Transfer Learning in Adaptive Filters: The Nearest Instance Centroid-Estimation Kernel Least-Mean-Square Algorithm

Kan Li, *Member, IEEE*, and José C. Príncipe, *Fellow, IEEE*

***Abstract*—We propose a novel nearest-neighbors approach to organize and curb the growth of radial basis function network in kernel adaptive filtering (KAF). The nearest-instance-centroid-estimation (NICE) kernel least-mean-square (KLMS) algorithm provides an appropriate time-space tradeoff with good performance. Its centers in the input/feature space are organized by quasi-orthogonal regions for greatly simplified filter evaluation. Instead of using all centers to evaluate/update the function approximation at every new point, a linear search among the iteratively-updated centroids determines the partial function to be used, naturally forming locally-supported partial functionals. Under this framework, partial functionals that compose the adaptive filter are quickly stored/retrieved based on input, each corresponding to a specialized "spatial-band" subfilter. The filter evaluation becomes the update of one of the subfilters, creating a content addressable filter bank (CAFB). This CAFB is incrementally updated for new signal applications with mild constraints, always using the past-learned partial filter sums, opening the door for transfer learning and significant efficiency for new data scenarios, avoiding training from scratch as have been done since the invention of adaptive filtering. Using energy conservation relation, we show the sufficient condition for mean square convergence of the NICE-KLMS algorithm and establish the upper and lower bounds of steady-state excess-mean-square-error (EMSE). Simulations on chaotic time-series prediction demonstrate similar levels of accuracy as existing methods, but with much faster computation involving fewer input samples. Simulations on transfer learning using both synthetic and real-world data demonstrate that NICE CAFB can leverage previously learned knowledge to related task or domain.**

***Index Terms*—Adaptive filter, kernel methods, kernel adaptive filtering (KAF), kernel least mean square (KLMS), least mean square, mean square convergence, spatial clustering, transfer learning, vector quantization.**

I. Introduction

KERNEL methods [1], such as support vector machine (SVM) [2], kernel principal component analysis (KPCA) [3], and Gaussian process (GP) [4], create a powerful unifying framework for classification, clustering, and regression, with many important applications in machine learning, signal processing, and biomedical engineering. In particular, the theory of adaptive signal processing is greatly enhanced through the integration of the theory of reproducing kernel Hilbert space (RKHS). By performing classical linear methods in a potentially infinite-dimensional feature space, kernel adaptive filtering (KAF) [5] removes the limitations of the linear model to provide general nonlinear solutions in the original input space. It bridges the gap between adaptive signal processing and feedforward artificial neural networks (ANNs), combining the universal approximation property of neural networks and the simple convex optimization of linear adaptive filters.

Since its introduction, KAF has rapidly gained traction in the scientific community, thanks to its usefulness coupled with simplicity. Despite its relative youth, KAF is already well-established in the literature for solving online nonlinear system identification. The kernel least-mean-square (KLMS) algorithm [6] is the simplest feedforward kernel method in the family of kernel adaptive filters. It can be viewed as a growing single-layer neural network, i.e., a finite impulse response (FIR) filter, trained using the LMS algorithm in the RKHS. Other well-known KAF algorithms include the kernel affine projection algorithms (KAPA) [7], kernel recursive least squares (KRLS) [8], and the extended kernel recursive least squares (EX-KRLS) algorithm [9]. While most research has focused on time-delayed feedforward implementations of kernel methods, recently, a recurrent formulation has been developed to solve nonlinear problems involving non-stationary dynamics. The kernel adaptive autoregressive-moving-average (KAARMA) algorithm [10] achieves the appropriate memory depth via internal states, by feeding back some or all of the outputs through time-delay units, at the input. As a result, the input and output are no longer independent stationary vectors, but correlated temporal sequences.

A major bottleneck of KAF algorithms is that computation scales with number of samples. When the reproducing kernel is Gaussian, kernel adaptive filters grow linearly like radial basis function (RBF) networks, which poses significant time-space complexity issues for continuous online adaptation. To address this issue, a variety of sparsification and quantization techniques have been proposed to curb the network growth [11]. In batch modes, sparsification has been addressed by pruning

Manuscript received January 19, 2017; revised June 21, 2017 and August 13, 2017; accepted August 21, 2017. Date of publication September 14, 2017; date of current version October 27, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Masahiro Yukawa. This work was supported by DARPA under Contracts N66001-10-C-2008 and N66001-15-1-4054. *(Corresponding author: Kan Li.)*

The authors are with the Computational NeuroEngineering Laboratory, University of Florida, Gainesville, FL 32611 USA (e-mail: likan@ufl.edu; principe@cnel.ufl.edu).

