Signal Processing 93 (2013) 1586–1596

Contents lists available at SciVerse ScienceDirect

Signal Processing

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

KIMEL: A kernel incremental metalearning algorithm

Chunguang Li*, Yan Ye, Qiuyuan Miao, Hui-Liang Shen

Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, People's Republic of China

ARTICLE INFO

Article history:
Received 17 January 2012
Received in revised form
30 June 2012
Accepted 11 July 2012
Available online 25 July 2012

Keywords:
Metalearning
Kernel method
Reproducing kernel Hilbert space
Least-mean-square
Image quality assessment

ABSTRACT

The Kernel method is a powerful tool for extending an algorithm from linear to nonlinear case. Metalearning algorithm learns the base learning algorithm, thus to improve performance of the learning system. Usually, metalearning algorithms exhibit faster convergence rate and lower Mean-Square Error (MSE) than the corresponding base learning algorithms. In this paper, we present a kernelized metalearning algorithm, named KIMEL, which is a metalearning algorithm in the Reproducing Kernel Hilbert Space (RKHS). The convergence analyses of the KIMEL algorithm are performed in detail. To demonstrate the effectiveness and advantage of the proposed algorithm, we firstly apply the algorithm to a simple example of nonlinear channel equalization. Then we focus on a more practical application in blind Image Quality Assessment (IQA). Experimental results show that the KIMEL algorithm has superior performance.

1. Introduction

Metalearning means learning of the base learning system, hence to improve the performance of the base learning system. The Delta-Bar-Delta (DBD) algorithm [1] is one of the most well-known metalearning algorithms, which consists of a weight update and a learning rate update both based on the delta rule. In [2], as an extension of the DBD algorithm, the Incremental Delta-Bar-Delta (IDBD) algorithm was presented, which is applicable to incremental tasks, that is, supervised learning tasks in which the examples are processed one-by-one and then discarded. In the IDBD, both the weight update rule and the learning rate update rule are derived by gradient descent. The base learning algorithm in the IDBD is the well-known Least-Mean-Square (LMS) algorithm. It was shown that, compared with the LMS, the IDBD exhibits faster convergence rate and lower Mean-Square Error (MSE). Besides, the IDBD is biologically plausible, and can play the role of a nervous metaplasticity model [3].

However, both the LMS and the IDBD algorithms are developed for the simple, linear learning systems. If the mapping between output and input is highly nonlinear, both algorithms get poor performance. The kernel method is a powerful tool to overcome this limitation. The kernel method firstly transforms the input data into a high-dimensional feature space via a reproducing kernel such that the inner product operation in the feature space can be computed efficiently through the kernel evaluations. Then, appropriate linear methods are applied to the transformed data. The Kernelized version of the Least-Mean-Square algorithm (KLMS) was proposed in [4]. Kernel adaptive filtering algorithms were systematically introduced in [5]. It is a natural question whether we can combine the kernel method and the IDBD algorithm to present a kernelized version of the IDBD or a similar algorithm, so that it is applicable to nonlinear learning systems. In this paper, inspired by the idea of the IDBD, we derive a Kernel Incremental MEtaLearning (KIMEL) algorithm. We perform convergence analysis of the algorithm in detail. Note that the KIMEL algorithm is not simply the KLMS with a variable step-size. Since the learning-rate update depends on the inputs of the system, it is also computed in RKHS.

* Corresponding author. Tel.: +86 571 87951577.
E-mail address: cgli@zju.edu.cn (C. Li).

0165-1684/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sigpro.2012.07.023

Since the IDBD algorithm can be seen as a Variable Step-Size (VSS) LMS algorithm, it is appropriate to compare the performance of the KIMEL algorithm with that of other kernelized VSS-LMS algorithms. Many researchers have proposed variable step-size algorithms based on the standard LMS weight update recursion, such as [13–16]. The robust VSS-LMS algorithm proposed in [14] is one of the well-known and successful VSS-LMS algorithms. In this paper, we compare the performance of the KIMEL with that of the Kernelized version of this VSS-LMS (KVSS-LMS). Note that, although the KVSS-LMS is actually also derived by us in this paper, we treat it as a competing algorithm.

Automatical assessment of perceptual image quality is critical for amounts of image processing applications. It is becoming increasingly important. The KIMEL algorithm proposed in this paper is excellent in solving nonlinear fitting or regression problems, which can be used to construct good approximation of the function relationship between the input and output data. The blind Image Quality Assessment (IQA) problem based on the machine learning can be seen as constructing a relationship between the distorted images and the final scores [31]. Thus, the KIMEL algorithm is applicable to the blind IQA problem. In this paper, we perform this study. The performance of the KIMEL algorithm is tested on the LIVE IQA database [41] and is compared with that of the KLMS algorithm, the KVSS-LMS algorithm, and the recent NR IQA algorithms in [31,32,38]. The Spearman Rank-Order Correlation Coefficient (SROCC), the (Pearson's) Linear Correlation Coefficient (LCC), and the Root MSE (RMSE) are used to evaluate the performance of these algorithms. Experimental results show that the performance of the proposed algorithm is superior to that of the competing methods.

The paper is organized as follows. In Section 2, we briefly introduce the IDBD algorithm, the kernel method, as well as the KLMS algorithm to make the paper self-contained, and present the KVSS-LMS algorithm for the purpose of comparison. In Section 3, the KIMEL algorithm is formulated and the convergence analyses are performed. In Section 4, a simple application in nonlinear channel equalization is presented to illustrate the effectiveness and advantage of the proposed algorithm. In Section 5, we apply the KIMEL algorithm to blind IQA problem, which is a more practical application. Finally, conclusions are drawn in Section 6.

2. Preliminary knowledge and algorithms for comparison

In this paper, the KIMEL algorithm to be presented is inspired by the IDBD algorithm [2] and based on the famed kernel trick, thus in this section we briefly introduce them, and two kernel algorithms as well, which will be used for comparison in the application parts.

2.1. Incremental Delta-Bar-Delta algorithm

In [2], the IDBD algorithm was derived, which can be regarded as a VSS-LMS algorithm with an individual step-size parameter for each weight dimension and these parameters change according to a metalearning process. The IDBD is developed for learning linear systems. After defining the input vector $X(n)=[x_1(n),x_2(n),\ldots,x_N(n)]^T$, the weight vector $W(n)=[w_1(n),w_2(n),\ldots,w_N(n)]^T$, and the desired output $d(n)$, the output of the linear system can be expressed as $y(n)=W(n)^TX(n)=\sum_{i=1}^{N}w_i(n)x_i(n)$ and the estimation error can be expressed as $e(n)=d(n)-y(n)$. The procedure of the algorithm is as follows:

$$
\begin{aligned}
&b_i(n+1)=b_i(n)+\theta e(n)x_i(n)h_i(n),\\
&a_i(n+1)=e^{b_i(n+1)},\\
&w_i(n+1)=w_i(n)+a_i(n+1)e(n)x_i(n),\\
&h_i(n+1)=h_i(n)[1-a_i(n+1)x_i(n)^2]^+ +a_i(n+1)e(n)x_i(n),
\end{aligned}
\tag{1}
$$

where $[\cdot]^+$ is a half-rectified function. In this algorithm, the step-size parameters a_i of all weight dimensions have an exponential relationship with the memory parameters b_i, which ensures a_i a positive value and provides a mechanism for making geometric steps in a_i. Note that h_i, an additional per-input memory parameter, is a decaying trace of the cumulative sum of recent changes to w_i, thus, the increment to b_i is proportional to the correlation between the current weight change $e(n)x_i(n)$ and a trace of recent weight changes $h_i(n)$. If the current step is positively correlated with past steps, indicating that the past steps should have been larger, the memory parameter b_i, as well as the step-size a_i, is increased. If the current step is negatively correlated with past steps, indicating that the past steps should have been smaller, the memory parameter b_i, as well as the step-size a_i, is decreased. The IDBD algorithm is a metalearning algorithm in the sense that it learns the step-size parameter based on previous learning experience.

It was shown that the IDBD algorithm outperforms the ordinary LMS algorithm and in fact it finds the optimal step-size parameters [2]. In IDBD, both the weight update rule and the learning rate update rule are derived by gradient descent, which is the origin of the ideology of our new algorithm to be presented in the following.

2.2. Kernel method

In order to learn a nonlinear relationship by a linear learning machine, a nonlinear feature set needs to be chosen, that is to say, transform the input data into a high-dimensional feature space using a certain nonlinear mapping and then the linear learning machine is applied in the feature space. Thus, the output of the learning machine has the form

$$
f(\boldsymbol{x})=\sum_{i=1}^{N}w_i\varphi_i(\boldsymbol{x}),
\tag{2}
$$

where $\boldsymbol{x}$ is the input data in original space, $\boldsymbol{\varphi}:\mathbb{X}\rightarrow\mathbb{F}$ is a nonlinear mapping from the input data space to a certain feature space, and w_i is the ith weight component in the feature space, assumed to have N dimensions.

