The Kernel Least-Mean-Square Algorithm

Weifeng Liu, *Student Member, IEEE*, Puskal P. Pokharel, *Student Member, IEEE*, and Jose C. Principe, *Fellow, IEEE*

***Abstract*—The combination of the famed kernel trick and the least-mean-square (LMS) algorithm provides an interesting sample-by-sample update for an adaptive filter in reproducing kernel Hilbert spaces (RKHS), which is named in this paper the KLMS. Unlike the accepted view in kernel methods, this paper shows that in the finite training data case, the KLMS algorithm is well posed in RKHS without the addition of an extra regularization term to penalize solution norms as was suggested by Kivinen [Kivinen, Smola and Williamson, "Online Learning With Kernels," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004] and Smale [Smale and Yao, "Online Learning Algorithms," *Foundations in Computational Mathematics*, vol. 6, no. 2, pp. 145–176, 2006]. This result is the main contribution of the paper and enhances the present understanding of the LMS algorithm with a machine learning perspective. The effect of the KLMS step size is also studied from the viewpoint of regularization. Two experiments are presented to support our conclusion that with finite data the KLMS algorithm can be readily used in high dimensional spaces and particularly in RKHS to derive nonlinear, stable algorithms with comparable performance to batch, regularized solutions.**

***Index Terms*—Kernel methods, least mean square, Tikhonov regularization.**

I. Introduction

THE solid mathematical foundation and considerable experimental successes are making kernel methods very popular. However, kernel methods (support vector machines [3], regularization networks [4], kernel principal component analysis (PCA) [5], etc.) are usually derived in batch mode requiring $O(N^3)$ operations along with $O(N^2)$ storage to perform the inversion or the singular value decomposition of the Gram matrices [4]. This excludes kernel methods from many real-world applications; therefore, it becomes important to exploit computationally efficient online extensions. The least-mean square (LMS) is the workhorse of adaptive signal processing due to its simplicity and elegance [6] and it can be expressed solely in terms of inner products. Hence, it became naturally one of the candidates for on-line kernel based learning. We study such a kernel based extension in this paper and call it the kernel least mean square (kernel LMS or KLMS) because it adapts filter parameters using a stochastic gradient approximation in reproducing kernel Hilbert spaces (RKHS) [16].

There have been many studies on kernel-based online learning algorithms such as kernel Hebbian algorithm [7] to solve the kernel PCA, kernel Adaline [8], kernel recursive least squares (RLS) [9], and kernel online learning [1]. Except [1], all works mentioned above focus on algorithm design and none of them provide a well-posedness analysis even though algorithms are implicitly cast in very high dimensional spaces [10]. Moreover, the KLMS is different from all these solutions because it naturally creates a *growing* radial basis function network (RBF) with a learning strategy similar to *resource allocating networks* (RAN) proposed by Platt [11]. Several heuristic techniques are employed in RANs to adapt the network parameters, which makes theoretical analysis intractable, while as we will see we can still formulate mathematically the use of resources in KLMS.

The concept of well-posedness was proposed by Hadamard [12]. Regularization as a remedy for ill-posedness became widely known due to the work of Tikhonov [13]. In least-squares problems, Tikhonov regularization is essentially a tradeoff between fitting the training data and reducing solution norms. More recently, the significance of well-posedness and its relation to generalization has been revealed in statistical learning theory [3], [14]. Further important results are proven by Bousquet and Elisseeff [15], where *Tikhonov regularization*-based methods have been shown to possess a strong L_∞ stability with respect to changes of single samples of the training set, and exponential bounds have been proved for the generalization error in terms of empirical error. Following this line of reasoning, the online kernel learning algorithm in [1] adopted a stochastic gradient approach to solve the *regularized risk functional* in the RKHS. The analysis in [1] only focuses on the probabilistic upper bound for convergence and concludes that the regularization term appears essential for its well-posedness.

This paper employs the conventional LMS theory to prove that when finite data is used in the training, *the KLMS is well-posed and therefore does not need explicit regularization* enriching the viewpoint outlined in [1], which simplifies the implementation and has the potential to provide better results because regularization biases the optimal solution as is well known. With this message at the core, the asymptotic behavior of the KLMS is examined in the framework of the small-step-size theory [6], which shows that it has a "self-regularization" mechanism due to its different convergence speeds along eigendirections. Furthermore, with these results a bound on the solution norm is established on the basis of the H^∞ stability theory [6], which implies immediately the well-posedness of the KLMS using arguments from Tikhonov regularization and regularization networks theory [14]. These results offer an understanding of the LMS algorithm from a learning theory perspective which to the

Manuscript received December 4, 2006; revised July 16, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jonathon Chambers. This work was partially supported by NSF Grant ECS-0601271. This work is a theoretical and extended study of the KLMS algorithm in the *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, vol. 3, pp. III-1421–III-1424, 2007.

The authors are with the Computational NeuroEngineering Laboratory, Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: weifeng@cnel.ufl.edu; pokharel@cnel.ufl.edu; principe@cnel.ufl.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2007.907881

best of our knowledge is not present in the adaptive filter literature. Although the KLMS algorithm was reported in [16], this paper presents for the first time the theoretical framework and the well-posedness results, as well as expands on the KLMS validation.

The organization of the paper is as follows. In Section II, a formulation of the KLMS algorithm is presented. Next, in Section III, its stochastic gradient descent behaviors are studied and important insights into its "self-regularization" mechanism are gained through the comparison with Tikhonov regularization. The bound on the solution norm is established and its implication is elaborated in Section IV. Two experiments are studied in Section V to support our theory. Finally, Section VI summarizes the conclusions and future lines of research.

II. Formulation of the Kernel Least-Mean-Square Algorithm

A. Learning Problem Setting

Suppose the goal is to uncover a function $f: \boldsymbol{U} \rightarrow \boldsymbol{R}$ based on a sequence $((u_1, y_1), \ldots, (u_N, y_N)) \in Z^N$ of examples $(u_i, y_i) \in Z = \boldsymbol{U} \times \boldsymbol{Y}$. $\boldsymbol{Y}$ is a compact subset of $\boldsymbol{R}$ and $\boldsymbol{U}$ is assumed as a compact subset of $\boldsymbol{R^M}$. To find the true underlying function f, we may try to solve the following empirical risk minimization (ERM) problem:

$$R_{\text{emp}}[f \in H, Z^N] = \sum_{i=1}^{N} (y_i - f(u_i))^2 \tag{1}$$

which is unfortunately ill-posed in general, depending on the choice of the hypothesis space $\boldsymbol{H}$. ERM is a sound strategy only if the function class is uniform Glivenko–Cantelli, that is, it satisfies the uniform law of large numbers [17]. Remarkably, this paper shows that the straightforward stochastic gradient descent algorithm (normally called the least mean square algorithm) to minimize directly the ERM principle (1) is well-posed in the sense of Hadamard unlike what was previously suggested in [1] and [2]. Although the literature on the LMS algorithm is extensive, this fact has not, to the best of our knowledge, being reported before.

B. Least-Mean-Square Algorithm

This section presents for completeness the LMS algorithm. In the LMS algorithm, the hypothesis space $\boldsymbol{H}_1$ consists of all the linear operators on $\boldsymbol{U}$, denoted by $w : U \rightarrow R$. Since $\boldsymbol{U}$ is embedded in a Hilbert space, the linear operator becomes the standard inner product.

The LMS algorithm minimizes the empirical risk:

$$\min_{w} R_{\text{emp}}[w \in H_1, Z^N] = \sum_{i=1}^{N} (y_i - w(u_i))^2. \tag{2}$$

Using the stochastic gradient, it is easy to show that [6]

$$\begin{aligned} w_0 &= 0 \\ e_n^a &= y_n - w_{n-1}(u_n) \\ w_n &= w_{n-1} + \eta e_n^a u_n \end{aligned} \tag{3}$$

where e_n^a is called the *a priori* error and η is the step size.

