least-squares (LS), are generally used for the design of KAFs thanks to their simple and convex natures. For example, the kernel least mean square (KLMS) algorithm is proposed by minimizing MSE [2] and the kernel recursive least-squares (KRLS) by minimizing LS of errors [3]. Although the quadratic loss based algorithms can provide desirable performance under Gaussian assumptions, they may suffer from performance degradation even divergence issues in non-Gaussian noises. To address these issues, the non-quadratic loss functions are introduced. The higher order error [4], [5] and lower order error [6], [7] based loss functions are the simplest two types of non-quadratic loss functions. Generally, the higher order error based loss can provide the performance improvement in light-tailed noises, e.g., uniform and binary noises, and the lower order error based loss can combat heavy-tailed noises, e.g., Laplace and α-stable noises. However, the higher order error based loss is not suitable for Gaussian and heavy-tailed noises [5], and the lower order error based loss may provide slow convergence rate in Gaussian noises [6]. Generally, the most representative correntropic loss (C-Loss) [8]–[10] and generalized correntropic loss (GC-loss) [11], [12] in information theory learning (ITL) [13] can combat non-Gaussian noises, efficiently. Recently, the kernel risk-sensitive loss (KRSL) [14] and kernel mean p-power error (KMPE) [15] have been proposed for robust learning. In addition, the logarithmic error loss including the Cauchy loss is another type of effective non-quadratic loss for non-Gaussian noises [16]–[20]. It is worth noting that the logarithmic

The associate editor coordinating the review of this article and approving it for publication was Donatella Darsena.

loss can provide better performance than C-Loss in specific environments [17], [18].

For the kernel selection in KAFs, the first issue is to choose the type of kernel. The commonly used kernels include Gaussian kernel, Laplace kernel, and polynomial kernel [21]. Thanks to its universal approximation capability, the Gaussian kernel is the default choice in the design of KAFs [1]. For the chosen Gaussian kernel, its bandwidth needs to be selected to achieve desirable approximation accuracy. There are several methods for the selection of kernel parameters, such as cross validation [22], penalizing functions [23], and plug-in methods [24]. However, these methods cannot be effectively applied for online learning owing to their huge computational cost. For efficient online learning, different methods for adaptively adjusting kernel parameters are introduced, which can change these parameters at different filtering stages, efficiently [25]. Another effective kernel selection strategy is the multikernel learning based method [26]–[28], which can combine several distinct kernels to achieve desirable performance. It is worth noting that the multikernel method can use different types of kernels rather than the same type of kernels with different kernel parameters. In addition, only a small number of kernels are needed to achieve desirable performance without incurring a huge computational burden in multikernel adaptive filtering algorithms [29]–[32].

For KAFs including multikernel algorithms, their growing network structures in adaptive filtering processes result in large computational burden, especially when the size of data set is very large. Different dimensionality reduction methods such as sparsification, vector quantization, and random Fourier features (RFF) are therefore proposed to address this issue. For the sparsification method, the redundant data are pruned according to the specific criteria such as novelty criterion [33], approximate linear dependency [3], surprise criterion [34], and prediction variance criterion [35] to generate a more compact network, thus reducing the computational complexity. The vector quantization method is the improvement of sparsification method, which utilizes the information of redundant data discarded by the sparsification method at the cost of increasing computation cost [36], [37]. Unlike sparsification and vector quantization methods, the RFF method is data-independent and provides a fixed-dimensional network structure [38]–[40]. The main drawback of RFF method is that its approximation accuracy depends on the dimensionality which needs to be determined for different applications in advance.

In this paper, the Cauchy loss [19], [20] is mainly considered owing to its robustness to large outliers. To improve the performance of Cauchy loss in some nonlinear tasks, a novel Cauchy kernel loss is developed by transforming the original data into the reproducing kernel Hilbert spaces (RKHS). In comparison with the conventional Cauchy loss, the proposed Cauchy kernel loss has a better performance surface, which leads to the performance improvement of Cauchy loss. Thus, the proposed Cauchy kernel loss can be extended to various robust learning, such as principal components analysis [41], face recognition [42], and other real-world applications [43], [44]. Here, the Cauchy kernel loss is mainly considered to develop a robust multikernel adaptive filtering algorithm for combating non-Gaussian noises and addressing kernel selection issues. To further reduce the computational complexity, a novel sparsification method based on the combination of online vector quantization method [36] and sliding-window method [45] is presented.

Three contributions of this paper are summarized as follows: (1) A novel Cauchy kernel loss is developed in RKHS to improve the performance of Cauchy loss, and some important properties of Cauchy kernel loss are also provided. (2) Based on the minimum Cauchy kernel loss criterion, the robust *multikernel minimum Cauchy kernel loss* (MKMCKL) algorithm is proposed in a multikernel method. The proposed MKMCKL algorithm can efficiently combat impulsive noises including large outliers and freely choose kernel parameters. (3) A novel sparsification method is introduced in MKMCKL to generate a sparse MKMCKL (SMKMCKL) algorithm. SMKMCKL using the online vector quantization method as well as the sliding-window method can reduce the complexity of MKMCKL, and thus can be applied for time-varying systems, efficiently.

The rest of this paper is organized as follows. We present the Cauchy kernel loss in Section II. Some important properties and performance surface of Cauchy kernel loss are also provided in this section. In Section III, the MKMCKL algorithm is proposed by using a multikernel method, and its sparsification algorithm is further proposed to reduce the complexity. In Section IV, simulations conducted in different applications illustrate the effectiveness of proposed algorithms. Finally, Section V concludes this paper.

II. MINIMUM CAUCHY KERNEL LOSS CRITERION

In this section, we first present the Cauchy kernel loss in the reproducing kernel Hilbert spaces (RKHS). Then, some important properties of Cauchy kernel loss and their proofs are provided. The performance surface of Cauchy kernel loss is discussed for comparison with that of Cauchy loss. Furthermore, the minimum Cauchy kernel loss criterion is introduced for design of robust learning algorithms.

A. CAUCHY KERNEL LOSS

Denote two variables as X and Y, and the Cauchy loss is defined by [19], [20]

$$L_C(X,Y) = E\left[\log\left(1+\frac{(X-Y)^2}{\gamma^2}\right)\right], \tag{1}$$

where $\gamma > 0$ is a constant and $E[\cdot]$ denotes the joint expectation of X and Y. The Cauchy loss has been successfully applied to learning algorithms thanks to its robustness to large outliers [19], [20]. However, it has been proven that the loss function developed in RKHS can improve the performance of the one in the original data space [8]–[10].

Thus, by using a kernel mapping to transform X and Y into RKHS, the Cauchy loss can be rewritten as follows:

$$L_{CK}(X,Y)=E\left[\log\left(1+\frac{\|\varphi(X)-\varphi(Y)\|^2}{\gamma^2}\right)\right], \tag{2}$$

where $\varphi(\cdot)$ is a nonlinear mapping induced by a Mercer kernel $\kappa(\cdot,\cdot)$, and $||\cdot||$ denotes the Euclidean norm. Note that we use the Euclidean metric to measure the distance between $\varphi(X)$ and $\varphi(Y)$ since they can be high-dimensional vectors.

Since the dimensionality of $\varphi(X)-\varphi(Y)$ may be very high, it is difficult to calculate the Euclidean norm (i.e., inner product) in (2). Thus, the kernel trick [21] is generally used to derive this inner product as

$$\begin{aligned}\|\varphi(X)-\varphi(Y)\|^2 &= \langle\varphi(X),\varphi(X)\rangle+\langle\varphi(Y),\varphi(Y)\rangle\\ &\quad -2\langle\varphi(X),\varphi(Y)\rangle\\ &= \kappa(X,X)+\kappa(Y,Y)-2\kappa(X,Y),\end{aligned} \tag{3}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product operator. Thanks to its universal approximation capability, the Gaussian kernel is used here, which is given by

$$\kappa(X,Y)=\exp\left(-\frac{\|X-Y\|^2}{2\sigma^2}\right), \tag{4}$$

where $\sigma > 0$ is the kernel bandwidth. Then, by using the Gaussian kernel and the kernel trick, the Cauchy kernel loss is defined as follows:

$$\begin{aligned}L_{CK}(X,Y) &= \log\left(1+\frac{1}{\gamma^2}\right)\\ &\quad -E\left[\log\left(1+\frac{1-\frac{1}{2}\|\varphi(X)-\varphi(Y)\|^2}{\gamma^2}\right)\right]\\ &= \log\left(1+\frac{1}{\gamma^2}\right)\\ &\quad -E\left[\log\left(1+\frac{\exp\left(-\frac{\|X-Y\|^2}{2\sigma^2}\right)}{\gamma^2}\right)\right],\end{aligned} \tag{5}$$

where the minimum value of $L_{CK}(X,Y)$ is 0.

