Performance and Complexity of Online Kernel Algorithms

Anthony Kuh* and Danilo Mandic†
* Dept. of Electrical Eng., University of Hawaii, Honolulu, HI 96822, USA E-mail: kuh@hawaii.edu
† Dept. Electrical and Electronic Eng., Imperial College, London SW72BT, UK
E-mail: d.mandic@imperial.ac.uk

***Abstract*—This paper discusses some recent advances in the development of nonlinear adaptive filtering. Specifically, it studies online kernel adaptive filters. We study the performance and complexity of a suite of kernel online algorithms from kernel recursive least square subspace (KRLSS) algorithms to kernel least mean square (KLMS) algorithms. A key to the kernel algorithms is that updating is done in the dual space via evaluation of kernels, the number of support vectors is controlled by using an information criterion, and the number of information vectors is controlled for KRLSS These algorithms have advantages in that they are nonlinear filters that are relatively easy to implement and have a number of parameters that can be adjusted to tradeoff performance for complexity.**

I. Introduction

In recent years there have been substantial advances in the area of nonlinear adaptive filtering. This paper looks at the development of online kernel algorithms using quadratic cost functions. Specifically, we examine a variety of kernel learning algorithms from the kernel recursive least squares subspace (KRLSS) algorithm to the kernel least mean square (KLMS) algorithm. The online algorithms are updated in the dual observation space by evaluating kernel functions. Performance and computational complexity can be traded off by adjusting different parameters of the algorithms. This includes criteria for choosing support vectors and information vectors. The algorithms are nonlinear adaptive filters that are easily implemented and different adaptive kernel algorithms are simulated on a nonlinear time series example.

Kernel methods have become a popular and widely used machine learning paradigm as they are based on sound theoretical principles of statistical learning theory, structural risk minimization, and reproducing kernel Hilbert spaces (RKHS)). Kernel methods solve nonlinear problems by transforming inputs and solving convex optimization methods. The solution can also be found in the dual space via the kernel trick [3], [15], [24]. Kernel methods have been used in a wide range of applications from communication to text recognition to biomedical applications [3], [18], [19], [23]. Commonly used kernel methods are the Support Vector Machine (SVM) with different formulations for classification and regression problems. For binary classification problem the common cost function for SVM is the hinge loss function and for regression functions the common cost function for SVM is the ϵ insensitive loss function [3], [15], [24]. Both problems involve solving quadratic optimization problems with linear inequality constraints. Most solutions involve solving problems in a batch mode where a set of training examples is given to the algorithm and then the algorithm finds the appropriate parameters; in primal space, the weight vector, w and threshold value, b or in dual space, the support values α, the associated support vectors $x(i)$, $1 \leq i \leq m$, and threshold value b.

In many signal processing applications such as time series prediction it is often necessary to implement algorithms that are online and adaptive. There has been some previous research implementing SVM in an online manner, [1], [8], [20] with these papers focusing on classification problems. In this paper we focus on SVM regression that uses a quadratic cost function which is referred to as the least squares SVM (LSSVM) [23]. The LSSVM solution is found by solving a set of linear equations in either the primal or dual space [23]. A problem with implementing the LSSVM solution in the dual space is that the dimensionality of the system is m, the number of training examples. For moderate to large training data sets the number of support vectors need to be reduced. The solution can be made sparse in the number of support vectors by using various methods including constrained subspace approaches [9], [2]. For these methods the weight vector is constrained to lie in the subspace generated by the support vectors with the solution again involving solving a set of linear equations.

These kernel algorithms can then be easily implemented in an online manner by using matrix identities and methods used for recursive least squares (RLS) filters [7]. This has been done in [5], [6], [9]. In [6] a criteria is chosen based on only inputs called the approximate linear dependence (ALD) which adds support vectors when the kernel vector is almost linearly independent of other support kernel vectors. Other criteria can also be used base on inputs and outputs such as training error criteria discussed in [5], [9]. There is similar research developed by [17] and for online Gaussian processes by [4]. More recently, two kernel stochastic approximation algorithms have been developed based on the LMS algorithm [14], [16]. These algorithms have the advantage of having lower computational complexity per update and being simpler to implement.