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

Digital Object Identifier 10.1109/TSP.2017.2752695

[12], [13] and fixed-size approaches [14], [15]. The focus in this paper is on online adaptive methods. Existing online sample evaluation and selection criteria include the approximate linear dependency (ALD) [8], novelty [16], prediction variance [17], surprise [18], and coherence [19]. Recent development include the dictionary-refining kernel adaptive filter using iterative shrinkage/projection [20]. The central theme has been to form a compact structure by either eliminating redundant information or minimizing information loss.

One of the most successful method to date, because of simplicity and information preservation, is the vector quantization (VQ) technique introduced in the quantized KLMS (QKLMS) algorithm [21], which was shown to outperform pruning techniques using the novelty surprise criteria, ALD, and prediction variance criteria. Rather than discarding the information associated with redundant data points, VQ updates the coefficients locally within a responsive domain. In practice, only the coefficient of its nearest neighbor is updated. A modified version (M-QKLMS) was introduced in [22] by computing the exact gradient when performing the VQ coefficient update. Nevertheless, these methods require the participation of all dictionary centers, to evaluate or update the function approximation at any new data sample.

A. Orthogonal Decomposition Using Exponentially Decaying Kernel

In this paper, we introduce the concept of a simple instance-based learning data-structure that self organizes essential data points. The nearest-instance-centroid-estimation (NICE) algorithm is complementary to existing sparsification and VQ techniques. Whereas the others bound the network size from below, by eliminating redundant basis function centers, NICE bounds the network structure from above, by ignoring centers outside a certain responsive domain. NICE divides the growing sum that defines the filter functional into partial sums (subfilters) that have tight support, i.e., that have nonzero output only in a subregion of the RKHS, naturally forming a compactly-supported reproducing kernel functional. The concept is supported by the fact that a Gaussian function has exponential decay to zero, therefore, if the samples are organized in sufficiently distant clusters, each is approximately orthogonal to the others.

Formally, for a function approximation of the form

$$\hat{f} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{u}_i, \cdot) \tag{1}$$

where the approximating function $\hat{f}$ is represented as a sum of N Gaussian functions ϕ, each associated by a different center $\mathbf{u}_i$, and weighted by a corresponding coefficient α_i. Although the Gaussian function has nonzero values over the full space, our computation has finite precision, so the Gaussian tails are effectively zero. Theoretically, this means, we can project $\hat{f}$ onto the subspace defined by N'

$$\mathrm{span}\{\phi(\mathbf{u}_\mathrm{j}, \cdot) : 1 \leq \mathrm{j} \leq \mathrm{N}'; \text{ and } \mathrm{N}' < \mathrm{N}\} \tag{2}$$

obtaining $\hat{f}_s$ (component in the subspace) and $\hat{f}_\perp$ (component perpendicular to the subspace), i.e.,

$$\hat{f} = \hat{f}_s + \hat{f}_\perp. \tag{3}$$

Using this decomposition, we can partition the basis functions into m orthogonal sets (at the machine precision), i.e.,

$$\begin{aligned}\hat{f}(\mathbf{u}) &= \sum_{j=1}^{N^{(1)}} \alpha_j^{(1)} \phi(\mathbf{u}_j, \mathbf{u}) + , \cdots , + \sum_{j=1}^{N^{(m)}} \alpha_j^{(m)} \phi(\mathbf{u}_j, \mathbf{u}) \\ &= \sum_{j=1}^{N^*} \alpha_j^* \phi(\mathbf{u}_j, \mathbf{u})\end{aligned} \tag{4}$$

where $\sum_{i=1}^{m} N^{(i)} = N$ and $\langle \phi(\mathbf{u}_j), \phi(\mathbf{u}) \rangle = 0$ for all $j \notin N^*$. For Gaussian functions, from the kernel trick $\langle \phi(\mathbf{u}_j), \phi(\mathbf{u}) \rangle = \phi(\|\mathbf{u}_j - \mathbf{u}\|)$, we can approximate orthogonality using the squared norm to define pseudo-normal bases or neighborhoods, by relaxing the orthogonality constraint to $\langle \phi(\mathbf{u}_j), \phi(\mathbf{u}) \rangle < \epsilon$ or equivalently approximate $N^* \approx \{\mathbf{u}_j : \|\mathbf{u}_j - \mathbf{u}\| < d_\epsilon\}$, where ϵ is an arbitrarily small positive quantity and d_ϵ is the corresponding distance value.