An important characteristic of a linear learning machine is that it can be expressed in dual [6], which means that the weight of the linear learning machine can be expressed as a

linear combination of the training data, i.e., $\boldsymbol{w}=\sum_{j=1}^{l}\alpha_j y_j\boldsymbol{\varphi}(\boldsymbol{x}_j)$, where $\{\boldsymbol{\varphi}(\boldsymbol{x}_j),y_j\}$ $(j=1,\ldots,l)$ are the training examples and α_j is the corresponding coefficient in the feature space. Thus, the output of the linear learning machine can be expressed by the inner product of the transformed test data $\boldsymbol{\varphi}(\boldsymbol{x})$ and training data $\boldsymbol{\varphi}(\boldsymbol{x}_j)$,

$$f(\boldsymbol{x})=\sum_{j=1}^{l}\alpha_j y_j\langle\boldsymbol{\varphi}(\boldsymbol{x}_j),\boldsymbol{\varphi}(\boldsymbol{x})\rangle. \tag{3}$$

If the inner product $\langle\boldsymbol{\varphi}(\boldsymbol{x}_i),\boldsymbol{\varphi}(\boldsymbol{x})\rangle$ can be computed directly in the feature space, we can construct a nonlinear learning machine without exactly knowing the mapping. The well-known kernel method is such a direct computing method.

A Mercer kernel [9,10] is a continuous, symmetric, positive-definite function κ: $\mathbb{X}\times\mathbb{X}\to\mathbb{R}$, where $\mathbb{X}$ is the input domain, a compact subset of $\mathbb{R}^L$. By the Mercer theorem [9,10], any Mercer kernel $\kappa(\boldsymbol{x},\boldsymbol{x}')$ can be expanded as follows,

$$\kappa(\boldsymbol{x},\boldsymbol{x}')=\sum_{i=1}^{\infty}\varsigma_i\varphi_i(\boldsymbol{x})\varphi_i(\boldsymbol{x}'), \tag{4}$$

where ς_i and φ_i are the non-negative eigenvalues and eigenfunctions, respectively. Therefore, a mapping $\boldsymbol{\varphi}$ can be constructed as

$$\boldsymbol{\varphi}:\mathbb{X}\to\mathbb{F}, \tag{5a}$$

$$\boldsymbol{\varphi}(\boldsymbol{x})=[\sqrt{\varsigma_1}\varphi_1(\boldsymbol{x}),\sqrt{\varsigma_2}\varphi_2(\boldsymbol{x}),\ldots]. \tag{5b}$$

By the construction of the mapping, an important implication is

$$\boldsymbol{\varphi}(\boldsymbol{x})^T\boldsymbol{\varphi}(\boldsymbol{x}')=\kappa(\boldsymbol{x},\boldsymbol{x}'). \tag{6}$$

Among all the kernels, the widely used ones are the Gaussian kernel (7) and the polynomial kernel (8),

$$\kappa(\boldsymbol{x},\boldsymbol{x}')=\exp(-a^0\|\boldsymbol{x}-\boldsymbol{x}'\|^2), \tag{7}$$

$$\kappa(\boldsymbol{x},\boldsymbol{x}')=(\boldsymbol{x}^T\boldsymbol{x}'+1)^p, \tag{8}$$

of which the Gaussian kernel has the feature vector with infinite dimensionality and is implicitly known.

Based on the above theory, we can choose a Mercer kernel to replace the inner product $\langle\boldsymbol{\varphi}(\boldsymbol{x}_i),\boldsymbol{\varphi}(\boldsymbol{x})\rangle$, so as to compute the equation (3) directly without knowing the feature map.

2.3. Kernel Least-Mean-Square algorithm

In order to overcome the limitation of the ordinary LMS algorithm in dealing with nonlinear problems, the Kernel Least-Mean-Square (KLMS) algorithm was proposed in [4]. By the kernel-induced mapping (5), firstly transform the input $\boldsymbol{X}(i)$ into RKHS as $\boldsymbol{\varphi}(\boldsymbol{X}(i))$, then use the LMS algorithm on the new example sequence $\{\boldsymbol{\varphi}(\boldsymbol{X}(i)),d(i)\}$. Let $\boldsymbol{W}(1)=\boldsymbol{0}$, then the learning procedure of the KLMS algorithm is as follows:

$$y(n)=\boldsymbol{W}(n)^T\boldsymbol{\varphi}(\boldsymbol{X}(n))=\mu\sum_{j=1}^{n-1}e(j)\kappa(\boldsymbol{X}(j),\boldsymbol{X}(n)),$$

$$e(n)=d(n)-y(n),$$

$$\boldsymbol{W}(n+1)=\boldsymbol{W}(n)+\mu e(n)\boldsymbol{\varphi}(\boldsymbol{X}(n)), \tag{9}$$

where μ is the step-size parameter of the KLMS algorithm.

If a test input point $\boldsymbol{X}_*$ is given at iteration i, the output of the system is

$$y_*=f(\boldsymbol{X}_*)=\mu\sum_{j=1}^{i}e(j)\kappa(\boldsymbol{X}(j),\boldsymbol{X}_*). \tag{10}$$

2.4. Kernel variable step-size LMS algorithm

The robust variable step-size algorithm proposed in [14] was proven to have fast convergence while ensure small misadjustment. It was used by many researchers in various applications. We briefly introduce the algorithm and then derive its kernelized version, for the purpose of comparison.

The weight update of the VSS-LMS algorithm [14] is

$$\boldsymbol{W}(n+1)=\boldsymbol{W}(n)+\mu(n)e(n)\boldsymbol{X}(n), \tag{11}$$

where the variables are defined the same as those in Section 2.1. The variable step size $\mu(n)$ is adjusted by using an estimation of the autocorrelation between $e(n)$ and $e(n-1)$,

$$\mu(n+1)=\alpha\mu(n)+\gamma p^2(n),$$

$$p(n)=\beta p(n-1)+(1-\beta)e(n)e(n-1), \tag{12}$$

where $0<\alpha<1$, $0<\beta<1$ and $\gamma>0$. Usually, the value of α is chosen to be close to 1 and γ is small. In stationary environment, β, which is an exponential weighting parameter, should be $\beta\approx1$, and for nonstationary environment, $\beta<1$. The step size $\mu(n+1)$ is set to $\mu_{\min}$ or $\mu_{\max}$ when it is below the lower bound or above the upper bound. In general, the constant $\mu_{\min}$ is selected as a compromise between steady-state misadjustment and convergence rate; and the value of $\mu_{\max}$ is chosen near the point of instability of the LMS algorithm to provide the maximum possible convergence rate [13,14].

In describing the above VSS-LMS algorithm, a linear finite impulse response filter is assumed. If the mapping between output and input is highly nonlinear, poor performance can be expected from the algorithm. To overcome the limitation, we extend the algorithm to RKHS to present a kernelized version. Here, we call the kernelized algorithm KVSS-LMS.

The derivation of the KVSS-LMS algorithm is as follows. Firstly, utilizing the Mercer theorem (4), we transform the data $\boldsymbol{X}(n)$ into a high-dimensional feature space $\mathbb{F}$ as $\boldsymbol{\varphi}(\boldsymbol{X}(n))$. Denote $\boldsymbol{\varphi}(n)=\boldsymbol{\varphi}(\boldsymbol{X}(n))$ for simplicity. Then using the VSS-LMS algorithm [14] on the new example sequence $\{\boldsymbol{\varphi}(n),d(n)\}$ yields

$$e(n)=d(n)-\boldsymbol{W}^T(n)\boldsymbol{\varphi}(n),$$

$$\boldsymbol{W}(n+1)=\boldsymbol{W}(n)+\mu(n)e(n)\boldsymbol{\varphi}(n), \tag{13}$$

where $\boldsymbol{W}(n)$ denotes the estimate of the weight vector at iteration n in $\mathbb{F}$. By letting $\boldsymbol{W}(1)=\boldsymbol{0}$ and using the weight update equation repeatedly, we have

$$\boldsymbol{W}(n+1)=\sum_{j=1}^{n}\mu(j)e(j)\boldsymbol{\varphi}(j). \tag{14}$$

However, $\boldsymbol{\varphi}$ is only implicitly known, and its dimensionality is high (infinite in the case of the Gaussian kernel). So, we need an alternative way to carry out the computation. Here, we adopt the same method as in the derivation of the KLMS algorithm [4]. That is, when there is a new input $\boldsymbol{X}(n+1)$, the output $y(n+1)$ of the system can be expressed in terms of inner products between transformed inputs, then by the kernel trick (6), we can efficiently compute the output through kernel evaluations

$$\begin{aligned} y(n+1) &= \boldsymbol{W}(n+1)^T\boldsymbol{\varphi}(n+1) \\ &= \sum_{j=1}^{n}\mu(j)e(j)\boldsymbol{\varphi}(j)^T\boldsymbol{\varphi}(n+1) \\ &= \sum_{j=1}^{n}\mu(j)e(j)\kappa(j,n+1). \end{aligned} \tag{15}$$