This paper discusses algorithm design, performance, and complexity issues associated with these online kernel algorithms and is organized as follows. In Section 2 we discuss the kernel least squares subspace KLSS optimization algorithm and solution. We then give an online formulation of the

KLSS algorithm called the kernel recursive least squares subspace KRLSS algorithm in Section 3. The algorithm is considered for time series prediction with updating of an estimate consisting of two steps; first adding and deleting information vectors, then testing to see if the support vectors are modified. Section 4 then discusses different criteria for adding and possibly deleting support vectors. In Section 5 we discuss approximations to kernel least squares algorithms. We focus on two algorithms; the primal kernel least mean square PKLMS algorithm, [14] and the dual kernel least mean square (DKLMS) algorithm, [16]. Section 6 gives a simulation example illustrating the tradeoffs between performance and complexity for both the KRLSS and DKLMS algorithms. Finally, Section 7 summarizes results of the paper and suggests further directions for this research.

II. Kernel Least Squares Regression

In this section we present the LSSVM developed in [23] and the subspace method presented in [9]. Here we are given training examples $(x(i), y(i))$, $1 \leq i \leq m$ where $x(i) \in \mathcal{R}^n$ and $y(i) \in \mathcal{R}$. We represent the data compactly as $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x} = [x(1)|\ldots|x(m)]$ and $\mathbf{y} = [y(1), \ldots, y(m)]^T$. The inputs are transformed from input space to feature space via kernel functions $\phi(x)$ that map inputs from $\mathcal{R}^n$ to feature space $\mathcal{R}^d$. Let $\mathbf{Z} = \Phi(\mathbf{x}) = [\phi(x(1))|\ldots|\phi(x(m))]$. Then the system can be formulated by a least squares optimization problem given by

$$\min J(w,b) = \min_{w,b} \frac{1}{2}||w||^2 + \frac{\gamma}{2}||e||^2 \tag{1}$$

subject to

$$e = \mathbf{y} - \mathbf{Z}^T w - \mathbf{1}b \tag{2}$$

where $w \in \mathcal{R}^d$ is the weight vector, $\mathbf{1}$ is a vector of 1s, and $b \in \mathcal{R}$ is the threshold value. Equation (1) contains two terms with the first term controlling complexity and the second term controlling squared error with $\gamma > 0$ denoting the regularization value that weights the squared error.

It is easily shown that the solution to this optimization problem in primal or dual space involves solving a set of linear equations. We cannot always work in the primal space as the dimensionality can be infinite if we work with certain kernels such as Gaussian kernels. However, a problem with working in the dual space is that the solution depends on the number of training examples, m which could be quite large. Unlike the standard SVM solution, the LSSVM solution uses all training examples as support vectors. Some method is needed to sparsify the training data set.

Let us pick m_s columns of $\mathbf{Z}$ to form a matrix $\mathbf{Z_s}$ with S denoting the set of m_s support vectors. The training inputs associated with these columns are the support vectors and methods for choosing these support vectors are discussed in [6], [9], [14], [16]. There are now two algorithms that we use to solve the LSSVM problem. One method is called the reduced system method and the second method is called the subspace method. We present solutions to both methods, but the paper focuses on the subspace method.

A. Reduced System Method

The reduced system method solves the LSSVM for $\mathbf{Z_s}$. The solution is given by

$$\begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & K_{SS} + I/\gamma \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}. \tag{3}$$

where $K_{SS} = \mathbf{Z_s}^T \mathbf{Z_s}$. Assume that $A = K_{SS} + I/\gamma$ is invertible, then

$$b = \mathbf{1}^T A^{-1} \mathbf{y} / m_s \tag{4}$$

and

$$\alpha = A^{-1}(\mathbf{y} - \mathbf{1}b). \tag{5}$$

B. Subspace Method

For this method the weights are constrained to lie in the subspace generated by the chosen columns of $\mathbf{Z}$ and this constraint can be expressed as

$$w_a = \mathbf{Z_s}\alpha. \tag{6}$$

where α is a m_s vector weighting the training feature vectors. Again let $K_{SS} = \mathbf{Z_s}^T \mathbf{Z_s}$ and introduce $K_S = \mathbf{Z_s}^T \mathbf{Z}$. By substituting equations (2,6) in equation (1) we have that

$$\min Q(\alpha, b) = \min \frac{1}{2}\alpha^T K_{SS}\alpha + \frac{\gamma}{2}||\mathbf{y} - {K_S}^T\alpha - \mathbf{1}b||^2. \tag{7}$$