The repeated application of the weight-update equation yields

$$w_n = \eta \sum_{i=1}^{n} e_i^a u_i. \tag{4}$$

Therefore, after n-step training, the weight is expressed as the linear combination of the previous and present input data weighted by the *a priori* errors. More important, the input–output operation of this learning system can be solely expressed in terms of inner products

$$\tilde{y} = w_n(\tilde{u}) = \eta \sum_{i=1}^{n} e_i^a < u_i, \tilde{u} >_U \tag{5}$$

$$e_n^a = y_n - \eta \sum_{i=1}^{n-1} e_i^a < u_i, u_n >_U . \tag{6}$$

Therefore, by the kernel trick, the LMS algorithm can be readily extended to RKHS [16].

C. Kernel LMS Algorithm

A kernel [18] is a continuous, symmetric, positive-definite function $\kappa: \boldsymbol{U} \times \boldsymbol{U} \rightarrow \boldsymbol{R}$. The commonly used kernels include the Gaussian kernel (7) and the polynomial kernel (8) among many others:

$$\kappa(u_i, u_j) = \exp(-a\|u_i - u_j\|^2) \tag{7}$$

$$\kappa(u_i, u_j) = \left(u_i^T u_j + 1\right)^p. \tag{8}$$

By the Mercer's theorem [18], for any kernel, there exists a mapping Φ such that

$$\kappa(u_1, u_2) = \Phi(u_1)\Phi^T(u_2), \quad \text{for } \forall u_1, u_2 \in U. \tag{9}$$

We utilize this theorem to transform the data u_i into the feature space $\boldsymbol{F}$ as $\Phi(u_i)$ and interpret (9) as the usual dot product. Let the hypothesis space $\boldsymbol{H}_2$ be the dual space of $\boldsymbol{F}$. Then, the KLMS is nothing but the LMS performed on the example sequence $\{(\Phi(u_1), y_1), \ldots, (\Phi(u_N), y_N)\}$, which is summarized as follows:

$$\begin{aligned} \Omega_0 &= 0 \\ e_n^a &= y_n - \Omega_{n-1}(\Phi(u_n)) \\ \Omega_n &= \Omega_{n-1} + \eta e_n^a \Phi(u_n). \end{aligned} \tag{10}$$

It may be difficult to have direct access to the weight and the transformed data in the transformed space. Using (5), (6), and (9), we have

$$\Omega_{n-1}(\Phi(u_n)) = \eta \sum_{i=1}^{n-1} e_i^a \kappa(u_i, u_n) \tag{11}$$

$$e_n^a = y_n - \eta \sum_{i=1}^{n-1} e_i^a \kappa(u_i, u_n) \tag{12}$$

and the final input–output relation (after N step training) of the learning algorithm is

$$\Omega_N = \eta \sum_{i=1}^{N} e_i^a \Phi(u_i) \tag{13}$$

$$\tilde{y} = \eta \sum_{i=1}^{N} e_i^a \kappa(u_i, \tilde{u}). \tag{14}$$

Although the training uses the stochastic gradient and it is still online, KLMS loses the original LMS simplicity due to the inability to work directly with the weights in the RKHS. By (11), it is seen that the KLMS allocates a new kernel unit when a new training data comes in with the input u_i as the center and the difference between the output of the network and the desired signal as the coefficient (scaled by the learning rate). The algorithm is summarized as follows.

Algorithm 1 Kernel Least-Mean-Square

Input: data $(\mathrm{u_t}, \mathrm{y_t})$, size N

Initialization

$f^0 = 0$,

η: learning step,

a: kernel width parameter,

Variables used in the loop

c_{new}: center of the new unit,

h_{new}: coefficient of the new unit

loop over input-output pairs $(\mathbf{u}, \mathbf{y})$

{

evaluate output of network $f^{t-1}(u)$

computer error $e_t^a = y - f^{t-1}(u)$

allocate new unit, $c_{\mathrm{new}} = u, h_{\mathrm{new}} = \eta e_t^a$

$f^t = h_{\mathrm{new}} \kappa_a(c_{\mathrm{new}}, \cdot) + f^{t-1}$

}

Notice that the KLMS algorithm (10) is a stochastic gradient approximation to minimize the following risk:

$$\min_{\Omega} R_{\mathrm{emp}}[\Omega \in H_2, Z^N] = \sum_{i=1}^{N} (y_i - \Omega(\Phi(u_i)))^2 \tag{15}$$

which is equivalent to

$$\min_{f} R_{\mathrm{emp}}[f \in H_3, Z^N] = \sum_{i=1}^{N} (y_i - f(u_i))^2. \tag{16}$$

The equivalence is easily recognized by identifying

$$f = \sum_{u_i \in U} \alpha_i \kappa(u_i, \cdot) = \sum_{u_i \in U} \alpha_i \Phi(u_i) = \Omega. \tag{17}$$

Equation (17) really translates the isometric isomorphism between $\boldsymbol{H}_2$ and $\boldsymbol{H}_3$ ($f = \Omega \circ \Phi$ will be more rigorous).

Conventionally, the LMS is applied to relatively small dimensional input space (a few to hundreds of taps) so the well-posedness issue due to the lack of training data does not pose a big problem. Since the KLMS is derived in a possibly infinite dimensional space with finite training data, its well-posedness study becomes crucial. From the viewpoint of machine learning theory, the algorithm itself would be incomplete without such a well-posedness study.

As pointed out in regularization theory, the ERM of (16) is usually ill-posed when the hypothesis space H_3 is unconstrained [14]. A question follows immediately: Does the KLMS converge in such a high-dimensional space? The question is actually twofold: 1) Is the solution of the KLMS close to the least-squares (LS) solution when (15) is well-posed? and 2) Does the KLMS give reasonable approximation even when (15) is ill-posed? The answers to these two questions also indicate the double meaning of convergence here: 1) Is the KLMS solution close to the batch mode solution from the viewpoint of stochastic gradient approximation? and 2) Is the KLMS solution close to the TRUE underlying function in the context of empirical risk minimization consistency? Before we answer these questions for the KLMS algorithm, it is necessary to briefly review the least squares method and Tikhonov regularization theory, as well as to establish the notation used throughout the paper.

D. Least-Squares and Regularization Theory

In LS, the multiple linear regression model is hypothesized as

$$y_n = \Omega^o(\Phi_n) + v(n). \tag{18}$$

From now on, Φ_n is used for $\Phi(u_n)$ unless the role of u_n is emphasized and $v(n)$ is the modeling uncertainty. Denote the transformed data matrix as $D_\Phi^T = [\Phi(u_1), \Phi(u_2), \ldots, \Phi(u_N)]$, the correlation matrix $R_\Phi = D_\Phi^T D_\Phi / N$, the cross correlation vector $p_\Phi = \sum_{i=1}^{N} \Phi(u_i) y_i / N = D_\Phi^T \vec{y} / N$ and the Gram matrix $G_\Phi = D_\Phi D_\Phi^T = [\kappa(u_i, u_j)]_{N \times N}$. The well-known LS solution is [19], [20]

$$\hat{\Omega} = D_\Phi^+ \vec{y} \tag{19}$$

where D_Φ^+ is the general pseudoinverse of D_Φ.

$$D_\Phi^+ = P \begin{bmatrix} S^{-1} & 0 \\ 0 & 0 \end{bmatrix} Q^T \tag{20}$$

and matrices P, Q, and S are given by the singular value decomposition (SVD) of D_Φ [19]

$$D_\Phi = Q \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} P^T. \tag{21}$$

P and Q are orthogonal matrices and $S = \mathrm{diag}\{s_1, s_2, \ldots, s_r\}$ assuming the rank of D_Φ is r. s_i are the singular values of D_Φ and assumed $s_1 \geq s_2 \geq \cdots \geq s_r > 0$.

