KERNEL LMS

Puskal P. Pokharel, Weifeng Liu, Jose C. Principe

Computational NeuroEngineering Laboratory
Department of Electrical and Computer Engineering, University of Florida

ABSTRACT

In this paper a nonlinear adaptive algorithm based on a kernel space least mean squares (LMS) approach is presented. With most of the neural network based methods for time series modeling it is difficult to implement a sample-by-sample adaptation method. This puts a serious limitation on the applicability of adaptive nonlinear filters in many optimal signal processing and communication applications where data arrives sequentially. This paper shows that the Kernel LMS algorithm provides a computational simple and an effective algorithm to train nonlinear systems for system modeling without the need for regularization, without convergence to local minima and without the need for a separate bock of data as a training set.

Index Terms— LMS, kernel trick, stochastic gradient

1. INTRODUCTION

The least mean squares (LMS) filter is widely used in all areas of adaptive learning from system identification to channel equalization. The popularity of this algorithm since its introduction by Widrow and Hoff [1] in the 1960s has grown immensely because of its simplicity and effectiveness. The idea is an intelligent simplification of the gradient decent method for learning [2] by using the local estimate of the mean square error. In other words, the LMS algorithm is said to employ a stochastic gradient instead of the deterministic gradient used in the method of steepest decent. By design, it avoids the estimation of correlation functions and matrix inversions. Unfortunately these characteristic do not extend to nonlinear adaptive filters. Nonlinear system adaptation based on local gradients require separate blocks of data, called the training set, to be made available before operation. Of course, the LMS algorithm can still be easily used to adapt the linear output layer of nonlinear models as the radial basis function (RBF) network. However, the centers of the basis are selected by the training data (either by centering each Gaussian on a sample or through clustering [3]), so a block of data is still necessary. For multilayer perceptrons (MLPs) the LMS stochastic gradient has to be heavily modified to take into consideration the nonlinearities (the backpropagation algorithm [3]). In MLPs the need for a training set is not fundamental, i.e. backpropagation can be applied one sample at a time, but since all the parameters of the system change with each sample, the performance is so poor early on and the convergence so slow that a training set is practically required. Moreover, the performance surface is most often nonconvex with many local minima, which further complicates training.

Kernel methods have also been proposed to produce nonlinear algorithms from linear ones expressed with inner products by employing the famed *kernel trick* [4]. More recently, there has been an interest in the machine learning community to train kernel regressors or classifiers one sample at a time to counteract the size of the huge datasets of some real world applications. Important theoretical results have proved the convergence of on-line learning algorithms with regularization in reproducing kernel Hilbert spaces [5], and a simple stochastic gradient based algorithm for adaptation has been developed [6]. In [7], a similar formulation is presented, but weighting coefficients are adapted in the input space. In this paper, we shall derive an LMS algorithm directly in kernel feature space and employ the kernel trick to obtain the solution in the input space. Moreover, the impressive results demonstrate that explicit regularization as used in [5, 6] may be redundant.

The next section provides a short introduction to kernel methods and the LMS algorithm. Sections 3 and 4 present the kernel LMS algorithm and some experimental results respectively. Finally, section 5 summarizes the main conclusions and points out some lines for further research.

2. BACKGROUND

2.1. Kernel methods

In the past years a number of kernel methods, including Support Vector Machines (SVM) [4], kernel principal component analysis (K-PCA) [8], and kernel independent component analysis (K-ICA) [9] have been proposed and applied to machine learning and signal processing problems. The basic idea of kernel algorithms is to transform the data $\mathbf{x}_i$ from the input space to a high dimensional feature space of vectors $\Phi(\mathbf{x}_i)$, where the inner products can be computed using a positive definite kernel function satisfying Mercer's conditions [4]: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$. This simple idea allows us to obtain nonlinear versions of any linear algorithm expressed in terms of inner products, without even knowing the exact mapping Φ. A particularly interesting characteristic

This work was supported in part by NSF grant ECS-0601271.

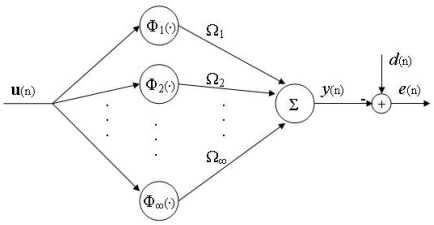

Fig. 1. the linear filter structure with feature vectors in the feature space.

of the feature space is that it is a *reproducing kernel Hilbert space* (RKHS): i.e., the span of functions $\{\kappa(\cdot, \mathbf{x}) : \mathbf{x} \in \mathcal{X}\}$ defines a unique functional Hilbert space [10], [11]. The crucial property of these spaces is the *reproducing property* of the kernel

$$f(\mathbf{x}) = \langle \kappa(\cdot, \mathbf{x}), f \rangle, \qquad \forall f \in \mathcal{F}.$$

