# SPARSE KERNEL RECURSIVE LEAST SQUARES USING $L_1$ REGULARIZATION AND A FIXED-POINT SUB-ITERATION

*Badong Chen[1], Nanning Zheng[1], Jose C. Principe[2]*

1. School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China
2. Department of Electrical and Computer Engineering, University of Florida, Gainesville, USA
(chenbd@mail.xjtu.edu.cn, principe@cnel.ufl.edu)

## ABSTRACT

A new kernel adaptive filtering (KAF) algorithm, namely the sparse kernel recursive least squares (SKRLS), is derived by adding a $l_1$-norm penalty on the center coefficients to the least squares (LS) cost (i.e. the sum of the squared errors). In each iteration, the center coefficients are updated by a fixed-point sub-iteration. Compared with the original KRLS algorithm, the proposed algorithm can produce a much sparser network, in which many coefficients are negligibly small. A much more compact structure can thus be achieved by pruning these negligible centers. Simulation results show that the SKRLS performs very well, yielding a very sparse network while preserving a desirable performance.

***Index Terms***— Kernel adaptive filtering, KRLS, SKRLS, $l_1$ regularization, fixed-point iteration

## 1. INTRODUCTION

Kernel adaptive filtering (KAF) is an on-line nonlinear learning technique, which is a natural generalization of linear adaptive filtering in reproducing kernel Hilbert spaces (RKHS) [1]. The KAF algorithms are derived in RKHS by using the linear structure and inner product of this space to implement the linear adaptive filtering algorithms and obtain the nonlinear algorithms in original input space. With a radial kernel (i.e. Gaussian kernel), the KAF algorithms create naturally a growing radial basis function (RBF) network, where the weights are related to the errors at each sample. Typical KAF algorithms include the kernel least mean square (KLMS) [2], kernel affine projection algorithms (KAPAs) [3], kernel recursive least squares (KRLS) [4], etc. The KAF algorithms have some distinguishing features: a) the learning process is on-line; b) the learning process is convex with no local minima; c) the learning process is universal if selecting a strictly positive-definite (SPD) kernel; d) the learning process requires moderate complexity if properly controlling the network size.

A huge challenge of the KAF algorithms, obviously, is their linearly growing structure, which results in increasing computational complexity and memory requirements especially when the sample size is very large. To overcome this problem, a variety of *sparsification* strategies are proposed to insert only the *important* input data into the *dictionary*. The novelty criterion (NC) [5], approximate linear dependency (ALD) criterion [4], coherence criterion (CC) [6], and surprise criterion (SC) [7] are among the popular sparsification criteria. On-line quantization methods can also be applied to constrain the network size [8, 9].

A new and possibly more efficient approach to the KAF sparsification is to borrow the idea of *compressive sensing* to produce a network with sparse coefficients (of which many values are negligibly small) by adding a sparsity inducing penalty term (e.g. $l_0$ or $l_1$ norm) to the adaptation cost. A compact network can then be achieved by pruning (in off-line or on-line manners) those centers (nodes) with negligible small coefficients. This sparsification strategy has been applied in the quantized KLMS (QKLMS) algorithm [10]. In the present paper, we apply such strategy to develop a new KAF algorithm, namely the sparse kernel recursive least squares (SKRLS), which is derived by adding a $l_1$ norm penalty on the center coefficients to the least squares cost and applying a fixed-point sub-iteration to update the center coefficients. Simulation results confirm the efficiency of the proposed algorithm. Note that in the literature the sparsity inducing penalty terms have been successfully used in linear adaptive filtering algorithms, such as least mean square (LMS) and recursive least squares (RLS) [11-17]. These sparsity regularized algorithms are shown to be highly efficient for sparse signal estimation or system identification.