1) Nearest Neighbor Search: Nearest neighbor search is a computationally intensive operation, especially in high dimensional spaces. Data-space partitioning and search data-structure can be utilized, however, on average, naive linear search outperforms space partitioning approaches on higher dimensional spaces, due to the *curse of dimensionality* [23]. The incremental nature of the representer theorem at the core of KAF algorithms allows a very simple solution that is heavily based on instantaneous computations. The key is to compare the current sample with a few representatives of the existing data, rather than every individual sample, and since kernel methods are inherently instance-based learning, there is diminished return for finer data structures. We trade the need to maintain complex search data structures for a sequentially-formed, depth-1 forest with the centroid of each cluster at the roots. The NICE network learns the clusters directly from data, using an intuitive kernel bandwidth metric, and updates their centroid locations through iterative update. To perform an evaluation, a linear search among the centroids determines the local support-of-interest, as shown in Fig. 1. When updating the filter weights, if the distance between the input data and its nearest-neighbor centroid exceeds a predefined threshold, a new cluster is formed with the input as its centroid. To avoid large discontinuities in learning, all the centers or dictionary and corresponding coefficients (prior knowledge) in this nearest-neighbor cluster are copied or transfered into the newly formed cluster, with one exception: they are not used to update the centroid. On the other hand, if the distance is below the predefined threshold, the input data is added to the nearest subfilter dictionary, and a one-step centroid update is performed. Without loss of generality, we apply this spatial clustering approach to the simplest kernel adaptive filter using LMS update, and name it the NICE-KLMS algorithm. To show its complementary property to existing RBF network reduction algorithms, we also introduce the NICE-QKLMS algorithm.

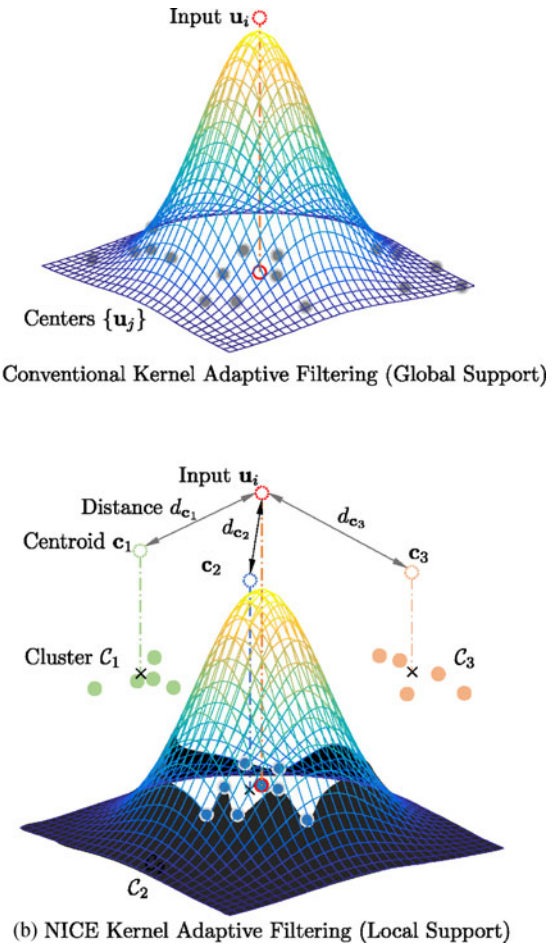


Fig. 1. A NICE network with locally-supported exponentially decaying kernel forming quasi-orthogonal regions or clusters.

The nearest-neighbor search and computation used in NICE is similar to the k-nearest neighbors (k-NN) algorithm [24] only in the sense that the function is approximated locally with respect to its nearest neighbors. However, rather than computing the distances from the test sample to all stored instances and applying weighted averaging, NICE computes the distances between the input and the set of centroids, then applies a standard KAF algorithm on the instances belonging to the nearest-neighbor cluster. Also, the number of centers in each cluster or neighborhood is not predefined, but rather instance-learned directly from data.

Along similar lines, unlike k-means clustering [25], which aims to partition the observations or centers into a fixed number of clusters with each center belonging to the cluster with the nearest mean, over several epochs, the clusters in NICE are formed instantaneously, using only the predefined distance threshold.