This problem is solved by finding the solution to the following set of linear equations,

$$\begin{bmatrix} m & \mathbf{1}^T {K_S}^T \\ K_S\mathbf{1} & K_{SS}/\gamma + K_S {K_S}^T \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} \mathbf{1}^T\mathbf{y} \\ K_S\mathbf{y} \end{bmatrix}. \tag{8}$$

Assume $A = K_{SS}/\gamma + K_S {K_S}^T$ is invertible. By elimination we then get that

$$b = \frac{\mathbf{1}^T\mathbf{y} - \mathbf{1}^T {K_S}^T A^{-1} K_S \mathbf{y}}{m - \mathbf{1}^T {K_S}^T A^{-1} K_S \mathbf{1}} \tag{9}$$

and

$$\alpha = A^{-1} K_S(\mathbf{y} - \mathbf{1}b). \tag{10}$$

We principally use the subspace method as it has advantages over the reduced system method. Both methods have the same dimensionality $m_s + 1$, however the subspace method uses more information and data, K_S as opposed to just K_{SS}. This results in improved performance for the subspace method at the cost of slightly higher computations.

To get the estimated output for either method given an input x, let $z = \Phi(x)$.

The estimated output is then given by

$$\hat{y}(z) = w^T z + b \tag{11}$$

or we can represent in dual space using kernel by

$$\hat{y}(z) = \alpha^T K_{Sx} + b \tag{12}$$

where $K_{Sx} = \mathbf{Z_s}^T z$.

III. Online Kernel RLS Algorithms

We consider prediction of time series data. Data is given by a sequence of data, $x(k) \in \mathcal{R}$. Here the feature input $z(k) = [\phi(x(k)) \dots \phi(x(k-n+1))]$ and the associated output is $y(k) = x(k+L)$ where $L \geq 1$ is the prediction step. For kernel algorithms we consider online recursive algorithms where at time k the support vector parameters are given by $\alpha(k)$. Without loss of generality we assume that the threshold value $b = 0$. The estimate of $y(k)$ is

$$\hat{y}(k) = \alpha(k)^H K_{Sx(k)} \tag{13}$$

The kernel online algorithm is then given the output $y(k)$ and updates the data matrices to compute the inverse of A and then update the support vector values. In addition to the length of the input vector, n we must determine the number of support vectors m_s and the number of information vectors m that are used in the kernel subspace algorithm. In Section 4 we discuss criteria for adding and deleting support vectors. The algorithm is then given by

1) Initialization of parameters and α.
2) Add and possibly delete information vectors. Update K_S. Then compute A^{-1} and update α.
3) Use criterion to establish whether information vector should be a support vector. If criteria is satisfied add information vector to set of support vectors and augment α. Update K_S and K_{SS}. Then compute A^{-1} and update α. If necessary, also delete support vector and associated quantities.
4) Form estimate and go to 2.

In step 2. information vectors are added and possibly deleted. This involves changing the columns of K_S. Computing A^{-1} can be done by using the Sherman Morrison Woodbury formula given by

$$(F + uv^T)^{-1} = F^{-1} - F^{-1}u(1 + v^T F^{-1}u)^{-1}v^T F^{-1} \tag{14}$$

where F is the old value of A and u and v contain information vectors to be added and possibly deleted. This results in $\mathcal{O}(m_S)^2$ computations.

In step 3. if the number of support vectors is increased by one, computing A^{-1} involves increasing the dimension of kernel vectors by one and using the following formula for inverting block matrices given by

$$\begin{bmatrix} F & G \\ E & H \end{bmatrix}^{-1} = \begin{bmatrix} F^{-1} & 0 \\ 0 & 0 \end{bmatrix} + v\delta^{-1}u \tag{15}$$

where F is the old value of A, and G, E, H contain new information. $v = \binom{F^{-1}G}{-1}$, $u = [EF^{-1}, -1]$ and $\delta = H - EF^{-1}G$. This results in $\mathcal{O}(mm_S)$ computations. The computation involved in adding a support vector is more than adding and deleting information vectors. The computational savings involved in computing A^{-1} using equation (14,15) instead of direct inversion becomes larger as m_S grows.

Note that in [6], [16] they prove that for using the ALD or the coherence criteria that an updating algorithm for stationary data consisting of just adding support vectors eventually results in a finite number of support vectors being added even if the number of training data grows arbitrarily large. This eliminates the need for deleting support vectors if the data in the time series is stationary. In practice training data may be nonstationary and improved performance can be achieved by not only adding support vectors, but occasionally deleting support vectors. Note that making more changes to the support vector set results in increased computations. In Section 6 we show through simulations the tradeoffs between performance and computational complexity as we vary the number of support vectors.

