AN EFFICIENT KERNEL NORMALIZED LEAST MEAN SQUARE ALGORITHM WITH COMPACTLY SUPPORTED KERNEL

*Osamu Toda and Masahiro Yukawa**

Dept. Electronics and Electrical Engineering, Keio University, Japan.
E-mail: toda@ykw.elec.keio.ac.jp, yukawa@elec.keio.ac.jp

ABSTRACT

We investigate the use of compactly supported kernels (CSKs) for the kernel normalized least mean square (KNLMS) algorithm proposed initially by Richard *et al.* in 2009. The use of CSKs yields sparse kernelized input vectors, offering an opportunity for complexity reduction. We propose a simple two-step method to compute the kernelized input vectors efficiently. In the first step, it computes an over-estimation of the support of the kernelized input vector based on a certain ℓ_1-ball. In the second step, it identifies the exact support by detailed examinations based on an ℓ_2-ball. Also, we employ the identified support given by the second step for coherence construction. The proposed method reduces the amount of ℓ_2-distance evaluations, leading to the complexity reduction. The numerical examples show that the proposed algorithm achieves significant complexity reduction.

Index Terms— Kernel learning, Gaussian kernel, Radial basis function, Positive definite function, Compactly supported function.

1. INTRODUCTION

Gaussian kernel is one of the radial basis functions which depend on the distance between two samples and does not depend on the direction. This particular property is one of the reasons for the popularity of Gaussian kernel for approximating unknown functions. In the literature of kernel adaptive filtering (kernel online learning), the Gaussian kernel has widely been employed [1–7]. From the computational aspects, the Gaussian kernel is not necessarily perfect because many practical applications such as kernel ridge regression require the inversion of the kernel matrix which is dense in the case of Gaussian kernels. Compactly supported and positive definite functions have been studied in statistics [8–12]. These functions have been applied to scattered data approximation and surface reconstruction as well as machine learning problems [13–17]. A remarkable advantage of the compactly supported and positive definite functions is that the kernel matrix becomes sparse, which reduces the computational burden of kernel methods [17].

The kernel normalized least mean square (KNLMS) algorithm [4] is one of the simplest examples of kernel adaptive filtering algorithms. The algorithm is free from the computation of the kernel matrix inversion since it projects the current estimate onto the zero-instantaneous hyperplane in the Euclidean space. However, the length of the kernelized input vector, which gives the normal vector of the hyperplane, is the size of the dictionary, which implies that the complexity for computing the kernelized input vector and the projection increases linearly with the dictionary size.

In this paper, we focus on the sparseness of the kernelized input vector for compactly supported kernels (CSKs) and propose an efficient KNLMS algorithm with CSKs. The support of the kernelized input vector corresponds to those data points in the dictionary which are contained by the ℓ_2-ball centered at the current data; the radius is determined by the support of the CSK employed. To avoid computing the ℓ_2 distances from the current data to all the dictionary data, we propose a simple two-step method. In the first step, it computes an over-estimate of the support based on the concentric circumscribed ℓ_1-ball of the ℓ_2-ball. In the second step, it identifies the exact support by computing the ℓ_2 distances from the current data to only those dictionary data which are contained by the ℓ_1-ball. Also, we employ the identified support given by the second step for coherence construction. This proposed method reduces the amount of ℓ_2-distance evaluations drastically, leading to a substantial reduction of the computational loads. The numerical examples show that the proposed algorithm achieves significant complexity reduction.

*This work was partially supported by JSPS Grants-in-Aid (24760292).

2. PRELIMINARIES

This section introduces the notation and the compactly supported kernels [8, 9, 11].