## 2. SPARSE KRLS

### 2.1. Kernel Least Squares Regression

Suppose the goal is to learn a continuous mapping $f:\mathbb{U}\to\mathbb{R}$ based on a sequence of available input-output

This work was supported by National Natural Science Foundation of China (No. 61372152).

examples (training data) $\{\boldsymbol{u}(j), d(j)\}_{j=1}^{i-1}$, where $\mathbb{U} \subset \mathbb{R}^d$ is the input space. The hypothesis space for learning is assumed to be a RKHS $\mathcal{H}_k$ associated with a Mercer kernel $\kappa(\boldsymbol{u}, \boldsymbol{u}')$, a continuous, symmetric, and positive-definite function $\kappa: \mathbb{U} \times \mathbb{U} \to \mathbb{R}$ [18]. To search such a function $f$, one may solve the regularized least squares regression in RKHS:

$$\min_{f \in \mathcal{H}_k} \sum_{j=1}^{i-1} \left(d(j) - f(\boldsymbol{u}(j))\right)^2 + \gamma \|f\|_{\mathcal{H}_k}^2 \tag{1}$$

where $\gamma \geq 0$ is the regularization factor that controls the smoothness of the solution. By the *Representer Theorem* [18], the function $f$ minimizing (1) can be expressed as a linear combination of the kernels in terms of available data

$$f(.) = \sum_{j=1}^{i-1} \alpha_j \kappa(\boldsymbol{u}(j), .) \tag{2}$$

Let $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_{i-1}]^T$ be the coefficient vector. The learning problem can also be defined as finding $\boldsymbol{\alpha} \in \mathbb{R}^{i-1}$ such that

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^{i-1}} \|\boldsymbol{d} - \mathbf{K}\boldsymbol{\alpha}\|^2 + \gamma \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha} \tag{3}$$

where $\boldsymbol{d} = [d(1), \cdots, d(i-1)]^T$, $\mathbf{K} \in \mathbb{R}^{(i-1)\times(i-1)}$ is the Gram matrix with elements $\mathbf{K}_{j_1 j_2} = \kappa(\boldsymbol{u}(j_1), \boldsymbol{u}(j_2))$.

The above least squares problem can alternatively be formulated in a *feature space*. According to Mercer's theorem, any Mercer kernel $\kappa(\boldsymbol{u}, \boldsymbol{u}')$ induces a mapping $\boldsymbol{\varphi}$ from the input space $\mathbb{U}$ to a high (possibly infinite) dimensional feature space $\mathbb{F}$. The inner product in the feature space can be calculated using the well-known *kernel trick*: $\boldsymbol{\varphi}(\boldsymbol{u})^T \boldsymbol{\varphi}(\boldsymbol{u}') = \kappa(\boldsymbol{u}, \boldsymbol{u}')$. The learning problem in feature space is then to find a high-dimensional weight vector $\boldsymbol{\Omega} \in \mathbb{F}$ that minimizes

$$\min_{\boldsymbol{\Omega} \in \mathbb{F}} \sum_{j=1}^{i-1} \left(d(j) - \boldsymbol{\Omega}^T \boldsymbol{\varphi}(\boldsymbol{u}(j))\right)^2 + \gamma \|\boldsymbol{\Omega}\|_{\mathbb{F}}^2 \tag{4}$$

The feature space $\mathbb{F}$ is isometric-isomorphic to the RKHS $\mathcal{H}_k$ induced by the kernel. This can be easily recognized by identifying $\boldsymbol{\varphi}(\boldsymbol{u}) = \kappa(\boldsymbol{u}, .)$ and $f = \boldsymbol{\Omega}$. The solution of (4) is

$$\boldsymbol{\Omega} = \boldsymbol{\Phi}\boldsymbol{\alpha}^* = \boldsymbol{\Phi}\left(\mathbf{K} + \gamma \mathbf{I}\right)^{-1} \boldsymbol{d} \tag{5}$$

where $\boldsymbol{\Phi} = \left[\boldsymbol{\varphi}(\boldsymbol{u}(1)), \cdots, \boldsymbol{\varphi}(\boldsymbol{u}(i-1))\right]$ (note that $\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \mathbf{K}$), $\boldsymbol{\alpha}^* = \left(\mathbf{K} + \gamma \mathbf{I}\right)^{-1} \boldsymbol{d}$ is the solution of (3), and $\mathbf{I}$ denotes an identity matrix with appropriate dimension. The aim of the KRLS algorithm is to update the solution $\boldsymbol{\alpha}^* = \left(\mathbf{K} + \gamma \mathbf{I}\right)^{-1} \boldsymbol{d}$ recursively as new data $\left(\boldsymbol{u}(i), d(i)\right)$ become available [].