Compared with standard compactly-supported kernels [26] and efficient over-estimation method [27], there is no fixed cut-off distance or range on all possible pairs of centers. The concept of a cut-off is only loosely associated with the minimum centroid distance: NICE-KLMS uses a finite subset of centers or local support rather than using a compactly-supported kernel, e.g., a truncated kernel. Simple thresholding technique used to sparsify an RBF kernel by setting a cut-off distance produces a sparse Gram matrix, but often destroys its positive definiteness. Furthermore, with knowledge transfer, in which out-of-range but close-by centers are copied to form the initial dictionary of a new cluster, NICE evaluations can extend beyond the neighborhood defined by the minimum centroid distance.

By partitioning the centers into distinct quasi-orthogonal regions, each cluster of NICE can be thought of as a separate filter or subfilter, specializing in different parts of the input/feature space or patterns. From this perspective, the NICE framework becomes a content addressable filter bank (CAFB). Instead of frequency bands, the filters are organized by amplitude bands. This CAFB can be incrementally updated for more and more new applications, always using the past-learned filters, opening the door for transfer learning and much more efficient training for new data scenarios, avoiding training from scratch as we have been doing since the invention of adaptive filtering.

Compared with multiple [28], [29] and mixture kernel learning [30], NICE-KLMS uses a single kernel (fixed RKHS) across filters. The appropriate filter (set of weights) is selected based on the minimum centroid distance. In this perspective, the NICE-KLMS can be viewed as a single-kernel multiple- or mixture-filter algorithm. In terms of time-space complexity, instead of running multiple learning algorithms in parallel, as is the case in the mixture model, only one filter is updated by NICE-KLMS at any given time step.

Compared to local-structure based KAF, such as the fixed-budget (FB) QKLMS [31], the network size of NICE-QKLMS is not determined *a priori*, but rather learned directly from the complexity or dynamic range of the data. The minimum description length (MDL) criterion can be used to adapt the network size [32], rather than a fixed constant, however it depends on prior knowledge of the locally stationary environment or window size. The only free parameter in NICE, the centroid distance threshold, is conditionally independent of the data, given the appropriate kernel parameter. Since it relates directly to the kernel bandwidth and the shape of the Gaussian is well-understood, it can be set very intuitively. In addition, the two major drawbacks of the existing algorithms are knowledge retention and computational complexity. NICE does not throw away previously learned structures, but rather naturally tucks them away for future use. When the environment changes back to a previous state, QKLMS-FB or QKLM-MDL has no inherent mechanism for recall and has to relearn the structure from scratch. The centroid computation is also significantly more simple to compute than the respective significance measures, e.g., MDL. Furthermore, the NICE paradigm is complementary to most network reduction algorithms and can be used in conjunction with significance measures.

The rest of this paper is organized as follows. In Section II, we provide a brief overview of the KLMS algorithm, then introduce the novel NICE-KLMS. We present the mean square convergence analysis for NICE-KLMS using the energy conservation relation in section III. Performance of the proposed algorithm is evaluated in Section IV, with special emphasis on the associative filter storage property of the CAFB framework. Section V concludes the paper, with future research.

II. Nearest-Instance-Centroid-Estimation Kernel Least-Mean-Square Algorithm

First, we briefly discuss the KLMS algorithm, then introduce the NICE extension for KLMS and QKLMS. In machine learning, supervised learning can be grouped into two broad categories: classification and regression. For a set of N data points $\mathcal{D} = \{\mathbf{u}_i, y_i\}_{i=1}^N$, the desired output y is either categorical variables (e.g., $y \in \{-1, +1\}$), in the case of binary classification, or real numbers (e.g., $y \in \mathbb{R}$) for the task of regression or interpolation, where $\mathbf{X}_1^N \triangleq \{\mathbf{u}_i\}_{i=1}^N$ is the set of M-dimensional input vectors, i.e., $\mathbf{u}_i \in \mathbb{R}^M$, and $\mathbf{y}_1^N \triangleq \{y_i\}_{i=1}^N$ is the corresponding set of desired vectors or observations. In this paper, we will focus on the latter problem, although the same approach can be used for classification. The task is to infer the underlying function $y = f(\mathbf{u})$ from the given data $\mathcal{D} = \{\mathbf{X}_1^N, \mathbf{y}_1^N\}$ and predict its value, or the value of a new observation y', for a new input vector $\mathbf{u}'$. Note that the desired data may be noisy in nature, i.e., $y_i = f(\mathbf{u}_i) + v_i$, where v_i is the noise at time i, which we assume to be independent and identically distributed (i.i.d.) Gaussian random variable with zero-mean and unit-variance, i.e., $V \sim \mathcal{N}(0, 1)$.