However, the LS problem can be ill-posed due to the nature of the problem, small data size or severe noise. The Tikhonov regularization [13] is widely used to address this issue. A regularization term is introduced in the LS cost function which penalizes the solution norm

$$\min_{\Omega} R_{\mathrm{reg}}[\Omega \in H_2, Z^N] = \|\vec{y} - D_\Phi \Omega\|^2 + \lambda \|\Omega\|_{H_2}^2. \tag{22}$$

In the Bayesian interpretation, the error square term is the likelihood and the regularized term is some *a priori* knowledge on the solution norm [21]. The solution is given by [13]

$$\hat{\Omega}_{\mathrm{TR}} = \left(D_\Phi^T D_\Phi + \lambda I\right)^{-1} D_\Phi^T \vec{y}. \tag{23}$$

More insights can be obtained through the SVD

$$\hat{\Omega}_{\mathrm{TR}} = P\,\mathrm{diag}\left(\frac{s_1}{s_1^2+\lambda},\ldots,\frac{s_r}{s_r^2+\lambda},0,\ldots,0\right)Q^T\vec{y}. \tag{24}$$

Comparing (24) with the pseudoinverse solution

$$\hat{\Omega} = P\,\mathrm{diag}\left(s_1^{-1},\ldots,s_r^{-1},0,\ldots,0\right)Q^T\vec{y} \tag{25}$$

we see that the Tikhonov regularization modifies the singular values through the following regularization function (reg-function):

$$H_{\mathrm{TR}}(x) = x^2/(x^2+\lambda). \tag{26}$$

Notice that $H_{\mathrm{TR}}(x)\to 1$ when x large and $H_{\mathrm{TR}}(x)\to 0$ when x small. In this sense, the Tikhonov regularization *smoothly filters* out the singular components that are small (relative to λ). With this understanding, the so-called truncated pseudo-inverse regularization is nothing but the following hard cut-off reg-function:

$$H_{\mathrm{HC}}(x) = \begin{cases} 1, & \text{if } x > a \\ 0, & \text{if } x \le a \end{cases} \tag{27}$$

where a is the cut-off threshold. If $s_m > a \ge s_{m+1}$, the solution becomes

$$\hat{\Omega}_{\mathrm{HC}} = P\,\mathrm{diag}\left(s_1^{-1},\ldots,s_m^{-1},0,\ldots,0\right)Q^T\vec{y}. \tag{28}$$

This method is equivalent to applying PCA to the data and using the first m principal components to represent the original data. Under reasonable signal-noise-ratio (SNR), the small singular value components are purely associated with the noise. Discarding these spurious features can effectively prevent over-learning. A similar idea can be found in [9] and [22]. There are many regularization techniques (e.g., nonsmooth regularization [23]), but the concept of regularization considered in this paper refers only to Tikhonov.

The optimization problem (22) is equivalent to

$$\min_f R_{\mathrm{emp}}[f\in H_3, Z^N] = \sum_{i=1}^{N}(y_i - f(u_i))^2 + \lambda\|f\|_{H_3}^2. \tag{29}$$

The solution to (29) is given by the regularization networks [10]

$$f_\lambda = D_\Phi^T(G_\Phi + \lambda I)^{-1}\vec{y}. \tag{30}$$

Some relevant properties are listed below.

Theorem 2.1: For a fixed positive λ, the norm of the solution is bounded in the following way:

$$\|f_\lambda\|_H^2 \le \frac{\|\vec{y}\|^2}{4\lambda}. \tag{31}$$

Theorem 2.2: The optimization problem (29) is equivalent to the following problem:

$$\min_f R_{\mathrm{emp}}[f\in H, z^N] = \sum_{i=1}^{N}(y_i - f(u_i))^2$$
$$\text{subject to } \|f\|_H^2 \le C \tag{32}$$

in the sense that for any positive number C there exists a unique λ_C such that (29) and (32) lead to the same solution. Moreover, λ_C monotonically decreases as C increases.

The proofs of these theorems are straightforward based on the regularization network theory and thus not included. Theorem 2.2 tells that *adding the regularization term is equivalent to constraining the norm of the solution* in LS problems, which plays a key role in our conclusion outlined in Section IV. From the viewpoint of machine learning, well-posedness heavily depends on the complexity of the hypothesis space. Boundedness in (32) implies directly the compactness of the hypothesis space which is conceptually similar to a finite VC dimension [3], [14]. As pointed out by Poggio in [14], the bounded hypothesis space ensures the well-posedness of empirical risk minimization. To conclude, the key of Tikhonov regularization is to examine how the small singular value components are taken care of and how the norm of the solution is constrained.

III. Stochastic Gradient Analysis of the Kernel LMS

In this section, we will examine some important properties of the KLMS algorithm in light of the results obtained in Section II-D. We will show that the KLMS algorithm is well-posed under usual conditions in the sense of Hadamard, using Tikhonov regularization theory. We assume the dimensionality of $\boldsymbol{F}$ is d. Our discussion is based on the finite dimension assumption for simplicity. Since $\boldsymbol{F}$ is a Hilbert space, all the discussion extends to infinite dimension gracefully except for different terminology (operator theory) [24].

Proposition 3.1: The KLMS algorithm (10) converges *asymptotically* in the mean sense to the optimal solution (19) under the "small-step-size" condition.

Remarks: Asymptotically, i.e., as N approaches infinity, the LS solution converges to the optimal Wiener solution very quickly. Therefore, we use the LS estimator as the optimal solution in the asymptotic analysis.

Proof: From the data SVD (21), we know that P is the eigenvector matrix of R_Φ and the corresponding eigenvalues are $\varsigma_i = s_i^2/N$. Express the weight error in the following way:

$$\Omega_n - \Omega^o = \sum_{i=1}^{d}\varepsilon_i(n)P_i \tag{33}$$

where P_i is the ith column of $P, \varepsilon_i(n)$ denotes the distance between Ω_n and Ω^o in the i th eigenvector direction. It has been shown in [6] that

$$E[\varepsilon_i(n)] = (1-\eta\varsigma_i)^n\varepsilon_i(0) \tag{34}$$

$$E[|\varepsilon_i(n)|^2] = \frac{\eta J_{\min}}{2-\eta\varsigma_i} + (1-\eta\varsigma_i)^{2n}\left(|\varepsilon_i(0)|^2 - \frac{\eta J_{\min}}{2-\eta\varsigma_i}\right) \tag{35}$$

where $J_{\min} = E[|v(n)|^2]$ is the irreducible mean square error.

In the case of $\varsigma_i = 0$, (34) and (35) yield

$$E[\varepsilon_i(n)] = \varepsilon_i(0) \tag{36}$$

$$E[|\varepsilon_i(n)|^2] = |\varepsilon_i(0)|^2. \tag{37}$$

That means $\varepsilon_i(n)$ will stay in the neighborhood of the initial value forever. Therefore, if the initialization is zero and the weight is decomposed as

$$\hat{\Omega} = \sum_{i=1}^{d} \hat{\Omega}_i P_i \tag{38}$$

the KLMS converges asymptotically (in the mean sense) to

$$\hat{\Omega}_i = \begin{cases} \Omega_i^o, & i = 1, \ldots, r \\ 0, & i = r+1, \ldots, d. \end{cases} \tag{39}$$

Equation (39) exactly states that the KLMS algorithm converges asymptotically (in the mean sense) to the LS pseudoinverse solution. This is expected because the LMS algorithm is only able to *seek a solution in the data subspace* and is not affected by the null subspace. For small eigenvalues the convergence speed is very slow, but it does converge asymptotically. ■

Notice that the above analysis is based on the asymptotic assumption, i.e., N approaches to infinity. However, in practice, the KLMS as formulated in (10) is applied for finite data and convergence would be never attained for *small* eigenvalue components. For example, assuming the step size $\eta = 0.1$, the eigenvalue is $\varsigma_i = 0.001$ and $N = 500$, we have $(1 - \eta\varsigma_i)^N = 0.95$, which means the corresponding component is attenuated by 0.05. In the *finite data sample case*, the LMS solution will differ significantly from the LS solution, with very important properties. More specifically, we have the following.

Proposition 3.2: If the KLMS algorithm is trained with N data points, under the small-step-size condition, the solution is (in the mean sense)

$$\hat{\Omega}_i = \begin{cases} [1 - (1 - \eta\varsigma_i)^N]\varsigma_i^{-1}(P^T p_\Phi)_i, & i = 1, \ldots, r \\ 0, & i = r+1, \ldots, d \end{cases} \tag{40}$$

where $\hat{\Omega}_i$ is defined in (38).