In practical applications, the Cauchy kernel loss is estimated using N samples $\{x_i,y_i\}_{i=1}^N$, also called the empirical Cauchy kernel loss, as follows:

$$\begin{aligned}\hat{L}_{CK}(X,Y) &= \log\left(1+\frac{1}{\gamma^2}\right)\\ &\quad -\frac{1}{N}\sum_{i=1}^{N}\log\left(1+\frac{\exp\left(-\frac{\|x_i-y_i\|^2}{2\sigma^2}\right)}{\gamma^2}\right).\end{aligned} \tag{6}$$

B. PROPERTIES

Based on the proposed Cauchy kernel loss (5), some important properties are provided as follows.

Property 1: The Cauchy kernel loss is symmetric, positive, and bounded.

Proof: Since $\kappa(X,Y)=\kappa(Y,X)$, $L_{CK}(X,Y)=L_{CK}(Y,X)$ is straightforward obtained, i.e., $L_{CK}(X,Y)$ is symmetric. Since $0<\kappa(X,Y)\le 1$, $0<L_{CK}(X,Y)\le\log(1+1/\gamma^2)$ can be easily obtained, i.e., $L_{CK}(X,Y)$ is positive and bounded. ■

Property 2: Some other loss functions can be generalized by the Cauchy kernel loss, which is specifically described as follows:

a) The Cauchy kernel loss reduces to the correntropic loss [8]–[10], when γ is large enough or σ is small enough.

b) The Cauchy kernel loss reduces to the Cauchy loss [19], [20], when σ is large enough.

c) The Cauchy kernel loss reduces to the mean square error [2], when σ and γ are large enough.

Proof: a) When γ is large enough or σ is small enough, using the fact that $\log(1+x)\approx x$ for small enough x, we obtain

$$L_{CK}(X,Y)\approx\log\left(1+\frac{1}{\gamma^2}\right)-\frac{1}{\gamma^2}E\left[\exp\left(-\frac{\|X-Y\|^2}{2\sigma^2}\right)\right],$$

which is a correntropic loss.

b) When σ is large enough, using the fact that $\exp(-x)\approx 1-x$ for small enough x, we obtain

$$\begin{aligned}L_{CK}(X,Y) &\approx \log\left(1+\frac{1}{\gamma^2}\right)\\ &\quad -E\left[\log\left(1+\frac{1-\frac{1}{2\sigma^2}\|X-Y\|^2}{\gamma^2}\right)\right],\end{aligned}$$

which can be seen as a Cauchy loss in the original data space.

c) Combining proofs of a) and b) above, we obtain

$$L_{CK}(X,Y)\approx\log\left(1+\frac{1}{\gamma^2}\right)-\frac{1}{\gamma^2}+\frac{1}{2\sigma^2\gamma^2}E\left[\|X-Y\|^2\right],$$

which is the mean square error regarding $X-Y$. ■

Property 3: Denote the error as $\mathbf{e}=X-Y=[e_1,e_2,\ldots,e_N]^T$ with $e_i=x_i-y_i$, $i=1,2,\ldots,N$. Then, the convex conditions of empiric Cauchy kernel loss are summarised as follow:

a) When $\max\limits_{i=1,2,\ldots,N}|e_i|\le\sigma$, the empiric Cauchy kernel loss is convex.

b) When $\min\limits_{i=1,2,\ldots,N}|e_i|>\sigma$ and γ is smaller than a certain value, the empiric Cauchy kernel loss is convex.

Proof: The Hessian matrix of empiric Cauchy kernel loss $\hat{L}_{CK}(X,Y)$ regarding $\mathbf{e}$ can be derived as

$$\mathbf{H}_{\hat{L}_{CK}(X,Y)}(\mathbf{e})=\mathrm{diag}[\theta_1,\theta_2,\ldots,\theta_N], \tag{7}$$

where

$$\theta_i=\zeta_i\left(\gamma^2\left(1-\frac{e_i^2}{\sigma^2}\right)+\exp\left(-\frac{e_i^2}{2\sigma^2}\right)\right), \tag{8}$$

with

$$\zeta_i=\frac{\exp\left(-\frac{e_i^2}{2\sigma^2}\right)}{N\sigma^2\left(\gamma^2+\exp\left(-\frac{e_i^2}{2\sigma^2}\right)\right)^2}\ge 0.$$

Then, according to (7) and (8), we have

a) $\mathbf{H}_{\hat{L}_{CK}(X,Y)}(\mathbf{e}) \geq 0$ when $\max_{i=1,2,\ldots,N} |e_i| \leq \sigma$, i.e., $\hat{L}_{CK}(X, Y)$ is convex at $\max_{i=1,2,\ldots,N} |e_i| \leq \sigma$.

b) $\mathbf{H}_{\hat{L}_{CK}(X,Y)}(\mathbf{e}) \geq 0$ when $\min_{i=1,2,\ldots,N} |e_i| > \sigma$ and

$$\gamma \leq \min_{|e_i|>\sigma, i=1,2,\ldots,N} \left\{ \sqrt{\frac{\sigma^2}{e_i^2 - \sigma^2} \exp\left(-\frac{e_i^2}{2\sigma^2}\right)} \right\},$$

which means that $\hat{L}_{CK}(X, Y)$ is convex in this case. ∎

Remark 1: According to Property 3, we have that the convexity of $\hat{L}_{CK}(X, Y)$ depends on e_i, σ, and γ. Especially, the value of γ can control the convex range of $\hat{L}_{CK}(X, Y)$, and a smaller γ can generate a larger convex range generally. Thus, we choose an appropriately small γ to guarantee the convexity of Cauchy kernel loss in practice. In comparison with C-Loss [8], a larger convex range of Cauchy kernel loss can be obtained, which is analyzed in the Appendix.

C. MINIMUM CAUCHY KERNEL LOSS CRITERION

We consider the following nonlinear system

$$d_t = f^*(\mathbf{u}_t) + n_t, \tag{9}$$

where $\mathbf{u}_t \in \mathbb{R}^d$ is the d-dimensional input vector at discrete time t, d_t is the desired output response, n_t is the noise, and $f^*(\cdot)$ is the optimal estimate of d_t.

When $f^*(\cdot)$ is modeled by a finite impulse response (FIR) system with an optimal weight vector $\mathbf{w}_o \in \mathbb{R}^d$, it can be estimated using weight vector $\mathbf{w} \in \mathbb{R}^d$ and the estimated output at discrete time t is given by $\hat{y}_t = \mathbf{w}^T\mathbf{u}_t$ with $(\cdot)^T$ being the transpose. Thus, the cost function using the Cauchy kernel loss can be defined by

$$\begin{aligned} J(e_t) &= \log\left(1 + \frac{1}{\gamma^2}\right) - \log\left(1 + \frac{\exp\left(-\frac{e_t^2}{2\sigma^2}\right)}{\gamma^2}\right) \\ &= \log\left(1 + \frac{1}{\gamma^2}\right) - \log\left(1 + \frac{\exp\left(-\frac{(d_t - \mathbf{w}^T\mathbf{u}_t)^2}{2\sigma^2}\right)}{\gamma^2}\right), \end{aligned} \tag{10}$$

where $e_t = d_t - \mathbf{w}^T\mathbf{u}_t$ is the estimation error at discrete time t. Let the value of γ for both Cauchy kernel loss and Cauchy loss be 1, and the comparison of their cost functions versus e_t is shown in Fig. 1, where CL denotes the Cauchy loss and CKL the Cauchy kernel loss for simplicity. From Fig. 1, we see that the value of σ can control the shape of Cauchy kernel loss based cost function, and the Cauchy kernel loss with an appropriately large σ can lead to a smoother curve than Cauchy loss when the error is small, which means that Cauchy kernel loss can provide better smoothness to the steady-state error than Cauchy loss. In addition, the curve of Cauchy kernel loss based cost function is smoother than that of Cauchy loss when the error is large, which implies