For a parametric approach or weight-space view to regression, the estimated latent function $\hat{f}(\mathbf{u})$ is expressed in terms of a parameters vector or weights $\mathbf{w}$. In the standard linear form

$$\hat{f}(\mathbf{u}) = \mathbf{w}^\intercal \mathbf{u}. \tag{5}$$

To overcome the limited expressiveness of this model, we can project the M-dimensional input vector $\mathbf{u} \in \mathbb{U} \subseteq \mathbb{R}^M$ (where $\mathbb{U}$ is a compact input domain in $\mathbb{R}^M$) into a potentially infinite-dimensional feature space $\mathbb{F}$. Define a $\mathbb{U} \to \mathbb{F}$ mapping $\varphi(\mathbf{u})$, the parametric model (5) becomes

$$\hat{f}(\mathbf{u}) = \mathbf{\Omega}^\intercal \varphi(\mathbf{u}) \tag{6}$$

where $\mathbf{\Omega}$ is the weight vector in the feature space.

Using the Representer Theorem [33] and the "kernel trick", (6) can be expressed as

$$\hat{f}(\mathbf{u}) = \sum_{i=1}^{N} \alpha_i \mathcal{K}(\mathbf{u}_i, \mathbf{u}) \tag{7}$$

where $\mathcal{K}(\mathbf{u}, \mathbf{u}')$ is a Mercer kernel, corresponding to the inner product $\langle \varphi(\mathbf{u}), \varphi(\mathbf{u}') \rangle$, and N is the number of basis functions or training samples. Note that $\mathbb{F}$ is equivalent to the reproducing kernel Hilbert spaces (RKHS) induced by the kernel if we identify $\varphi(\mathbf{u}) = \mathcal{K}(\mathbf{u}, \cdot)$. The most commonly used kernel is the Gaussian kernel

$$\mathcal{K}_a(\mathbf{u}, \mathbf{u}') = \exp\left(-a\|\mathbf{u} - \mathbf{u}'\|^2\right) \tag{8}$$

where $a > 0$ is the kernel parameter. Without loss of generality, we focus on the kernel least-mean-square algorithm [6], which is the simplest KAF algorithm.

The learning rule for the KLMS algorithm in the feature space follows the classical linear adaptive filtering algorithm, the LMS

$$\begin{cases} \mathbf{\Omega}_0 = \mathbf{0} \\ e_i = y_i - \langle \mathbf{\Omega}_{i-1}, \varphi(\mathbf{u}_i) \rangle \\ \mathbf{\Omega}_i = \mathbf{\Omega}_{i-1} + \eta e_i \varphi(\mathbf{u}_i) \end{cases} \tag{9}$$

which, in the original input space, becomes

$$\begin{cases} \hat{f}_0 = 0 \\ e_i = y_i - \hat{f}_{i-1}(\mathbf{u}_i) \\ \hat{f}_i = \hat{f}_{i-1} + \eta e_i \mathcal{K}(\mathbf{u}_i, \cdot) \end{cases} \tag{10}$$

where e_i is the prediction error in the i-th time step, η is the learning rate or step-size, and f_i denotes the learned mapping at iteration i. Using KLMS, we can estimate the mean of y with linear per-iteration computational complexity $O(N)$, making it an attractive online algorithm.

A. Nearest Instance Centroid Estimation

As described in the previous section, the NICE algorithm is developed under the framework of subspace decomposition by organizing new sample points into existing clusters (quasi-orthogonal regions) or forming new ones based on the minimum centroid distance $d_{\min}^{(c)}$ and the threshold distance d_c. For continuous online adaptation of the KLMS algorithm, we use the first data sample to initialize a cluster, which also serve as its centroid and the weight of the KAF associated with the first cluster. For each subsequent data point, the minimum centroid distance is computed, resulting in two types of operations:

1) *Cluster:* If the minimum centroid distance is less than the predefined threshold, i.e., $d_{\min}^{(c)} < d_c$, the sample is assigned to its nearest-neighbor cluster. It is then used to update the corresponding filter's weights and the centroid location.
2) *Split:* Otherwise, the sample is used to form a new cluster, its centroid, and the corresponding weights of a new filter.