Proof: The initialization is zero so

$$\varepsilon_i(0) = -\Omega_i^o. \tag{41}$$

By (34)

$$\begin{aligned} E[\hat{\Omega}_i(n)] &= E\left[\Omega_i^o + \varepsilon_i(n)\right] = \Omega_i^o + E[\varepsilon_i(n)] \\ &= [1 - (1 - \eta\varsigma_i)^n]\Omega_i^o. \end{aligned} \tag{42}$$

On the other hand, by (21), the LS pseudoinverse solution can be decomposed in the eigendirections as

$$\Omega_{i,\mathrm{LSPI}} = \begin{cases} \varsigma_i^{-1}(P^T p_\Phi)_i, & i = 1, \ldots, r \\ 0, & i = r+1, \ldots, d \end{cases} \tag{43}$$

and then using (43) as the optimal solution, we complete the proof. ■

Writing (40) into matrix form yields

$$\begin{aligned} \hat{\Omega} &= P\,\mathrm{diag}\left([1 - (1 - \eta\varsigma_1)^N]\varsigma_1^{-1}, \ldots, \right. \\ &\quad \left. [1 - (1 - \eta\varsigma_r)^N]\varsigma_r^{-1}, 0, \ldots, 0\right) P^T p_\Phi \\ &= P\,\mathrm{diag}\left([1 - (1 - \eta\varsigma_1)^N]\varsigma_1^{-1}, \ldots, \right. \\ &\quad \left. [1 - (1 - \eta\varsigma_r)^N]\varsigma_r^{-1}, 0, \ldots, 0\right) P^T D_\Phi^T \vec{y}/N \\ &= P\,\mathrm{diag}\left(\left[1 - \left(1 - \eta s_1^2/N\right)^N\right] s_1^{-1}, \ldots, \right. \\ &\quad \left. \left[1 - \left(1 - \eta s_r^2/N\right)^N\right] s_r^{-1}, 0, \ldots, 0\right) Q^T \vec{y}. \end{aligned} \tag{44}$$

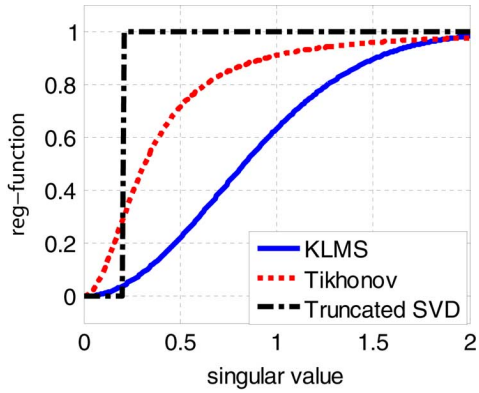

Fig. 1. Reg-functions of the KLMS ($\eta = 1, N = 500$), Tikhonov ($\lambda = 0.1$) and Truncated SVD ($a = 0.2$).

Comparing with (24) and (25), the finite sample KLMS algorithm modifies the singular values by the following reg-function:

$$H_{\mathrm{LMS}}(x) = 1 - (1 - \eta x^2/N)^N. \tag{45}$$

Notice that $\lim_{x\to 0} H_{\mathrm{KLMS}}(x) \cdot x^{-1} = 0$ by the L'Hopital's rule. Indeed the KLMS trained with finite data filters out the small singular value components as prescribed by the Tikhonov regularization (see Fig. 1). In other words, the KLMS of (10) provides well-posed solutions even when the empirical risk cost (15) is ill-posed.

In this section, we established important KLMS algorithm properties in parallel with the conventional LS solutions. The two propositions are based on the small-step-size theory and the conclusions are in the mean sense. The analysis reveals a convergence behavior for the KLMS trained with finite data akin of algorithms that are regularized with a weight decaying Tikhonov regularization technique. This is particularly interesting since the KLMS is a stochastic, on-line algorithm, while the LS solution utilizes an analytic, batch mode approach.

It is clear from the above analysis that the well-posedness of KLMS is dependent upon the step size and training data size which also affects two fundamental characteristics of on-line, steep descent algorithms: the speed of convergence and the misadjustment. When seen from the point of view of well-posedness, the goal is to find a small step-size so that the algorithm converges along large eigenvalue directions and is insensitive along small eigenvalue directions. Of course this depends on the eigenvalue spread of the problem, but the user can still select the stopping criterion and the step size. Intrinsically, the selection of an appropriate step size for good generalization collides with the requirement for fast convergence, and corroborates solutions with small misadjustment. More specifically, the misadjustment is [6], [25]

$$\mathrm{Mt} = \frac{J(\infty) - J_{\min}}{J_{\min}} = \frac{\eta}{2}\mathrm{tr}[R_\Phi] = \frac{\eta}{2N}\mathrm{tr}[G_\Phi]. \tag{46}$$

And for shift-invariant kernels, i.e.,

$$\langle \Phi(u_i), \Phi(u_i) \rangle_H = \|\Phi(u_i)\|^2 = g_0 \tag{47}$$

the misadjustment of the KLMS simply equals $\eta g_0/2$, which is data independent but is proportional to the step size. However,

the best solution from the point of view of convergence is not obtained with the smallest possible step size, as in the misadjustment and regularization. Ultimately, the step size is required to satisfy the following condition for the algorithm to stay stable:

$$\eta < 1/\varsigma_1. \tag{48}$$

Since $\varsigma_1 < \mathrm{tr}[R_\Phi] = \mathrm{tr}[G_\Phi]/N$, a conservative upper bound for the step size is

$$\eta < N/\mathrm{tr}[G_\Phi]. \tag{49}$$

In the case of shift-invariant kernels, the conservative upper bound is trivially g_0, which is also verified by the condition in H^∞ stability discussed in the following section.

IV. Solution Norm Analysis of the Kernel LMS With Finite Data

In this section, we will give a rigorous analysis of the norm of the solution obtained from the KLMS, thus complementing the insightful analysis of Section III. We will show that the KLMS algorithm trained with finite data should be regarded as the stochastic gradient approximation of a *constrained* least-squares optimization

$$\min_{\Omega} J(\Omega) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \Omega\Phi_i)^2 \quad \text{subject to } \|\Omega\|^2 \le C \tag{50}$$

instead of the conventional view of a stochastic approximation to the unregularized empirical risk minimization. Moreover, we will explicitly estimate C.

Theorem 4.1: (H^∞ stable [25]) Given training data $\{\Phi_i, y_i\}_{i=1}^N$ that satisfy the linear regression model (18) for any unknown vector Ω^o and finite energy noise sequence $\{v(i)\}$ without any statistical assumption. Use the KLMS algorithm to estimate $s(i) = \Omega^o(\Phi_i)$ by $\hat{s}(i\,|\,i-1) = \Omega_{i-1}(\Phi_i)$.

The following inequality holds:

$$\frac{\sum_{j=1}^{i}|\hat{s}(j|j-1) - s(j)|^2}{\eta^{-1}\|\Omega^o\|^2 + \sum_{j=1}^{i-1}|v(j)|^2} < 1, \quad \text{for all } i = 1, 2, \ldots, N \tag{51}$$

if and only if the matrices $\{\eta^{-1}I - \Phi_i\Phi_i^T\}$ are positive-definite for $i = 1, 2, \ldots, N$. ∎

This theorem will be used to prove the following results that, to the best of our knowledge, are not in the present literature.

Theorem 4.2: Under the H^∞ stable condition, the *a priori* error defined in (10) satisfies the following inequality:

$$\|\vec{e}^a\|^2 < \eta^{-1}\|\Omega^o\|^2 + 2\|\vec{v}\|^2 \tag{52}$$

where $\vec{e}^a = [e_1^a, e_2^a, \ldots, e_N^a]^T$ and $\vec{v} = [v(1), v(2), \ldots, v(N)]^T$.