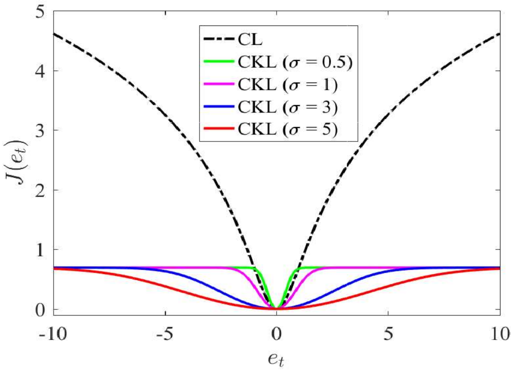


FIGURE 1. The cost functions of cauchy kernel loss and cauchy loss versus e_t.

that Cauchy kernel loss can provide better robustness to large outliers. To further show the superiority of Cauchy kernel loss over Cauchy loss, their performance surfaces versus $\mathbf{w} = [w_1; w_2]$ are shown in Fig. 2, where the optimal weight vector is $\mathbf{w}_o = [10; 10]$; $\mathbf{u}_t$ and n_t are zero-mean Gaussian variables with unit variance; $\gamma = 1$ is used for Cauchy loss; $\gamma = 1$ and $\sigma = 4$ are used for Cauchy kernel loss. From Fig. 2, we see that the Cauchy kernel loss has smaller gradient than the Cauchy loss near the optimal solution, which implies that the Cauchy kernel loss can achieve a lower misadjustment than the Cauchy loss.

Based the cost function of Cauchy kernel loss, the optimal solution can be obtained by solving the following minimizing optimization problem:

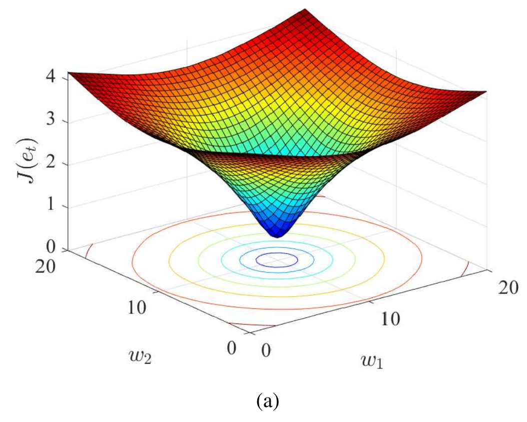
$$\begin{aligned} &\min_{\mathbf{w}} \left\{ \log\left(1 + \frac{1}{\gamma^2}\right) - \log\left(1 + \frac{\exp\left(-\frac{(d_t - \mathbf{w}^T\mathbf{u}_t)^2}{2\sigma^2}\right)}{\gamma^2}\right) \right\} \\ \Leftrightarrow &\min_{\mathbf{w}} \left\{ -\log\left(1 + \frac{\exp\left(-\frac{(d_t - \mathbf{w}^T\mathbf{u}_t)^2}{2\sigma^2}\right)}{\gamma^2}\right) \right\}, \end{aligned} \tag{11}$$

which is also called the minimum Cauchy kernel loss criterion.

Based on the minimum Cauchy kernel loss criterion, different robust learning algorithms can be developed. In this work, we mainly focus on its application for robust multikernel adaptive filtering.

III. PROPOSED ALGORITHMS

In this section, we first briefly introduce the multikernel adaptive filtering. Then, based on the minimum Cauchy kernel loss criterion, a novel robust *multikernel minimum Cauchy kernel loss* (MKMCKL) algorithm is proposed to address the stability issues of traditional multikernel algorithms. In addition, a sparsification is further introduced in MKMCKL to reduce the complexity caused by the growing network size, generating the sparse MKMCKL (SMKMCKL) algorithm.



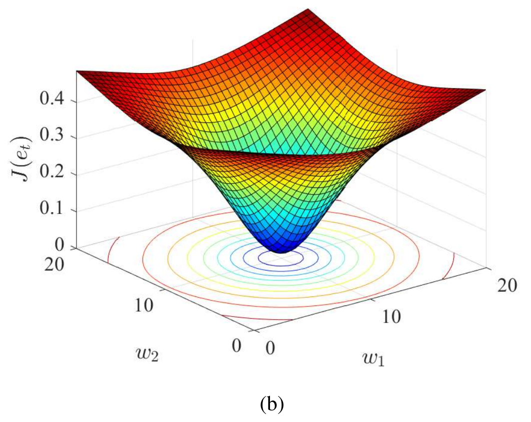


FIGURE 2. Performance surfaces of (a) Cauchy loss and (b) Cauchy kernel loss.

A. MULTIKERNEL ADAPTIVE FILTERING

For the nonlinear system (9), a kernel adaptive filter (KAF) [1] is a sequential estimator of $f(\cdot)$ by performing linear operations in RKHS. Denote $\varphi(\cdot)$ as a nonlinear mapping induced by a Mercer kernel $\kappa(\cdot,\cdot)$. According to the kernel trick, $f(\cdot)$ can be estimated in a linear form as

$$\hat{f}(\cdot) = \mathbf{\Omega}^T\varphi(\cdot), \tag{12}$$

where $\mathbf{\Omega}$ is the weight vector in RKHS. The Gaussian kernel is generally used owing to its universal approximation capability, which is given by

$$\kappa(\mathbf{u},\mathbf{u}') = \exp\left(-\frac{\left\|\mathbf{u}-\mathbf{u}'\right\|^2}{2h^2}\right), \tag{13}$$

where $h > 0$ is the kernel bandwidth. Here, we use kernel bandwidth h to distinguish kernel bandwidth σ used in Cauchy kernel loss (5) to avoid ambiguities.

Using the kernel trick, the estimated output at discrete time t can be expressed as follows:

$$\hat{f}(\mathbf{u}_t) = \sum_{j=1}^{t-1} a_j\kappa(\mathbf{u}_j,\mathbf{u}_t), \tag{14}$$

where a_j is the coefficient. KAFs can be applied for online nonlinear tasks but cannot be effectively applied to non-stationary ones since they only use a single kernel function. The multikernel adaptive filters using multiple kernels are therefore introduced to address this issue [29], [30].

For multikernel adaptive filtering, multiple Gaussian kernels with different kernel bandwidths are generally used. Hence, (14) can be rewritten as

$$\hat{f}(\mathbf{u}_t) = \sum_{i=1}^{k}\sum_{j=1}^{t-1} a_j^{(i)}\kappa^{(i)}(\mathbf{u}_j,\mathbf{u}_t), \tag{15}$$

where $k \geq 1$ is the number of kernels.

Since the quadratic similarity measure, i.e., mean square error (MSE), is adopted in traditional multikernel adaptive filters [29], [30], the convergence cannot be guaranteed in impulsive noise environments. To this end, we present a novel robust multikernel adaptive filter based on the minimum Cauchy kernel loss criterion for stability improvement in the following.

B. MKMCKL ALGORITHM

Using the multikernel method, the estimation error can be described by

$$e_t = d_t - \mathbf{\Omega}_t^T\Phi(\mathbf{u}_t), \tag{16}$$

where $\mathbf{\Omega}_t$ is the weight vector in RKHS at discrete time t and $\Phi(\cdot)$ is a nonlinear mapping induced by multiple kernels to transform $\mathbf{u}_t$ into RKHS. Thus, the minimum Cauchy kernel loss criterion (11) is rewritten as

$$\min_{\mathbf{\Omega}}\left\{-\log\left(1+\frac{\exp\left(-\frac{(d_t-\mathbf{\Omega}_t^T\Phi(\mathbf{u}_t))^2}{2\sigma^2}\right)}{\gamma^2}\right)\right\}. \tag{17}$$

Further, the cost function existing in (17) is written as

$$J(e_t) = -\log\left(1+\frac{\exp\left(-\frac{e_t^2}{2\sigma^2}\right)}{\gamma^2}\right). \tag{18}$$