After detailed derivation, we found that the update rule of the step-size parameter in the KVSS-LMS algorithm stays the same as that in (12). Thus, by letting $\boldsymbol{W}(1)=\boldsymbol{0}$, we can summarize the KVSS-LMS algorithm as follows:

$$\begin{aligned} &y(n) = \boldsymbol{W}(n)^T\boldsymbol{\varphi}(\boldsymbol{X}(n)) = \mu\sum_{j=1}^{n-1}e(j)\kappa(\boldsymbol{X}(j),\boldsymbol{X}(n)), \\ &e(n) = d(n)-y(n), \\ &\boldsymbol{W}(n+1) = \boldsymbol{W}(n)+\mu(n)e(n)\boldsymbol{\varphi}(\boldsymbol{X}(n)), \\ &p(n) = \beta p(n-1)+(1-\beta)e(n)e(n-1), \\ &\mu(n+1) = \alpha\mu(n)+\gamma p^2(n). \end{aligned} \tag{16}$$

Compared with the KLMS algorithm, the KVSS-LMS algorithm has an adaptive step-size parameter and we do not need to adjust it by hand.

At iteration i, given a test input point $\boldsymbol{X}_*$, the output of the system is

$$y_* = f(\boldsymbol{X}_*) = \sum_{j=1}^{i}\mu(j)e(j)\kappa(\boldsymbol{X}(j),\boldsymbol{X}_*). \tag{17}$$

Since the update of the step-size parameter in the KVSS-LMS algorithm is in the original space other than in RKHS, the convergence analysis of the algorithm is quite similar to that in [14]. Limited to the length of this paper, we omit it here.

3. Kernel Incremental MEtaLearning Algorithm (KIMEL)

3.1. Algorithm formulation

Inspired by the derivation of the IDBD algorithm [2], we propose an incremental metalearning algorithm in RKHS, named KIMEL. Many well-known learning algorithms were derived by using gradient decent, including the LMS algorithm and the Back-Propagation (BP) learning algorithm. Here we derive the KIMEL algorithm also by using gradient descent. The newly proposed algorithm is not directly the IDBD algorithm [2] in RKHS. The original IDBD algorithm has an individual step-size parameter for each weight component of a weight vector, while the KIMEL algorithm has a common step-size parameter for all weight components of a weight vector. Because in RKHS, the weight dimension can be very high, for example, it is infinite for the Gaussian kernel, adjusting individual step-size parameter on each weight dimension is unrealistic. Thus, we adopt the common step-size parameter adjusting strategy on all weight components of a weight vector here. This is also the main reason that we do not call the new algorithm KIDBD. We admit that an algorithm with a common step-size parameter is not as good as the counterpart with different step-size parameters. This is a compromise strategy. Fortunately, from the simulations shown in the next sections, we see that the KIMEL algorithm with a common step-size parameter still has excellent performance.

We are trying to minimize the expected value of the square error $e^2(n)$, where $e(n)=d(n)-y(n)$. Because the expected error is unknown, we use instead the gradient of the sample error $e(n)$. The update of the weight vector is

$$\begin{aligned} \boldsymbol{W}(n+1) &= \boldsymbol{W}(n)-\frac{1}{2}\mu(n+1)\frac{\partial e^2(n)}{\partial \boldsymbol{W}(n)} \\ &= \boldsymbol{W}(n)-\mu(n+1)e(n)\frac{\partial e(n)}{\partial \boldsymbol{W}(n)} \\ &= \boldsymbol{W}(n)-\mu(n+1)e(n)\frac{\partial [d(n)-y(n)]}{\partial \boldsymbol{W}(n)} \\ &= \boldsymbol{W}(n)+\mu(n+1)e(n)\frac{\partial y(n)}{\partial \boldsymbol{W}(n)} \\ &= \boldsymbol{W}(n)+\mu(n+1)e(n)\frac{\partial}{\partial \boldsymbol{W}(n)}[\boldsymbol{W}(n)^T\boldsymbol{\varphi}(n)] \\ &= \boldsymbol{W}(n)+\mu(n+1)e(n)\boldsymbol{\varphi}(n), \end{aligned} \tag{18}$$

where $\boldsymbol{\varphi}(n)$ is the projection of the input $\boldsymbol{X}(n)$ in RKHS.

We use an alternative function of the step-size parameter as follows, other than the exponential function shown in (1):

$$\mu(n) = \frac{1}{1+e^{-\beta(n)}}. \tag{19}$$

By using this function, the step-size parameter μ is guaranteed positive, and the range is (0,1), which is the range of the step-size parameter usually in. Thus, we do not need to set the upper and the lower bound additionally.

The sample error is also a function of β. Using gradient descent to minimize the sample error $e(n)$ with respect to β, we have:

$$\begin{aligned} \beta(n+1) &= \beta(n)-\frac{1}{2}\theta\frac{\partial e^2(n)}{\partial \beta} \\ &= \beta(n)-\frac{1}{2}\theta\frac{\partial e^2(n)}{\partial \boldsymbol{W}(n)}\frac{\partial \boldsymbol{W}(n)}{\partial \beta} \\ &= \beta(n)+\theta e(n)\boldsymbol{\varphi}(n)^T\frac{\partial \boldsymbol{W}(n)}{\partial \beta} \\ &= \beta(n)+\theta e(n)\boldsymbol{\varphi}(n)^T\boldsymbol{h}(n), \end{aligned} \tag{20}$$

where the positive constant θ is the so-called metalearning rate [2]. In this equation, the partial derivative with respect to β without a time index should be explained as the derivative with respect to an infinitesimal change in β at all time steps. The vector $\boldsymbol{h}(n)$ is defined as $\partial \boldsymbol{W}(n)/\partial\beta$, which is a memory parameter. The update rule for $\boldsymbol{h}(n)$ is in turn derived as follows:

$$\begin{aligned} \boldsymbol{h}(n+1) &= \frac{\partial \boldsymbol{W}(n+1)}{\partial \beta} = \frac{\partial}{\partial \beta}\left[\boldsymbol{W}(n)+\frac{1}{1+e^{-\beta(n+1)}}e(n)\boldsymbol{\varphi}(n)\right] \\ &= [\boldsymbol{I}-\mu(n+1)\boldsymbol{\varphi}(n)\boldsymbol{\varphi}(n)^T]\boldsymbol{h}(n)+\mu(n+1)[1-\mu(n+1)]e(n)\boldsymbol{\varphi}(n), \end{aligned} \tag{21}$$

where $[\boldsymbol{I}-\mu(n+1)\boldsymbol{\varphi}(n)\boldsymbol{\varphi}(n)^T]\boldsymbol{h}(n)$ should be interpreted as a forgetting term, which causes a decay of $\boldsymbol{h}(n)$. To make the algorithm computationally tractable in RKHS, we use a constant forgetting factor $\lambda(0<\lambda<1)$. Thus the first term becomes $\lambda\boldsymbol{h}(n)$. Getting rid of the half-rectified operator (see (1)) also brings convenience for theoretical analysis. Since $\mu(n+1)[1-\mu(n+1)]$ is always positive, the second term increments $\boldsymbol{h}(n)$ by the last weight change. The repeated application of (21) through iterations yields

$$\boldsymbol{h}(n)=\sum_{j=1}^{n-1}\lambda^{n-1-j}\mu(j+1)[1-\mu(j+1)]e(j)\boldsymbol{\varphi}(j). \tag{22}$$

Thus, the memory vector $\boldsymbol{h}(n)$ is a decaying trace of the cumulative sum of recent changes to $\boldsymbol{W}(n+1)$.

Similar to the analysis in the previous algorithm, $\boldsymbol{\varphi}$ is only implicitly known. So, we need an alternative way to carry out the computation. To make the KIMEL algorithm computable, the step-size parameter should be computable in RKHS. By substituting (22) into (20), the update of β is:

$$\begin{aligned}\beta(n+1)&=\beta(n)+\theta e(n)\boldsymbol{\varphi}(n)^T\sum_{j=1}^{n-1}\lambda^{n-1-j}\mu(j+1)[1-\mu(j+1)]e(j)\boldsymbol{\varphi}(j)\\&=\beta(n)+\theta e(n)\sum_{j=1}^{n-1}\lambda^{n-1-j}\mu(j+1)[1-\mu(j+1)]e(j)\boldsymbol{\varphi}(n)^T\boldsymbol{\varphi}(j)\\&=\beta(n)+\theta e(n)\sum_{j=1}^{n-1}\lambda^{n-1-j}\mu(j+1)[1-\mu(j+1)]e(j)\kappa(n,j).\end{aligned} \tag{23}$$

From (23), we see that the memory parameter β is computable, then the step-size parameter μ is also computable in RKHS. Unlike in the IDBD algorithm, β in (23) does not depend on $\boldsymbol{h}(n)$ explicitly any more. Let $\boldsymbol{W}(1)=\boldsymbol{0}$, the KIMEL algorithm is summarized below.