IV. Criteria for adding Support Vectors

In the previous section we discussed KRLSS algorithms. A key to these algorithms is when to add support vectors. There are several approaches that can be taken. We categorize these approaches into three groups; data independent, input dependent, and input/output dependent.

Examples of data independent are choosing support vectors at random or choosing the most recent information vectors as support vectors by a windowing scheme. The advantages of data independent methods is the cost of the data independent methods is low. Data independent methods in general are not used as the performance is inferior to other methods and computational costs could also be high (e.g. support vector set will change at each update for most recent time window method which will result in higher computational costs.)

Input dependent methods have achieved popularity due to their good performance. Examples of input dependent methods are the ALD test in [6] and the coherence test used in [16]. The ALD test involves first finding

$$\delta(k) = \min_a ||K_{Sx(k)} - K_{SS}a||^2 \tag{16}$$

This results in $\delta(k) = z(k)^T z(k) - K_{Sx(k)}^T (K_{SS})^{-1} K_{Sx(k)}$. If $\delta(k) \leq \mu_0$, then the current training input is added as a support vector. With knowledge of $(K_{SS})^{-1}$ the ALD test takes $\mathcal{O}(m_s)^2$ operations. The coherence test normalizes kernels so that $K(x, x) = 1$, then tests the magnitudes of the components of $K_{Sx(k)}$. if all of the components are less than a threshold value μ_0, then current training input is added as a support vector. The coherence test takes $\mathcal{O}(m_s)$ operations.

Input/output dependent methods have also been developed. These criteria depend on both input training example data and their outputs. These methods achieve good performance and can achieve better performance than just input dependent methods, but require slightly higher computations. Let I be the set of m information vectors, then a training error based criteria examines the angle between $e = \mathbf{y} - \alpha^T K_S$ with the new kernel vector $K_{\mathrm{I}x(k)}$. If this angle is small, then current training input is added as a support vector [9]. This method takes $\mathcal{O}(mm_s)$ operations. Other methods such as the surprise criterion combine ALD with training error and are based on approximating processes by Gaussian processes and looking at training examples that have high information content, [12].

Key concerns for all these methods is the performance versus computation requirement tradeoff. It is not possible to

choose the optimal set of support vectors [22], but if we can choose a reasonable set of support vectors we can still get good performance. By changing threshold parameter values we can also tradeoff performance for higher computations and more support vectors.

Besides controlling the number of support vectors we also have to control the number of information vectors given by the set I. For time series problems we typically do this by using the m most recent training examples as belonging to I. Note in this case that S is not a subset of I.

V. Kernel LMS Algorithm

Recently, two kernel LMS algorithms have been introduced and are discussed here. The first is formulated in the primal space and is referred to as the PKLMS algorithm [14] and the second is formulated in the dual space and is referred to as the DKLMS algorithm [16]. Both algorithms perform updates using kernel operations and depend on m_S, the number of support vectors chosen.

A. Primal Kernel LMS Algorithm

The PKLMS algorithm performs LMS in the feature space and is given by

$$w(k+1) = w(k) + \eta e(k) z(k) \tag{17}$$

where η is the step size, $e(k)$ is the error, $z(k) = \phi(x(k))$, and $w(0) = 0$ (zero initial conditions). The problem with calculating the PKLMS in the primal weight space is that $w(k)$ could be very high dimensional and difficult to calculate or not possible to calculate (e.g. Gaussian kernel). However, in [14] show that the PKLMS algorithm can be easily computed in the dual observation space. Instead of updating weights, we can update the error term with

$$e(k+1) = y(k+1) - \sum_{i=1}^{k} \eta e(i) K(x(i), x(k+1)) \tag{18}$$

and the estimate is given by

$$\hat{y}(k) = \sum_{i=1}^{k-1} \eta e(i) K(x(i), x(k)). \tag{19}$$

Some additional remarks about the algorithm:

Remark 1: Note that $\alpha_k(k) = \eta e(k)$. PKLMS updates each support vector value only once (when that training example is present). This algorithm performs LMS in the primal space with updates depending only on the current feature input $\phi(x(k))$ and is a subsystem algorithm.

Remark 2: The algorithm can be run similarly to the KRLSS algorithms in that the weight vector is updated only when training examples satisfy a criteria specified in the previous section and hence become support vectors. The number of support vectors, m_s is then a small fraction of the training examples.