Proof: First, notice that

$$e_i^a - v(i) = s(i) - \hat{s}(i\,|\,i-1). \tag{53}$$

Substituting (53) into (51), we have

$$\frac{\sum_{j=1}^{i}\left|e_j^a - v(j)\right|^2}{\eta^{-1}\|\Omega^o\|^2 + \sum_{j=1}^{i-1}|v(j)|^2} < 1, \quad \text{for all } i = 1, 2, \ldots, N \tag{54}$$

or, equivalently,

$$\sum_{j=1}^{i}\left|e_j^a - v(j)\right|^2 < \eta^{-1}\|\Omega^o\|^2 + \sum_{j=1}^{i-1}|v(j)|^2, \quad \text{for all } i = 1, 2, \ldots, N. \tag{55}$$

By the triangle inequality

$$\begin{aligned}\sum_{j=1}^{i}|e_j^a|^2 &\le \sum_{j=1}^{i}|e_j^a - v(j)|^2 + \sum_{j=1}^{i}|v(j)|^2 \\ &< \eta^{-1}\|\Omega^o\|^2 + \sum_{j=1}^{i-1}|v(j)|^2 + \sum_{j=1}^{i}|v(j)|^2, \\ &\quad \text{for all } i = 1, 2, \ldots, N.\end{aligned} \tag{56}$$

In term of norms

$$\|\vec{e}^a\|^2 < \eta^{-1}\|\Omega^o\|^2 + 2\|\vec{v}\|^2. \tag{57}$$

∎

Remark: Based on the above result, we can show further that

$$\|\vec{e}^a\|^2 < 2\|\vec{y}\|^2. \tag{58}$$

The proof is in the Appendix.

Theorem 4.3: Under the H^∞ stable condition, the norm of the solution Ω_N given by (13) is upper-bounded:

$$\|\Omega_N\|^2 < \sigma_1\eta(\|\Omega^o\|^2 + 2\eta\|\vec{v}\|^2) \tag{59}$$

where σ_1 is the largest eigenvalue of G_Φ.

Proof: By (13) and (57)

$$\begin{aligned}\|\Omega_N\|^2 &= \eta^2(\vec{e}^a)^T G_\Phi \vec{e}^a \le \sigma_1\eta^2\|\vec{e}^a\|^2 \\ &< \sigma_1\eta(\|\Omega^o\|^2 + 2\eta\|\vec{v}\|^2).\end{aligned} \tag{60}$$

∎

We can better appreciate the result obtained in (60), by noting that the upper bound smoothly depends on the training data, which exactly agrees with the stability condition of well-posedness according to the definition of Hadamard [12]. The effect of choosing a smooth kernel is demonstrated by the dependency of the upper bound on the largest eigenvalue. In the case of the Gaussian kernel, the largest eigenvalue is well upper bounded

regardless of the input signal energy unlike the polynomial kernel. On the other hand, in the linear case, the LMS operates in a relatively small dimensional space and the ill-posedness due to insufficient training data is unlikely to happen.

Alternatively, in light of these facts, KLMS trained with finite data has also a very nice interpretation in terms of Tikhonov regularization theory. Instead of a stochastic gradient approximation of the unconstrained least-squares optimization

$$\min_{\Omega} J(\Omega) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \Omega\Phi_i)^2 \tag{61}$$

KLMS is a stochastic gradient approximation of the following *constrained* least-squares optimization:

$$\min_{\Omega} J(\Omega) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \Omega\Phi_i)^2$$
$$\text{subject to } \|\Omega_N\|^2 < \sigma_1\eta(\|\Omega^o\|^2 + 2\eta\|\vec{v}\|^2) \tag{62}$$

which according to Theorem 2.2 is equivalent to Tikhonov regularization.

The significance of an upper bound for the solution norm is well studied by Poggio and Girosi in the context of regularization network theory [14], [26]. A constraint on the solution norm like (62) ensures well-posedness of the problem through the resulting compactness of the effective hypothesis space. In Poggio's words, compactness of the hypothesis space is sufficient for consistency of empirical error minimization, forcing smoothness and stability [14].

Note that the upper bound obtained in (31) and (58) are both data dependent. The dependence on $\vec{y}$ is understandable since at the core, both are essentially solving an approximate linear system "$A\vec{x} \approx \vec{y}$" and the solution is of the form "$A^*\vec{y}$", where "A^*" is some reasonable approximation of the matrix inverse. As pointed out in [13] and [27], the ill-posedness lies exactly in this *inverse* operation, and different kinds of regularization techniques can be viewed as different ways of getting meaningful inverse approximations. For example, in an unconstrained RBF network, the solution norm is *reciprocally* proportional to the minimum singular value (see Theorem 2.1), which is surely problematic when it is very close to zero. In other words, a small change in the minimum singular value due to small changes of training data may cause a huge difference in the solution.

Another comment follows from the Bayesian interpretation of regularization [14], [28]. Under the Gaussian assumption, an optimal choice of the regularization parameter in (29) is $2\sigma_y^2/N$, where σ_y is the noise variance in the observations and N is the number of training points. This indicates the bound on the solution norm should be relaxed with more training data available according to Theorem 2.2, which also validates the data dependence of the bound from this perspective. The independent and identically distributed (i.i.d.) assumption of this result also suggests training data should not be repeatedly used, which may jeopardize well-posedness.

The results outlined in Theorem 4.3 depend on the finite data assumption. In practice, some linear dependency [9] or novelty criterion [11] will be employed in the selection of kernel centers.

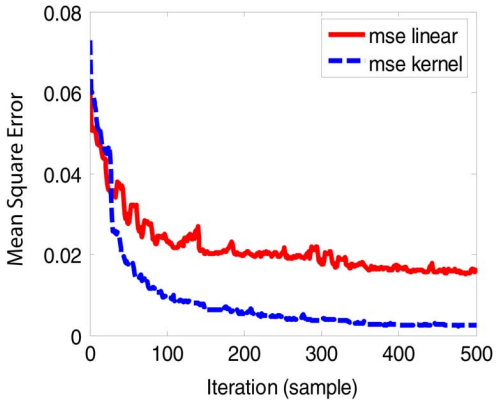

Fig. 2. Learning curves of the LMS and kernel LMS (learning rate 0.2 for both).

Therefore, it is guaranteed to have finite kernel centers during training with the compactness assumption on the input space. On the other hand, we can also assume the noise energy is finite in the model (18) [25].

V. Simulations

A. Time Series Prediction

The first example is the short-term prediction of the Mackey-Glass chaotic time series with parameter $\tau = 30$, and the sampling period 6 s. We compare the performance of a linear combiner trained with LMS, the KLMS and a regularization network (RN), which implicitly uses a radial basis function network topology specified by the kernel utilized. The time embedding is 10 for all the systems and a segment of 500 samples is used as the training data and another 100 as the test data. All the data is corrupted by Gaussian noise with zero mean and 0.04 variance. A Gaussian kernel with kernel width 1 is chosen for both RN and KLMS. In RN, every input point is used as the center and the training is done in batch mode. There are 100 Monte Carlo simulations that are run with different realizations of noise. The results are summarized in subsequent tables. Fig. 2 is the learning curves for the LMS and KLMS with a learning rate of 0.2 for both algorithms. As expected, the KLMS converges to a smaller value of MSE due to its nonlinear nature. Surprisingly, the rate of decay of both learning curves is basically the same, which implies that the effective eigenvalue spread of the data matrix in the input and projected spaces are rather similar.