Algorithm. The kernel incremental metalearning algorithm (KIMEL)

Input: training data $(\boldsymbol{X}(1),d(1)),(\boldsymbol{X}(2),d(2)),\ldots,(\boldsymbol{X}(N),d(N))$
Initialization: choose the metalearning rate θ, the forgetting factor λ, the initial step-size $\{\mu(1),\mu(2)\}$, the kernel $\kappa(\cdot,\cdot)$, the initial center $\mathcal{C}(1)=\{\boldsymbol{X}(1)\}$, and the initial output $y(1)=0$.
Iteration:
for $n=2:N$
(1) compute the filter output
$y(n)=\boldsymbol{W}(n)^T\boldsymbol{\varphi}(n)=\sum_{j=1}^{n-1}\mu(j+1)e(j)\kappa(j,n)$
(2) compute the error
$e(n)=d(n)-y(n)$
(3) compute the new step-size
$\beta(n+1)=\beta(n)+\theta e(n)\cdot\sum_{j=1}^{n-1}\lambda^{n-1-j}\mu(j+1)(1-\mu(j+1))e(j)\kappa(n,j)$
$\mu(n+1)=\frac{1}{1+e^{-\beta(n+1)}}$
(4) store the new center
$\mathcal{C}(n)=\{\mathcal{C}(n-1),\boldsymbol{X}(n)\}$
end

Finally, we note that the increment to β is proportional to the product of the current weight change $e(n)\boldsymbol{\varphi}(n)$ and a trace of recent weight changes. If the current step is positively correlated with past steps, it indicates that the past steps should have been larger (and equation (20) accordingly increases β, then the step-size parameter μ increases); if the current step is negatively correlated with past steps, it indicates that the past steps were too large (and equation (20) accordingly decreases β, then the step-size parameter μ decreases). As we know from Section 2.1, such are the basic features of the IDBD algorithm, which are inherited in the KIMEL algorithm. Note that this algorithm, as well as the former two kernel algorithms, are computed without using the weights. We write explicitly the expressions of the weight iterations simply for clarity and for keeping consistent with the usual (non-kernel) adaptive algorithms.

3.2. Convergence analysis of the mean weight vector

The KIMEL algorithm given in *Algorithm* is difficult to analyze exactly. To simplify the analysis, we introduce the following assumption [13]:

$$E[\mu(n+1)e(n+1)\boldsymbol{\varphi}(n)]=E[\mu(n+1)]E[e(n)\boldsymbol{\varphi}(n)]. \tag{24}$$

The condition for this assumption to be true is that $\mu(n+1)$ is a constant. It cannot really hold for the KIMEL algorithm. However, normally, the metalearning rate θ is chosen as a small value, thus the step-size parameter varies slowly around its mean value and we can say that this assumption is approximately true. This assumption was widely adopted in analyzing variable step-size adaptive algorithms.

Now we provide the convergence analysis of the proposed algorithm when operating in stationary environment. The analysis method is a standard method in the field.

The desired output signal $d(n)$ is given by

$$d(n)=\boldsymbol{W}_*^T\boldsymbol{\varphi}(n)+\xi(n), \tag{25}$$

where $\xi(n)$ is a zero-mean independent disturbance.

From the above assumption, we have the mean weight vector in RKHS:

$$\begin{aligned}E[\boldsymbol{W}(n+1)]&=E[\boldsymbol{W}(n)]+E[\mu(n+1)]E[e(n)\boldsymbol{\varphi}(n)]\\&=E[\boldsymbol{W}(n)]+E[\mu(n+1)]E[(d(n)-y(n))\boldsymbol{\varphi}(n)]\\&=E[\boldsymbol{W}(n)]+E[\mu(n+1)]E[(\boldsymbol{\varphi}(n)^T\boldsymbol{W}_*\\&\quad+\xi(n)-\boldsymbol{\varphi}(n)^T\boldsymbol{W}(n))\boldsymbol{\varphi}(n)]\\&=E[\boldsymbol{W}(n)]+E[\mu(n+1)]\mathbf{R}E[\boldsymbol{W}_*-\boldsymbol{W}(n)],\end{aligned} \tag{26}$$

where $\mathbf{R}=E[\boldsymbol{\varphi}(n)\boldsymbol{\varphi}(n)^T]$ is the autocorrelation matrix of the transformed input data. Let $\tilde{\boldsymbol{W}}(n)=\boldsymbol{W}(n)-\boldsymbol{W}_*$ represent the weight-error vector, then by subtracting $E[\boldsymbol{W}_*]$ from both sides of (26), we have

$$E[\tilde{\boldsymbol{W}}(n+1)]=[\mathbf{I}-E[\mu(n+1)]\mathbf{R}]E[\tilde{\boldsymbol{W}}(n)]. \tag{27}$$

The mean weight vector is converged if and only if

$$\prod_{n=0}^{N}[\boldsymbol{I}-E[\mu(n+1)]\mathbf{R}]\to 0,\quad \text{as } N\to\infty. \tag{28}$$

From (28), we obtain a sufficient condition to ensure the convergence of the mean weight vector:

$$0<E[\mu(n+1)]<\frac{2}{\lambda_{\max}(\mathbf{R})}, \tag{29}$$

where $\lambda_{\max}(\mathbf{R})$ is the maximum eigenvalue of the matrix $\mathbf{R}$. However, the upper bound in (29) is incomputable as a result of $\mathbf{R}=E[\boldsymbol{\varphi}(n)\boldsymbol{\varphi}(n)^T]$. In order to compute it conveniently, we use the estimation version of the autocorrelation matrix. Denoting the transformed data matrix

$\mathbf{\Phi}=[\boldsymbol{\varphi}_1,\boldsymbol{\varphi}_2,\ldots,\boldsymbol{\varphi}_M]$, where M is the size of the training data, $\mathbf{R}_\varphi=(1/M)\mathbf{\Phi}\mathbf{\Phi}^T$ its autocorrelation matrix, and $\mathbf{G}_\varphi=\mathbf{\Phi}^T\mathbf{\Phi}$ its Gram matrix which is an $M\times M$ matrix with $\kappa(\boldsymbol{X}(i),\boldsymbol{X}(j))$ as its (i,j)th component. We use the following facts: $\mathbf{R}_\varphi$ and $\mathbf{G}_\varphi$ are both positive semidefinite; if $\mathbf{R}_\varphi$ has m nonzero eigenvalues $\{\lambda_j\}_{j=1}^m$, $\mathbf{G}_\varphi$ has m nonzero eigenvalues, which are $\{M\lambda_j\}_{j=1}^m$. Therefore, the estimation version of the convergence range for $E[\mu(n+1)]$ is

$$0<E[\mu(n+1)]<\frac{2M}{\lambda_{\max}(\mathbf{G}_\varphi)}. \tag{30}$$

3.3. Mean-square-error behavior and steady-state analysis

In the following, we conduct the MSE behavior and steady-state analysis. To assess the performance of the system, we need to derive the expression of the misadjustment.

Since $\mathbf{R}$, the autocorrelation matrix of the transformed data $\boldsymbol{\varphi}$, is symmetric, there exist matrices $\mathbf{Q}$ and $\mathbf{\Lambda}$, where $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues of $\mathbf{R}$ and $\mathbf{Q}\mathbf{Q}^T=\mathbf{Q}^T\mathbf{Q}=\mathbf{I}$, such that $\mathbf{R}=\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$. Let $\boldsymbol{V}(n)=\mathbf{Q}^T\tilde{\boldsymbol{W}}(n)$ and $\boldsymbol{\varphi}'(n)=\mathbf{Q}^T\boldsymbol{\varphi}(n)$, this algorithm in terms of $\boldsymbol{V}(n)$ is as follows:

$$\boldsymbol{V}(n+1)=[\boldsymbol{I}-\mu(n+1)\boldsymbol{\varphi}'(n)\boldsymbol{\varphi}'(n)^T]\boldsymbol{V}(n)+\mu(n+1)\xi(n)\boldsymbol{\varphi}'(n). \tag{31}$$