Clearly, the *Cluster* operation does not change the behavior of the KLMS algorithm, except that instead of updating the weights of a global filter, each new sample is assigned to a local filter or subfilter associated with its nearest cluster or quasi-orthogonal region in the input/feature space. The *Split* operation, on the other hand, carves out a new local region. If we allow the kernel adaptive filter associated with this new cluster to be initialized from scratch with just one sample, it results in a performance discontinuity in time. For continuous learning, this jump becomes insignificant in the long run. However, for short term update, we can avoid these by copying some or all of the weights and centers from its nearest-neighbor cluster (out-of-range in terms of the centroid distance threshold, but spatially, still the closest). This can be viewed as a smoothing procedure. In the worst case, the last cluster will retain a dictionary size equivalent to KLMS (if it is passed from one cluster to the next in its entirety), however, with probability zero. For this to happen, the data would have to be preorganized by cluster and presented to the algorithm in order. An exponentially decaying term λ can be used to gradually diminish the effects of the

Algorithm 1: The NICE-KLMS Algorithm

Initialization:

d_c: centroid distance threshold
a: kernel parameter
η: learning rate
$\boldsymbol{\Omega}_1 = \eta y_1 \varphi(\mathbf{u}_1)$: initial weight
$\boldsymbol{\Omega} = \{\boldsymbol{\Omega}_1\}$: set of filters (weights)
$c_1 = \mathbf{u}_1$: centroid for cluster 1
$s_1 = 1$: effective size of cluster 1 for centroid update
$\mathcal{C}_1 = \{\mathbf{u}_1\}$: center of cluster 1
$\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1\}$: set of clusters

Computation:

while $\{\boldsymbol{u}_i, y_i\}$ *available* **do**
  Compute minimum centroid distance
  $d_{\min}^{(c)} = \min_{1 \le j \le |\boldsymbol{\mathcal{C}}|} \|\mathbf{u}_i - c_j\|^2$
  Select nearest-neighbor cluster
  $j^* = \arg\min_{1 \le j \le |\boldsymbol{\mathcal{C}}|} \|\mathbf{u}_i - c_j\|^2$
  Compute the output of the j^*-th filter, then error
  $\hat{y}_i = \sum_{\ell=1}^{|\mathcal{C}_{j^*}|} \alpha_{j^*}^{(\ell)} \mathcal{K}_a(\mathcal{C}_{j^*}^{(\ell)}, \mathbf{u}_i)$
  $e_i = y_i - \hat{y}_i$
  Nearest-Instance-Centroid-Estimation:
  if $d_{\min}^{(c)} < d_c$ **then**
    Update the weights of j^*-th filter
    $\boldsymbol{\Omega}_{j^*} = \boldsymbol{\Omega}_{j^*} + \eta e_i \varphi(\mathbf{u}_i)$
    Update cluster j^*
    $\mathcal{C}_{j^*} = \{\mathcal{C}_{j^*}, \mathbf{u}_i\}$
    Update cluster j^* centroid, then effective size
    $c_{j^*} = \frac{s_{j^*} c_{j^*} + \mathbf{u}_i}{s_{j^*}+1}$
    $s_{j^*} = s_{j^*} + 1$
  else
    Form new cluster/filter
    $\mathcal{C}_{|\boldsymbol{\mathcal{C}}|+1} = \{\mathbf{u}_i\}$: new cluster
    $c_{|\boldsymbol{\mathcal{C}}|+1} = \mathbf{u}_i$: centorid
    $s_{|\boldsymbol{\mathcal{C}}|+1} = 1$: effective size
    $\boldsymbol{\mathcal{C}} = \{\boldsymbol{\mathcal{C}}, \mathcal{C}_{|\boldsymbol{\mathcal{C}}|+1}\}$
    Knowledge Transfer
    $\boldsymbol{\Omega}_{|\boldsymbol{\mathcal{C}}|+1} = \boldsymbol{\Omega}_{j^*} + \eta e_i \varphi(\mathbf{u}_i)$
    Update set of filters
    $\boldsymbol{\Omega} = \{\boldsymbol{\Omega}, \boldsymbol{\Omega}_{|\boldsymbol{\mathcal{C}}|+1}\}$

copied coefficients in that particular part of the space. We can also remove these initial samples when their contributions fall below a certain threshold, as new samples are brought into the cluster. More elaborate schemes such as MDL can be used to further reduce the cluster size. Note that the out-of-range centers associated with these weights will never be used to update the centroid location. Since the centroid is the geometric mean of the vector space, we can update its location easily with a one-step operation using its previous location, the number of existing within-cluster centers, and the new data point.