As one can observe in Table I, the performance of the KLMS is approximately five times better than the linear LMS as can be expected (the MG is a nonlinear system) and is comparable to the RN with suitable regularization. This is indeed surprising since the RN can be viewed as a batch mode kernel regression method versus the KLMS which is a straight stochastic gradient approach implemented in the RKHS. All the results in the tables are in the form of "average $\pm$ standard deviation." It is interesting to compare the design and performance of KLMS with different step sizes and the RN with different regularizations since each controls the stability of the obtained solution. First of all, when the regularization parameter is zero, the RN performance in the test set is pretty bad (worse than the linear

TABLE I
PERFORMANCE COMPARISON WITH DIFFERENT STEP SIZES AND REGULARIZATION PARAMETERS

Algorithms	Linear LMS	KLMS (η=0.1)	KLMS (η=0.2)	KLMS(η=0.6)	RN (λ=0)	RN (λ=1)	RN (λ=10)
Training MSE	0.021±0.002	0.0074±0.0003	0.0054±0.0004	0.0062±0.0012	0±0	0.0038±0.0002	0.011±0.0001
Testing MSE	0.026±0.007	0.0069±0.0008	0.0056±0.0008	0.0058±0.0017	0.012±0.004	0.0039±0.0008	0.010±0.0003

TABLE II
ALGORITHM COMPLEXITY COMPARISON

Algorithms	Linear LMS	KLMS	RN
Computation (training)	O(N)	$O(N^2)$	$O(N^3)$
Memory (training)	O(M)	O(N)	$O(N^2)$
Computation (test)	O(M)	O(N)	O(N)
Memory (test)	O(M)	O(N)	O(N)

TABLE III
SOLUTION NORM COMPARISON

Algorithms	KLMS(η=0.1)	KLMS(η=0.2)	KLMS(η=0.6)	RN (λ=0)	RN (λ=1)	RN (λ=10)
solution norm	0.84±0.02	1.14±0.02	1.73±0.06	3375±639	1.47±0.03	0.55±0.01

TABLE IV
PERFORMANCE COMPARISON WITH DIFFERENT NOISE LEVELS

Algorithms	Linear LMS	KLMS (η=0.1)	RN (λ=1)
Training MSE ($\sigma = .005$)	0.017±5e-5	0.0050±2e-5	0.0014±1e-5
Testing MSE ($\sigma = .005$)	0.018±0.0002	0.0041±0.0001	0.0012±6e-5
Training MSE ($\sigma = .02$)	0.018±0.0002	0.0055±0.0001	0.0020±6e-5
Testing MSE ($\sigma = .02$)	0.018±0.0007	0.0046±0.0004	0.0016±0.0002
Training MSE ($\sigma = .04$)	0.021±0.002	0.0074±0.0003	0.0038±0.0002
Testing MSE ($\sigma = .04$)	0.026±0.007	0.0069±0.0008	0.0039±0.0008
Training MSE ($\sigma = .1$)	0.033±0.001	0.019±0.001	0.010±0.0005
Testing MSE ($\sigma = .1$)	0.031±0.005	0.018±0.003	0.017±0.003
Training MSE ($\sigma = .5$)	0.326±0.015	0.252±0.010	0.097±0.003
Testing MSE ($\sigma = .5$)	0.363±0.057	0.332±0.052	0.331±0.052

solution), which indicates that the problem is ill-posed (i.e., that the inverse is ill conditioned) and needs regularization. The RN is capable of outperforming the KLMS in the test set with the proper regularization parameter $(\lambda = 1)$, but the difference is small and at the expense of a more complex solution (see Table II) as well as with a careful selection of the regularization parameter. Table II summarizes the computational complexity of the three algorithms. The KLMS effectively reduces the computational complexity and memory storage when compared with the RN. And if we have direct access to the feature space, the complexity can be reduced even further to the dimensionality of the feature space if it is not prohibitively large. Table III supports our theory that the norm of the KLMS solution is well-bounded. As we see, increasing the step size in the KLMS increases the norm of the solution but fails to increase the performance because of the gradient noise in the estimation (misadjustment).

Next, different noise variance σ^2 is used in the data to further validate KLMS applicability. As we see in Table IV, the KLMS performs consistently on both the training and test with different noise level and degrades gracefully with increasing noise, which is an illustration of its well-posedness. It is observed that at severe noise level $(\sigma = .5)$, all methods fall apart since the noise component will no longer correspond to the smallest singular value as required by Tikhonov regularization. With small noise, the regularization network outperforms the KLMS since misadjustment plays a more critical role here. This is a quite good illustration of the difficulty the KLMS may face to balance among convergence, misadjustment and regularization. But remember the KLMS is a much simpler, online algorithm and the performance gap compared with RN is the price to pay. Throughout this set of simulation, the kernel used in the KLMS and the RN is the Gaussian kernel with kernel size 1. The learning step is 0.1 for both the linear LMS and the KLMS. The regularization parameter of the RN is set at the best value $(\lambda = 1)$.

Any kernel method, including the KLMS, needs to choose a suitable kernel. In the third set of simulations, the effect of different kernels and different kernel parameters on KLMS is

TABLE V
PERFORMANCE COMPARISON OF DIFFERENT KERNEL SIZES OF THE GAUSSIAN KERNEL

	Linear LMS	KLMS (a=10)	KLMS (a=2)	KLMS(a=0.2)	RN (a=10)	RN (a=2)	RN (a=0.2)
Training MSE	0.022±0.001	0.0085±0.0005	0.0061±0.0003	0.017±0.0007	0.0040±0.0002	0.0043±0.0002	0.0098±0.0003
Testing MSE	0.022±0.001	0.0078±0.0010	0.0056±0.0014	0.016±0.0010	0.0068±0.0009	0.0047±0.0006	0.0092±0.0005

TABLE VI
PERFORMANCE COMPARISON OF DIFFERENT ORDERS OF THE POLYNOMIAL KERNEL

	KLMS (p=2, η=0.1)	KLMS (p=5, η=0.01)	KLMS (p=8, η=0.0006)	RN (p=2, λ=1)	RN (p=5, λ=1)	RN (p=8, λ=1)
Training MSE	0.010±0.001	0.0099±0.0006	0.027±0.009	0.0064±0.0005	0.0034±0.0003	0.0014±0.0001
Testing MSE	0.009±0.002	0.0099±0.0007	0.025±0.009	0.0066±0.0008	0.0059±0.0007	0.0078±0.0004

TABLE VII
PERFORMANCE COMPARISON OF DIFFERENT TRAINING DATA SIZE

	Linear LMS (N=1000)	Linear LMS (N=2000)	Linear LMS (N=4000)	KLMS (N=1000)	KLMS (N=2000)	KLMS (N=4000)
Training MSE	0.020±0.0004	0.019±0.0004	0.019±0.0003	0.0060±0.0002	0.0058±0.0002	0.0054±0.0001
Testing MSE	0.019±0.0015	0.018±0.0009	0.020±0.0016	0.0062±0.0009	0.0053±0.0010	0.0058±0.0007

demonstrated. In the case of the Gaussian kernel (7), we choose three kernel parameters: 10, 2, and 0.2. The learning rate is 0.1 for both the linear LMS and the KLMS and the regularization parameter of the RN is 1 throughout the simulation. The results are summarized in Table V. As expected, kernel sizes that are too small or too large hurt performance for both KLMS and RN. In this problem, a kernel size around 1 gives the best performance on the test set. In the case of the polynomial kernel, the order is set to 2, 5, and 8. The learning rate is chosen accordingly in the KLMS as listed in Table VI (recall the relation between the learning rate and the trace of the Gram matrix). It is observed that the performance deteriorates substantially when p is too large (>8) for the KLMS. This is also validated by the misadjustment formula (46).

The general kernel selection problem has received a lot of attention in the literature (see [29]–[31] and references therein). The general idea is to formulate the problem as a convex optimization through parameterization of the kernel function family. It will be interesting to investigate this problem specifically for the KLMS and we will pursue the research in future work. One of the interesting observations is the automatic normalization operated by the Gaussian kernel (or any shift-invariant kernel) in the KLMS. This also explains the robustness with regard to step size in the case of the Gaussian kernel, and the difficulties encountered in the polynomial case.

It is noted in the theoretical analysis that the number of training data affects the well-posedness of the solution. In the last set of simulations, we choose different training data sizes to see how the KLMS performs. The noise variance is set at 0.05 and the number of training data is 1000, 2000, and 4000, respectively. Other parameters are the same as in the first set of simulations. As presented in Table VII, the performance of the KLMS is consistently better on the training and test sets with increasing number of training data. Because, when the number of training data increases, the ill-posedness of the empirical risk minimization (15) itself is relieved.

