Knowledge-Based Systems 59 (2014) 21–32

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

# Online learning with kernel regularized least mean square algorithms

Haijin Fan *, Qing Song, Sumit Bam Shrestha

*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore*

ARTICLE INFO

*Article history:*
Received 30 March 2013
Received in revised form 31 January 2014
Accepted 1 February 2014
Available online 8 February 2014

*Keywords:*
Kernel method
Dictionary
Cumulative coherence
Diagonally dominant
Weight convergence

ABSTRACT

In this paper, we propose a novel type of kernel least mean square algorithm with regularized structural risk for online learning. In order to curb the continuous growing of kernel functions, a new dictionary selection method based on the cumulative coherence measure is applied to perform the sparsification procedure, which can obtain a dictionary with diagonally dominant Gram matrix under certain conditions. On the updating of the kernel weight, the linear least mean square algorithm is generalized into the reproducing kernel Hilbert space (RKHS) with minimized updating structural risk and it results in a kernel regularized least mean square (KRLMS) algorithm. A simplified version of the KRLMS algorithm is also presented by applying only partial updating information to train the algorithm at each iteration, which reduces the computational complexity. Theoretical analysis of their convergence issues is examined and variable learning rates are adopted in the training process which can guarantee the weight convergence of the algorithm in terms of a bounded measurement error. Several experiments are carried out to prove the effectiveness of the proposed algorithm for online learning compared to some existing kernel algorithms.

## 1. Introduction

Kernel methods have been widely applied in nonlinear signal processing applications in the reproducing kernel Hilbert space (RKHS). The basic idea is the use of Mercer kernels which nonlinearly transform the original input feature into a high or even infinite dimensional RKHS. In such a transformed feature space, the solution is linear in the RKHS [1]. The popular "kernel trick" makes the inner product between two high-dimensional transformed features able to be obtained easily and it makes the kernel methods more efficient. The well developed kernel methods include support vector machines (SVMs) [2–4], support vector regression (SVR) [5,6], Gaussian process theory [7], different kinds of kernel least mean square algorithms [7,8]. These kernel methods have been popularly used for batch learning or online learning in various applications. In a batch learning model, the computational complexity and memory required usually grow superlinearly with the number of training samples. However, the high complexity of these batch learning methods makes them unsuitable for online learning.

In kernel online learning context, to reduce the computational complexity, the sparsification method was designed to prevent the size of kernel functions being too large. In the last decade, different sparsification methods have been proposed. They aimed to select a compact dictionary with finite size using different criteria including the novelty criterion [9,10], the approximate linear dependency (ALD) criterion [11], the coherence-based criterion [12], the information theoretic criterion [10,13], the significance-based criterion [14]. By relying on these sparsification methods, many kernel algorithms were proposed for online learning. These algorithms include the kernel normalized least mean square (KNLMS) algorithm [12], the kernel affine projection (KAP) algorithm [12,15], the kernel recursive least square (KRLS) algorithm [11], the quantized kernel least mean square (QKLMS) algorithm [16], where linear least square algorithms were generalized into the RKHS and the solutions became the linear combinations of selected kernel functions. For sparse multi-kernel learning, a novel method integrating feature selection and multi-kernel learning were developed for sparse coding [17]. In online learning settings, the training samples are available one by one and at each training iteration, only one training sample is present to update both the dictionary and kernel coefficients of the algorithm. Underlying this learning fashion, the existing sparsification rules update the dictionary by deciding whether adding a new kernel function characterized by the new training sample or not while remaining the old dictionary members unchanged. They are kind of *constructive* methods without applying the *pruning* method, where the old dictionary member may be deleted [18].

* Corresponding author. Tel.: +65 8552 5016.
*E-mail address:* hfan1@e.ntu.edu.sg (H. Fan).

http://dx.doi.org/10.1016/j.knosys.2014.02.005
0950-7051/© 2014 Elsevier B.V. All rights reserved.