2.1. Notation and the KNLMS Algorithm

Throughout this paper, let $\mathbb{R}$, $\mathbb{N}$, and $\mathbb{N}^*$ denote the sets of all real numbers, nonnegative integers, and positive integers, respectively. We denote by $(\cdot)^\mathsf{T}$ the transpose. Also we denote by $\|\cdot\|_1$ and $\|\cdot\|_2$ the ℓ_1 norm and the ℓ_2 norm, respectively, of vectors. The task of kernel adaptive filtering is to estimate a nonlinear function $\psi : \mathcal{U} \to \mathbb{R}$ in an online fashion with sequentially arriving data $(\boldsymbol{u}_n, d_n)_{n\in\mathbb{N}} \subset \mathcal{U} \times \mathbb{R}$, where $\mathcal{U} \subset \mathbb{R}^L$ and $\mathbb{R}$ are the input and output spaces, respectively, and n is the time index. Let $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ denote a positive definite kernel. Let $\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j\in\mathcal{J}}$ be the dictionary indicated by the index set $\mathcal{J}_n := \left\{j_1^{(n)}, j_2^{(n)}, \cdots, j_{r_n}^{(n)}\right\} \subset \{0, 1, \cdots, n-1\}$, where $r_n \in \mathbb{N}^*$ is the dictionary size. A kernel adaptive filter is then given by

$$\varphi_n(\boldsymbol{u}) = \sum_{j\in\mathcal{J}_n} h_j^{(n)} \kappa(\boldsymbol{u}, \boldsymbol{u}_j), \quad \boldsymbol{u} \in \mathcal{U}, \quad n \in \mathbb{N}, \tag{1}$$

where $h_j^{(n)} \in \mathbb{R}$. An estimate of d_n can be expressed in a vector form as follows:

$$\hat{d}_n := \varphi_n(\boldsymbol{u}_n) = \boldsymbol{h}_n^\mathsf{T} \boldsymbol{k}_n, \tag{2}$$

where the ith components of $\boldsymbol{h}_n \in \mathbb{R}^{r_n}$ and $\boldsymbol{k}_n \in \mathbb{R}^{r_n}$ are given by $[\boldsymbol{h}_n]_i := h_{j_i^{(n)}}^{(n)}$ and $[\boldsymbol{k}_n]_i := \kappa\left(\boldsymbol{u}_n, \boldsymbol{u}_{j_i^{(n)}}\right)$, respectively. Initialize the dictionary index set $\mathcal{J}_{-1} := \emptyset$ and the filter $\boldsymbol{h}_0 := \emptyset$. KNLMS [4] designs the dictionary index set $\mathcal{J}_n$ based on the coherence cri-

Table 1. Gaussian and compactly supported functions for $L = 2$, $\rho := \|\boldsymbol{x}-\boldsymbol{y}\|_2$, $\alpha := 1/2\sigma^2$, and $\tilde{\rho} := \rho^2/c^2$; $(x)_+ := \max\{x, 0\}$ for $x \in \mathbb{R}$.

Name	Function
Gaussian	$\psi_{\mathrm{G}}(\rho) = \exp\left(-\alpha\rho^2\right)$
Matérn's CSK [8]	$\psi_{\mathrm{Mrt}}(\rho) = (1-\tilde{\rho})_+^2$
Wendland's CSK [9]	$\psi_{\mathrm{Wnd}}(\rho) = (1-\tilde{\rho})_+^6 \left(35\tilde{\rho}^2+18\tilde{\rho}+3\right)/3$
Wu's CSK [11]	$\psi_{\mathrm{Wu}}(\rho) = (1-\tilde{\rho})_+^4 \left(3\tilde{\rho}^3+12\tilde{\rho}^2+16\tilde{\rho}+4\right)/4$

terion as follows:

$$\mathcal{J}_{n+1} := \begin{cases} \mathcal{J}_n \cup \{n\}, & \text{if } \min_{j\in\mathcal{J}_n} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_j)| \le \mu_0, \\ \mathcal{J}_n, & \text{otherwise,} \end{cases} \tag{3}$$