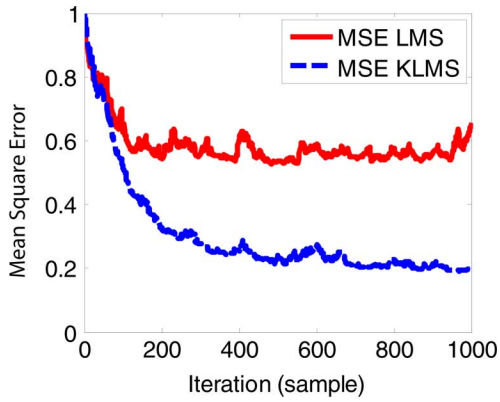

Fig. 3. Learning curves of the LMS $(\eta = 0.005)$ and kernel LMS $(\eta = 0.1)$ in the nonlinear channel equalization $(\sigma = 0.4)$.

B. Nonlinear Channel Equalization

The LMS algorithm is widely used in channel equalization. The problem setting is as follows: A binary signal $\{s_1, s_2, \ldots, s_L\}$ is fed into a nonlinear channel. At the receiver end of the channel, the signal is further corrupted by additive i.i.d. Gaussian noise and is then observed as $\{r_1, r_2, \ldots, r_L\}$. The aim of channel equalization (CE) is to construct an "inverse" filter that reproduces the original signal with as low an error rate as possible. It is easy to formulate it as a regression problem, with examples $\{(r_{t+D}, r_{t+D-1}, \ldots, r_{t+D-l+1}), s_t\}$, where l is the time embedding length, and D is the equalization time lag.

In this experiment, the nonlinear channel model is defined by $z_t = s_t + 0.5s_{t-1}, r_t = z_t - 0.9z_t^2 + n_\sigma$, where n_σ is the white Gaussian noise with a variance of σ^2. The learning curve is plotted in Fig. 3, which is similar to the previous example. Testing was performed on a 5000-sample random test sequence. We compare the performance of the conventional LMS, the KLMS and the RN as a batch mode baseline. The Gaussian

TABLE VIII
PERFORMANCE COMPARISON IN NCE WITH DIFFERENT NOISE LEVELS σ

Algorithms	Linear LMS (η=0.005)	KLMS (η=0.1)	RN (λ=1)
BER ($\sigma = .1$)	0.162±0.014	0.020±0.012	0.008±0.001
BER ($\sigma = .4$)	0.177±0.012	0.058±0.008	0.046±0.003
BER ($\sigma = .8$)	0.218±0.012	0.130±0.010	0.118±0.004

kernel with $a = 0.1$ is used in the KLMS for best results, and $l = 5$ and $D = 2$. The results are presented in Table VIII; each entry consists of the average and the standard deviation for 100 repeated independent tests. The results in Table VIII show that the RN outperforms the KLMS in terms of the bit error rate (BER) but not by much, which is surprising since one is a batch method, and the other is online. They both outperform the conventional LMS substantially as can be expected because the channel is nonlinear. The regularization parameter for the RN and the learning rate of KLMS were set for optimal results.

VI. DISCUSSION AND CONCLUSION

This paper proposes the KLMS algorithm which is intrinsically a stochastic gradient methodology to solve the LS problem in RKHS. Since the update equation of the KLMS can be written as inner products, KLMS can be efficiently computed in the input space. The good approximation ability of the KLMS stems from the fact that the transformed data $\Phi(u_n)$ includes possibly infinite different features of the original data. In the framework of stochastic projection, the space spanned by $\{\Phi(u_i)\}$ is so large that the projection error of the desired signal y_n could be very small [32], as is well known from Cover's theorem [10]. This capability includes modeling of nonlinear systems, which is the main reason why the kernel LMS can achieve good performance in the Mackey–Glass system prediction and nonlinear channel equalization.

However, this projection to potentially infinite dimensional spaces brings two types of concerns. First, since one works in practice with finite data sets, is the solution well-posed in the feature space? Second, since gradient descent algorithms depend upon the eigenvalue spread of the data correlation matrix, and the misadjustment is also a function of the trace of this matrix, does the KLMS provide practical solutions?

This paper provides a conclusive answer to the first question that is rather surprising, in light of the results and methods available in the kernel methods literature. In fact, the KLMS trained with finite data provides well-posed solutions, even when the original ERM problem is ill-posed. We show this in two ways for clarity. First, we use the idea of singular value filtering derived from Tikhonov regularization to show that the KLMS "filters" the small singular values of the data matrix dependent upon the step size and the data size. More formally we show that the norm of the KLMS solution is always bounded. Therefore, the KLMS is well posed, being "self-regularized" by the step size which effectively works as the regularization parameter.

These results seem to imply that the KLMS provides a "free lunch." Unfortunately, this is not the case, but there are important implications for the design and implementation of adaptive solutions in high dimensional spaces. In terms of design, there is no need to regularize the LS cost function when using the KLMS algorithm. In terms of implementation, the KLMS is far simpler to implement, both in terms of computational complexity and memory storage compared with other kernel methods, and it is intrinsically online, inherited from the LMS algorithm. The reason there is no free lunch, is that the generalization performance is dependent upon the step size which also affects two fundamental characteristics of the KLMS algorithm: the speed of convergence and the misadjustment. The requirements for regularization, convergence speed, and misadjustment do indeed conflict among themselves, and it is important to understand how regularization changes the well-known speed–misadjustment compromise.

Regarding the second concern, further work should be conducted to elucidate the eigenstructure of the data in feature space, which as we know limits the speed of convergence of the algorithm. As demonstrated in Figs. 2 and 3, at least for the problems investigated here, the eigenstructure must be similar because the two curves converge at the same rate. For a complete discussion of convergence, a rigorous eigenspread analysis after the nonlinear transformation is essential. However, as shown in most of the kernel principal component analysis applications, the transformed data in the feature space are usually well clustered in a small subspace (say, 95% power) [5], [33] such that fast convergence is attained for the kernel LMS algorithm, which is also corroborated in our experiments.

Demonstrated by the experiments, the KLMS has general applicability due to its simplicity since it is impractical to work with batch mode kernel methods in large data sets. The KLMS is very useful in problems like nonlinear channel equalization, nonlinear system identification, nonlinear active noise control, where online learning is a necessity. It also provides a possible tool to tackle problems in the area of nonlinear nonstationary signal processing.

To conclude, this paper provides a significant theoretical result concerning the well-posedness property of the KLMS, and meanwhile provides a clear guideline on how to design the LMS algorithm in high-dimensional space and how to choose the stepsize considering convergence, misadjustment *and regularization*, which has, to the best of our knowledge, been lacking or improperly addressed. However, this paper still does not address many implementation details that are important and that should be thoroughly investigated in future research, such as the issue of the kernel design (shape and bandwidth parameters), convergence and optimal learning rates and its implication in regularization and misadjustment, as well as its behavior in nonstationary environments.

APPENDIX I

Proof of Theorem 4.2: Notice (57) is valid for any unknown vector Ω^o and finite energy noise sequence $\{v(i)\}$ and a tradeoff is there to achieve the tightest bound. A trivial Ω^o can be chosen with large noise term in (18). On the other hand, a "precise" Ω^o can be chosen such that $\|\vec{v}\| = 0$, while the norm of this choice $\|\Omega^o\|$ may be very large. For this problem, $\Omega^o = 0$ can be chosen and $v(i) = y_i$ follows. Substituting them into (57) yields

$$\|\vec{e}^a\|^2 < 2\|\vec{y}\|^2. \tag{63}$$

We can utilize the results from the regularization networks to show that this is actually a very tight bound. Using the results from the regularization networks, we have

$$\|\Omega^o\|^2 = \vec{\alpha}^T G_\Phi \vec{\alpha} = \vec{y}^T (G_\Phi + \lambda I)^{-1} G_\Phi (G_\Phi + \lambda I)^{-1} \vec{y} \tag{64}$$

$$\|\vec{v}\|^2 = \|\vec{y}\|^2 - 2\vec{y}^T G_\Phi \vec{\alpha} + \vec{\alpha}^T G_\Phi G_\Phi \vec{\alpha}. \tag{65}$$