As in [12], the MSE is defined as

$$\begin{aligned}E[e^2(n)]&=E[\xi^2(n)-2\xi(n)\boldsymbol{\varphi}'(n)^T\boldsymbol{V}(n)+\boldsymbol{V}(n)^T\varphi'(n)\varphi'(n)^T\boldsymbol{V}(n)]\\&=E[\xi^2(n)]+\mathrm{tr}(\mathbf{\Lambda}E[\boldsymbol{V}(n)\boldsymbol{V}(n)^T])\\&=\epsilon_{\min}+\epsilon_{ex}(n),\end{aligned} \tag{32}$$

where $\epsilon_{\min}=E[\xi^2(n)]$ is the minimum value of the MSE, and $\epsilon_{ex}(n)=\mathrm{tr}(\mathbf{\Lambda}E[\boldsymbol{V}(n)\boldsymbol{V}(n)^T])$ is the excess MSE. From equation (32), we know that the MSE is related to the diagonal elements of $E[\boldsymbol{V}(n)\boldsymbol{V}(n)^T]$. To calculate the expression of $E[\boldsymbol{V}(n)\boldsymbol{V}(n)^T]$, as widely accepted and used in this field (see, e.g. [7,8]), we assume that the weight-error vector $\boldsymbol{V}(n)$ and the input vector $\boldsymbol{\varphi}(n)$ are statistically independent. Then by postmultiplying both sides of (31) by $\boldsymbol{V}(n+1)^T$ and taking expectations, we have

$$\begin{aligned}E[\boldsymbol{V}(n+1)\boldsymbol{V}(n+1)^T]\approx{}&E[\boldsymbol{V}(n)\boldsymbol{V}(n)^T]-\mathbf{\Lambda}E[\mu(n+1)]E[\boldsymbol{V}(n)\boldsymbol{V}(n)^T]\\&-E[\mu(n+1)]E[\boldsymbol{V}(n)\boldsymbol{V}(n)^T]\mathbf{\Lambda}+E[\mu^2(n+1)]\mathbf{\Lambda}E[\boldsymbol{V}(n)\boldsymbol{V}(n)^T]\mathbf{\Lambda}\\&+E[\mu^2(n+1)]\mathbf{\Lambda}\epsilon_{min}.\end{aligned} \tag{33}$$

Note that the relationship between $\mu(n)$ and $\beta(n)$ is nonlinear, referring to *Algorithm*. In order to compute $E[\mu(n)]$ given the expectation of $\beta(n)$, we firstly introduce a conclusion: For a continuous random variable X with expectation χ and variation σ^2, and Y is some function of X, denoted as $Y=H(X)$, then the expectation of Y can be computed approximately by

$$E[Y]\approx H(\chi)+\frac{H''(\chi)}{2}\sigma^2. \tag{34}$$

This approximation can be obtained easily by Taylor series expansion around $E(X)$. Apply this conclusion to the analysis of $E[\mu(n)]$ and $E[\mu^2(n)]$, and meanwhile we notice that, as mentioned above, $\beta(n)$ can be regarded approximately as a constant in the small θ case, thus the variation of $\beta(n)$ is very small. Using (34), we have

$$E[\mu(n)]\approx\frac{1}{1+e^{-E[\beta(n)]}}, \tag{35}$$

and

$$E[\mu^2(n)]\approx\left(\frac{1}{1+e^{-E[\beta(n)]}}\right)^2. \tag{36}$$

From *Algorithm*, the mean of $\beta(n)$ is

$$\begin{aligned}E[\beta(n+1)]={}&E[\beta(n)]\\&+E\left[\theta e(n)\sum_{j=1}^{n-1}\lambda^{n-1-j}\mu(j+1)[1-\mu(j+1)]e(j)\kappa(n,j)\right].\end{aligned} \tag{37}$$

To study the steady-state performance of the proposed algorithm, in the following analysis we assume that the proposed algorithm has converged. In this case, the sample of the error $e(n)$ can be assumed uncorrelated, i.e., $E[e(n-i)e(n-j)]=0$ for $i\neq j$ and the step-size parameter $\mu(n)$ is approximately independent of $e(n)$ and $\boldsymbol{\varphi}(n)$ [2], hence, the last term of (37) is zero. This means that the value of $E[\beta(n)]$ is a constant in the converged condition. Denoting $E[\beta(\infty)]=B$, from (35) and (36), we obtain the mean and mean-square behaviors of $\mu(n)$ and $\mu^2(n)$,

$$E[\mu(\infty)]\approx\frac{1}{1+e^{-B}}, \tag{38}$$

and

$$E[\mu^2(\infty)]\approx\left(\frac{1}{1+e^{-B}}\right)^2, \tag{39}$$

where $E[\mu^2(\infty)]$ and $E[\mu(\infty)]$ are the steady-state values of $E[\mu^2(n)]$ and $E[\mu(n)]$, respectively. Following the same argument as in [13], a sufficient condition that guarantees the convergence of the MSE is

$$0<\frac{E[\mu^2(\infty)]}{E[\mu(\infty)]}<\frac{2}{3\ \mathrm{tr}(\mathbf{R})}. \tag{40}$$

From (38) and (39), we have

$$\frac{E[\mu^2(\infty)]}{E[\mu(\infty)]}\approx\frac{1}{1+e^{-B}}\approx E[\mu(\infty)]=C, \tag{41}$$

hence, the condition to ensure the convergence of the MSE in the KIMEL algorithm is

$$0<E[\mu(\infty)]<\frac{2}{3\ \mathrm{tr}(\mathbf{R})}, \tag{42}$$

i.e.,

$$E[\beta(\infty)]>\ln\frac{2}{3\ \mathrm{tr}(\mathbf{R})-2}. \tag{43}$$

The misadjustment is defined as [11]

$$M=\frac{\epsilon_{ex}(\infty)}{\epsilon_{\min}}. \tag{44}$$

If the algorithm satisfies the condition (42) or (43), substituting (32) and (33) into (44), we have

$$M=\sum_{i=1}^{N}\frac{E[\mu^2(\infty)]\lambda_i}{2E[\mu(\infty)]-E[\mu^2(\infty)]\lambda_i}. \tag{45}$$

Rearranging (45) and substituting (41) into (45) yield

$$M=\sum_{i=1}^{N}\frac{C\lambda_i}{2-C\lambda_i}. \tag{46}$$

In the event of small value of misadjustment so that $\sum_{i=1}^{N} C\lambda_i \ll 1$, we have the following expression of misadjustment:

$$M\approx\frac{C}{2}\,\mathrm{tr}(\mathbf{R}). \tag{47}$$

Similarly, the misadjustment expressed in (47) is incomputable as a result of $\mathbf{R}=E[\boldsymbol{\varphi}(n)\boldsymbol{\varphi}(n)^T]$, but we can obtain the estimation version of the misadjustment of the KIMEL as

$$M\approx\frac{C}{2}\,\mathrm{tr}(\mathbf{R}_\varphi)=\frac{C}{2M}\,\mathrm{tr}(\mathbf{G}_\varphi). \tag{48}$$

4. A simple application in nonlinear channel equalization

The KLMS algorithm has been shown to have good performance in nonlinear channel equalization [4]. Now we also test the KIMEL algorithm in this problem and compare it with the KLMS and KVSS-LMS algorithms.

We first describe the example used in [5] briefly. The nonlinear channel model is shown in Fig. 1, which consists of a serial connection of a linear filter and a memoryless nonlinearity unit. A binary signal $\{s(1),s(2),\ldots,s(N)\}$ is fed into the channel. At the receiver of the channel, the signal is corrupted by additive white Gaussian noise and the output is $\{r(1),r(2),\ldots,r(N)\}$. The aim of channel equalization is to construct an "inverse" filter that reproduces the original signal with as low error as possible. It can be formulated as a regression problem with examples $\{[r(i),r(i+1),\ldots,r(i+l)],s(i-D)\}$, where l is a time-embedding length and D is a time lag. As in [5], we set $l=5$ and $D=2$. The nonlinear channel model is defined by $x(i)=s(i)+0.5s(i-1)$ and $r(i)=x(i)-0.9x(i)^2+n(i)$, where $n(i)$ is the white Gaussian noise with a variance of σ^2.

Fig. 1. Basic structure of a nonlinear channel.

We first compare the performance among the KLMS, KVSS-KLMS, and KIMEL algorithms with the varying of the (meta-)learning rates μ, γ and θ, respectively. In this simulation, for the KVSS-LMS algorithm, we set $\alpha=0.99$ (which is found to work well in this application), $\beta=0.99$ (for stationary environment it should be close to 1 as mentioned above), $\mu_{\max}=0.5$, $\mu_{\min}=0.001$. For the KIMEL algorithm, $\lambda=0.8$ is used. We found that the choice of λ has little effect on the performance as long as it is close to and less than 1. Since the step-size parameters in the KVSS-LMS and KIMEL are both adaptive, the initial step-size parameters can be set as arbitrary small values. In our simulation, they are both set as 0.5. In all the three algorithms, the Gaussian kernel with $a^0=0.1$ is chosen.