Then, taking the gradient of $J(e_t)$ regarding $\mathbf{\Omega}_t$ and using the stochastic gradient descent method generate the following weight update form:

$$\mathbf{\Omega}_t = \mathbf{\Omega}_{t-1} - \mu\nabla_{\mathbf{\Omega}_t}J(e_t) = \mathbf{\Omega}_{t-1} + \mu h(e_t)\Phi(\mathbf{u}_t), \tag{19}$$

where $\mu > 0$ is the step-size parameter and $h(e_t)$ is given by

$$h(e_t) = \frac{\exp\left(-e_t^2/(2\sigma^2)\right)e_t}{1+\exp\left(-e_t^2/(2\sigma^2)\right)/\gamma^2}. \tag{20}$$

In (19), the dimension of $\Phi(\mathbf{u}_t)$ may be very high such that it is difficult to derive $\mathbf{\Omega}_t$ using (19) directly. Thus, an alternative method in a recursive form is presented as follows:

$$\begin{aligned}\mathbf{\Omega}_t &= \mathbf{\Omega}_{t-1} + \mu h(e_t)\Phi(\mathbf{u}_t)\\ &= \mathbf{\Omega}_{t-2} + \mu h(e_{t-1})\Phi(\mathbf{u}_{t-1}) + \mu h(e_t)\Phi(\mathbf{u}_t)\\ &\cdots\\ &= \mathbf{\Omega}_0 + \mu\sum_{j=1}^{t-1}\sum_{i=1}^{k} h(e_j)\Phi_i(\mathbf{u}_j)\end{aligned}$$

Algorithm 1 MKMCKL Algorithm

```
Input: ${\mathbf{u}_t, d_t} \in \mathbb{R}^d \times \mathbb{R}$ $(t = 1, 2, \ldots)$.
Initialization: $\mu > 0, \gamma > 0, \sigma > 0,$
$h > 0, \kappa^{(i)}(\cdot,\cdot)(i = 1, 2, \ldots, k),$
$\mathbf{a}_1(1) = \mu d_1; \mathcal{C}_1 = \{\mathbf{u}_1\}, y_1 = \sum_{i=1}^{k} \mathbf{a}_1(1)\kappa^{(i)}(\mathbf{u}_1, \cdot).$
Computation:
While $\{\mathbf{u}_t, d_t\}$ is available, do
  $y_t = \sum_{j=1}^{t-1}\sum_{i=1}^{k} \mathbf{a}_j(t-1)\kappa^{(i)}(\mathbf{u}_j, \mathbf{u}_t),$
  $e_t = d_t - y_t,$
  $\mathcal{C}_t = \{\mathcal{C}_{t-1}, \mathbf{u}_t\},$
  $\mathbf{a}_t(t) = \mu \frac{\exp(-e_t^2/(2\sigma^2))e_t}{1+\exp(-e_t^2/(2\sigma^2))/\gamma^2}.$
end
```

$$= \mu \sum_{j=1}^{t-1}\sum_{i=1}^{k} h(e_j)\kappa^{(i)}(\mathbf{u}_j, \cdot), \tag{21}$$

where $\mathbf{\Omega}_0$ is assumed to be a zero vector in the last line. Then, the estimated output at discrete time t is given by

$$\begin{aligned} y_t &= \mathbf{\Omega}_t^T \Phi(\mathbf{u}_t) = \langle \mathbf{\Omega}_t, \Phi(\mathbf{u}_t)\rangle \\ &= \mu \sum_{j=1}^{t-1}\sum_{i=1}^{k} h(e_j)\left\langle \kappa^{(i)}(\mathbf{u}_j, \cdot), \kappa^{(i)}(\mathbf{u}_t, \cdot)\right\rangle, \end{aligned} \tag{22}$$

where $y_t = \hat{f}(\mathbf{u}_t)$ denotes the estimated output at discrete time t for simplicity, and $\langle\cdot,\cdot\rangle$ denotes the inner product. Using the kernel trick [1], $\langle \kappa^{(i)}(\mathbf{u}_j, \cdot), \kappa^{(i)}(\mathbf{u}_t, \cdot)\rangle = \kappa^{(i)}(\mathbf{u}_j, \mathbf{u}_t)$ can be obtained efficiently. Thus, the estimated output can be rewritten as follows:

$$y_t = \mu \sum_{j=1}^{t-1}\sum_{i=1}^{k} h(e_j)\kappa^{(i)}(\mathbf{u}_j, \mathbf{u}_t). \tag{23}$$

Finally, let $\mathbf{a}(t)$ and $\mathcal{C}_t$ be the coefficient vector and the corresponding dictionary at discrete time t, respectively, and $\mathbf{a}_j(t)$ the jth element of $\mathbf{a}(t)$, then we summarise the MKMCKL algorithm in *Algorithm* 1, specifically.

Remark 2: Thanks to the use of Cauchy kernel loss, the proposed MKMCKL can combat impulsive noises including large outliers, efficiently. The used multiple kernels also lead to the efficiency improvement of MKMCKL in non-stationary systems. In addition, when a single kernel is used, the MKMCKL reduces to the standard kernel minimum Cauchy kernel loss (KMCKL) algorithm, which can be applied to stationary systems, effectively.

C. SMKMCKL ALGORITHM

Similar to the traditional KAFs, MKMCKL generates a growing network structure. Especially when the size of input data is very large, MKMCKL results in huge computational and storage burdens, which leads to the inefficiency of MKMCKL in practical applications. Thus, we present a sparsification method for MKMCKL to reduce the complexity.

Algorithm 2 SMKMCKL Algorithm

```
Input: $\{\mathbf{u}_t, d_t\} \in \mathbb{R}^d \times \mathbb{R}$ $(t = 1, 2, \ldots)$.
Initialization: $\mu > 0, \delta > 0, \gamma > 0, \sigma > 0, D > 0,$
$h > 0, \kappa^{(i)}(\cdot,\cdot)(i = 1, 2, \ldots, k), M = 1$
$\mathbf{a}_1(1) = \mu d_1, \mathcal{C}_1 = \{\mathbf{u}_1\}, y_1 = \sum_{i=1}^{k} \mathbf{a}_1(1)\kappa^{(i)}(\mathbf{u}_1, \cdot).$
Computation:
While $\{\mathbf{u}_t, d_t\}$ is available, do
  $y_t = \sum_{j=1}^{\mathrm{size}(\mathcal{C}_{t-1})}\sum_{i=1}^{k} \mathbf{a}_{t-1}(j)\kappa^{(i)}(\mathcal{C}_{t-1}(j), \mathbf{u}_t),$
  $e_t = d_t - y_t, h(e_t) = \frac{\exp(-e_t^2/(2\sigma^2))e_t}{1+\exp(-e_t^2/(2\sigma^2))/\gamma^2},$
  $dis = \min_{j\in \mathrm{size}(\mathcal{C}_{t-1})} \mathrm{dis}(\mathcal{C}_{t-1}(j), \mathbf{u}_t),$
  $j^* = \arg\min_{j\in \mathrm{size}(\mathcal{C}_{t-1})} \mathrm{dis}(\mathcal{C}_{t-1}(j), \mathbf{u}_t),$
  if $dis > \delta$ & $M \geq D$
   $M = D,$
   $\mathcal{C}_t = \{\breve{\mathcal{C}}_{t-1}, \mathbf{u}_t\}, \mathbf{a}(t) = \left[\breve{\mathbf{a}}(t-1), \mu h(e_t)\right]^T,$
  else if $dis \leq \delta$
     $\mathcal{C}_t = \mathcal{C}_{t-1}, \mathbf{a}_{j^*}(t) = \mathbf{a}_{j^*}(t-1) + \mu h(e_t),$
    else
     $M = M + 1,$
     $\mathcal{C}_t = \{\mathcal{C}_{t-1}, \mathbf{u}_t\}, \mathbf{a}(t) = [\mathbf{a}(t-1), \mu h(e_t)]^T.$
    end
  end
end
```

Owing to the improved computational efficiency and tracking performance, the online vector quantization method [36] and the sliding-window method [45] are combined into MKMCKL to generate the SMKMCKL algorithm. Let $D > 0$ be the sliding-window size, $M > 0$ the dictionary size, and $\mathrm{dis}(\mathcal{C}_{t-1}, \mathbf{u}_t)$ the distance (Euclidean metric) between input $\mathbf{u}_t$ and dictionary $\mathcal{C}_{t-1}$. Then, the following sparsification criterion is presented:

$$\begin{cases} \min\limits_{j\in \mathrm{size}(\mathcal{C}_{t-1})} \mathrm{dis}(\mathcal{C}_{t-1}(j), \mathbf{u}_t) > \delta \\ M \geq D, \end{cases} \tag{24}$$

where $\delta > 0$ is the quantization size and $\mathcal{C}_{t-1}(j)$ denotes the jth element of $\mathcal{C}_{t-1}$. Thus, when the input $\mathbf{u}_t$ arrives, if (24) is satisfied, the sliding-window method [45] is used, and otherwise the traditional online vector quantization method [36] is used. Specifically, the SMKMCKL algorithm is described as follows.