Since the Gaussian kernel is isotropic, and we know the interval estimation and coverage probability of a normal distribution, for convenience and intuition, we express the NICE centroid distance threshold d_c in terms of the unnormalized standard deviation. The unnormalized Gaussian-kernel standard deviation σ_k is defined with respect to the kernel parameter a in (8) as

$$\sigma_k \triangleq \sqrt{\frac{1}{2a}}. \tag{11}$$

The proposed NICE-KLMS algorithm is summarized in Algorithm 1.

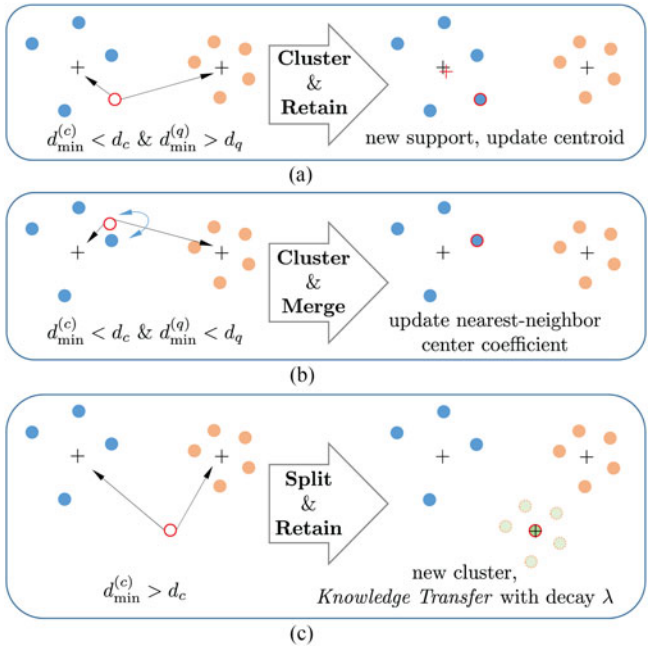


Fig. 2. Three main types of operations in the NICE-QKLMS algorithm.

The NICE-KLMS algorithm behaves identically to KLMS when the number of clusters is fixed at one, i.e., infinite centroid-distance threshold or $d_c = \infty$. In practice, it runs much faster than KLMS, since the number of centers needed per cluster/subfilter is significantly fewer, and on average, the number of clusters (operations need to select the nearest subfilter) is significantly smaller than the average size of individual clusters, i.e., $|\boldsymbol{\mathcal{C}}| \ll |\bar{\mathcal{C}}|$.

B. Vector Quantization

As noted above, the vector quantization technique in QKLMS is complementary to NICE, and can be combined to further reduce the network structure and run-time complexity. Each of the within-cluster centers can be viewed as a mini centroid and compacted using a quantization distance threshold d_q. We present the NICE-QKLMS algorithm in Algorithm 2. Just like KLMS can be viewed as a special case of NICE-KLMS, when the centroid distance threshold is infinity, i.e., $d_c = \infty$, the NICE-KLMS algorithm is a special case of NICE-QKLMS, when the quantization distance threshold is zero, $d_q = 0$. Together, the two thresholds bound the RBF network size from above and below. This generalized approach consists of three types of operations (shown in Fig. 2):

1) *Cluster & Retain:* If the minimum centroid distance is less than the predefined threshold, i.e., $d_{\min}^{(c)} < d_c$, the sample is assigned to its nearest-neighbor cluster. Furthermore, if the minimum quantization distance is greater than the predefined threshold ($d_{\min}^{(q)} > d_q$), computed inside the nearest-neighbor cluster, the data sample is retained and used to update the corresponding subfilter's weights and the centroid location. Note, that $d_{\min}^{(q)}$ can be obtained efficiently during the filtering procedure, and does not require a separate loop.