The filters are trained with 1000 data and fixed afterward, then 200 sample random test data are used to test the performance of the three algorithms. Since the three MSE curves corresponding to the three algorithms have three different (meta-)learning parameters (horizontal axes), plotting them altogether in a panel is impossible. We plot them in two panels, as shown in Fig. 2. Each point in the figure is obtained by averaging over 50 Monte Carlo independent tests. The result in Fig. 2(a) shows that the KVSS-LMS outperforms the KLMS in terms of the MSE in a certain range of γ. However, if the value of γ approaches zero, the performance of the KVSS-LMS is worse than that of the KLMS, which is of no surprise because the step-size parameter $\mu(n)$ decreases exponentially when γ is small, and after some iterations, it become so small that the forthcoming training data has little contribution to the MSE reduction. In Fig. 2(b), it can be seen that the KIMEL outperforms the KLMS in a wide range of θ, and the choice of θ has little influence on the performance of the algorithm. The performance of the KIMEL for a wide range

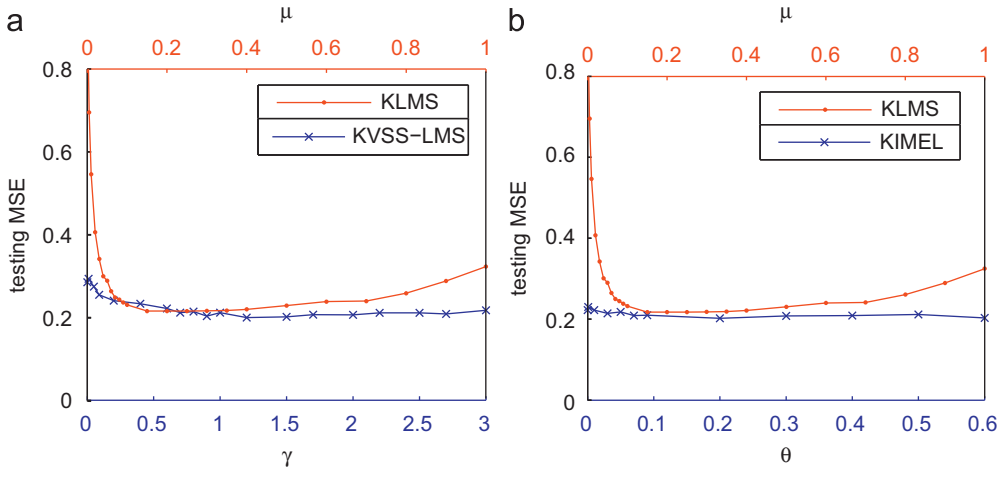

Fig. 2. (a) The influence of different γ on the testing MSE of the KVSS-LMS and the KLMS and (b) the influence of different θ on the testing MSE of the KIMEL and the KLMS.

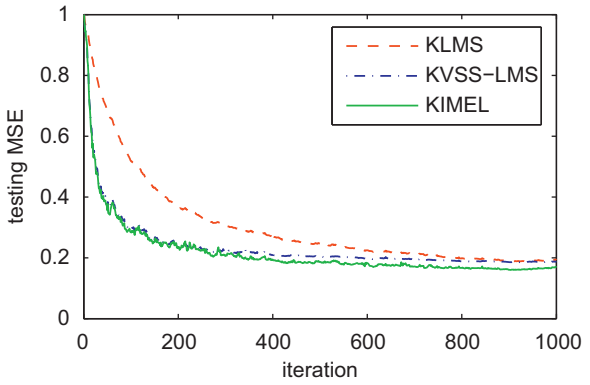

Fig. 3. The learning curves of the KLMS, the KVSS-LMS and the KIMEL algorithms.

of θ is as good as that of the KVSS-LMS with the optimal γ value.

Next, we compare the convergence behaviors of these algorithms. The filters are also trained with 1000 data and tested with another 200 sample random test set. At each iteration, the testing MSE is computed on the test set using the filters resulting from the training, which is the same as the computing method in [5]. The step-size parameter of the KLMS is set as 0.1 which is almost the optimal value for the KLMS algorithm refereing to Fig. 2. To guarantee that the final MSE of the KVSS-LMS is approximately the same as that of the KLMS, $\gamma = 0.5$ is used in the KVSS-LMS. As for the KIMEL, due to the little influence of θ on the final MSE, we choose $\theta = 0.5$ to ensure that the initial convergence rate is approximately equal to that of the KVSS-LMS. The setting of the other parameters is the same as above. Fig. 3 is a typical plot of the learning curves, averaged over 10 Monte Carlo independent tests. It shows that the KIMEL algorithm provides a faster convergence rate than the KLMS algorithm, and the lowest final MSE among the three algorithms.

5. Application in blind image quality assessment

We live in an increasingly visual digital world. How to assess the quality of an image becomes more and more important. An intuitive method is subjective testing, which measures perceived quality by asking many human assessors to score the quality of a set of test images. Although this method can achieve "accurate" results, it is time-consuming and very difficult to model in a deterministic way. Therefore, there is an increasing demand to develop objective measurement techniques that can assess image quality automatically.

Objective image quality assessment can be divided into three categories depending on the amount of information provided to the algorithm. Full-Reference (FR) methods [19–21] need whole information of the original undistorted image and the distorted image whose quality is to be assessed. Reduced-Reference (RR) approaches [22–25] are provided with the distorted image and some additional information of the original undistorted image, such as visual features. No-Reference (NR) /blind methods [26–32] need no information of the original undistorted image. In fact, we often meet the case that the original images are not given in advance. So the development of NR methods is highly desirable.

Recently, various NR IQA algorithms were presented. In [26], it uses Nature Scene Statistic (NSS) to blindly evaluate the quality of images compressed by JPEG2000 image coder. A similar method was proposed in [28], which is based on NSS of the DCT coefficients. In [29], the image quality assessment method based on the circular back propagation neural model was presented to measure the effects of image enhancement filters, using general pixel-based image features. In the above mentioned NR IQA algorithms, they are intended to assess a single specific type of distortion, such as distortions in JPEG compressed image. Naturally, more general NR IQA methods are required to deal with multiple types of distortions. In [32], the Blind Image Quality Index (BIQI) was proposed, which is a two-step framework for NR IQA based on NSS. Recently, machine learning techniques have also been applied to IQA, such as Nonnegative Matrix Factorization (NMF), subspace learning and neural network methods [33–37]. In [31], an image quality assessment algorithm based on the General Regression Neural Network (GRNN) was proposed, which uses the relevant perceptual features. In this paper, the underlying IQA problem belongs to the NR/blind category. In the following, firstly we introduce several relevant perceptual features and describe an image quality assessment scheme used in this paper, then the experiments are given to demonstrate the effectiveness of the proposed algorithm.

5.1. Relevant perceptual features

There are many methods about feature extraction, such as Independent Component Analysis (ICA) [17] and Direct Kernel Biased Discriminant Analysis (DKBDA) [18]. In this paper, we choose several image features from [27,31] which possess relevant information-bearing. These features are complementary aspects on the image content: phase congruency, local information, gradient and zero-crossing rate. The first feature measures the degree of coherency of the local frequencies comprising the image; the second feature reflects the available local information contents of the image; the third feature measures the relevant rate of change of image luminance; and the last feature reflects the activity of the image. Any one of these is less relevant without the other three. Phase congruency loses relevance in the case that the image information and activity is reduced. If the local phase is random-like or the activity is low, the image information will not reflect perceptual features. The gradient and zero-crossing rate are less critical when the image has less local information and structure (phase congruency).

5.1.1. Phase congruency

Phase congruency is a relatively new concept as an image feature. The underlying principle of phase congruency is that perceptually significant image features occur at spatial locations where the important Fourier components are maximally in-phase with one another.

Phase congruency has many applications in image processing, including feature detection, segmentation, face recognition, etc.

Phase congruency has two popular kinds of definitions. The first definition was proposed by Morrone and Owens [39], who defined its function in terms of the Fourier series expansion of a signal I at a location x to be

$$PC_I(x)=\max_{\overline{\varphi}(x)\in[0,2\pi]}\frac{\sum_n A_n\cos[\varphi_n(x)-\overline{\varphi}(x)]}{\sum_n A_n}, \tag{49}$$

where A_n is the amplitude of the nth Fourier component of the image signal I, $\varphi_n(x)$ is the local phase of the Fourier component at x, and $\overline{\varphi}(x)$ is the average phase at x. The value of $\overline{\varphi}(x)$ that maximizes the equation (49) is the amplitude weighted mean local phase angle of all the Fourier terms at x.

The second measure of phase congruency was proposed by Kovesi [40], which is easier to compute

$$PC_{II}(x)=\frac{\sum_n W(x)\lfloor A_n(x)\Delta\varphi_n(x)-T\rfloor}{\sum_n A_n(x)+\varepsilon}, \tag{50}$$

where $\lfloor\rfloor$ denotes that the enclosed quantity is equal to itself when its value is nonnegative, and zero otherwise; the term $W(x)$ is a factor that weights for frequency spread; $A_n(x)=\sqrt{e_n(x)^2+o_n(x)^2}$ is the amplitude at a given wavelet scale, $[e_n(x),o_n(x)]=[I(x)*M_n^e,I(x)*M_n^o]$, I is the image signal, M_n^e is even-symmetric wavelet at a scale n, while M_n^o is odd-symmetric; $\Delta\varphi_n(x)=\cos[\varphi_n(x)-\overline{\varphi}(x)]-|\sin[\varphi_n(x)-\overline{\varphi}(x)]|$ is a sensitive measure of phase deviation; T is an estimate of the noise level which is determined from the statistics of the filter responses to the image; and ε is a small value which prevents division from zero. The MATLAB code of computing phase congruency can be found at http://www.csse.uwa.edu.au/ pk/Research/MatlabFns/index.html.