Case 1: When (24) is satisfied, the first element of dictionary $\mathcal{C}_{t-1}$ is removed and $\mathbf{u}_t$ is added as the Dth element, i.e., $\mathcal{C}_t = \{\breve{\mathcal{C}}_{t-1}, \mathbf{u}_t\}$ with $\breve{\mathcal{C}}_{t-1}$ being the dictionary obtained by removing the first element from $\mathcal{C}_{t-1}$. And the coefficient is updated as

$$\mathbf{a}(t) = \left[\breve{\mathbf{a}}(t-1), \mu h(e_t)\right]^T, \tag{25}$$

where $\breve{\mathbf{a}}(t-1)$ denotes the coefficient obtained by removing the first element from $\mathbf{a}(t-1)$.

Case 2: When (24) is not satisfied, if $\min_{j\in\text{size}(\mathcal{C}_{t-1})} \text{dis}(\mathcal{C}_{t-1}(j), \mathbf{u}_t) \leq \delta$, keep the dictionary unchanged, i.e., $\mathcal{C}_t = \mathcal{C}_{t-1}$, and update the coefficient as

$$\mathbf{a}_{j^*}(t) = \mathbf{a}_{j^*}(t-1) + \mu h(e_t), \tag{26}$$

where $j^* = \arg\min_{j\in\text{size}(\mathcal{C}_{t-1})} \text{dis}(\mathcal{C}_{t-1}(j), \mathbf{u}_t)$. Otherwise, $\mathbf{u}_t$ is added into the dictionary, i.e., $\mathcal{C}_t = \{\mathcal{C}_{t-1}, \mathbf{u}_t\}$, and the coefficient is updated as

$$\mathbf{a}(t) = [\mathbf{a}(t-1), \mu h(e_t)]^T. \tag{27}$$

Finally, the SMKMCKL algorithm is summarised in *Algorithm* 2, specifically.

Remark 3: The SMKMCKL algorithm is obtained by combining the online vector quantization method [36] and the sliding-window method [45], leading to the improvement of computational efficiency and tracking performance. In addition, the other sparsification methods [3], [33]–[35] and fixed-budget methods [46], [47] can also be applicable for MKMCKL, which is beyond the scope of this work.

IV. SIMULATIONS RESULTS

Monte Carlo (MC) simulations on the prediction of a Mackey-Glass (MG) chaotic time-series and the nonlinear regression are performed to validate the effectiveness of the proposed MKMCKL and SMKMCKL algorithms, and the other typical algorithms are used for performance comparison. Specifically, KLMS [2], KLMP [7], KMC [10], and MKLMS [30] are used for comparison with MKMCKL, and QKLMP [7] and QKMC [37] for comparison with SMKMCKL. The mean square error (MSE) defined by $\text{MSE}\,(\text{dB}) = 10\log_{10}\left(\sum_{t=1}^{L}(d_t - y_t)^2/L\right)$, where d_t denotes the desired output, y_t denotes the predicted output, and L is the length of testing data, is used for performance evaluation. To demonstrate the robustness against impulsive noises, the desired output is contaminated by the noise modeled with $n_t = n_{1,t} + b_t n_{2,t}$, where $n_{1,t}$ is an ordinary zero-mean Gaussian noise with variance $\sigma_1^2 = 0.01$, $n_{2,t}$ is an α-stable noise with parameters $\{1.2, 0, 1, 0\}$ [48] to generate large outliers, and b_t is a Bernoulli random process with $Pr\{b_t = 1\} = c$, $Pr\{b_t = 0\} = 1 - c$, and $0 \leq c \leq 1$ ($c = 0.1$ is chosen here) [16]. In the following simulations, the Gaussian kernels with different bandwidths are used to represent multiple kernels, and all the results are obtained as averages over 50 independent runs.

A. MG TIME-SERIES PREDICTION

Mackey-Glass time-series is used for prediction tasks in KAFs thanks to its periodic and chaotic natures, which is decribed from the following differential equation [1]:

$$\frac{dx(t)}{dt} = -bx(t) + \frac{ax(t-\tau)}{1 + x(t-\tau)^n}, \tag{28}$$

with $a = 0.2$, $b = 0.1$, and $n = 10$. In (28), the value of time lag τ can reflect its dynamic nature, and we use $\tau = 30$ to generate a chaotic time-series discretized at a sampling period of 6 seconds for prediction. And the prediction task is to use the previous seven time-series to predict the current one, i.e., the input-output pair is denoted by $\{\mathbf{u}_t, d_t\}$ with $\mathbf{u}_t = [x(t-7), x(t-6), \ldots, x(t-1)]^T$ and $d_t = x(t)$. The noisy MG time-series of length 2000 are used for training and the other clean time-series of length 200 for testing.

1) PERFORMANCE OF MKMCKL

We first validate the influence of the number of kernels on the performance of MKMCKL. For fair comparison, free parameters $\gamma = 1$ and $\sigma = 1$, and step-size $\mu = 0.1$ are configured in MKMCKL with different kernels. The testing MSEs of MKMCKL with different kernels are shown in Fig. 3. From Fig. 3, we see that the number of kernels can affect the convergence speed and filtering accuracy of MKMCKL, and MKMCKL using two kernels can achieve desirable performance. Especially, Fig. 3 illustrates that MKMCKL using multiple kernels outperforms MKMCKL using a single kernel (i.e., a standard KAF). Therefore, in the following simulations, two kernels are used in multikernel algorithms unless otherwise specified.

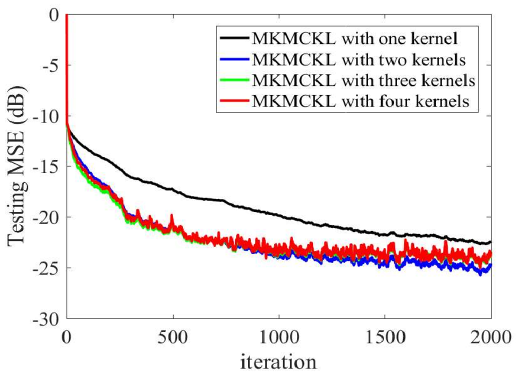


FIGURE 3. Testing MSEs of MKMCKL with different kernels in MG time-series prediction.

Then, we validate the influence of γ on the performance of MKMCKL. Two kernel parameters are set to $h_1 = 0.5$ and $h_2 = 1.5$, and $\sigma = 1$ and $\mu = 0.1$ are configured in MKMCKL. The testing MSEs of MKMCKL with different γ are shown in Fig. 4. It can be seen from Fig. 4 that a large γ can improve the convergence rate and reduce the steady-state error. However, when γ is larger than a certain value ($\gamma = 1.5$), the performance of MKMCKL is not changed. The effect of σ on the performance of MKMCKL is also validated. $\gamma = 1$ is configured in MKMCKL, and other parameters are the same as those in Fig. 4. The testing MSEs of MKMCKL with different σ are shown in Fig. 5. From Fig. 5, we see that $\sigma = 0.5$ can achieve desirable performance. Hence, in practical applications, we choose the values of γ and σ of MKMCKL to achieve desirable performance by trials.

To show the superiority of MKMCKL over other algorithms, we compare the MSE performance of MKMCKL, KLMS, KLMP, KMC, and MKLMS. The same two kernels as those in Fig. 4 are configured in MKMCKL and MKLMS,

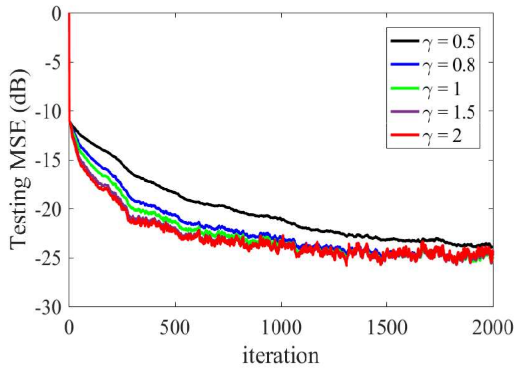


FIGURE 4. Testing MSEs of MKMCKL with different γ in MG time-series prediction.