In particular, a nonlinear mapping from the input space to an RKHS can be defined as $\Phi(\mathbf{x}) = \kappa(\cdot, \mathbf{x})$ such that

$$\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle = \langle \kappa(\cdot, \mathbf{x}), \kappa(\cdot, \mathbf{y}) \rangle = \kappa(\mathbf{x}, \mathbf{y}),$$

and thus $\Phi(\mathbf{x}) = \kappa(\cdot, \mathbf{x})$ defines the Hilbert space associated with the kernel, and can be thought as a nonlinear transformation from the input to feature space. Without loss of generality, in this paper we will only consider the translation-invariant radial basis (Gaussian) kernel, which is the most widely used Mercer kernel.

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp - \left(\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2} \right) \tag{1}$$

2.2. Least mean squares (LMS) algorithm

The LMS algorithm, introduced in 1960 by Widrow, is a very simple and elegant method of training a linear adaptive system to minimize mean square error. Given a quadratic cost function $J_{\mathbf{w}}(n)$ (where $\mathbf{w}$ is the tap-weight vector and n is the time index), it can be shown that with exact measurements of the gradient vector $\nabla J_{\mathbf{w}}(n)$ and a suitable chosen step-size parameter η, the weight vector updated by using the steepest-descent algorithm converges in the mean to the optimum Wiener solution. The LMS algorithm, instead of using the exact gradient to update the weight vector, uses the instantaneous estimate given by $\nabla \hat{J}_{\mathbf{w}}(n) = -2e(n)\mathbf{u}(n)$ resulting in the following stochastic gradient descent update rule.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\eta e(n)\mathbf{u}(n). \tag{2}$$

The detailed analysis including that of convergence and misadjustment is given in [12] and will not be the focus of this paper.

3. KERNEL LMS

In this section we present the kernel LMS algorithm. The basic idea is to perform the linear LMS algorithm given by (2) in the kernel feature space. For this let us assume that Φ maps the point $\mathbf{u}(n)$ in input space to $\Phi(\mathbf{u}(n))$ in the kernel feature space with $\langle \Phi(\mathbf{u}(n)), \Phi(\mathbf{u}(m)) \rangle = \kappa(\mathbf{u}(n), \mathbf{u}(m))$, where $\langle \cdot, \cdot \rangle$ represents the inner product in the kernel Hilbert space. This transformation to feature space for the most widely used kernels is nonlinear, and depending on the choice of the kernel this space can be infinite dimensional. The Gaussian kernel that we use here will correspond to an infinite dimensional Hilbert space. Since this feature space is linear, $\Phi(\mathbf{u}(n))$ can be considered an infinite dimensional column vector with usual vector inner products. Let Ω be the weight vector in this space such that the output is $y(n) = \langle \Omega(n), \Phi(u(n)) \rangle$. $\Omega(n)$ is Ω at time n. Let $d(n)$ be the desired response. Figure 1 shows the nonlinear filtering scheme with the input vector $\mathbf{u}(n)$ being transformed to the infinite feature vector $\Phi(\mathbf{u}(n))$, whose components are then linearly combined by the infinite dimensional weight vector. Note that there is only one layer of weights in this nonlinear filter, but since the size of feature space is potentially infinite it is a universal approximator [13]. Now, due to the linear structure of the RKHS cost function $J_{\mathbf{\Omega}}(n) = E[(d(n) - y(n))^2]$ can be minimized with respect to Ω. This can be done in the same way as done in (2) using the stochastic instantaneous estimate of the gradient vector, which yields

$$\Omega(n+1) = \Omega(n) + 2\eta e(n)\Phi(\mathbf{u}(n)). \tag{3}$$

Like before, η is the step-size parameter that controls the convergence, speed, and misadjustment of the adaptation algorithm [13, 12]. The only catch here is that Ω in (3) is in the infinite dimensional feature space and it would be practically impossible to update for Ω directly. Instead we shall use (3) to relate each $\Omega(n)$ to its initialization $\Omega(0)$. This would easily give

$$\Omega(n) = \Omega(0) + 2\eta \sum_{i=0}^{n-1} e(i)\Phi(\mathbf{u}(i)). \tag{4}$$

For convenience we shall choose $\Omega(0)$ to be zero (hence $e(0) = d(0)$). The final expression for $\Omega(n)$ becomes

$$\Omega(n) = 2\eta \sum_{i=0}^{n-1} e(i)\Phi(\mathbf{u}(i)). \tag{5}$$

It is here we shall exploit the *kernel trick*. Given $\Omega(n)$ from (5) and the input $\Phi(\mathbf{u}(n))$ the output at n is given by

$$\begin{aligned} y(n) &= \langle \Omega(n), \Phi(\mathbf{u}(n)) \rangle = \eta \sum_{i=0}^{n-1} e(i) \langle \Phi(\mathbf{u}(i)), \Phi(\mathbf{u}(n)) \rangle \\ &= \eta \sum_{i=0}^{n-1} e(i)\kappa(\mathbf{u}(i), \mathbf{u}(n)). \end{aligned} \tag{6}$$

We call (6) the Kernel LMS algorithm. It is clear that, given the kernel, Kernel LMS has a unique solution because it is solving a quadratic problem in feature space. Notice also that the weights of the nonlinear filter are never explicitly used in the Kernel LMS algorithm, so the order of the filter is not user

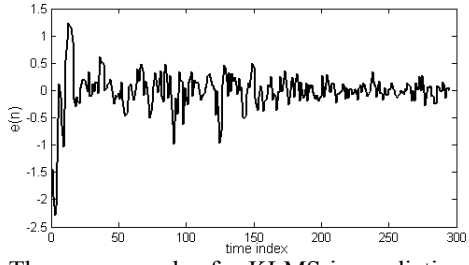

Fig. 2. The error samples for KLMS in predicting Mackey-Glass time series.

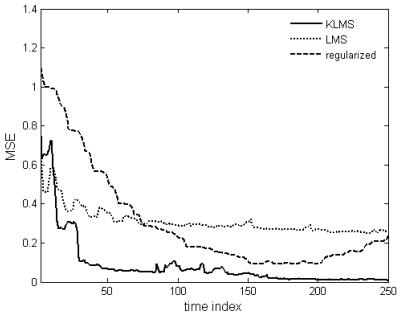

Fig. 3. The learning curves for the LMS, the KLMS and the regularized solution [6].

controllable. More importantly, since the present output is determined solely by previous inputs and all the previous errors, it can be readily computed in the input space! These error samples are similar to *innovation* terms in sequential state estimation [13], as they add new information to improve the output estimate. Each new input sample results in an output and hence a corresponding error, which is never modified further and it is incorporated in the estimate of the next output. This recursive computation makes Kernel LMS especially useful for on-line nonlinear signal processing, but the complexity of the algorithm increases linearly as new error samples are used. The error samples obtained for the Mackey-Glass time series (used in our experiments) are shown in figure 2, and they show a surprisingly fast convergence. The initial errors in the adaptation tend to prevent the algorithm from over fitting the data. The initial errors are the most dominant since the errors decay quickly and seem to have a prominent role in the estimates of the output. We believe this nonparametric improvement in the output samples is a reason for not requiring any kind of explicit regularization. In practice, if the errors are satisfactorily small after p input samples, then the upper index in the summation in (6) can be fixed and the computation complexity for each successive output will be $O(p)$. This is, of course, larger than that for the linear LMS algorithm, but still is smaller than most nonlinear algorithms. The kernel size (the variance parameter in the radial basis function) is a free parameter which affects the overall performance of the algorithm like in any kernel based algorithm and can be chosen through a quick cross validation step. The Silverman's rule [14] of thumb is another alternative. For our simulations the kernel size chosen by the variance of the data works reasonable well. Here, just like for the linear LMS algorithm, the stepsize controls the convergence, speed, and misadjustment of Kernel LMS. Through linear adaptive filter theory it can be expected that for convergence, η is upper bounded by the largest eigenvalue of the data covariance (in the feature space) which is difficult to estimate and dependent upon the kernel size.

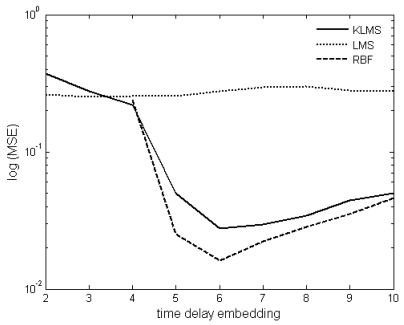

Fig. 4. Comparison of the mean square error for the three methods with varying embedding dimension (filter order for LMS) of the input.

4. EXPERIMENTS AND RESULTS

To demonstrate the performance of the proposed method we will present some simulation results. The mean square error will be used to compare the performance of the Kernel LMS algorithm (KLMS) and that of the linear LMS algorithm for the one step prediction of the Mackey-Glass (MG30) time series. The simulations implement equation (6) for the KLMS and the equation (2) for the linear LMS. The kernel size and the step size were determined for best results after scanning the parameters. The data was normalized to unity variance before hand. The kernel size, σ^2 was chosen to be 1 for these experiments. It was also observed that the performance was not very sensitive to the kernel sizes between 1 and 4. The values of the step size for the LMS and the KLMS algorithms were chosen as 0.01 and 0.5 respectively.

The plots presented include the learning curve and comparisons of MSE values for different embedding dimensions. To plot the learning curve after each update the learning was frozen and a new batch of 300 data samples was used to estimate the mean square error. Figure 3 shows the learning curve for both algorithms (the step size values were chosen for fastest convergence). Surprisingly, the speeds of convergence of both methods (LMS and KLMS) are comparable (i.e. the two learning curves are basically parallel to each other) even though the KLMS is working in an infinite dimensional Hilbert space, where theoretically the eigenvalue spread is unconstrained and statistical reasoning requires regularization to be performed [15]. This can be attributed to the fact that the scattering of the data, although existing in an infinite dimensional space in theory, is such that far lesser canonical directions are dominant (the corresponding covariance matrix has a few eigenvalues that are dominant, while the others are zero for practical purposes). Since the LMS only searches the

space of significant eigenvalues anyway, it tends to concentrate on the signal subspace and suggests that explicit regularization is not required. Figure 3 also includes the regularized case [6] where results were only satisfactory when the regularization parameter was close to zero. Here we used a regularization parameter of 0.7 and stepsize of 0.08. These parameters were chosen after a set of trials to obtain the best MSE after convergence. A systematic way of choosing the regularization parameter for this kind of stochastic learning is still lacking.

The KLMS filter achieves the best result for the embedding dimension of 6 (figure 4), which is the optimal value for the MG time series according to Takens embedding theorem [16]. For the plots, the LMS and the KLMS algorithms were both trained (on-line) for 200 data samples and were tested using a testing set of 300 new data points. As a reference we have also included the results using an RBF network (trained with the least squares method) with 200 RBFs centered at the samples, and 300 new samples used for testing. The results obtained with KLMS is surprisingly close to that of the RBF network considering the 200 "weights" are used so differently: in the RBF all the 200 weights are optimized for the data set, while in the KLMS the errors are computed and fixed for the subsequent samples (i.e. the KLMS is basically nonparametrically trained). Therefore, the on-line use of the data results in a very small degradation of performance. Notice also that training the KLMS was much simpler with no need of matrix computation and inversion, which makes it very practical for real world nonlinear filtering applications.

5. CONCLUSION

This paper derived directly Widrow's least mean squares algorithm in an infinite dimensional kernel Hilbert space. The algorithm uses the kernel trick to obtain the final output, effectively resulting in the adaptation of a nonlinear filter without the complexities of backpropagation and utilizing the data pretty much as the conventional linear filter trained with LMS. Although stochastic learning is known in the machine learning community using regularization, here we found out that for time series modeling the regularization is not necessary to obtain a practical algorithm and a very good solution. Since the LMS tends to search the signal manifold because it stalls in the directions where data has small variance, the problem is solved effectively in a much smaller space spanned by the dominant eigen-directions and does not require external regularization.

The kernel LMS algorithm basically solves the least squares problem (in the feature space) implying that the gradient search is on a smooth quadratic performance surface, resulting in reliable convergence without the hassles of local minima. Another interesting fact when the Gaussian kernel is utilized, is that the transformed data lies on the surface of a sphere, i.e. the transformed data is automatically normalized, which has a great advantage for the LMS algorithm. Indeed, this LMS becomes automatically normalized LMS.

6. REFERENCES

[1] B. Widrow, *Adaptive filters I: Fundamentals (TR 6764-6)*, Stanford Electronics Laboratories, Stanford, CA, 1966, Technical Report.

[2] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

[3] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.

[4] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, 1995.

[5] S. Smale and Y. Yao, "Online learning algorithms," *Foundations of computational mathematics*, vol. 6, pp. 145–170, 2006.

[6] J. Kivinen, A. Smola, and R. Williamson, "Online learning with kernels," *IEEE Transactions on Signal Processing*, vol. 52, pp. 2165–2176, 2004.

[7] B. Mitchinson and R. F. Harrison, "Adaptive kernel-based equalization for non-stationary digital communications channels," *International Journal of System Science*, vol. 34, pp. 693–703, 2003.

[8] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[9] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," *Journal of Machine Learning Research*, vol. 3, pp. 1–48, 2002.

[10] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, vol. 68, pp. 337–404, 1950.

[11] S. Saitoh, *Theory of Reproducing Kernels and its Applications*, Longman Scientific and Technical, U.K., 1988.

[12] A. H. Sayed, *Fundamentals of Adaptive Filtering*, John Wiley, New Jersey, 2003.

[13] Simon S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Upper Saddle River, NJ 07458, USA, fourth edition, 2002.

[14] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, 1986.

[15] A. N. Tikhonov and V. Y. Arsenin, *Solution of Ill-Posed Problems*, Wiley, New York, 1977.

[16] F. Takens, "Detecting strange attractors in turbulence," in *Lecture Notes in Mathematics*, vol. 898 of *Dynamical Systems and Turbulence*, pp. 366–381. Springer Verlag, 1981.