Remark 3: Each weight update require $\mathcal{O}(m_s)$ computations. Testing for whether an update takes place should also be $\mathcal{O}(m_s)$ computations. Criteria that could be used are the coherence criteria or simplified ALD or surprise criteria.

Remark 4: The PKLMS algorithm is shown to converge to the RKHS least square solution for proper step sizes and does not need regularization [14].

B. Dual Kernel LMS Algorithm

Consider the kernel least squares problem without regularization and zero bias. The problem involves finding the α to minimize $||K_S\mathbf{y} - K_S K_S^T \alpha||^2$. This looks similar to the least squares problem in primal space where the object is to find w to minimize $||Rw - P||^2$ where $R = \mathbf{x}\mathbf{x}^T$ is the correlation matrix and $P = \mathbf{x}\mathbf{y}$ is the cross-correlation vector. The LMS algorithm is a stochastic gradient approximation algorithm derived from this equation. Similarly we can derive the DKLMS algorithm in the dual kernel space discussed in [16]:

1) Initialization of algorithm
2) Get current kernel vector $h(x(k)) = K_{Sx(k)}$ and test coherence
3) If $h(x(k))$ has small coherence current input $x(k)$ becomes a support vector. We then have $S = S + \{x(k)\}$ and $\alpha(k+1) = [\alpha(k)^T, 0]^T$.
4) Update weights,
$$\alpha(k+1) = \alpha(k) + \eta e(k) h(x(k)) \tag{20}$$
5) Increment k and go to 2.

Here again η is the step size and $e(k)$ is the error term given by

$$e(k) = y(k) - \alpha(k)^T h(x(k)) \tag{21}$$

Some remarks about the algorithm:

Remark 5: Algorithm performs kernel LMS in dual space. Here inputs are given by kernel vector where kernels are evaluated between set of support vectors and current input. Here each component of $\alpha(k)$ is modified with each update.

Remark 6: DKLMS has α vector updated with each training example. It differs from LMS in that dimension of α increases by one when coherence criteria is satisfied. From Section 3 note that the dimensionality of α which is the number of support vectors remains finite.

Remark 7: See **Remark 3:**.

Remark 8: DKLMS has similar behavior as LMS in that it converges to least squares solution for small enough step sizes. Additional variations to the algorithm can be made by having update depending on k most recent kernel vectors. This algorithm is call the kernel affine projection algorithm and has been discussed in [13], [21], [16].

VI. Simulations

To illustrate the behavior of the algorithm we simulate the following nonlinear time series estimation problem with additive noise. This example was also presented in [16].

$$y(k) = (.8 - .5\exp(-y(k-1)^2))y(k-1)$$

$$-(.3+\exp(-y(k-1)^2)y(k-2)+.1\sin(\pi y(k-1))+v(k) \quad (22)$$

with $v(k)$ being additive white Gaussian noise with standard deviation .1 and with initial conditions $y(0) = y(-1) = .1$. Here we simulated the KRLSS algorithm and DKLMS algorithm with Gaussian kernels. Simulations were run for 10000 iterations and repeated 200 times. The coherence criteria was used and the DKLMS algorithm was run using regularization value of $\epsilon = .03$, stepsize of $\eta = .09$. A normalized form of the LMS algorithm was run to control weight magnitudes..

In Fig. 1 we show a comparison between the KRLSS and DKLMS algorithms. The MSE curves look similar to standard RLS and LMS curves. The KRLSS converges in much fewer iterations and converges to a slightly lower MSE value than DKLMS algorithm. However KRLSS takes about three times the computations as DKLMS per iteration. In Fig. 2 we compare the DKLMS algorithm for different coherence threshold values. For $\mu_0 = .3$ we get an average of 16 support vectors, for $\mu_0 = .5$ we get an average of 25 support vectors, and for $\mu_0 = .7$ we get an average of 45 support vectors. The computation time increases roughly linearly as we increase the number of support vectors. The MSE improves as we increase the number of support vectors, but the difference in MSE between 25 and 45 support vectors is small. In Fig. 3 we vary the number of information vectors for the KRLSS algorithm. We see that the MSE decreases rapidly when the number of information vectors is increased from 50 to 200 and then the rate of decrease becomes slower as we make further increases to the number of information vectors. The computation time increases roughly linearly as we increase the number of information vectors.