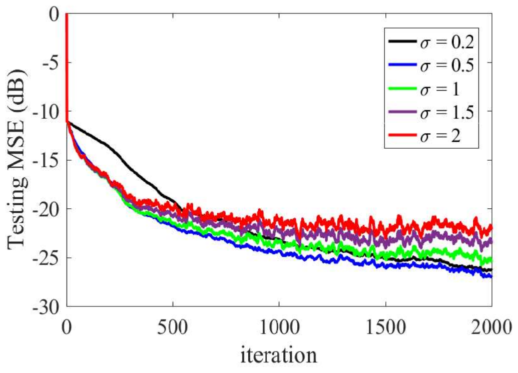


FIGURE 5. Testing MSEs of MKMCKL with different σ in MG time-series prediction.

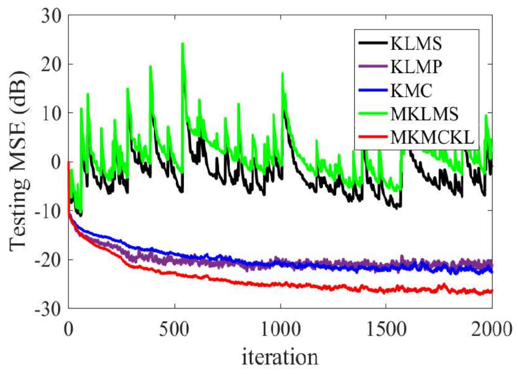


FIGURE 6. Testing MSEs of KLMS, KLMP, KMC, MKLMS, and MKMCKL in MG time-series prediction.

and the Gaussian kernel with kernel bandwidth $h = 1$ is configured in KLMS, KLMP, and KMC. The other parameters of algorithms are chosen such that each algorithm achieves desirable performance, which are specifically set as: step-size $\mu = 0.1$ for all algorithms; $p = 1.2$ for KLMP; $\sigma = 1.8$ for KMC; $\gamma = 2$ and $\sigma = 0.5$ for MKMCKL. The compared testing MSEs are shown in Fig. 6. From Fig. 6, we see that the quadratic loss based algorithms (i.e., KLMS and MKLMS) cannot combat impulsive noises, and the proposed

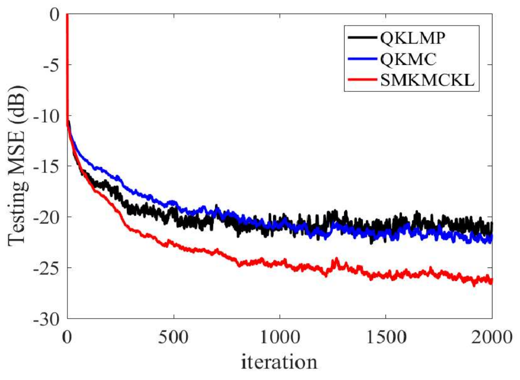


FIGURE 7. Testing MSEs of QKLMP, QKMC, and SMKMCKL in MG time-series prediction.

MKMCKL can combat impulsive noises efficiently and provides the fastest convergence speed and best filtering accuracy in all the compared algorithms. Therefore, we only use the robust algorithms or their sparsification forms for performance comparison in the following simulations.

2) PERFORMANCE OF SMKMCKL

Since SMKMCKL only prunes the redundant data in MKMCKL, the influence of kernels on the performance in SMKMCKL is the same as that in MKMCKL. Hence, we only perform the performance comparison of SMKMCKL with other robust sparse KAFs (i.e., QKLMP and QKMC). For fair comparison, the parameters of all algorithms are chosen such that each algorithm achieves desired performance. Specifically, quantization size $\delta = 0.1$ is configured for QKLMP, QKMC, and SMKMCKL, and sliding-window size $D = 100$ for SMKMCKL. And the other parameters are the same as those in Fig. 6. The compared testing MSEs are shown in Fig. 7, and we see from this figure that SMKMCKL outperforms QKLMP and QKMC from the aspects of convergence speed and filtering accuracy.

In addition, to further illustrate the superiority of SMKMCKL in terms of the choice of kernel parameters, we compare the steady-state MSE performance of SMKMCKL, QKLMP, and QKMC under different kernel parameters. In QKLMP and QKMC, the kernel parameter is changed from 0.1 to 2, and in SMKMCKL, h_1 is fixed to 0.5 and h_2 is changed in the same way as that in QKLMP and QKMC. The other parameters are the same as those in Fig. 7. Fig. 8 shows the compared steady-state MSEs of all algorithms, where the steady-state MSEs are calculated as averages over the last 200 iterations. We see from Fig. 8 that the steady-state MSE of SMKMCKL is less sensitive to kernel parameters and SMKMCKL has the best filtering accuracy in the compared algorithms.

B. NONLINEAR REGRESSION

The example of nonlinear regression is considered, which can be described as the following dynamic nonlinear

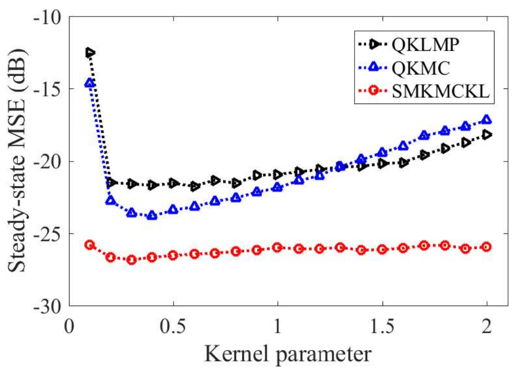


FIGURE 8. Testing steady-state MSEs of QKLMP, QKMC, and SMKMCKL under different kernel parameters in MG time-series prediction.

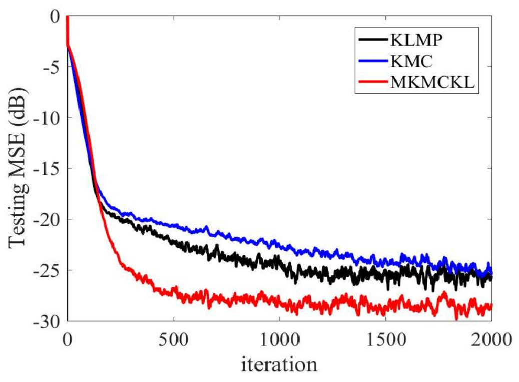


FIGURE 9. Testing MSEs of KLMP, KMC, and MKMCKL in nonlinear regression.

system [49]:

$$\begin{aligned} x(t) = &(0.8 - 0.5\exp(-x(t-1)^2))x(t-1) \\ &-(0.3 + 0.9\exp(-x(t-1)^2))x(t-2) \\ &+0.1\sin(x(t-1)\pi), \end{aligned} \tag{29}$$

with $x(1) = 0.1$ and $x(2) = 0.1$. The input-output pair of this example is given by $\{\mathbf{u}_t, d_t\}$ with $\mathbf{u}_t = [x(t-2), x(t-1)]^T$ and $d_t = x(t)$. In the following simulations, the noisy time-series of length 2000 are used for training and the other clean time-series of length 200 for testing.

1) PERFORMANCE OF MKMCKL

Similar to Fig. 3, the performance of MKMCKL under different numbers of kernels is validated, of which results are not presented here to conserve the space. The simulation results show that the MKMCKL using two kernels can achieve desirable performance. Then, we compare the MSE performance of MKMCKL with those of KLMP and KMC. The selection of kernel in this simulation is the same as that in Fig. 6, and the other parameters are chosen such that each algorithm achieves desirable performance, which are specifically as follows: $\mu = 0.05$ and $p = 1.2$ for KLMP; $\mu = 0.1$ and $\sigma = 1.4$ for KMC; $\mu = 0.1$, $\gamma = 3$, and $\sigma = 0.5$ for MKMCKL. The compared results are shown in Fig. 9. From Fig. 9, we see that MKMCKL provides the fastest convergence rate and best filtering accuracy in all the compared algorithms

2) PERFORMANCE OF SMKMCKL

The MSE performance of SMKMCKL is compared to those of QKLMP and QKMC. For fair comparison, the quantization size $\delta = 0.02$ is configured in these three algorithms, and the other parameters are the same as those in Fig. 9. The compared MSEs are shown in Fig. 10, and the obtained simulation results also show that SMKMCKL provides the best performance in terms of convergence rate and filtering accuracy in all the compared algorithms.