5.1.2. Image entropy

The sample entropy of the image I is defined as

$$E_I=-\sum_n p(n)\log_2 p(n), \tag{51}$$

where $p(n)$ is the empirical probability of luminance value n. We can use the MATLAB function entropy() to calculate the entropy of an image. It has been used in a variety of ways, but only a few applications are relevant to IQA.

5.1.3. Image gradient

The image gradient is a directional change in the intensity or color in an image. Image gradient is a normal way used to extract information from an image. If luminance changes significantly in the image I, the gradient $\nabla I=[\nabla I_x,\nabla I_y]$ is large. A simple and robust measurement of the horizontal and vertical components of the gradient of I is attained by convolving I with the 3×3 Sobel operator. We can use MATLAB function gradient() to compute it. As usual, the gradient amplitude is estimated as the square root of the sum of the direction derivation estimates.

5.1.4. Zero-crossing rate

Although blurring is difficult to be evaluated without the reference image, it causes the reduction of signal activity. We can use the activity measures to get more insight for the relative blur in the image, in addition to that reflected in the phase congruency. One of the activity measures is the Zero-Crossing (ZC) rate. Denote the test image signal as $I(m,n)$ where $m\in[1,M]$ and $n\in[1,N]$. For $n\in[1,N-2]$, we define

$$z_h(m,n)=\begin{cases}0 & \text{horizontal ZC at } d_h(m,n),\\ 1 & \text{otherwise,}\end{cases} \tag{52}$$

where $d_h(m,n)=I(m,n+1)-I(m,n)$ reflects a difference signal along each horizontal line. The horizontal ZC rate then can be estimated as

$$Z_h=\frac{1}{M(N-2)}\sum_{i=1}^{M}\sum_{j=1}^{N-2}z_h(m,n). \tag{53}$$

We can calculate the vertical feature of ZC rate Z_v by using the similar method. Finally the feature is given by

$$Z=\frac{Z_h+Z_v}{2} \tag{54}$$

5.2. NR/blind image quality assessment scheme

In Fig. 4, we show a schematic diagram for NR/blind image quality assessment which is used in our experiment. It consists of four parts: the first part is a number of images from an image database; the second part is feature extraction and the extracting results contains image information; the third part is regarded as a processor which can assess the quality of images by using these features as inputs; and the last one is the output of the processor which is image quality score.

5.3. Experiments

We use five perceptually motivated features as the input according to the above illustration: (1) the mean value of the phase congruency of the distorted image; (2) the entropy of the phase congruency of the distorted image; (3) the entropy of the gray value of the distorted image; (4) the mean value of the gradient amplitude of the distorted image; and (5) the average of the ZC rate.

In our experiment, we use the popular LIVE IQA Database [41]. There are five types of distorted images in the database: JPEG, JPEG2000 (JP2K), White Noise (WN), Blur, and Fast-Fading (FF). The database contains a total of 982 images. Besides, this database includes Differential Mean Opinion Scores (DMOS), which are given by subjective methods for each distorted image. In the following experiments, we remove all reference images (totally 203) from the database, leaving 779 distorted images for training and testing. The LIVE IQA Database (totally 779 images) consists of 29 groups of

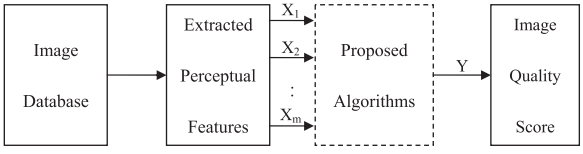

Fig. 4. Schematic diagram for NR image quality assessment.

images having identical content but different levels of distortion. In each experiment, we randomly select 5 groups as the testing dataset, and the other groups as the training dataset. Thus, no content is shared between these two datasets. To illustrate the training and testing method, we show one division of the datasets in Table 1. In the following simulations, the results are all obtained by averaging over 30 independent experiments.

In the experiments, we extract the features of the images as the inputs, and we use the DMOS as the desired outputs. After training, we use the algorithms to attain the scores of the test images, and we calculate the values of the following measures. To show the efficiency of the KIMEL algorithm, we compare it with the KLMS, KVSS-LMS, and three NR IQA algorithms proposed recently (GRNN-based IQA algorithm [31], BIQI [32] and BLIINDS [38]).

In order to compare the performance of these algorithms, we introduce three measures according to VQEG [42]. The first one is the Spearman Rank Order Correlation Coefficient (SROCC), which measures the monotonicity of the quality index. The larger the SROCC value is, the better. The second one is the (Pearson's) Linear Correlation Coefficient (LCC) (the larger, the better). The third one is the Root MSE (RMSE) (the smaller, the better).

In the first simulation, we test the RMSE performance of the KLMS, KVSS-LMS and KIMEL with the varying of their (meta-)learning rates, respectively. In this simulation, for the KVSS-LMS algorithm, we set $\alpha=0.997$, $\beta=0.9$, $\mu_{\max}=0.5$, $\mu_{\min}=0.05$. The KVSS-LMS with these parametric values is found to work well in this application. For the KIMEL algorithm, $\lambda=0.8$ is used. The initial step-size parameters for the KVSS-LMS and the KIMEL are both set as 0.5. In all the three algorithms, the Gaussian kernel with $a^0=10$ is chosen.

The results are presented in Fig. 5. From Fig. 5, we see that the choices of the γ and θ values in broad ranges have little influence on the performance of the KVSS-LMS and the KIMEL algorithms. The KIMEL outperforms the KLMS and the KVSS-LMS algorithms in terms of RMSE.

In the second simulation, the parameters of the KVSS-LMS are set as $\alpha=0.997$, $\beta=0.9$, $\gamma=1$, $\mu_{\max}=0.5$, $\mu_{\min}=0.05$ and the parameters of the KIMEL are set as $\theta=0.1$, $\lambda=0.8$. The step-size parameter of the KLMS is set as 0.2 and the spread parameter of the GRNN for fitting is set as 0.04. The purpose of these parameter setting is to make the algorithms have respectively excellent performance in NR IQA. Using the trained models, image quality is assessed, yielding the averaged results shown in Table 2.

From Table 2, we see that the KIMEL outperforms the KLMS and the KVSS-LMS algorithms in terms of both the SROCC, LCC and RMSE measures. Among the kernel algorithms and other NR IQA algorithms (GRNN-based IQA algorithm, BIQI and BLIIND indices), the KIMEL algorithm has the best overall performance.

Table 1
Image categories for different datasets.

Data	Image categories
Training Dataset	bikes, house, paintedhouse, sailing1, statue, dancers caps, cemetery, manfishing, lighthouse, sailing4, coinsinfountian carnivaldolls, monarch, studentsculpture, ocean, parrots, sailing2 womanhat, flowersonih35, lighthouse2, rapids, building2, plane
Testing Dataset	churchandcapitol, buildings, stream, woman, sailing3

6. Conclusion

In this paper, we proposed a kernel incremental meta-learning algorithm named KIMEL, which was derived by

Table 2
NR IQA Results on LIVE IQA Database.