Substituting (64) and (65) into (57) yields

$$\|\vec{e}^a\|^2 < \vec{y}^T [(G_\Phi + \lambda I)^{-1}(2G_\Phi + \eta^{-1})G_\Phi (G_\Phi + \lambda I)^{-1} + 2 - 4G_\Phi (G_\Phi + \lambda I)^{-1}]\vec{y}. \tag{66}$$

Define

$$M(\lambda) =: [(G_\Phi + \lambda I)^{-1}(2G_\Phi + \eta^{-1})G_\Phi (G_\Phi + \lambda I)^{-1} + 2 - 4G_\Phi (G_\Phi + \lambda I)^{-1}]. \tag{67}$$

Suppose $G_\Phi = Q \sum Q^T$ and $\sum = \mathrm{diag}(\sigma_1, \ldots, \sigma_N)$, where $\sigma_1 \geq \cdots \geq \sigma_N \geq 0$

$$M(\lambda) = Q \left[\mathrm{diag} \left\{ \frac{\sigma_1 \eta^{-1} + 2\lambda^2}{(\sigma_1 + \lambda)^2}, \ldots, \frac{\sigma_N \eta^{-1} + 2\lambda^2}{(\sigma_N + \lambda)^2} \right\} \right] Q^T. \tag{68}$$

Since (57) is valid for any λ, we try to find the tightest upper-bound.

Let us analyze the following function:

$$h_i(\lambda) = \frac{\sigma_i \eta^{-1} + 2\lambda^2}{(\sigma_i + \lambda)^2}. \tag{69}$$

Notice that $\lambda = \infty$ corresponds to the trivial solution $\Omega^o = 0$. It is easy to check that the minimum is at $\lambda_* = 1/2\eta$. Therefore, specifically

$$\|\vec{e}^a\|^2 < 2\|Q^T \vec{y}\|^2 = 2\|\vec{y}\|^2. \tag{70}$$

■

REFERENCES

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

[2] S. Smale and Y. Yao, "Online learning algorithms," *Found. Comput. Math.*, vol. 6, no. 2, pp. 145–170, 2006.

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

[4] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Comput.*, vol. 7, pp. 219–269, 1995.

[5] B. Scholkopf, A. Smola, and K. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, pp. 1299–1319, 1998.

[6] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[7] K. Kim, M. O. Franz, and B. Scholkopf, "Iterative kernel principal component analysis for image modeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 9, pp. 1351–1366, Sep. 2005.

[8] T. Frieb and R. Harrison, "A kernel-based adaline," in *Proc. Eur. Symp. Artificial Neural Networks (ESANN)*, Bruges, Belgium, Apr. 21–23, 1999, pp. 245–250, D-Facto public., ISBN 2-600049-9-X.

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

[10] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1998.

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

[12] J. Hadamard, "Sur les problèmes aux dérivées partielles et leur signification physique," *Princeton Univ. Bull.*, pp. 49–52, 1902.

[13] A. Tikhonov and V. Arsenin, *Solution of Ill-Posed Problems*. Washington DC: Winston & Sons, 1977.

[14] T. Poggio and S. Smale, "The mathematics of learning: Dealing with data," *Amer. Math. Soc. Notice*, vol. 50, no. 5, pp. 537–544, 2003.

[15] O. Bousquet and A. Elisseeff, "Stability and generalization," *J. Mach. Learn. Res.*, vol. 2, pp. 499–526, 2002.

[16] P. Pokharel, W. Liu, and J. Principe, "Kernel LMS," in *Proc. Int. Conf. Acoustics, Speech, Signal Process. (ICASSP)*, 2007, vol. 3, pp. III-1421–III-1424.

[17] A. Caponnetto and A. Rakhlin, "Stability properties of empirical risk minimization over Donsker classes," *J. Mach. Learn. Res.*, vol. 7, pp. 2565–2583, 2006.

[18] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, pp. 337–404, 1950.

[19] G. Golub and C. Loan, *Matrix Computations*, 2nd ed. Baltimore, MD: The John Hopkins Univ. Press, 1989.

[20] G. W. Stewart, *Introduction to Matrix Computations*. New York: Academic, 1973.

[21] A. Català and C. Angulo, "A comparison between the Tikhonov and the Bayesian approaches to calculate regularization matrices," *Neural Process. Lett.*, vol. 11, no. 3, pp. 185–195, Jun. 2000.

[22] R. Fierro, G. H. Golub, P. C. Hansen, and D. P. O'Leary, "Regularization by truncated total least squares," *SIAM J. Sci. Comput.*, vol. 18, no. 4, pp. 1223–1241, 1997.

[23] C. R. Vogel, "Nonsmooth regularization," in *Inverse Problems Geophys. Appl.*, H. W. Engl, A. K. Louis, and W. Rundell, Eds. Philadelphia, PA: SIAM, 1997, pp. 1–11.

[24] W. Rudin, *Functional Analysis*. New York: McGraw-Hill Science, 1991.

[25] A. Sayed, *Fundamentals of Adaptive Filtering*. New York: Wiley, 2003.

[26] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Comput.*, vol. 7, no. 2, pp. 219–269, 1995.

[27] A. Neumaier, "Solving ill-conditioned and singular linear systems: A tutorial on regularization," *SIAM Rev.*, vol. 40, no. 3, pp. 636–666, 1998.

[28] T. Evgeniou, M. Pontil, and T. Poggio, "Regularization networks and support vector machines," *Adv. Comput. Math.*, vol. 13, pp. 1–50, 2000.

[29] C. Micchelli and M. Pontil, "Learning the kernel function via regularization," *J. Mach. Learn. Res.*, vol. 6, pp. 1099–1125, 2005.

[30] A. Argyriou, C. A. Micchelli, and M. Pontil, "Learning convex combinations of continuously parameterized basic kernels," in *Proc. 18th Conf. Learning Theory*, 2005, pp. 338–352.

[31] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," in *Machine Learning*. New York: Springer, 2002.

[32] E. Parzen, "Statistical methods on time series by Hilbert space methods," Applied Mathematics and Statistics Laboratory, Stanford University, Stanford, CA, Tech. Rep. no. 23, 1959.

[33] T. Zhang, "Learning bounds for kernel regression using effective data dimensionality," *Neural Comput.*, vol. 17, no. 9, pp. 2077–2098, Sep. 2005.

Weifeng Liu (S'06) was born in Shanghai, China. He received the B.S. and M.S. degrees in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2003 and 2005, respectively. He is currently working towards the Ph.D. degree at the Computational NeuroEngineering Laboratory at the University of Florida, Gainesville.

His research interests are signal processing, machine learning and pattern recognition.

Puskal P. Pokharel (S'04) was born in Lalitpur, Nepal, in 1981. He received the Bachelor of Technology degree in electronics and communication engineering from the Indian Institute of Technology (IIT), Roorkee, India, in 2003 and the M.S. degree in electrical and computer engineering from the University of Florida, Gainesville, in 2005. Presently, he is working towards the Ph.D. degree at the University of Florida Compuational NeuroEngineering Laboratory (CNEL).

His current research interests include digital signal processing, machine learning, information theoretic learning, and their applications.

Jose C. Principe (M'83–SM'90–F'00) is Distinguished Professor of Electrical and Biomedical Engineering at the University of Florida, Gainesville, where he teaches advanced signal processing and artificial neural networks (ANNs) modeling. He is BellSouth Professor and Founder and Director of the University of Florida Computational NeuroEngineering Laboratory (CNEL). He is involved in biomedical signal processing, in particular the electroencephalogram (EEG) and the modeling and applications of adaptive systems. He has more than 129 publications in refereed journals, 15 book chapters, and over 300 conference papers. He has directed over 50 Ph.D. dissertations and 61 Master's degree theses.

Dr. Principe is an AIMBE Fellow and a recipient of the IEEE Engineering in Medicine and Biology Society Career Service Award. He is also a former member of the Scientific Board of the Food and Drug Administration, and a member of the Advisory Board of the McKnight Brain Institute at the University of Florida. He is Past Editor-in-Chief of the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, Past President of the International Neural Network Society, and former Secretary of the Technical Committee on Neural Networks of the IEEE Signal Processing Society.