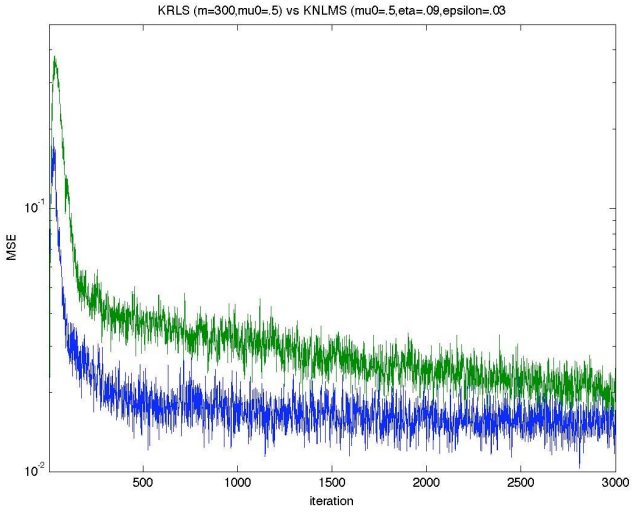

Fig. 1. Comparison between KRLSS and DKLMS algorithms. Bottom curve is KRLSS and top curve is DKLMS. Average number of support vectors is roughly $m_s = 25$.

VII. Summary and Further Directions

This paper discusses a variety of nonlinear online kernel algorithms. One set of algorithms is based on the RLS algorithm and a second set of algorithms is based on the LMS algorithm. Both algorithms run in the dual observation space and involve computation of kernels. To control the number of support vectors an information criterion is used. The number of information vectors is controlled by windowing data. These algorithms achieve good performance, are simple to implement, and performance can be traded off with higher computational complexity by adjusting different parameters of the algorithm. We envision that these suite of algorithms will have wide applicability in many arenas from time series prediction to nonlinear parametric estimation to system identification problems.

There are many further directions for this research from analysis of the algorithms to further development of algorithms. We have previously looked at extensions of these online kernel algorithms to complex filtering applications [10] and distributed learning [11].

References

[1] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, Fast Kernel Classifiers with Online and Active Learning, *Journal of Machine Learning Research*, vol. 6, pp. 1579-1619, Sep. 2005.

[2] G. C. Cawley, N. L. C. Talbot. A greedy training algorithm for sparse least-squares support vector machines, *Proceedings of the International Conference on Artificial Neural Networks ICANN 2002*, Madrid, Spain, 681–686, 2002.

[3] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, U.K., 2000.

[4] L. Csato and M. Opper. Sparse online Gaussian processes, *Neural Computation*, Vol. 14-3, 641-669, 2002.

[5] B. de Kruif and T. deVries. Pruning Error Minimization in Least Squares Support Vector Machines. *IEEE Trans. on Neural Networks*, vol. 14, 3, 696 -702, 2003.

[6] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm, *IEEE Trans. Signal Proc.*, Vol. 52, 8, 2275-2285, Aug., 2004.

[7] S. Haykin. *Adaptive filter theory, 4th Ed.*, Prentice Hall, Englewood Cliffs, NJ, 2003.

[8] J. Kivinen, A. Smola, R. Williamson. Online Learning with Kernels, *IEEE Trans. on Signal Proc.*, Vol. 52, 8, 2165-2176, Aug. 2004.

[9] A. Kuh, Intelligent recursive kernel subspace estimation algorithms, in *Proceedings of the 39th annual Conference on Information Sciences and Systems (CISS 2005)*, 216-221, Baltimore, MD, USA, Mar., 2005.

[10] A. Kuh and D. Mandic, Applications of Complex Augmented Kernels to Wind Profile Prediction, *2009 International Conference on Acoustics, Speech, and Signal Processing*, 3581-3584, Taipei, Taiwan, Apr. 2009.

[11] A. Kuh and C. Zhu. Sensor Network Localization Using Least Squares Kernel Regression, *Signal Processing Techniques for Knowledge Extraction and Information Fusion,* D. Mandic, M. Golz, A. Kuh, D. Obradovic, and T. Tanaka, Editors, 77–96, Springer, Apr. 2008.