Previously, we developed sparsification methods for online kernel learning such as the significance-based method, which was incorporated into the famous recursive least square algorithm [14], and the mutual information concept based sparsification rule [13]. Both are type of constructive sparsification methods. In this paper, a novel sparsification rule is proposed for dictionary selection which combines the *constructive* and the *pruning* methods for kernel online learning. A cumulative coherence concept is adopted to measure the sparsity of a dictionary as an extension of the coherence concept, which can provide a deeper measure of the dictionary [19,20]. In sparse approximation problems, a dictionary with small enough cumulative coherence guarantees that the desired signal can be constructed by the sparse signal exactly [21,22]. In kernel algorithms, if the cumulative coherence of a dictionary is less than one, it is obvious that its Gram matrix is diagonally dominant and such a matrix is nonsingular and invertible [23]. The proposed sparsification method is based on the cumulative coherence measure. It is a *constructive* and *pruning* combined method with two stages. First, it examines whether the dictionary can be expanded with a new kernel function; otherwise, it decides whether to replace one of the old dictionary member with the new kernel function. In this way, the old dictionary member can be replaced if it cannot represent the new data well. It is shown that the selected dictionary satisfies several important properties in sparse approximation problems.

Based on the selected dictionary, the least mean square (LMS) algorithm is extended into the RKHS and a kernel regularized least mean square (KRLMS) algorithm is derived. The novel contribution is that the kernel algorithm is updated under a regularization that the change of its structure risk in RKHS is minimized at each training iteration. Motivated by the partial update linear algorithm (see [24,25] and therein), a method to reduce the complexity in algorithm updating with only part of its parameters updated at each step, we propose a complexity reduced partial information updated KRLMS (PIU-KRLMS) algorithm. It uses only part of the updating information to tune itself at each iteration. Furthermore, to speed up the convergence of the algorithm, a novel method based on the bound of the measurement error is applied to determine the variable learning rates for both the KRLMS and the PIU-KRLMS algorithms, which can make a tradeoff between the algorithm's convergence speed and accuracy.

The organization of the rest of this paper is as follows. In Section 2, some fundamental ideas of kernel methods are discussed. In Section 3, the proposed dictionary selection method is presented and some of its properties are exploited. In Section 4, the KRLMS and the PIU-KRLMS algorithms are proposed followed by the theoretical analysis of their weight convergence. In Section 5, experimental results of several examples are presented and finally conclusion is given in Section 6.

Some notations used throughout the paper are as follows:

| | |
|---|---|
| $\boldsymbol{x}$, $\boldsymbol{x}(i)$ | column vector, the $i$(th) element of $\boldsymbol{x}$, |
| $\boldsymbol{X}$, $\boldsymbol{X}(i,j)$ | matrix, the entry of $\boldsymbol{X}$ in the $i$(th) row and $j$(th) column, |
| $\boldsymbol{X}(:,i)$, $\boldsymbol{X}^{-ii}$ | the $i$(th) column of $\boldsymbol{X}$, the sub-matrix of $\boldsymbol{X}$ with the $i$(th) row and column deleted. |

## 2. Fundamentals of kernel methods

By using a nonlinear mapping function, which maps the low dimensional feature space into the high dimensional RKHS, linear models can be found in the RKHS for many applications. Suppose that $\mathcal{H}$ is a Hilbert Space and $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ denotes the inner product in the Hilbert Space. A mapping function $\varphi(\cdot)$ will transfer the input feature space $\mathcal{U}$ to a high dimensional feature space in $\mathcal{H}$. The inner product between two input features $\boldsymbol{u}_i$ on the point $\boldsymbol{u}_j$ becomes [26]:

$$\langle\varphi(\boldsymbol{u}_i),\varphi(\boldsymbol{u}_j)\rangle_{\mathcal{H}} = \langle\kappa(\boldsymbol{u}_i,\cdot),\kappa(\boldsymbol{u}_j,\cdot)\rangle_{\mathcal{H}} = \kappa(\boldsymbol{u}_i,\boldsymbol{u}_j), \tag{1}$$