where $\mu_0 \in (0, 1)$ is the coherence threshold. If $\mathcal{J}_{n+1} \neq \mathcal{J}_n$, then we let $h_{j_n^{(n)}} := 0$ and define augmented vectors $\tilde{\boldsymbol{h}}_n := [\boldsymbol{h}_n^{\mathsf{T}}, 0]^{\mathsf{T}} \in \mathbb{R}^{r_{n+1}}$ and $\tilde{\boldsymbol{k}}_n := [\boldsymbol{k}_n^{\mathsf{T}}, 1]^{\mathsf{T}} \in \mathbb{R}^{r_{n+1}}$. Otherwise, we let $\tilde{\boldsymbol{h}}_n := \boldsymbol{h}_n$ and $\tilde{\boldsymbol{k}}_n := \boldsymbol{k}_n$. The update equation is given as:

$$\boldsymbol{h}_{n+1} := \tilde{\boldsymbol{h}}_n + \eta_n \frac{d_n - \tilde{\boldsymbol{h}}_n^{\mathsf{T}}\tilde{\boldsymbol{k}}_n}{\tilde{\boldsymbol{k}}_n^{\mathsf{T}}\tilde{\boldsymbol{k}}_n}\tilde{\boldsymbol{k}}_n, n \in \mathbb{N}, \tag{4}$$

where $\eta_n \in [0, 2]$ is the step size.

2.2. Compactly Supported Kernels

The Gaussian kernel can be defined as [18, 19]

$$\kappa_{\mathrm{G}}(\boldsymbol{x}, \boldsymbol{y}) := \psi_{\mathrm{G}}(\rho) := \exp\left(\frac{-\rho^2}{2\sigma^2}\right) \tag{5}$$

where $\sigma > 0$ is the kernel parameter and

$$\rho := \|\boldsymbol{x} - \boldsymbol{y}\|_2 \tag{6}$$

denotes the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{y}$. This means that the Gaussian kernel depends only on the distance of two samples, but not on the direction of their difference vector. Such functions are called *radial*, and its special case is *compactly supported radial functions*, which we shall denote by $\psi_{\mathrm{C}} : \mathbb{R} \to \mathbb{R}$. A CSK can be defined as $\kappa_{\mathrm{C}}(\boldsymbol{x}, \boldsymbol{y}) := \psi_{\mathrm{C}}(\rho)$. Table 1 presents some known CSKs studied by Matérn [8], Wendland [9], and Wu [11] for the smoothness parameter set to 2 with the cutoff parameter $c > 0$, which controls the support region (see Fig. 1a). Fig. 1b shows the Gaussian kernel for $\sigma = 0.2$ and the CSKs for $c = 2\sigma(= 0.4)$. The functions presented in Table 1 are strictly positive over the range $[0, c)$ and zero over $[c, \infty)$.

3. PROPOSED ALGORITHM FOR COMPLEXITY REDUCTION

Due to the compactness of the support of ψ_C, the kernelized input vector $\boldsymbol{k}_n$ becomes sparse with the use of an appropriately chosen c; in other words, c controls the sparsity of $\boldsymbol{k}_n$. Without the sparse structure, the complexity for obtaining $\boldsymbol{k}_n$ and for updating $\boldsymbol{h}_n$ is linear in the dictionary size r_n. However, under the sparseness, once we know the support

$$\begin{aligned} \mathrm{supp}(\boldsymbol{k}_n) &:= \left\{\iota \in \mathcal{J}_n \mid [\boldsymbol{k}_n]_\iota := \kappa(\boldsymbol{u}_n, \boldsymbol{u}_\iota) \neq 0\right\} \\ &= \left\{\iota \in \mathcal{J}_n \middle| \, \|\boldsymbol{u}_n - \boldsymbol{u}_\iota\|_2 < c\right\}, \end{aligned} \tag{7}$$

the complexity for the remaining computational tasks to obtain $\boldsymbol{k}_n$ and to update $\boldsymbol{h}_n$ is linear in the support size. This is the core of the complexity reduction.