Method	SROCC	LCC	RMSE
KIMEL	0.8407	0.8439	8.7457
KVSS-LMS	0.8294	0.8281	9.5088
KLMS	0.8184	0.8197	9.7040
GRNN	0.8297	0.8301	8.8900
BIQI [32]	0.8195	0.8205	15.6223
BLIINDS [38]	0.7996	N/A	N/A

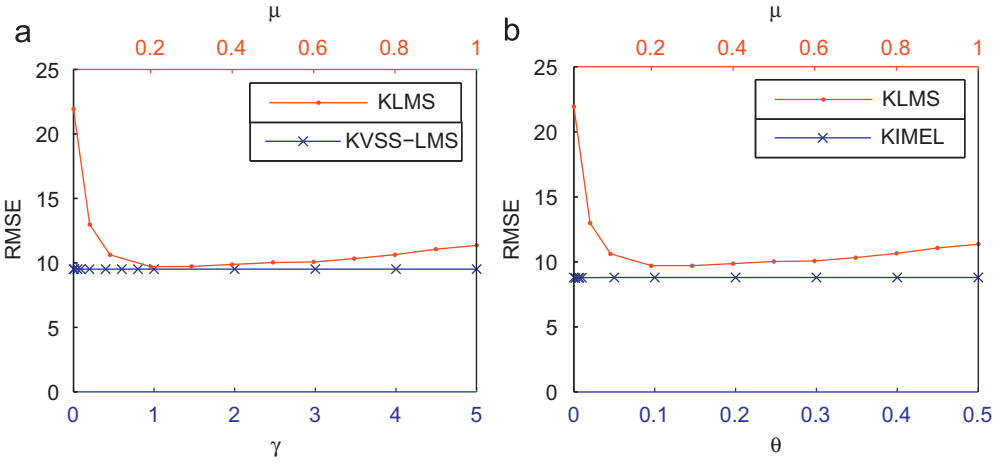

Fig. 5. (a) The influence of different γ on the RMSE performance of the KVSS-LMS and the KLMS and (b) the influence of different θ on the RMSE performance of the KIMEL and the KLMS.

utilizing the gradient descent method. Detailed convergence analyses were presented. One of the advantages of the KIMEL algorithm is that it has only two parameters, the metalearning rate θ and the forgetting factor λ, which have quite wide ranges for choice to produce excellent performance. Thus in using the KIMEL algorithm, we do not need to choose the parameters scrupulously, which is a quite tedious task in using some other algorithms. The proposed algorithm was applied successfully to the nonlinear channel equalization and the NR IQA problem. Experimental results showed that the performance of the KIMEL algorithm is superior to that of the competing methods.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61171153 & 61101045), the Foundation for the Author of National Excellent Doctoral Dissertation of PR China (Grant No. 2007B42), the Scientific Research Foundation for the Returned Overseas Chinese Scholars, the Zhejiang Provincial Natural Science Foundation of China (Grant No. LR12F01001), the National Basic Research Program of China (Grant No. 2009CB320801), and Open Research Grants of the Information Processing and Automation Technology Prior Discipline of Zhejiang Province.

References

[1] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Network 1 (4) (1988) 295–307.
[2] R.S. Sutton, Adapting bias by gradient descent: an incremental version of delta-bar-delta, in: Proceedings of the Tenth National Conference on Artificial Intelligence, MIT Press, 1992, pp. 171–176.
[3] N. Schweighofer, M.A. Arbib, A model of cerebellar metaplasticity, Learning & Memory 4 (5) (1998) 421–428.
[4] W. Liu, P. Pokharel, J.C. Príncipe, The kernel least mean square algorithm, IEEE Transactions on Signal Processing 56 (2) (2008) 543–554.
[5] W. Liu, J.C. Príncipe, S. Haykin, Kernel Adaptive Filtering: A Comprehensive Introduction, John Wiley & Sons, Hoboken, 2010.
[6] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, 2000.
[7] E. Walach, B. Widrow, The least mean fourth (LMF) adaptive algorithm and its family, IEEE Transactions on Information Theory 30 (2) (1984) 275–283.
[8] B. Widrow, P.E. Mantey, L.J. Griffiths, B.B. Goode, Adaptive antenna systems, IEEE Transactions on Information Theory 55 (12) (1967) 2143–2159.
[9] N. Aronszajn, Theory of reproducing kernels, Transactions of the American Mathematical Society 68 (3) (1950) 337–404.
[10] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (2) (1998) 121–167.
[11] S. Haykin, Adaptive Filter Theory, 4th ed. Prentice-Hall, Englewood Cliffs, NJ, 2002.
[12] B. Widrow, J.M. McCool, M.G. Larimore, C.R. Johnson Jr., Stationary and nonstationary learning characteristics of the LMS adaptive filter, Proceedings of the IEEE 64 (8) (1976) 1151–1162.
[13] R.H. Kwong, E.W.J. Johnston, A variable step size LMS algorithm, IEEE Transactions on Signal Processing 40 (7) (1992) 1633–1642.
[14] T. Aboulnasr, K. Mayyas, A robust variable step-size LMS-type algorithm: analysis and simulations, IEEE Transactions on Signal Processing 45 (3) (1997) 631–639.
[15] H.C. Shin, A.H. Sayed, W.J. Song, Variable step-size NLMS and affine projection algorithms, IEEE Signal Processing Letters 11 (2) (2004) 132–135.
[16] J.K. Hwang, Y.P. Li, Variable step-size LMS algorithm with a gradient-based weighted average, IEEE Signal Processing Letters 16 (12) (2009) 1043–1046.
[17] A. Hyvärinen, E. Oja, Independent component analysis: algorithms and applications, Neural Networks 13 (4–5) (2000) 411–430.
[18] D. Tao, X. Tang, X. Li, Y. Rui, Direct kernel biased discriminant analysis: a content-based image retrieval relevance feedback algorithm, IEEE Transactions on Multimedia 8 (4) (2006) 716–727.
[19] Z. Wang, A.C. Bovik, A universal image quality index, IEEE Signal Processing Letters 43 (12) (2002) 81–84.
[20] J.A. Saghri, P.S. Cheatham, A. Habibi, Image quality measure based on human visual system model, Optical Engineering 28 (7) (1989) 813–818.
[21] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Transactions on Signal Processing 13 (4) (2004) 600–612.
[22] Z. Wang, E.P. Simoncelli, Reduced-reference image quality assessment using a wavelet-domain natural image statistic model, in: Human Visual Electronic Imaging X, Proceeding of the SPIE, vol. 5666, 2005, pp. 149–159.
[23] X. Gao, W. Lu, D. Tao, X. Li, Image quality assessment based on multiscale geometric analysis, IEEE Transactions on Image Processing 18 (7) (2009) 1409–1423.
[24] D. Tao, X. Li, W. Lu, X. Gao, Reduce-reference IQA in contourlet domain, IEEE Transactions on Systems, Man, and Cybernetics, Part B 39 (6) (2009) 1623–1627.
[25] X. Li, D. Tao, X. Gao, W. Lu, A natural image quality evaluation metric, Signal Processing 89 (4) (2009) 548–555.
[26] H.R. Sheikh, A.C. Bovik, L. Cormack, No-reference quality assessment using natural scene statistics: JEPG2000, IEEE Transactions on Image Processing 14 (11) (2005) 1918–1927.
[27] Z. Wang, H. R. Sheikh, A. C.Bovik, No-reference perceptual quality assessment of JEPG compressed images, in: Proceedings of the IEEE International Conference on Image Processing, vol. 1, 2002, pp. I-477–I-480.
[28] T. Brandão, M.P. Queluz, No-reference image quality assessment based on DCT domain statistics, Signal Processing 88 (4) (2008) 822–833.
[29] P. Gastaldo, R. Zunino, I. Heynderickx, E. Vicario, Objective quality assessment of displayed images by using neural networks, Signal Processing: Image Communication 20 (7) (2005) 643–661.
[30] J. Zhang, T.M. Le, S.H. Ong, T.Q. Nguyen, No-reference image quality assessment using structural activity, Signal Processing 91 (11) (2011) 2575–2588.
[31] C.F. Li, A.C. Bovik, X.J. Wu, Blind image quality assessment using a general regression neural network, IEEE Transactions on Neural Networks 22 (5) (2011) 793–799.
[32] A.K. Moorthy, A.C. Bovik, A two-step framework for constructing blind image quality indices, IEEE Signal Processing Letters 17 (5) (2010) 513–516.
[33] D. Tao, X. Li, X. Wu, S.J. Maybank, Geometric mean for subspace selection, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2) (2009) 260–274.
[34] N. Guan, D. Tao, Z. Luo, B. Yuan, Online nonnegative matrix factorization with robust stochastic approximation, IEEE Transactions on Neural Networks and Learning Systems 23 (7) (2012) 1087–1099.
[35] N. Guan, D. Tao, Z. Luo, B. Yuan, NeNMF: an optimal gradient method for nonnegative matrix factorization, IEEE Transactions on Signal Processing 60 (6) (2012) 2882–2898.
[36] N. Guan, D. Tao, Z. Luo, B. Yuan, Non-negative patch alignment framework, IEEE Transactions on Neural Networks 22 (8) (2011) 1218–1230.
[37] S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd ed. Prentice-Hall, 1999.
[38] M.A. Saad, A.C. Bovik, C. Charrier, A DCT statistics-based blind image quality index, IEEE Signal Processing Letters 17 (6) (2010) 583–586.
[39] M.C. Morrone, R.A. Owens, Feature detection from local energy, Pattern Recognition Letters 6 (5) (1987) 303–313.
[40] P. Kovesi, Image features from phase congruency, International Journal of Computer Vision 1 (3) (1999) 1–26.
[41] H.R. Sheikh, Z. Wang, L.K. Cormack, A.C. Bovik, LIVE Image Quality Asssessment Database, Release 2, 2006 [Online]. Available: 〈http://live.ece.utexas.edu/research/quality〉.
[42] VQEG, 2000. Final Report from the Video Quality Experts Group on the Validation of Objective Models of Video Quality Assessment [Online]. Available: 〈http://www.vqeg.org/〉.