where $k(\cdot,\cdot)$ is a positive definite and symmetric kernel function. The inner product of two feature vectors in the high dimensional feature space can be easily computed by (1) without knowing the exact function of $\varphi(\cdot)$, which is usually called the *kernel trick*. The commonly used kernels include the Gaussian kernel $\kappa(\boldsymbol{u}_i,\boldsymbol{u}_j) = \exp\left(-\|\boldsymbol{u}_i-\boldsymbol{u}_j\|^2/2\sigma^2\right)$, the Laplacian kernel $\kappa(\boldsymbol{u}_i,\boldsymbol{u}_j) = \exp(-\|\boldsymbol{u}_i-\boldsymbol{u}_j\|/\sigma)$ with $\sigma$ being the kernel width and the polynomial kernel $\kappa(\boldsymbol{u}_i,\boldsymbol{u}_j) = \left(\eta+\boldsymbol{u}_i^{\top}\boldsymbol{u}_j\right)^q$, with $\eta \geqslant 0$ and $q \in \mathbb{N}$.

### 2.1. Kernel methods

Given the feature input-desired output sequence $\{\boldsymbol{u}_j, d_j\}_{j=1}^t$, the problem is to find a function $f(\cdot)$ to reconstruct the corresponding output $f_t(\boldsymbol{u}_t) = \langle f_t(\cdot), k(\cdot,\boldsymbol{u}_t)\rangle_{\mathcal{H}}$. By virtue of the representer theorem [26], the function $f_t(\cdot)$ can be expressed as a linear form in the RKHS

$$f_t(\cdot) = \boldsymbol{\omega}^T(t)\varphi(\cdot), \tag{2}$$

where $\boldsymbol{\omega}(t)$ is the weight coefficient and it can be expressed as a linear combination of the obtained feature vectors in the RKHS till the $(t)$th training iteration

$$\boldsymbol{\omega}(t) = \sum_{j=1}^{t}\alpha_j\varphi(\boldsymbol{u}_j). \tag{3}$$

Using the kernel trick, we have

$$f_t(\cdot) = \sum_{j=1}^{t}\alpha_j\kappa(\cdot,\boldsymbol{u}_j), \tag{4}$$

where $\kappa(\cdot,\boldsymbol{u}_j)$ is a kernel function with its center being the feature input vector $\boldsymbol{u}_j$ and $\alpha_j$ is the kernel coefficient. As a result, the function can be estimated implicitly by the feature input vectors with a Mercer kernel function. The problem of model (4) is that the number of kernel functions grows continuously, with its number equal to the number of training samples. If sparsification methods [10–12] are applied to reduce the kernel function number and suppose that a sparse dictionary $\mathcal{D}_t = \{\kappa(\boldsymbol{c}_1,\cdot),\ldots,\kappa(\boldsymbol{c}_{m_t},\cdot)\}$ with $m_t$ members is obtained and $\{\boldsymbol{c}_j\}_{j=1}^{m_t}$ is selected from the feature input vectors of training samples $\{\boldsymbol{u}_i\}_{i=1}^t$, the estimated function becomes

$$f_t(\cdot) = \sum_{j=1}^{m_t}\alpha_j\kappa(\cdot,\boldsymbol{c}_j) = \boldsymbol{K}_t\boldsymbol{\alpha}, \tag{5}$$

where the number of kernel functions is limited to the size of the dictionary, with $\boldsymbol{K}_t = [\kappa(\boldsymbol{c}_1,\cdot),\ldots,\kappa(\boldsymbol{c}_{m_t},\cdot)]$ representing the kernel functions and $\boldsymbol{\alpha} = [\alpha_1,\ldots,\alpha_{m_t}]^T$ being the kernel weight. The Gram matrix of the dictionary can be expressed as $\boldsymbol{G}_t = (\boldsymbol{K}_t)^T\boldsymbol{K}_t$ in RKHS and we have $\boldsymbol{G}_t(i,j) = \boldsymbol{K}_t^T(i)\boldsymbol{K}_t(j) = \langle\boldsymbol{K}_t(i),\boldsymbol{K}_t(j)\rangle_{\mathcal{H}}$.

## 3. A new dictionary selection method

As shown in the model (4), the kernel function number in kernel methods without sparse representation will increase with the number of training samples and this makes its computational complexity very high and a large memory is required for the information storage. However, with a sparsification procedure, its sparse model (5) is much less complicated in both computational or structure complexity. For most of the kernel online learning algorithms, one of the key issue is how to select a representable dictionary $\mathcal{D}_t$ to characterize the kernel model.