The influence of the selection of kernel parameters on the performance of SMKMCKL, QKLMP, and QKMC is also validated in this example. The same simulation settings as those in Fig. 9 are used and the kernel parameters are set in the same way as those in Fig. 8. The steady-state MSEs of SMKMCKL, QKLMP, and QKMC versus kernel parameters are shown in Fig. 11. From Fig. 11, we have that the steady-state MSE of SMKMCKL is not sensitive to kernel parameters while these of QKLMP and QKMC are sensitive. And, SMKMCKL provides the best filtering accuracy in all the compared algorithms under different kernel parameters.

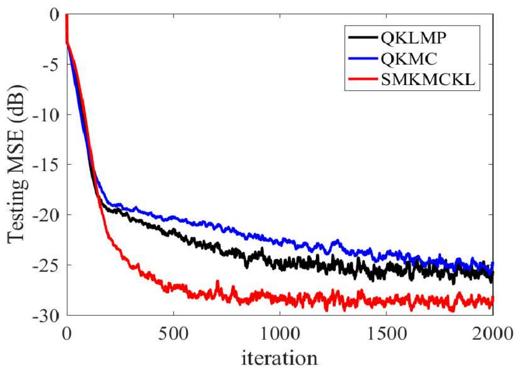


FIGURE 10. Testing MSEs of QKLMP, QKMC, and SMKMCKL in nonlinear regression.

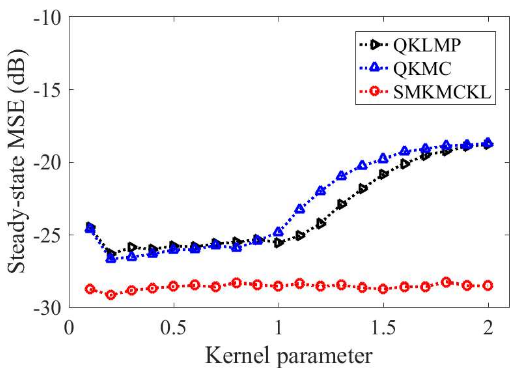


FIGURE 11. Testing steady-state MSEs of QKLMP, QKMC, and SMKMCKL under different kernel parameters in nonlinear regression.

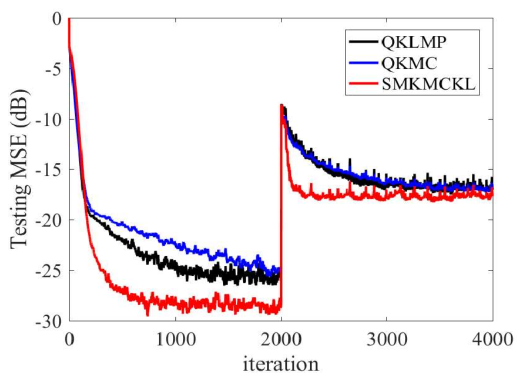


FIGURE 12. Testing MSEs of QKLMP, QKMC, and SMKMCKL in a non-stationary system.

Finally, the tracking performance of SMKMCKL, QKLMP, and QKMC in a non-stationary system is validated. To model such a non-stationary system, the nonlinear system (29) is changed to Mackey-Glass chaotic system (28) at the 2001th iteration. The other settings are the same as those in Fig. 10, and the testing MSEs of SMKMCKL, QKLMP, and QKMC are shown in Fig. 12. It can be seen from Fig. 12 that SMKMCKL can track the time-varying system efficiently and provides the best tracking performance in all the compared algorithms.

V. CONCLUSION

A novel Cauchy kernel loss is presented in this paper. Under the minimum Cauchy kernel loss criterion, a multikernel minimum Cauchy kernel loss (MKMCKL) algorithm is therefore proposed for robust learning. The proposed MKMCKL algorithm can combat impulsive noises including large outliers in non-stationary systems, effectively. To further reduce the complexity of MKMCKL, a new sparsification method is presented using the combination of online vector quantization method and sliding-window method for MKMCKL, generating a sparse MKMCKL (SMKMCKL) algorithm. By applying MKMCKL and SMKMCKL to the prediction of Mackey-Glass chaotic time-series and the nonlinear regression in the presence of impulsive noises, the proposed algorithms show the superiorities over other traditional algorithms from the aspects of robustness, filtering accuracy, choice of kernel parameters, and tracking performance.

APPENDIX

The correntropic loss (C-Loss) using a Gaussian kernel is defined by [8]

$$L_{C-Loss}(X,Y) = 1 - E\left[\exp\left(-\frac{(X-Y)^2}{2\sigma^2}\right)\right]. \quad (30)$$

Thus, the Cauchy kernel loss (5) can be seen as a logarithmic C-Loss. In addition, from Property 2 a), we have that the Cauchy kernel loss can reduce to the C-Loss when γ is large enough or σ is small enough. Hence, compared with the C-Loss, the Cauchy kernel loss is a more general loss.

To further show the superiority of Cauchy kernel loss over C-Loss, we derive the Hessian matrix of C-Loss by using the same method as that in the proof of Property 3 as follows:

$$\mathbf{H}_{\hat{L}_{C-Loss}(X,Y)}(\mathbf{e}) = \mathrm{diag}[\theta_1, \theta_2, \ldots, \theta_N], \quad (31)$$

where

$$\theta_i = \frac{1}{N\sigma^2}\left(1 - \frac{e_i^2}{\sigma^2}\right)\exp\left(-\frac{e_i^2}{2\sigma^2}\right). \quad (32)$$

Thus, to guarantee the convexity of C-Loss, $1-\frac{e_i^2}{\sigma^2} > 0$ should be satisfied. Compared with the convex condition of Cauchy kernel loss, i.e., $\gamma^2\left(1-\frac{e_i^2}{\sigma^2}\right) + \exp\left(-\frac{e_i^2}{2\sigma^2}\right) > 0$, a larger convex range of Cauchy kernel loss is directly obtained, especially when a small γ is chosen.

REFERENCES

[1] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. New York, NY, USA: Wiley, 2010.

[2] W. Liu, P. P. Pokharel, and J. C. Príncipe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 543–554, Feb. 2008.

[3] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[4] K. Xiong, S. Wang, and B. Chen, "Robust normalized least mean absolute third algorithms," *IEEE Access*, vol. 7, pp. 10318–10330, 2019.

[5] E. Eweda, "Stabilization of high-order stochastic gradient adaptive filtering algorithms," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 3948–3959, Aug. 2017.

[6] V. Mathews and S. Cho, "Improved convergence analysis of stochastic gradient adaptive filters using the sign algorithm," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-35, no. 4, pp. 450–454, Apr. 1987.

[7] W. Ma, J. Duan, W. Man, H. Zhao, and B. Chen, "Robust kernel adaptive filters based on mean p-power error for noisy chaotic time series prediction," *Eng. Appl. Artif. Intell.*, vol. 58, pp. 101–110, Feb. 2017.

[8] W. Liu, P. P. Pokharel, and J. C. Príncipe, "Correntropy: Properties and applications in non-Gaussian signal processing," *IEEE Trans. Signal Process.*, vol. 55, no. 11, pp. 5286–5298, Nov. 2007.

[9] A. Singh and J. C. Príncipe, "Using correntropy as a cost function in linear adaptive filters," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2009, pp. 2950–2955.

[10] Y. Liu and J. Chen, "Correntropy kernel learning for nonlinear system identification with outliers," *Ind. Eng. Chem. Res.*, vol. 53, no. 13, pp. 5248–5260, Nov. 2013.

[11] B. Chen, L. Xing, H. Zhao, N. Zheng, and J. C. Príncipe, "Generalized correntropy for robust adaptive filtering," *IEEE Trans. Signal Process.*, vol. 64, no. 13, pp. 3376–3387, Jul. 2016.

[12] Y. He, F. Wang, J. Yang, H. Rong, and B. Chen, "Kernel adaptive filtering under generalized maximum correntropy criterion," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 1738–1745.

[13] J. C. Príncipe, *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*. New York, NY, USA: Springer, 2010.