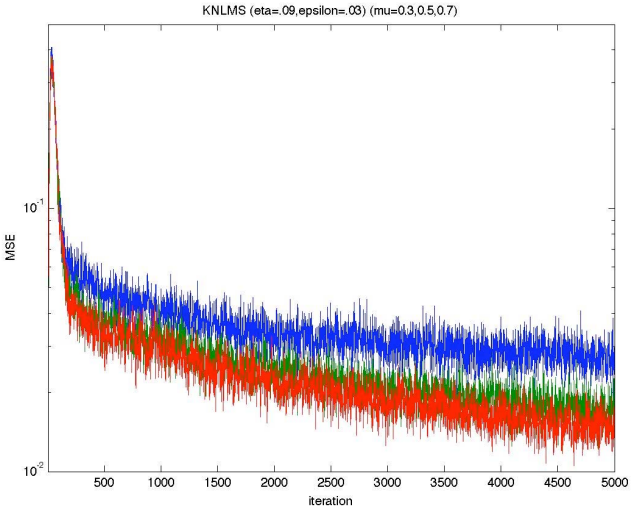

Fig. 2. DKLMS as we vary the coherence threshold value, μ_0. Top curve is $\mu_0 = .3$, middle curve is $\mu_0 = .5$, and bottom curve is $\mu_0 = .7$.

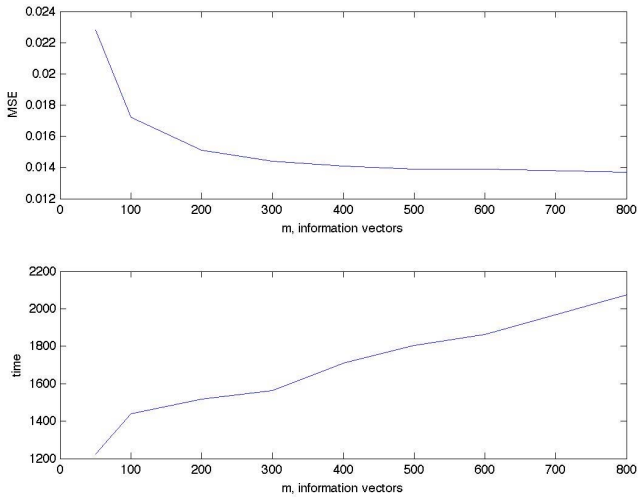

Fig. 3. KRLSS with varying information number of information vectors. Average number of support vectors is roughlyt $m_s = 25$

[12] W. Liu. *Adaptive Filtering in Reproducing Kernel Hilbert Spaces,* Ph.D. Dissertation, Univ. of Florida, 2008.
[13] W. Liu, J. Principe. Kernel Affine Projection Algorithms, *EURASIP Journal on Advances in Signal Processing, Special Issue: Emerging Machine Learning Techniques in Signal Processing*, Vol. 2008, Article ID 784292, 12 pages, 2008.
[14] W. Liu, P. Pokharel, J. Principe. The Kernel LMS Algorithm, *IEEE Trans. on Signal Proc.*, vol. 56, 2, 543-554, Feb. 2008.
[15] K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An Introduction to Kernel-Based Learning Algorithms. *IEEE Trans. on Neural Networks*, Vol. 12, #2, 181-202, 2001.
[16] C. Richard, J.-C. Bermudez, P. Honeine. Online prediction of time series data with kernels. *IEEE Trans. on Signal Proc.*, vol. 57, 3 1058-1067, Mar. 2009.
[17] R. Rifkin. *Learning with kernels: Support vector machines, regularization, optimization and beyond*, Ph.D. thesis, MIT, 2002.
[18] B. Schölkopf, C. Burges, and A. Smola. *Advances in kernel methods support vector learning*. MIT Press, Cambridge, MA, 1999.
[19] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond* . MIT Press, Cambridge, MA, 2001.
[20] S. Shalev-Shwartz and Y. Singer. A Primal-Dual Perspective of Online Learning Algorithms, *Machine Learning* vol. 69, 2-3 115-142, 2007.
[21] K. Slavakis and S. Theodoridis. Sliding window generalized kernel affine projection algorithm using projection mappings. *EURASIP Journal on Advances in Signal Processing, Special Issue: Emerging Machine Learning Techniques in Signal Processing*, Vol. 2008, Article ID 735351, 16 pages, 2008.
[22] A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th International Conference on Machine Learning*, Morgan Kaufmann Publishers, 911-918, 2000.
[23] J. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least squares support vector machines*, World Scientific Publishing Co., Singapore, 2000.
[24] V. Vapnik. *Statistical learning theory*. John Wiley, New York City, NY, 1998.