### 2.2. SKRLS Algorithm

Now we derive the SKRLS algorithm. Let $\boldsymbol{\Omega}(i)$, $\boldsymbol{\alpha}(i)$, $\boldsymbol{\Phi}(i)$, $\boldsymbol{K}(i)$ and $\boldsymbol{d}(i)$ denote the related vectors or matrices at iteration $i$. We propose the following optimization cost:

$$J\left(\boldsymbol{\Omega}(i)\right) = \frac{1}{2} \sum_{j=1}^{i} e^2(j) + \gamma \left|\boldsymbol{\alpha}(i)\right|_1 \tag{6}$$

where $e(j) = d(j) - \boldsymbol{\Omega}(i)^T \boldsymbol{\varphi}(j)$, $\boldsymbol{\varphi}(j) = \boldsymbol{\varphi}\left(\boldsymbol{u}(j)\right)$, and $|.|_1$ denotes the $l_1$ norm. Then we have (although here the evaluation of the gradient is not rigorous since the cost is not differentiable)

$$\begin{aligned}
&\frac{\partial J\left(\boldsymbol{\Omega}(i)\right)}{\partial \boldsymbol{\Omega}(i)} \\
&= \sum_{j=1}^{i} e(j) \frac{\partial}{\partial \boldsymbol{\Omega}(i)} e(j) + \gamma \frac{\partial}{\partial \boldsymbol{\Omega}(i)} \left|\boldsymbol{\alpha}(i)\right|_1 \\
&= -\sum_{j=1}^{i} \left(d(j) - \boldsymbol{\Omega}(i)^T \boldsymbol{\varphi}(j)\right) \boldsymbol{\varphi}(j) + \lambda_1 \boldsymbol{\Omega}(i) - \lambda_1 \boldsymbol{\Omega}(i) + \gamma \frac{\partial \boldsymbol{\alpha}(i)^T}{\partial \boldsymbol{\Omega}(i)} \frac{\partial \left|\boldsymbol{\alpha}(i)\right|_1}{\partial \boldsymbol{\alpha}(i)} \\
&= -\sum_{j=1}^{i} d(j) \boldsymbol{\varphi}(j) + \sum_{j=1}^{i} \boldsymbol{\varphi}(j) \boldsymbol{\varphi}(j)^T \boldsymbol{\Omega}(i) + \lambda_1 \boldsymbol{\Omega}(i) \\
&\qquad - \lambda_1 \boldsymbol{\Phi}(i) \boldsymbol{\alpha}(i) + \gamma \boldsymbol{\Phi}(i) \boldsymbol{K}(i)^{-1} sign\left(\boldsymbol{\alpha}(i)\right) \\
&= -\boldsymbol{\Phi}(i) \boldsymbol{d}(i) + \left[\boldsymbol{\Phi}(i) \boldsymbol{\Phi}(i)^T + \lambda_1 \mathbf{I}\right] \boldsymbol{\Omega}(i) \\
&\qquad + \boldsymbol{\Phi}(i) \left(\gamma \boldsymbol{K}(i)^{-1} sign\left(\boldsymbol{\alpha}(i)\right) - \lambda_1 \boldsymbol{\alpha}(i)\right)
\end{aligned} \tag{7}$$

where $\lambda_1 > 0$ is a regularization factor in the matrix $\boldsymbol{Q}_1(i)$ that will be defined latter. Let $\frac{\partial J\left(\boldsymbol{\Omega}(i)\right)}{\partial \boldsymbol{\Omega}(i)} = 0$, we obtain the optimal weight vector in feature space:

$$\boldsymbol{\Omega}^*(i) = \left[\boldsymbol{\Phi}(i) \boldsymbol{\Phi}(i)^T + \lambda_1 \mathbf{I}\right]^{-1} \left[\boldsymbol{\Phi}(i) \boldsymbol{d}(i) - \boldsymbol{\Phi}(i) \left(\gamma \boldsymbol{K}(i)^{-1} sign\left(\boldsymbol{\alpha}(i)\right) - \lambda_1 \boldsymbol{\alpha}(i)\right)\right] \tag{8}$$

Applying the matrix inversion lemma yields

$$\boldsymbol{\Omega}^*(i) = \boldsymbol{\Phi}(i) \left[\boldsymbol{K}(i) + \lambda_1 \mathbf{I}\right]^{-1} \left[\boldsymbol{d}(i) - \left(\gamma \boldsymbol{K}(i)^{-1} sign\left(\boldsymbol{\alpha}(i)\right) - \lambda_1 \boldsymbol{\alpha}(i)\right)\right] \tag{9}$$

Then the optimal coefficient vector is

$$\begin{aligned}
\boldsymbol{\alpha}(i) &= \left[\boldsymbol{K}(i) + \lambda_1 \mathbf{I}\right]^{-1} \left[\boldsymbol{d}(i) - \left(\gamma \boldsymbol{K}(i)^{-1} sign\left(\boldsymbol{\alpha}(i)\right) - \lambda_1 \boldsymbol{\alpha}(i)\right)\right] \\
&\approx \left[\boldsymbol{K}(i) + \lambda_1 \mathbf{I}\right]^{-1} \left[\boldsymbol{d}(i) - \left(\gamma \left[\boldsymbol{K}(i) + \lambda_2 \mathbf{I}\right]^{-1} sign\left(\boldsymbol{\alpha}(i)\right) - \lambda_1 \boldsymbol{\alpha}(i)\right)\right] \\
&= \boldsymbol{Q}_1(i) \left[\boldsymbol{d}(i) - \left(\gamma \boldsymbol{Q}_2(i) sign\left(\boldsymbol{\alpha}(i)\right) - \lambda_1 \boldsymbol{\alpha}(i)\right)\right]
\end{aligned} \tag{10}$$

where $\boldsymbol{Q}_1(i) = \left[\boldsymbol{K}(i) + \lambda_1 \mathbf{I}\right]^{-1}$, $\boldsymbol{Q}_2(i) = \left[\boldsymbol{K}(i) + \lambda_2 \mathbf{I}\right]^{-1}$, $\lambda_2 > 0$ is a small positive regularization factor in $\boldsymbol{Q}_2(i)$ to avoid the singularity problem. From (10), we immediately propose a fixed-point sub-iteration to update the coefficient vector:

$$\boldsymbol{\alpha}^{k+1}(i) = g\left(\boldsymbol{\alpha}^k(i)\right) \tag{11}$$

where $g(\boldsymbol{\alpha}(i))=\boldsymbol{Q}_1(i)\left[\boldsymbol{d}(i)-\left(\gamma\boldsymbol{Q}_2(i)sign(\boldsymbol{\alpha}(i))-\lambda_1\boldsymbol{\alpha}(i)\right)\right]$, $k$ denotes the $k$ -th fixed-point sub-iteration. The initial value in the iteration (11) can be set at $\boldsymbol{\alpha}^0(i)=\left[\boldsymbol{\alpha}(i-1)^T,0\right]^T$ .

The matrices $\boldsymbol{Q}_1(i)$ and $\boldsymbol{Q}_2(i)$ can be updated using the basic recursion in the standard KRLS algorithm [1]:

$$\boldsymbol{Q}_l(i)=r_l(i)^{-1}\begin{bmatrix}\boldsymbol{Q}_l(i-1)r_l(i)+z_l(i)z_l(i)^T & -z_l(i)\\ -z_l(i)^T & 1\end{bmatrix},\ l=1,2 \quad (12)$$

where

$$z_l(i)=\boldsymbol{Q}_l(i-1)\boldsymbol{h}(i)$$

$$\boldsymbol{h}(i)=\boldsymbol{\Phi}(i-1)^T\boldsymbol{\varphi}(i)$$

$$r_l(i)=\lambda_l+\boldsymbol{\varphi}(i)^T\boldsymbol{\varphi}(i)-z_l(i)^T\boldsymbol{h}(i)$$

The pseudocode description of the proposed SKRLS algorithm is summarized in Table 1.

TABLE I. SKRLS ALGORITHM

***Initialization:***

Selecting the kernel $\kappa$, the regularization parameters $\gamma$, $\lambda_1$, and $\lambda_2$, and the fixed-point sub-iteration number $N_f$ .

Initializing the dictionary $\boldsymbol{D}(1)=\{\boldsymbol{u}(1)\}$, and the matrices $Q_1(1)=\left[\kappa(\boldsymbol{u}(1),\boldsymbol{u}(1))+\lambda_1\right]^{-1}$, $Q_2(1)=\left[\kappa(\boldsymbol{u}(1),\boldsymbol{u}(1))+\lambda_2\right]^{-1}$ and the coefficient $\boldsymbol{\alpha}(1)=Q_1(1)d(1)$.

***Computation:***

**while** $\{\boldsymbol{u}(i),y(i)\}$ ($i>1$)available **do**

Update the dictionary: $\boldsymbol{D}(i)=\{\boldsymbol{D}(i-1),\boldsymbol{u}(i)\}$,

Update the matrices $\boldsymbol{Q}_1(i)$ and $\boldsymbol{Q}_2(i)$ using (12),

Update the coefficient vector $\boldsymbol{\alpha}(i)$ as follows:

**for** $k=1:N_f$

$\boldsymbol{\alpha}^{k+1}(i)=g\left(\boldsymbol{\alpha}^k(i)\right)$, where $\boldsymbol{\alpha}^0(i)=\left[\boldsymbol{\alpha}(i-1)^T,0\right]^T$

**end**

**end while**

***Remark 1:*** Although a strict proof of the convergence is not available (a standard tool for such proof is the well-known *Banach Fixed-Point Theorem*), our simulation results show that the proposed fixed-point sub-iteration will always converge to the optimal solution in few iterations.

***Remark 2:*** In practice, the regularization factor $\lambda_1$ can be set at a larger value because it does not bias the solution, while $\lambda_2$ is in general set at a smaller value.

***Remark 3:*** The pruning of the SKRLS filter can be performed off-line or on-line. In off-line manner, the centers with small coefficients (usually below a preset threshold) will be discarded only after the training process is stopped. In on-line manner, however, the centers with small coefficients will be discarded during the training process. One can use the methods in [19] to derive the discarding criterion and update the matrices $\boldsymbol{Q}_1(i)$ and $\boldsymbol{Q}_2(i)$ when the discarding occurs.

## 3. SIMULATION RESULTS

Let's consider the IKEDA chaotic dynamic system given by the following complex map [20]:

$$z_{n+1}=A+Bz_n\exp\left(i\left(C-\frac{K}{1+|z_n|^2}\right)\right) \quad (13)$$

where $A=1$, $B=0.7$, $C=0.4$, $K=6$, and $z_0=0+0i$. In physics the IKEDA map is related to the laser dynamics. The attractor of this IKEDA system is shown in Fig. 1.

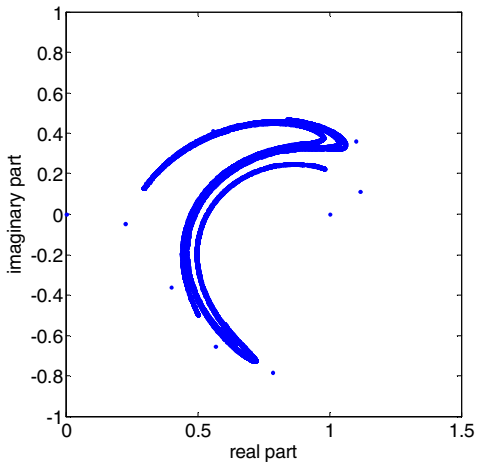


Fig. 1 The attractor of the IKEDA system ( $B=0.7$ )

In the simulation, the real-part time series is used and the generated data are corrupted by an additive white Gaussian noise with zero mean and variance 0.001. The goal is to predict the current value of the time series using the previous four points. The Gaussian kernel with kernel size 0.5 is selected as the Mercer kernel. In all the simulations, the parameter $\gamma=0.005$, $\lambda_1=10$, $\lambda_2=0.1$. For each Monte Carlo simulation, 1000 samples are used as the training data and another 50 points as the test data (the filter is fixed in the testing phase).

The learning curves (averaged over 100 Monte Carlo runs) of the KRLS and the SKRLS with different fixed-point sub-iteration numbers are shown in Fig. 2. As one can see clearly, the SKRLS yields almost the same convergence performance as the original KRLS. Table 2 lists the testing MSE at final iteration. The histograms of the coefficients values (at final iteration) are illustrated in Fig. 3 (where $N_f=5$). As expected, the SKRLS produces a much sparser network, since its coefficients are much more concentrated around zero.

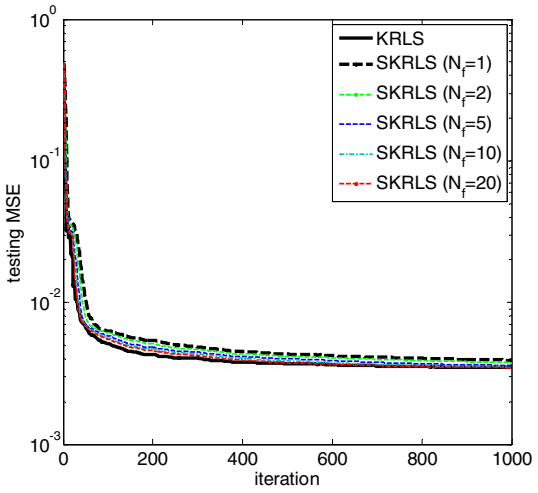


Fig. 2 Learning curves of KRLS and SKRLS

TABLE II. TESTING MSE AT FINAL ITERATION

| Algorithm | Testing MSE |
|---|---|
| KRLS | 0.0035±0.0007 |
| SKRLS ( $N_f = 1$ ) | 0.0039±0.0008 |
| SKRLS ( $N_f = 2$ ) | 0.0038±0.0008 |
| SKRLS ( $N_f = 5$ ) | 0.0036±0.0008 |
| SKRLS ( $N_f = 10$ ) | 0.0035±0.0007 |
| SKRLS ( $N_f = 20$ ) | 0.0035±0.0007 |

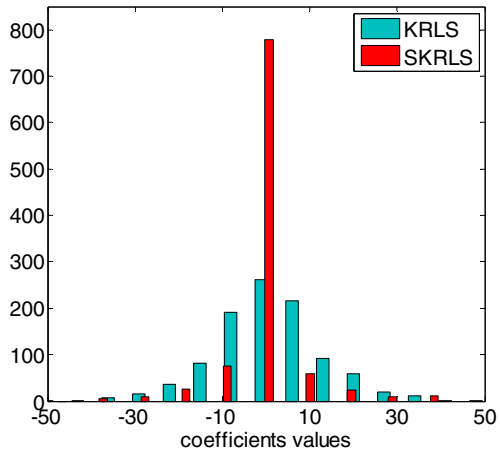


Fig. 3 Histogram plots of the coefficients values ( $N_f = 5$ )

Next, we demonstrate the performance of the SKRLS with online pruning ( $N_f = 5$ ). During the training, we add a new center into the dictionary only when the Euclidean distance between the newly arrived sample and the dictionary is larger than 0.1. Further, at each iteration we discard the center with the smallest coefficient provided the coefficient value is less than 0.05. Fig. 4 shows the learning curves of the SKRLS with online pruning and the KRLS with several sparsification methods (NC, ALD, SC). The parameters of these sparsification methods are chosen such that all the algorithms converge to the same steady state. The corresponding network growth curves are shown in Fig. 5. Evidently, the SKRLS with online pruning achieves the smallest network size at final iteration.

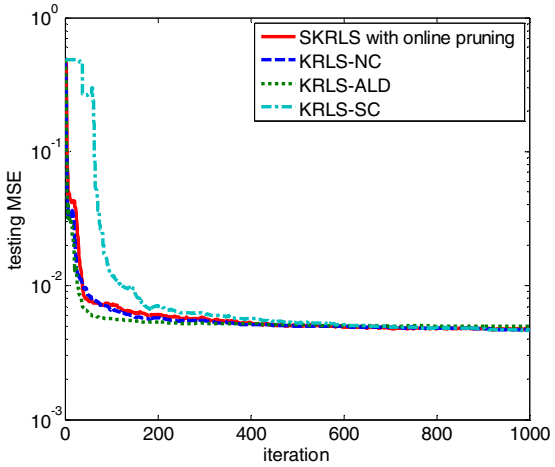


Fig. 4 Learning curves of the SKRLS with online pruning and the KRLS with several sparsification methods

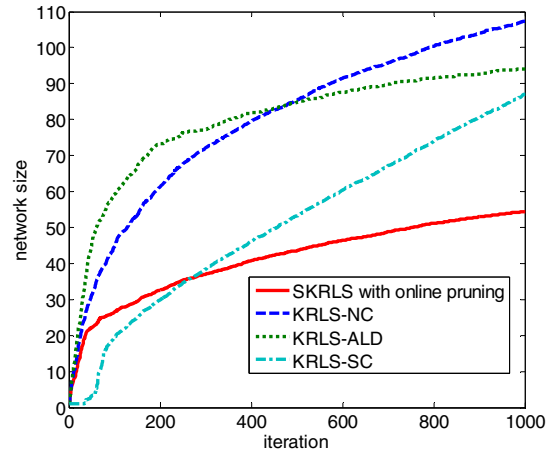


Fig. 5 Network growth curves of the SKRLS with online pruning and the KRLS with several sparsification methods

## 4. CONCLUSION

A new kernel adaptive filtering (KAF) algorithm, namely the sparse kernel recursive least squares (SKRLS) algorithm, is developed in this paper by adding a $l_1$-norm penalty to the least squares cost and applying a fixed-point sub-iteration to update the center coefficients. With an on-line (or off-line) pruning method, the proposed algorithm may yield a nonlinear adaptive filter with high accuracy and small network size. In future study, it is important to prove the convergence, and investigate how to select the proper parameters.

## REFERENCES


[1] W. Liu, J. C. Principe, S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*, Wiley, 2010.

[2] W. Liu, P. Pokharel, J. C. Principe, "The kernel least mean square algorithm," *IEEE Transactions on Signal Processing*, vol. 56, pp. 543-554, 2008.

[3] W. Liu, J. C. Principe, "Kernel affine projection algorithm," *EURASIP J. Adv. Signal Process.*, vol. 2008, Article ID 784292, 12 pages, doi: 10.1155/2008/784292.

[4] Y. Engel, S. Mannor, R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, pp. 2275-2285, 2004.

[5] J. Platt, "A resource-allocating network for function interpolation" *Neural Computation*, vol. 3, pp. 213-225, 1991.

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

[7] W. Liu, Il Park, J. C. Principe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Transactions on Neural Networks*, vol. 20, pp. 1950-1961, 2009.

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

[9] B. Chen, S. Zhao, P. Zhu, J. C. Principe, "Quantized kernel recursive least squares algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 9, pp. 1484-1491, 2013.

[10] B. Chen, S. Zhao, S. Seth, J. C. Principe, "Online efficient learning with quantized KLMS and $l_1$ regularization," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*. IEEE, 2012, June, pp 1-6.

[11] Y. Gu, J. Jin, S. Mei, "$l_0$ norm constraint LMS algorithm for sparse system identification," *IEEE Signal Process. Lett.*, vol. 16, pp. 774-777, 2009.

[12] Y. Chen, Y. Gu, A. O. Hero, "Sparse LMS for system identification," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on.* IEEE, 2009, pp. 3125-3128.

[13] E. M. Eksioglu, "Sparsity regularized recursive least squares adaptive filtering," *IET Signal Process.*, vol. 5, pp. 480-487, 2011.

[14] J. Jin, Y. Gu, S. Mei, "A stochastic gradient approach on compressive sensing signal reconstruction based on adaptive filtering frame- work, " *IEEE Journal on Selected Topics in Signal Processing,* vol. 4, no.2, pp. 409–420, 2010.

[15] B. Babadi, N. Kalouptsidis, and V. Tarokh, "SPARLS: The sparse RLS algorithm," *IEEE Trans. on Signal Process.*, vol. 58, no. 8, pp. 4013–4025, 2010.

[16] D. Angelosante, J.A. Bazerque, and G.B. Giannakis, "Online Adaptive Estimation of Sparse Signals: Where RLS Meets the $l_1$-Norm," *IEEE Trans. on Signal Process.*, vol. 58, no. 7, pp. 3436–3447, 2010.

[17] Y. Kopsinis, K. Slavakis, and S. Theodoridis, "Online Sparse System Identification and Signal Reconstruction using Projections onto Weighted $l_1$ Balls," *Arxiv preprint arXiv:1004.3040*, 2010.

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

[19] S. Van Vaerenbergh, I. Santamaria, W. Liu, J. C. Principe, "Fixed-budget kernel recursive least-squares," in *Acoustics, Speech and Signal Processing, 2010. ICASSP 2010. IEEE International Conference on.* IEEE, 2010, pp. 1882-1885.

[20] K. Ikeda, "Multiple-valued stationary state and its instability of the transmitted light by a ring cavity system," *Opt. Commun.*, vol. 30, pp. 257–261, 1979.