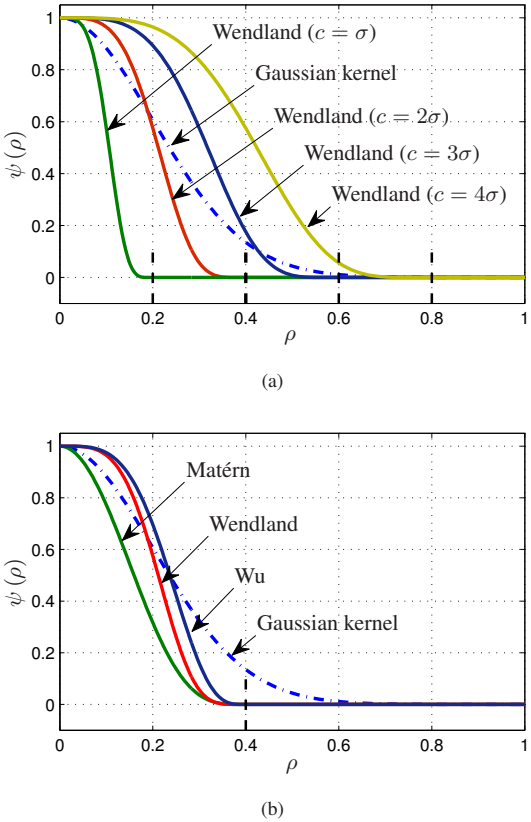


Fig. 1. Compactly supported kernels and a Gaussian kernel.

3.1. Efficient Support Identification through Over-estimation

The support estimation is constructed by using an ℓ_1-ball to reduce the complexity in terms of multiplications. The computational reduction is based on the fact that the evaluation of the ℓ_1 norm involves no multiplications (only absolute value evaluations and additions). The idea is to exclude, at the first step, (not all but) most of the 'off-support' dictionary elements which locate out of the support of the $\psi_C\left(\|\cdot - \boldsymbol{u}_n\|_2\right)$. Because of this, the evaluations of the Euclidean distance $\|\boldsymbol{u}_n - \boldsymbol{u}_\iota\|_2$ can be saved for the 'off-support' dictionary elements.

An open ℓ_1-ball centered at $\boldsymbol{u} \in \mathbb{R}^L$ with radius $\gamma > 0$ is defined as follows:

$$\mathcal{B}_1(\boldsymbol{u}, \gamma) := \left\{\boldsymbol{x} \in \mathbb{R}^L \mid \|\boldsymbol{x} - \boldsymbol{u}\|_1 < \gamma\right\}. \tag{8}$$

The proposed algorithm is composed of three stages.

Stage 1 : The support identification is accomplished by two steps.

Step 1 : Over-estimation based on ℓ_1-ball

An over-estimate of $\mathrm{supp}(\boldsymbol{k}_n)$ is defined by

$$\hat{\mathcal{I}}_n := \left\{\iota \in \mathcal{J}_n \mid \boldsymbol{u}_\iota \in \mathcal{B}_1\left(\boldsymbol{u}_n, \sqrt{L}c\right)\right\} \supset \mathrm{supp}(\boldsymbol{k}_n). \tag{9}$$

Here, the inclusion holds because $\sqrt{L}\|\boldsymbol{x}\|_1 \ge \|\boldsymbol{x}\|_2, \forall \boldsymbol{x} \in \mathbb{R}^L$.

Step 2 : Support identification based on ℓ_2-ball

The open ℓ_2-ball centered at $\boldsymbol{u}$ with radius $\gamma > 0$ is defined as

$$\mathcal{B}_2(\boldsymbol{u}, \gamma) := \left\{ \boldsymbol{x} \in \mathbb{R}^L \mid \|\boldsymbol{x} - \boldsymbol{u}\|_2 < \gamma \right\}. \quad (10)$$

Then, the support is identified under the inclusion in (9), as

$$\mathrm{supp}(\boldsymbol{k}_n) = \left\{ \iota \in \hat{\mathcal{I}}_n \mid \boldsymbol{u}_\iota \in \mathcal{B}_2(\boldsymbol{u}_n, c) \right\}. \quad (11)$$

Stage 2 : The dictionary is constructed on $\mathrm{supp}(\boldsymbol{k}_n)$ as follows:

$$\mathcal{J}_{n+1} := \begin{cases} \mathcal{J}_n \cup \{n\}, & \text{if } \boldsymbol{u}_\iota \notin \mathcal{B}_2(\boldsymbol{u}_n, \mu_1 c) \ \forall \iota \in \mathrm{supp}(\boldsymbol{k}_n), \\ \mathcal{J}_n, & \text{otherwise,} \end{cases} \quad (12)$$

where $\mu_1 \in (0, 1)$ is the threshold. Note that the condition in (12) can be checked only for the subset $\mathrm{supp}(\boldsymbol{k}_n)$ of $\mathcal{J}_n$.

Stage 3 : The filter is updated by (4). Note here that the off-support components are unchanged and therefore the complexity for the update is determined by the support size, not by the dictionary size.

The parameter μ_1 of the proposed algorithm is closely related to the μ_0 of KNLMS in (3). In fact, setting

$$\mu_1 = \frac{\sigma}{c}\sqrt{2\ln\left(\mu_0^{-1}\right)}, \quad (13)$$

one can obtain the same dictionary as that of KNLMS. The proposed algorithm is insensitive to the choice of μ_1 within a range in which the dictionary size is sufficiently large.

Example 1 *To illustrate the proposed algorithm, we show a simple example in Fig. 2. In Fig. 2a, the dictionary data are plotted by $r_n = 200$ dots. Note that those data are indicated by circles (rather than dots) in $\mathcal{B}_1\left(\boldsymbol{u}_n, \sqrt{2}c\right)$. In Fig. 2b, the data in $\mathcal{B}_2(\boldsymbol{u}_n, c)$ are indicated by filled ones.*

Stage 1 :

Step 1. *An over-estimate $\hat{\mathcal{I}}_n$ is attained by taking the 12 circles in $\mathcal{B}_1\left(\boldsymbol{u}_n, \sqrt{2}c\right)$; see Fig. 2a.*

Step 2. *The $\mathrm{supp}(\boldsymbol{k}_n)$ is indicated by the 9 filled circles, out of 12, in $\mathcal{B}_2(\boldsymbol{u}_n, c)$; see Fig. 2b.*

Stage 2 :

There exists some dots in $\mathcal{B}_2(\boldsymbol{u}_n, \mu_1 c)$ (see Fig. 2b), and hence $\mathcal{J}_{n+1} = \mathcal{J}_n$.

Stage 3 :

By using the $\mathrm{supp}(\boldsymbol{k}_n)$ obtained, the proposed algorithm finally updates the filter by (4).

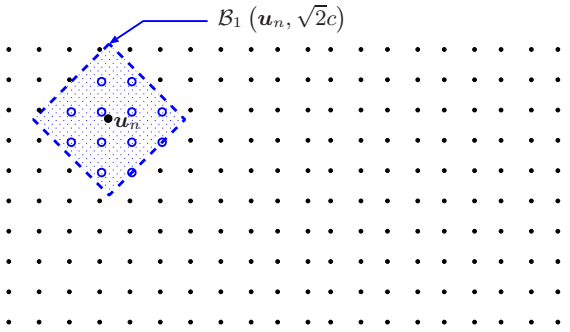
In this case, 12 evaluations of $\|\boldsymbol{u}_n - \boldsymbol{u}_\iota\|_2$ needs to be performed to obtain $\boldsymbol{k}_n$, and (3) and (4) are computed only for the length-9 on-support subvector of $\boldsymbol{k}_n$.

3.2. Complexity of the Proposed Algorithms with CSK

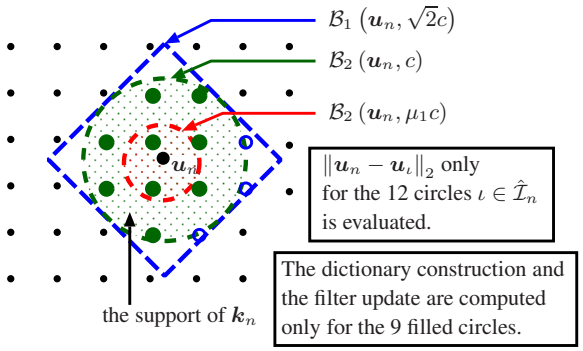
The total number of multiplications and comparisons of the proposed algorithm with Wendland's CSK is $(L+1)r_n + (L+1)\hat{s}_n + 14s_n + 2$ where r_n is the dictionary size, $\hat{s}_n$ is the size of the over-estimated support $\hat{I}_n$ and s_n is the support size. The complexity of KNLMS with Gaussian kernel is $(L+6)\,r_n + 3$. The proposed algorithm is more efficient than KNLMS when