[14] B. Chen, L. Xing, B. Xu, H. Zhao, N. Zheng, and J. C. Príncipe, "Kernel risk-sensitive loss: Definition, properties and application to robust adaptive filtering," *IEEE Trans. Signal Process.*, vol. 65, no. 11, pp. 2888–2901, Jun. 2017.

[15] B. Chen, L. Xing, X. Wang, J. Qin, and N. Zheng, "Robust learning with kernel mean *p*-power error loss," *IEEE Trans. Cybern.*, vol. 48, no. 7, pp. 2101–2113, Jul. 2018.

[16] M. O. Sayin, N. D. Vanli, and S. S. Kozat, "A novel family of adaptive filtering algorithms based on the logarithmic cost," *IEEE Trans. Signal Process.*, vol. 62, no. 17, pp. 4411–4424, Sep. 2014.

[17] K. Xiong and S. Wang, "Robust least mean logarithmic square adaptive filtering algorithms," *J. Franklin Inst.*, vol. 356, no. 1, pp. 654–674, Jan. 2019.

[18] S. Wang, W. Wang, K. Xiong, H. H. C. Iu, and C. K. Tse, "Logarithmic hyperbolic cosine adaptive filter and its performance analysis," *IEEE Trans. Syst., Man, Cybern. Syst.*, to be published. doi: 10.1109/TSMC.2019.2915663.

[19] M. Colin Gallagher, T. J. Fisher, and J. Shen, "A cauchy estimator test for autocorrelation," *J. Stat. Comput. Simul.*, vol. 85, no. 6, pp. 1264–1276, 2015.

[20] N. Guan, T. Liu, Y. Zhang, D. Tao, and L. S. Davis, "Truncated cauchy non-negative matrix factorization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 1, pp. 246–259, 2019.

[21] B. Scholkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002.

[22] S. An, W. Liu, and S. Venkatesh, "Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression," *Pattern Recognit.*, vol. 40, no. 8, pp. 2154–2162, Aug. 2007.

[23] W. Hardle, *Applied Nonparametric Regression*. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[24] E. Herrmann, "Local bandwidth choice in kernel regression estimation," *J. Comput. Graph. Statist.*, vol. 6, no. 1, pp. 35–54, Mar. 1997.

[25] B. Chen, J. Liang, N. Zheng, and J. C. Príncipe, "Kernel least mean square with adaptive kernel size," *Neurocomputing*, vol. 191, pp. 95–106, May 2016.

[26] B.-S. Shin, M. Yukawa, R. L. G. Cavalcante, and A. Dekorsy, "Distributed adaptive learning with multiple kernels in diffusion networks," *IEEE Trans. Signal Process.*, vol. 66, no. 21, pp. 5505–5519, Nov. 2018.

[27] S. S. Bucak, R. Jin, and A. K. Jain, "Multiple kernel learning for visual object recognition: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1354–1369, Jul. 2014.

[28] Y.-Y. Lin, T.-L. Liu, and C.-S. Fuh, "Multiple kernel learning for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 6, pp. 1147–1160, Jun. 2011.

[29] M. Yukawa, "Multikernel adaptive filtering," *IEEE Trans. Signal Process.*, vol. 60, no. 9, pp. 4672–4682, Sep. 2012.

[30] F. A. Tobar, S.-Y. Kung, and D. P. Mandic, "Multikernel least mean square algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 265–277, Feb. 2014.

[31] R. Pokharel, S. Seth, and J. C. Príncipe, "Mixture kernel least mean square," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Aug. 2013, pp. 1–7.

[32] Z. Cao, S. Yu, G. Xu, B. Chen, and J. C. Príncipe, "Multiple adaptive kernel size KLMS for Beijing PM2.5 prediction," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 1403–1407.

[33] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, 1991.

[34] W. Liu, I. Park, and J. C. Príncipe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 1950–1961, Dec. 2009.

[35] L. Csato and M. Opper, "Sparse on-line Gaussian processes," *Neural Comput.*, vol. 14, no. 3, pp. 641–668, Mar. 2002.

[36] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, "Quantized kernel least mean square algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 1, pp. 22–32, Jan. 2012.

[37] S. Wang, Y. Zheng, S. Duan, L. Wang, and H. Tan, "Quantized kernel maximum correntropy and its mean square convergence analysis," *Digit. Signal Process.*, vol. 63, pp. 164–176, Apr. 2017.

[38] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, vol. 20, Dec. 2007, pp. 1177–1184.

[39] S. Wang, L. Dang, B. Chen, S. Duan, L. Wang, and C. K. Tse, "Random Fourier filters under maximum correntropy criterion," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 10, pp. 3390–3403, Oct. 2018.

[40] K. Xiong and S. Wang, "The online random Fourier features conjugate gradient algorithm," *IEEE Signal Process. Lett.*, vol. 26, no. 5, pp. 740–744, May 2019.

[41] I. Jolliffe, *Principal Components Analysis*. Hoboken, NJ, USA: Wiley, 2002.

[42] R. He, W.-S. Zheng, and B.-G. Hu, "Maximum correntropy criterion for robust face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1561–1576, Aug. 2011.

[43] J. P. Rhinelander and X. P. Liu, "Truncation error compensation in kernel machines," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2013, pp. 1889–1894.

[44] X. You, W. Guo, S. Yu, K. Li, and J. C. Príncipe, and D. Tao, "Kernel learning for dynamic texture synthesis," *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4782–4795, Oct. 2016.

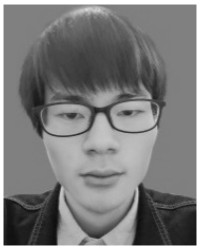
[45] S. Van Vaerenbergh, J. Via, and I. Santamaria, "A sliding-window kernel RLS algorithm and its application to nonlinear channel identification," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. Proc. (ICASSP)*, Toulouse, France, May 2006, pp. 789–792.

[46] B. J. de Kruif and T. J. A. de Vries, "Pruning error minimization in least squares support vector machines," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 696–702, May 2003.

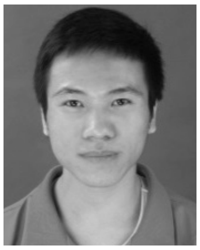
[47] S. V. Vaerenbergh, I. Santamaría, W. Liu, and J. C. Príncipe, "Fixed-budget kernel recursive least-squares," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Dallas, TX, USA, Mar. 2010, pp. 1882–1885.

[48] B. Weng and K. E. Barner, "Nonlinear system identification in impulsive environments," *IEEE Trans. Signal Process.*, vol. 53, no. 7, pp. 2588–2594, Jul. 2005.

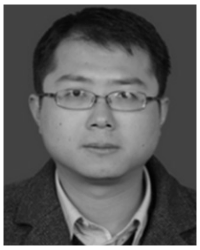
[49] M. A. Takizawa and M. Yukawa, "Adaptive nonlinear estimation based on parallel projection along affine subspaces in reproducing kernel Hilbert space," *IEEE Trans. Signal Process.*, vol. 63, no. 16, pp. 4257–4269, Aug. 2015.



WEI SHI received the B.Eng. degree in optoelectronic information science and engineering from Bengbu University, Bengbu, China, in 2017. He is currently pursuing the M.Eng. degree with the College of Electronic and Information Engineering, Southwest University, Chongqing, China. His current research interests include kernel adaptive filtering and information theoretic learning.



KUI XIONG received the B.Eng. degree in electronic information science and technology from Zhejiang Sci-Tech University, Hangzhou, China, in 2017. He is currently pursuing the M.Eng. degree with the College of Electronic and Information Engineering, Southwest University, Chongqing, China. His current research interests include kernel adaptive filtering and information theoretic learning.



SHIYUAN WANG (M'13–SM'18) received the B.Eng. and M.Eng. degrees in electronic and information engineering from Southwest Normal University, Chongqing, China, in 2002 and 2005, respectively, and the Ph.D. degree in circuit and system from Chongqing University, Chongqing, in 2011.

He was a Research Associate with Hong Kong Polytechnic University, from August 2012 to August 2013. He is currently a Professor with the College of Electronic and Information Engineering, Southwest University, China. He has published one book and more than 50 articles. His research interests include adaptive signal processing, nonlinear dynamics and chaos, and bioinformatics. He is serving as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS.