IMPROVING THE TRACKING ABILITY OF KRLS USING KERNEL SUBSPACE PURSUIT

Jad Kabbara and Ioannis N. Psaromiligkos

Department of Electrical and Computer Engineering, McGill University, Montreal, Canada
Email: jad.kabbara@mail.mcgill.ca; yannis@ece.mcgill.ca

ABSTRACT

We present a new Kernel Recursive Least Squares (KRLS) algorithm that is able to efficiently track time-varying systems. In order to alleviate the detrimental effect of a large dictionary size on the algorithm's tracking ability, we decouple the equality between dictionary size and weight vector size, an equality that has been encountered in all previous KRLS algorithms. In the proposed method, the maximum size of the weight vector is fixed and is independent from the dictionary size. We introduce the Kernel Subspace Pursuit algorithm which we use to choose a subset of the dictionary that tracks best the most recent received data samples. The selected dictionary elements are then used in the KRLS iterations. We show through simulations that our algorithm outperforms existing KRLS algorithms in tracking time-varying systems.

Index Terms— Online kernel methods, sparsification, least squares regression, subspace pursuit, tracking

1. INTRODUCTION

In recent years, kernel methods [1] have received major attention in areas such as machine learning and signal processing. They are powerful techniques based on non-linear transformations of data into a high-dimensional space where regression models of the transformed data are expected to be more accurate. "Kernelized" extensions of many well-known algorithms have been presented, including kernel SVMs [2], kernel PCA [3] and kernel Fisher discriminant analysis [4].

Engel et al. were the first to present in [5] a kernelized version of the popular online recursive least squares (RLS) adaptive algorithm, termed kernel RLS (KRLS). This algorithm is sequentially presented with input-output pairs and iteratively calculates a linear LS regressor in the high-dimensional feature space induced by the employed kernel. As is typical with kernel-based regression methods, the number of parameters that need to be calculated to obtain the LS solution is equal to the number of input vectors that grows without bound as the iterations progress. To address this issue, a "sparsification" procedure is proposed in [5] to be used in conjunction with KRLS, that forms the LS regressor using a carefully chosen subset, termed dictionary, of the input vectors. The dictionary is built iteratively: an incoming input is added to the dictionary only if its feature representation can't be written as an approximate linear combination of the feature representations of the input vectors already admitted to the dictionary. In surprise criterion (SC)-KRLS [6], Liu et al. approach the same issue from an information theoretic perspective by adopting as the admission criterion the "surprise" which measures how relevant the information that a new input-output pair can contribute to the already accumulated "knowledge" of the learning system is. Other algorithms such as sliding window (SW)-KRLS [7] and fixed budget (FB)-KRLS [8] adopt a more aggressive approach towards curbing the size of the dictionary by imposing a hard limit on it: When the bound is reached, the oldest input vector is discarded (SW-KRLS) or a "pruning" criterion is used to discard an element (FB-KRLS).

In standard KRLS problems, the LS solution is given in the form of a weighted linear combination of kernels evaluated at the input vectors. As a result, in all previous KRLS-type algorithms, the vector consisting of those weights is of equal size to that of the dictionary. This coupling of the parameter vector length to the dictionary size introduces an interesting trade-off. While a large dictionary is favorable as it would represent all the dynamics of the input-output relationship over time, it has a detrimental effect on the algorithm's ability to track changes in that relationship; it is well known that having to adjust a large number of weights significantly slows down adaptation. This trade-off highlights the need to decouple the size of the weight vector from the dictionary size and motivates the work presented in this paper.

In this work, we introduce a new KRLS-type algorithm designed for the specific purpose of tracking time-varying systems. Data samples are admitted to the dictionary only after passing the SC test, however, the maximum size M of the weight vector is fixed and independent from the dictionary size. As such, we gain the best of both worlds: on one hand, we are allowed to have a large dictionary that can accurately represent the dynamics of the input-output relationship; on the other, the weight vector retains a limited size which leads to an improved tracking ability. Our algorithm, when the dictionary size is less than M, is identical to SC-KRLS [6]. However, when the dictionary size exceeds M, we choose, from the dictionary elements, the M elements that will track best the N most recent received data samples. To this end, we extend the Subspace Pursuit (SP) algorithm [9] to non-linear regression problems and introduce the Kernel Subspace Pursuit (KSP).

In summary, the contributions of our work are two-fold: First, we present an online algorithm that is capable of tracking efficiently time-varying systems, by decoupling the dictionary size and weight vector size, the equality between which was encountered in all previous KRLS-type algorithms. Second, to the best of our knowledge, our work is the first to present a kernel version of the Subspace Pursuit algorithm and to use it in the context of tracking of time-varying systems.

The remainder of this paper is organized as follows: In Section 2, we formulate the problem considered in this work and present a brief overview of KRLS. In Section 3, we introduce the Kernel Subspace Pursuit algorithm that will be used as a building block of the proposed KRLS algorithm described in Section 4. The results of simulations studies are presented in Section 5. Finally, Section 6 concludes the paper.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

2. PROBLEM FORMULATION AND BACKGROUND

2.1. Problem Formulation

Consider an online prediction setup where we are sequentially given input-output pairs $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots\}$, with $\mathbf{x}_i \in \mathcal{X}$, for some input space $\mathcal{X}$, and $y_i \in \mathbb{R}$. Our goal is to predict y_{n+1} for a new input $\mathbf{x}_{n+1}$. This is done by identifying a function $f : \mathcal{X} \mapsto \mathbb{R}$ that minimizes at time n the sum of squared errors given by:

$$\mathcal{L} = \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2. \tag{1}$$

2.2. Kernel Recursive Least Squares

Kernels, first introduced in [10], are functions $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ that measure the similarity of two vectors in the input space $\mathcal{X}$. The class of positive definite kernels [1] is of particular interest because kernels belonging to this class can be written in the following form: $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$ where $\phi : \mathcal{X} \mapsto \mathcal{H}$ maps input vectors from $\mathcal{X}$ into a Hilbert space $\mathcal{H}$, referred to as the feature space, with the inner product $\langle \cdot, \cdot \rangle$: $\mathcal{H} \times \mathcal{H} \mapsto \mathbb{R}$. The substitution of a high-dimensional inner product $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ by $k(\mathbf{x}, \mathbf{x}')$ is commonly called the kernel trick. Learning techniques making use of it are referred to as kernel methods.

According to the Representer Theorem [11] any minimizer $f \in \mathcal{H}$ of $\mathcal{L}$ in (1) admits a representation of the form

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i) \tag{2}$$

with $\alpha_i \in \mathbb{R}$. The importance of this theorem is that it ensures that the minimizer of (1) can be expressed as a finite linear combination of kernels evaluated at the input vectors, and therefore, identifying $f \in \mathcal{H}$ reduces to identifying the coefficients α_i, $i = 1, \dots, n$. Indeed, substituting (2) in (1), $\mathcal{L}$ can be written as

$$\mathcal{L}(\boldsymbol{\alpha}) = \|\mathbf{K}_n \boldsymbol{\alpha} - \mathbf{y}_n\|^2 \tag{3}$$

where $\mathbf{y}_n = [y_1, \dots, y_n]^T$, $\mathbf{K}_n$ is the Gram matrix of the kernel k with entries $[\mathbf{K}_n]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_n]^T$ is a weight vector. Minimizing the loss function (3) gives the optimal vector $\boldsymbol{\alpha} = (\mathbf{K}_n)^\dagger \mathbf{y}_n$ where $(\cdot)^\dagger$ denotes the Moore-Penrose pseudo-inverse.

Engel et al. present in [5] an online kernel version of the RLS algorithm, namely Kernel RLS (KRLS), which is able to recursively solve (3). However, as the dimension of $\mathbf{K}_n$ and the length of $\boldsymbol{\alpha}$ are both n they grow without bound as the iterations progress; storing them in memory and performing computations on them soon becomes cumbersome. Moreover, the large length of the (generally dense) vector $\boldsymbol{\alpha}$ could lead to severe overfitting [5]. To address this issue, a "sparsification" procedure is commonly employed in conjunction with KRLS, that limits the size of K_n by forming it using a carefully chosen subset, termed dictionary, of the input vectors. More specifically, if we denote by $\mathcal{D}_n = [\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \dots, \tilde{\mathbf{x}}_{m_n}]$ the dictionary at time n containing m_n (out of n) input vectors, then the Gram matrix $\mathbf{K}_n$ has entries $[\mathbf{K}_n]_{i,j} = k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)$, $i, j = 1, \dots, m_n$. The dictionary is built iteratively; each new input-output pair is checked against an admission criterion. If the criterion is satisfied the new input vector is admitted to the dictionary, otherwise it is discarded. Of particular interest to our work are the Approximate Linear Dependency (ALD) Criterion and Surprise Criterion (SC) described next.

2.2.1. Approximate Linear Dependency Criterion

In ALD [5], an incoming data sample $\mathbf{x}_n$ is added to the dictionary only if its feature representation $\phi(\mathbf{x}_n)$ cannot be written as an approximate linear combination of the feature representations of the vectors $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \dots, \tilde{\mathbf{x}}_{m_{n-1}}$ previously admitted to the dictionary, i.e., if

$$\delta_n \triangleq \min_{a_1, \dots, a_{m_{n-1}}} \left\| \sum_{i=1}^{m_{n-1}} a_i \phi(\tilde{\mathbf{x}}_i) - \phi(\mathbf{x}_n) \right\|^2 > \nu \tag{4}$$

where ν is an accuracy parameter.

2.2.2. Surprise Criterion

In [6], Liu et al. propose to use as an admission criterion the surprise that quantifies how much information a new input-output pair contains relatively to the already accumulated knowledge of the learning system. For the nth input-output pair $\{\mathbf{x}_n, y_n\}$, the surprise is defined as $\mathcal{S}_n = -\ln p(\mathbf{x}_n, y_n | \mathcal{D}_{n-1})$ where $p(\mathbf{x}_n, y_n | \mathcal{D}_{n-1})$ is the posterior probability distribution of $\{\mathbf{x}_n, y_n\}$ given the dictionary $\mathcal{D}_{n-1}$. If the new input-output pair results in a small value for the surprise, it is deemed redundant, and so, it is not added to the dictionary. For KRLS, the surprise criterion is given by: $\mathcal{S}_n = \frac{1}{2} \ln \delta_n + \frac{(y_n - \mathbf{k}_n^T \boldsymbol{\alpha}_n)^2}{2\delta_n}$ where δ_n is the ALD measure, $\boldsymbol{\alpha}_n$ is the weight vector that solves (3) and $[\mathbf{k}_{n-1}]_i = k(\tilde{\mathbf{x}}_i, \mathbf{x}_n)$ where $i = 1, \dots, m_{n-1}$. The details of the derivation of $\mathcal{S}_n$ for KRLS can be found in [6].

3. KERNEL SUBSPACE PURSUIT

Subspace Pursuit (SP) was first introduced by Dai et al. in [9] in the context of compressive sensing (see also [12] for a similar method). It was originally proposed as an iterative method for the reconstruction of an unknown sparse signal from a set of linear measurements. In essence, SP is a low complexity method to obtain LS solutions with a pre-specified sparsity level.

In this section, we extend SP to non-linear regression problems and introduce Kernel SP (KSP) in the spirit of Kernel Matching Pursuit [13] and Kernel Basis Pursuit [14].

Consider again the setup introduced in Section 2.1. Let $\mathcal{D}_n = [\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_{m_n}]$ be the dictionary containing a subset of input vectors up to time n. As in KRLS, we are interested in obtaining a minimizer of (1) of the form $f(\mathbf{x}) = \sum_{i=1}^{m_n} \alpha_i k(\mathbf{x}, \tilde{\mathbf{x}}_i)$ but we now add the constraint that the vector of coefficients $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_{m_n}]^T$ is M-sparse, i.e., it contains exactly M non-zero coefficients.

Let $\mathbf{G}_n$ be the $m_n \times n$ Gram matrix obtained by evaluating the functions $k(\cdot, \tilde{x}_i)$, $i = 1, \dots, m_n$ at the input vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$, i.e.,

$$[\mathbf{G}_n]_{i,j} = k(\tilde{\mathbf{x}}_i, \mathbf{x}_j). \tag{5}$$

Using $\mathbf{G}_n$, we can formulate the constrained minimization of (1) considered here as:

$$\min_{\boldsymbol{\alpha}} \|\mathbf{G}_n \boldsymbol{\alpha} - \mathbf{y}_n\|^2 \text{ s.t. } \boldsymbol{\alpha} \text{ is } M - \text{sparse}. \tag{6}$$

Denote by $\mathbf{G}_T$, with T being a set of column indices, the matrix consisting of the columns of $\mathbf{G}_n$ with indices in T (to simplify presentation, for the rest of this section we will drop the subscript n of $\mathbf{G}_n$). Then (6) can be rewritten as

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^M, T} \|\mathbf{G}_T \boldsymbol{\alpha} - \mathbf{y}_n\|^2 \text{ s.t. cardinality of } T \text{ is } M. \tag{7}$$

KSP solves (7) by, initially selecting the M columns of $\mathbf{G}$ that exhibit the highest correlation with the vector $\mathbf{y}_n$, and then iteratively refining this set by discarding and adding columns. The refinement process is carried out by retaining "informative" columns of $\mathbf{G}$, i.e., the columns of $\mathbf{G}$ that better approximate $\mathbf{y}_n$, and discarding the less informative ones. In particular, at a given iteration, we begin by computing the correlation vector between $\mathbf{G}$ and the approximation error of $\mathbf{y}_n$ from the previous iteration (called the residue), and identifying the columns of $\mathbf{G}$ that correspond to that vector's M elements with the largest magnitudes. The new set of M columns is then merged with the set of columns of the previous iteration's approximation. We then project $\mathbf{y}_n$ onto the span of the columns of $\mathbf{G}$ in the merged set and we obtain a new set of indiced, say $\tilde{T}$, by identifying the columns corresponding to the M largest magnitude projection coefficients. Finally, a new LS estimate of $\mathbf{y}_n$, $\hat{\mathbf{y}}_n$, is acquired by projecting the vector $\mathbf{y}_n$ onto the span of the columns of $\mathbf{G}_{\tilde{T}}$:

$$\hat{\mathbf{y}}_n = \text{proj}(\mathbf{y}_n, \mathbf{G}_{\tilde{T}}) := \mathbf{G}_{\tilde{T}}\mathbf{G}_{\tilde{T}}^{\dagger}\mathbf{y}_n. \quad (8)$$

and a new residue is obtained as $\mathbf{y}_n - \hat{\mathbf{y}}_n$. Algorithm 1 summarizes KSP.

Algorithm 1 Kernel Subspace Pursuit Algorithm

Input: M, $\mathbf{y}_n$, $\mathcal{D}$.
Initialization:
Compute $\mathbf{G}$ using (5).
$T_0 = \{$Indices of the M largest magnitude entries in the vector $\mathbf{G}\mathbf{y}\}$
$\mathbf{r}_0 = \mathbf{y}_n - \text{proj}(\mathbf{y}_n, \mathbf{G}_{T_0})$
Loop: At iteration ℓ ($1 \leq \ell \leq 5$), execute the following:
$\tilde{T}_\ell = T_{\ell-1} \cup \{$Indices of the M largest magnitude entries in the vector $\mathbf{G}^T\mathbf{r}_{\ell-1}\}$
$T_\ell = \{$Indices of the M largest magnitude entries in $\mathbf{G}_{\tilde{T}_\ell}^{\dagger}\mathbf{y}_n\}$
$\mathbf{r}_\ell = \mathbf{y}_n - \text{proj}(\mathbf{y}_n, \mathbf{G}_{T_\ell})$
If $||\mathbf{r}_\ell||_2 > ||\mathbf{r}_{\ell-1}||_2$, let $T_\ell = T_{\ell-1}$ and exit the loop.
Output: T_ℓ

4. PROPOSED METHOD

In this section, we present the proposed Subspace Pursuit (SP)-KRLS-type algorithm tailored to tracking time-varying systems.

In SP-KRLS, the maximum size M of the weight vector $\boldsymbol{\alpha}$ is fixed and independent from the size of the dictionary. If, at the nth iteration, the input sample is admitted to the dictionary $\mathcal{D}_n$ and the dictionary size exceeds M, we use KSP to select M elements in order to form the LS regressor. More specifically, we consider the αM ($\alpha > 1$) most recent entries in the dictionary $\mathcal{D}_n = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{m_n}]$ (we assume that the elements of the dictionary are sorted by the time they were admitted to the dictionary - oldest first), from which we want to select the M elements that will lead to the best approximation of the most recent N received target values. In this context, the KSP gram matrix $\mathbf{G}_n$ is obtained by evaluating $k(\cdot, \tilde{\mathbf{x}}_i)$, $i = m_n - \alpha M + 1, \ldots, m_n$ at the N most recent inputs $\mathbf{x}_{n-N+1}, \ldots, \mathbf{x}_n$. The vector $\mathbf{y}_n$ consists of the most recent N target values, i.e., $\mathbf{y}_n = [y_{n-N+1}, \ldots, y_n]^T$. It is important to emphasize that KSP need not be run at every iteration (after the dictionary size had exceeded M). Indeed, KSP is only "triggered" when a new element has been added to the dictionary. We view the admission of a new element in the dictionary as an indication of a possible change in the input-ouput relationship being tracked, thus, requiring an update to the M-sized subset of the dictionary used in the KRLS computations.

For constructing our sparse dictionary, we adopt the surprise criterion introduced in Section 2.2.2. In our algorithm, when the system receives a new pair $\{\mathbf{x}_n, y_n\}$, it is checked against the SC test [6]. We are presented with two possible scenarios:

1. If it does not pass the SC test, the input vector is not added to the dictionary and the weight vector is updated appropriately via the KRLS recursions [5].
2. If the input vector is admitted to the dictionary, we are further presented with two cases:
   (a) If the size of the dictionary is less than or equal to M, the weight vector is updated appropriately via the KRLS recursions.
   (b) If the size of the dictionary is larger than M, we use KSP (Algorithm 1) to identify the M input vectors that will be used by the KRLS.

SP-KRLS is summarized in Algorithm 2.

5. SIMULATIONS

In this section, we experimentally test SP-KRLS and compare its tracking performance to that of the following algorithms: ALD-KRLS [5], SW-KRLS [7] and FB-KRLS [8]. We also test the performance of SP-KRLS in predicting the highly chaotic Mackey-Glass time series.

5.1. Tracking of a Time-Varying Wiener System

In this experiment, we consider a system composed of a time-varying linear filter followed by a static non-linearity.

The linear filter varies in time as follows along 3200 iterations: During the first 1500 iterations, its transfer function is given by: $H_1(z) = 1 - 0.37z^{-1} - 0.48z^{-2} + 0.81z^{-3}$. On iteration 1501, the filter is abruptly changed to $H_2(z) = 1 - 0.83z^{-1} + 0.67z^{-2} + 0.72z^{-3}$ which remains constant for 1200 iterations. At iteration 2701, the filter starts linearly changing, throughout 500 iterations, from $H_2(z)$ to $H_3(z) = 1 - 0.5z^{-1} - 0.25z^{-2} + 0.4z^{-3}$. The output of the linear filter is then passed through the static non-linear function $f(x) = \tanh(x)$. The resulting signal is finally corrupted with 20 dB of white Gaussian noise. 400 sample points are used as a test set (different in each phase of the scenario according to the linear filter) and a time embedding of 4 is chosen, i.e., $\mathbf{x}_n = [x_n, x_{n-1}, x_{n-2}, x_{n-3}]^T$. The input signal is normally distributed with mean 0 and variance 0.5.

For SP-KRLS, we set the thresholds for learnable data to $T_1 = 3$ and $T_2 = -3$. For SP-KRLS, FB-KRLS and SW-KRLS, we set $M = 200$. In addition, for SP-KRLS $N = 10$. For FB-KRLS, the step-size parameter is set to $\mu = 0.01$. The regularization parameter for SP-KRLS and FB-KRLS is set to $\lambda = 0.001$. The accuracy parameter for ALD-KRLS is set to $\nu = 0.001$. For all algorithms, a Gaussian kernel is used with a width $\sigma = 0.8$.

The results, averaged out over 50 Monte-Carlo simulations, are shown in Figure 1. The performance of ALD-KRLS is the worst as it is not designed to be a tracking algorithm. SP-KRLS outperforms both SW-KRLS and FB-KRLS in tracking the system: Following changes in the linear filter (indicated by the vertical lines in the plot), the MSE curve of SP-KRLS is the fastest to change, thus capturing the fastest the change in the system. Moreover, following changes in the system, the MSE of SP-KRLS converges in each phase to the smallest value among all three algorithms.

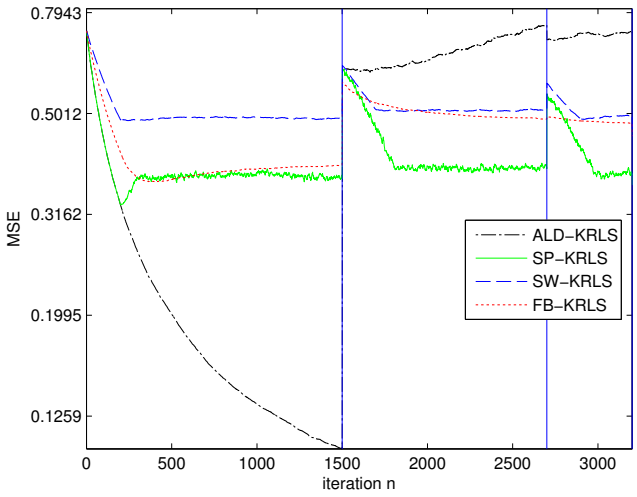

Fig. 1. Performance of SP-KRLS vs other KRLS algorithms on a time-varying Wiener system.

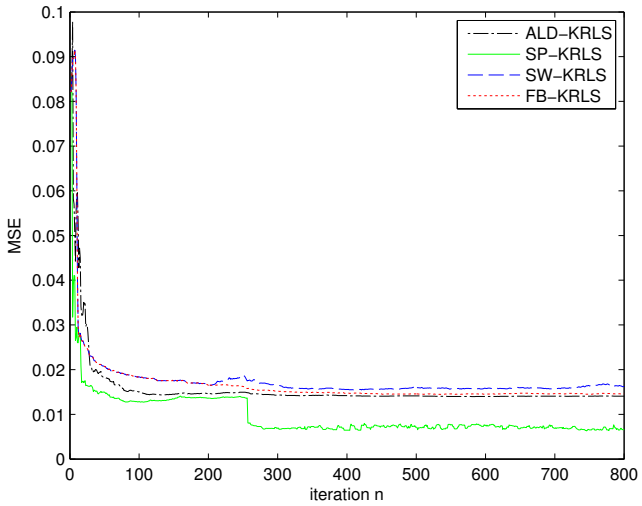

Fig. 2. Performance of SP-KRLS vs other KRLS algorithms in predicting a Mackey-Glass time series.

Algorithm 2 SP-KRLS

Parameters: T_1, T_2, M, N, α.
Initialize: $\mathbf{k}_1 = [k_{11}]$, $\mathbf{k}_1^{-1} = [1/k_{11}]$, $\boldsymbol{\alpha}_n = (y_1/k_{11})$, $\mathbf{P}_1 = [1]$, $m = 1$.
for $n = 2, 3, \ldots$ **do**
  Get new sample: $(\mathbf{x}_n, y_n)$
  $\mathbf{k}_{n-1}$ with $[\mathbf{k}_{n-1}]_i = k(\mathbf{x}_i, \mathbf{x}_n), i = 1, \ldots, m$.
  $k_{nn} = k(\mathbf{x}_n, \mathbf{x}_n)$
  SC Test:
  $\boldsymbol{a}_n = \mathbf{K}_{n-1}^{-1}\mathbf{k}_{n-1}$
  $\delta_n = k_{nn} - \mathbf{k}_{n-1}^T\boldsymbol{a}_n$
  $\mathcal{S}_n = \frac{1}{2}\ln\delta_n + \frac{(y_n - \mathbf{k}_{n-1}^T\boldsymbol{\alpha}_{n-1})^2}{2\delta_n}$
  if $(T_1 \geq \mathcal{S}_n \geq T_2)$ **then** {*Add* $\mathbf{x}_n$ *to dictionary*}
    if $m \leq M$ **then** {*SC-KRLS approach*}
      $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\mathbf{x}_n\}$, $\mathcal{D}_{y\,n} = \mathcal{D}_{y\,n-1} \cup \{\mathbf{y}_n\}$
      $\mathbf{K}_n^{-1} = \frac{1}{\delta_n}\begin{bmatrix}\delta_n\mathbf{K}_{n-1}^{-1} + \boldsymbol{a}_n\boldsymbol{a}_n^T & -\boldsymbol{a}_n \\ -\boldsymbol{a}_n^T & 1\end{bmatrix}$.
      $\mathbf{P}_n = \begin{bmatrix}\mathbf{P}_{n-1} & \mathbf{0} \\ \mathbf{0}^T & 1\end{bmatrix}$
      $\boldsymbol{\alpha}_n = \begin{bmatrix}\boldsymbol{\alpha}_{n-1} - \frac{\boldsymbol{a}_n}{\delta_n}\left(y_n - \mathbf{k}_{n-1}^T\boldsymbol{\alpha}_{n-1}\right) \\ \frac{1}{\delta_n}\left(y_n - \mathbf{k}_{n-1}^T\boldsymbol{\alpha}_{n-1}\right)\end{bmatrix}$
      $m = m + 1$
    else {*KSP approach*}
      $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\mathbf{x}_n\}$, $\mathcal{D}_{y\,n} = \mathcal{D}_{y\,n-1} \cup \{\mathbf{y}_n\}$
      $m = m + 1$
      Find the set, T_ℓ, of indices of the best M elements in $\mathcal{D}_n$ using Algorithm 1.
      $sub\mathcal{D}_n = \{\mathcal{D}_n$ entries with indices given by $T_\ell\}$
      $sub\mathcal{D}_{y\,n} = \{\mathcal{D}_{y\,n}$ entries with indices given by $T_\ell\}$
      Find $\mathbf{K}_n^{-1}$ where $[\mathbf{K}_n]_{i,j} = k(sub\mathcal{D}_{n\,i}, sub\mathcal{D}_{n\,j}), i, j = 1, \ldots, m$
      $\boldsymbol{\alpha}_n = \mathbf{K}_n^{-1} sub\mathcal{D}_{y\,n}$
    end if
  else {*SC-KRLS approach: Case "dictionary unchanged"*}
    $\mathcal{D}_n = \mathcal{D}_{n-1}$, $\mathcal{D}_{y\,n} = \mathcal{D}_{y\,n-1}$
    $\mathbf{q}_n = \frac{\mathbf{P}_{n-1}\boldsymbol{a}_n}{1 + \boldsymbol{a}_n^T\mathbf{P}_{n-1}\boldsymbol{a}_n}$
    $\mathbf{P}_n = \mathbf{P}_{n-1} - \mathbf{q}_n\boldsymbol{a}_n^T\mathbf{P}_{n-1}$
    $\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_{n-1} + \mathbf{K}_n^{-1}\mathbf{q}_n\left(y_n - \mathbf{k}_{n-1}^T\boldsymbol{\alpha}_{n-1}\right)$
  end if
end for
Output: $\mathcal{D}_t$, $\boldsymbol{\alpha}_n$

5.2. Prediction of the Mackey-Glass Time Series

In this experiment, we perform one-step prediction on the highly chaotic Mackey-Glass time series. The algorithm is trained online on 800 sample points, and the MSE performance is calculated at each iteration on a test set of 200 sample points. The Gaussian kernel used in this experiment has a width $\sigma = 8$. For SP-KRLS, we set the thresholds for learnable data to $T_1 = 200$ and $T_2 = -4$. All other parameters are the same as in the first experiment. In Figure 2, we can clearly see that our algorithm outperforms the other algorithm in predicting the Mackey-Glass time series as the MSE curve for SP-KRLS converges to a lower steady-state value.

6. CONCLUSION

We presented a new KRLS algorithm that is able to efficiently track time-varying systems. To overcome the trade-off between the algorithm's tracking ability and the dictionary size which, in all previous KRLS algorithms, is equal to the weight vector size, SP-KRLS decouples the dictionary size from the weight vector size, which is fixed independently from the dictionary size. We introduced Kernel Subspace Pursuit by which we are able to choose, at each iteration, from the most recent $2M$ dictionary elements, the M elements that are the most highly correlated with the most recent N received data samples. Simulations showed that the proposed algorithm outperformed other well-known KRLS algorithms in tracking time-varying systems.

7. REFERENCES

[1] A. Smola and B. Schölkopf, *Learning with kernels*, MIT Press, Cambridge, MA, 1 edition, 2002.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*, vol. 1, Springer New York, 2001.

[3] B. Schölkopf, A. Smola, and K. Müller, "Kernel principal component analysis," in *Artificial Neural Networks, International Conference on*, pp. 583–588. Springer, 1997.

[4] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Muller, "Fisher discriminant analysis with kernels," in *Neural Networks for Signal Processing, IEEE Signal Processing Society Workshop on*, 1999, pp. 41–48.

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

[6] W. Liu, I. Park, and J.C. Príncipe, "An information theoretic approach of designing sparse kernel adaptive filters," *Neural Networks, IEEE Transactions on*, vol. 20, no. 12, pp. 1950–1961, 2009.

[7] S. Van Vaerenbergh, J. Via, and I. Santamana, "A sliding-window kernel rls algorithm and its application to nonlinear channel identification," in *Acoustics Speech and Signal Processing (ICASSP), IEEE International Conference on*, may 2006, vol. 5, p. V.

[8] S. Van Vaerenbergh, I. Santamaría, W. Liu, and J.C. Príncipe, "Fixed-budget kernel recursive least-squares," in *Acoustics Speech and Signal Processing (ICASSP), IEEE International Conference on*, march 2010, pp. 1882–1885.

[9] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *Information Theory, IEEE Transactions on*, vol. 55, no. 5, pp. 2230–2249, 2009.

[10] M. Aizerman, E. Braverman, and L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning," 1964.

[11] B. Schölkopf, R. Herbrich, and A. Smola, "A generalized representer theorem," in *Computational learning theory*. Springer, 2001, pp. 416–426.

[12] D. Needell and J. Tropp, "Cosamp: iterative signal recovery from incomplete and inaccurate samples," *Communications of the ACM*, vol. 53, no. 12, pp. 93–100, 2010.

[13] P. Vincent and Y. Bengio, "Kernel matching pursuit," *J. Machine Learning*, vol. 48, no. 1-3, pp. 165–187, 2002.

[14] V. Guigue, A. Rakotomamonjy, and S. Canu, "Kernel basis pursuit," in *Machine Learning, European Conference on*. October 2005, pp. 146–157, Springer.