$$\hat{s}_n < \frac{5r_n + 1}{17} \left(\approx 0.29r_n + 0.07 < 0.3r_n\right). \quad (14)$$

The complexity of the algorithms is summarized in Table. 2.



(a) Over-estimation of $\mathrm{supp}(\boldsymbol{k}_n)$ by $\mathcal{B}_1\left(\boldsymbol{u}_n, \sqrt{2}c\right)$.



(b) Support identification and the dictionary construction.

Fig. 2. An illustration of the proposed algorithm.

4. NUMERICAL EXAMPLES

Computer simulations are conducted to verify the efficiency of the proposed algorithm. We use the Gaussian kernel and the Wendland's CSK which is a popular family of CSK; see Table 1. In our experiments, the other CSKs exhibited almost the same performance as Wendland's CSK, and thus we only show the results for Wendland's one. The three adaptive algorithms are compared: KNLMS [4] with Gaussian (KNLMS-Gaussian), KNLMS with CSK (KNLMS-CSK), and the proposed algorithm. The simulation settings are given as follows: $L = 2$, the output is generated as $d_n := \varphi(\boldsymbol{u}_n) + v_n, n \in \mathbb{N}$, with

$$\varphi(\boldsymbol{x}) := \sum_{j=1}^{4} h_j^* \exp\left(-\|\boldsymbol{x} - \boldsymbol{c}_j\|_2^2 / 2\sigma_j^*\right), \quad (15)$$

where $(h_1^*, h_2^*, h_3^*, h_4^*) = (1.5, -1.5, -0.5, 1.5)$, $(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*) = (0.4, 1, 0.4, 0.8)$, $\boldsymbol{c}_1 = [0.5, 0.5]^\mathsf{T}$, $\boldsymbol{c}_2 = [-0.5, -0.5]^\mathsf{T}$, $\boldsymbol{c}_3 = [0.5, -0.5]^\mathsf{T}$ and $\boldsymbol{c}_4 = [-0.5, 0.5]^\mathsf{T}$. The input $\boldsymbol{u}_n \in (-3, 3)^2$ obeys the i.i.d. uniform distribution. The noise v_n is additive white Gaussian noise with $\mathrm{SNR} := \frac{E\{\varphi^2(\boldsymbol{u}_n)\}}{E\{v_n^2\}} = 20\mathrm{dB}$. A normalized *Mean Squared Error (MSE)* is defined as follows:

$$MSE := \frac{E\left\{\left(d_n - \boldsymbol{h}_n^\mathsf{T}\boldsymbol{k}_n\right)^2\right\}}{E\{d_n^2\}} \quad (16)$$

which is approximately computed in our experiments by taking an arithmetic average over 300 independent trials. The step size for each algorithm is $\eta_n = 0.1, \forall n \in \mathbb{N}$. The parameter for KNLMS-

Table 2. Computational complexity of the algorithms.

Algorithm	Kernel	Multiplication	Addition	Comparison
KNLMS	Gaussian	$(L+5)\,r_n+2$	$(L+1)\,r_n-1$	r_n+1
	Matérn	$(L+5)\,r_n+2$	$(L+3)\,r_n-1$	$2r_n+1$
	Wendland	$(L+12)\,r_n+2$	$(L+5)\,r_n-1$	
	Wu	$(L+14)\,r_n+2$	$(L+6)\,r_n-1$	
Proposed	Matérn	$L\hat{s}_n+5s_n+2$	$(2L-1)\,r_n+(L-1)\,\hat{s}_n+4s_n-1$	$(L+1)r_n+\hat{s}_n+2s_n$
	Wendland	$L\hat{s}_n+12s_n+2$	$(2L-1)\,r_n+(L-1)\,\hat{s}_n+6s_n-1$	
	Wu	$L\hat{s}_n+14s_n+2$	$(2L-1)\,r_n+(L-1)\,\hat{s}_n+7s_n-1$	

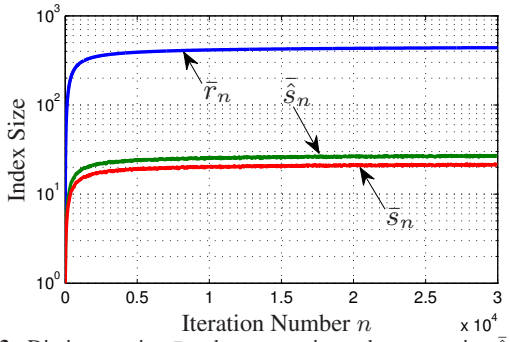


Fig. 3. Dictionary size $\bar{r}_n$, the over-estimated support size $\bar{\hat{s}}_n$ and the exact support size $\bar{s}_n$.

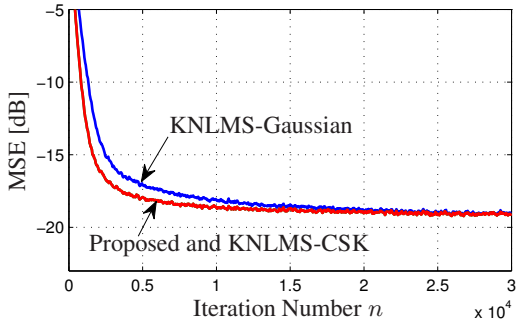


Fig. 4. MSE learning curves.

Gaussian is set to $\sigma = 0.2$. The parameters for KNLMS-CSK and the proposed algorithm are set to $c = 0.8$. We set $\mu_0 = 0.5$ for KNLMS and KNLMS-CSK, and $\mu_1 = 0.29$ for the proposed algorithm; the μ_1 is determined automatically with (13).

In Fig. 3, the average value of the final dictionary size $\bar{r}_{3\times10^4} = 439$ for all algorithms. the average over-estimated support size $\bar{\hat{s}}_n$ and the average support size $\bar{s}_n$ are plotted respectively by the green and red curves. The average and maximum values of $\mathrm{supp}\,(\boldsymbol{k}_n)$ were 29 and 31, respectively, and those for the over-estimated support were 36 and 38, respectively. The figure shows the efficiency of the ℓ_1-based over-estimation because $\hat{s}_n - s_n \approx 3 \ll r_n \approx 200$ (see also Fig. 2).

Fig 4 shows the MSE learning curves. It can be seen that all algorithms show comparable convergence performances. To see the complexity for each algorithm, we plot the number of multiplications and additions in Fig. 5. It can be seen that the number of multiplications of the proposed algorithm is smaller than the one of KNLMS; the number of multiplications plus comparisons are smaller as well.

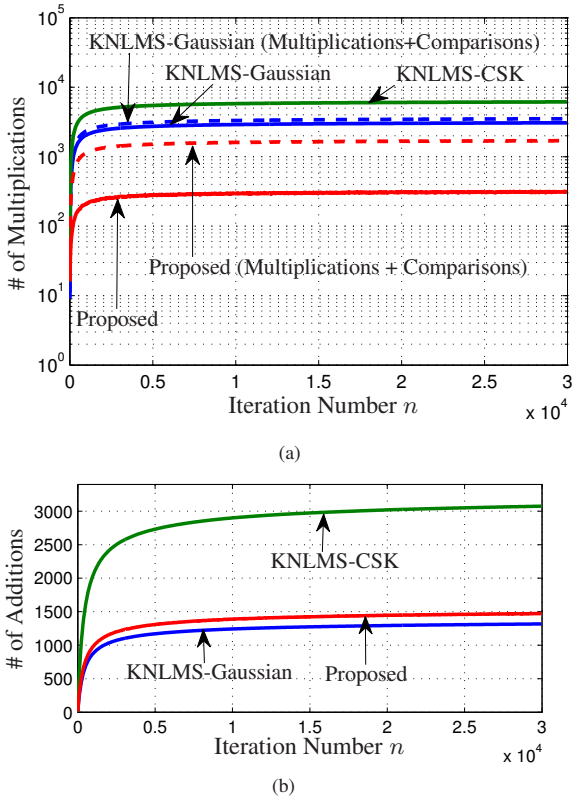


Fig. 5. The complexity of the algorithms at each iteration.

Consequently, the proposed algorithm can significantly reduce the complexity while keeping the estimation performance.

5. CONCLUSION

We presented an efficient KNLMS algorithm with the use of compactly supported kernels. The proposed algorithm was composed of three stages. In the first stage, the support of the kernelized input vector was identified by two steps through its over-estimate based on the circumscribed ℓ_1-ball. In the second stage, the dictionary was constructed based on the knowledge about the support identified in the first stage. In the third stage, the filter was updated based on the support information acquired in the first stage and the dictionary constructed in the second stage. Overall, the complexity was reduced significantly as demonstrated by the numerical examples.

6. REFERENCES

[1] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Processing.*, vol. 52, no. 8, pp. 2165–2176, 2004.

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

[3] W. Liu and J. C. Principe, "Kernel affine projection algorithms," *EURASIP J. Advances in Signal Processing*, vol. 2008, no. 1, pp. 784292, 2008.

[4] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Processing.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[5] K. Slavakis, S. Theodoridis, and I. Yamada, "Adaptive constrained learning in reproducing kernel hilbert spaces: The robust beamforming case," *IEEE Trans. Signal Processing.*, vol. 57, no. 12, pp. 4744–4764, Dec. 2009.

[6] B. Chen, S. Zhao, P. Zhu, and J. C. Principe, "Quantized kernel least mean square algorithm," *IEEE Trans. Neural Networks and Learning Systems.*, vol. 23, no. 1, pp. 22–32, Jan. 2012.

[7] W. Gao, J. Chen, C. Richard, and J. Huang, "Online dictionary learning for kernel LMS," *IEEE Trans. Signal Processing.*, vol. 62, no. 11, pp. 2765–2777, Jun. 2014.

[8] B. Matérn, *Spatial variation*, Lecture notes in statistics. Springer-Verlag, 1986.

[9] H. Wendland, "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree," *Advances in Computational Mathematics*, vol. 4, no. 1, pp. 389–396, 1995.

[10] H. Wendland, *Scattered Data Approximation*, Cambridge Univ. Press, 2004.

[11] Z. Wu, "Compactly supported positive definite radial functions," *Advances in Computational Mathematics*, vol. 4, no. 1, pp. 283–292, 1995.

[12] M. D. Buhmann, "Radial functions on compact support," *Proc. Edin. Math. Soc. 2*, vol. 41, pp. 33–46, Feb. 1998.

[13] M. S. Floater and A. Iske, "Multistep scattered data interpolation using compactly supported radial basis functions," *J. Comput. and App. Math.*, vol. 73, no. 12, pp. 65–78, 1996.

[14] G. E. Fasshauer, "Solving differential equations with radial basis functions: multilevel methods and smoothing," *Advances in Computational Mathematics*, vol. 11, no. 2-3, pp. 139–159, 1999.

[15] G. E. Fasshauer, *Meshfree Approximation Methods with MATLAB*, World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007.

[16] M. G. Genton, "Classes of kernels for machine learning: A statistics perspective," *Journal of Machine Learning Research*, vol. 2, no. 2, pp. 299–312, 2002.

[17] H. H. Zhang, M. Genton, and P. Liu, "Compactly supported radial basis function kernels," *Available at www4.stat.ncsu.edu/hzhang/research.html*, 2004.

[18] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, USA, 2001.

[19] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 4th edition, 2008.