Algorithm 2: The NICE-(M)-QKLMS Algorithm

Initialization:
d_c: centroid distance threshold
d_q: quantization distance threshold
a: kernel parameter
η: learning rate
$\mathbf{\Omega}_1 = \eta y_1 \varphi(\mathbf{u}_1)$: initial weight
$\boldsymbol{\Omega} = \{\mathbf{\Omega}_1\}$: set of filters (weights)
$c_1 = \mathbf{u}_1$: centroid for cluster 1
$s_1 = 1$: effective size of cluster 1 for centroid update
$\mathcal{C}_1 = \{\mathbf{u}_1\}$: center of cluster 1
$\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1\}$: set of clusters
Computation:
while $\{\boldsymbol{u}_i, y_i\}$ *available* **do**
  Compute minimum centroid distance
  $d_{\min}^{(c)} = \min_{1 \le j \le |\boldsymbol{\mathcal{C}}|} \|\mathbf{u}_i - c_j\|^2$
  Select nearest-neighbor cluster
  $j^* = \arg\min_{1 \le j \le |\boldsymbol{\mathcal{C}}|} \|\mathbf{u}_i - c_j\|^2$
  Compute the output of the j^*-th filter, then error
  $\hat{y}_i = \sum_{\ell=1}^{|\mathcal{C}_{j^*}|} \alpha_{j^*}^{(\ell)} \mathcal{K}_a(\mathcal{C}_{j^*}^{(\ell)}, \mathbf{u}_i)$
  $e_i = y_i - \hat{y}_i$
  Nearest-Instance-Centroid-Estimation:
  if $d_{\min}^{(c)} < d_c$ **then**
    Use minimum quantization distance from above
    $d_{\min}^{(q)} = \min_{1 \le m \le |\mathcal{C}_{j^*}|} \|\mathbf{u}_i - \mathcal{C}_{j^*}^{(m)}\|^2$
    Select nearest-neighbor center
    $m^* = \arg\min_{1 \le m \le |\mathcal{C}_{j^*}|} \|\mathbf{u}_i - \mathcal{C}_{j^*}^{(m)}\|^2$
    Perform Vector Quantization:
    if $d_{\min}^{(q)} < d_q$ **then**
      Update nearest-neighbor coefficient
      $\alpha_{j^*}^{(m^*)} = \alpha_{j^*}^{(m^*)} + \eta e_i$ (QKLMS) or
      $\alpha_{j^*}^{(m^*)} = \alpha_{j^*}^{(m^*)} + \eta e_i \mathcal{K}_a(\mathbf{u}_{j^*}^{(m^*)}, \mathbf{u}_i)$ (M-QKLMS)
    else
      Update the weights of j^*-th filter
      $\mathbf{\Omega}_{j^*} = \mathbf{\Omega}_{j^*} + \eta e_i \varphi(\mathbf{u}_i)$
      Update cluster j^*
      $\mathcal{C}_{j^*} = \{\mathcal{C}_{j^*}, \mathbf{u}_i\}$
      Update cluster j^* centroid, then effective size
      $c_{j^*} = \frac{s_{j^*} c_{j^*} + \mathbf{u}_i}{s_{j^*}+1}$
      $s_{j^*} = s_{j^*} + 1$
  else
    Form new cluster/filter
    $\mathcal{C}_{|\boldsymbol{\mathcal{C}}|+1} = \{\mathbf{u}_i\}$: new cluster
    $c_{|\boldsymbol{\mathcal{C}}|+1} = \mathbf{u}_i$: centorid
    $s_{|\boldsymbol{\mathcal{C}}|+1} = 1$: effective size
    $\boldsymbol{\mathcal{C}} = \{\boldsymbol{\mathcal{C}}, \mathcal{C}_{|\boldsymbol{\mathcal{C}}|+1}\}$
    Knowledge Transfer
    $\mathbf{\Omega}_{|\boldsymbol{\mathcal{C}}|+1} = \mathbf{\Omega}_{j^*} + \eta e_i \varphi(\mathbf{u}_i)$
    Update set of filters
    $\boldsymbol{\Omega} = \{\boldsymbol{\Omega}, \mathbf{\Omega}_{|\boldsymbol{\mathcal{C}}|+1}\}$

2) *Cluster & Merge:* From the above operation, if the newly assigned data sample's quantization distance is smaller than the threshold ($d_{\min}^{(q)} < d_q$), the sample is effective merged inside its nearest-neighbor center. Cluster size and centroid location remains the same. Only the coefficient of its nearest neighbor is updated, using the subfilter-output error.
3) *Split & Retain:* Otherwise, if $d_{\min}^{(c)} > d_c$, the new sample is used to form a new cluster, its centroid, and the corresponding weights of a new subfilter. To avoid short-term learning discontinuity, the nearest-neighbor cluster weights are carried over. The worst-case dictionary size does not necessarily scale linearly with the number of clusters, since any additional center has to always meet the global quantization constraint. The worst-case computational complexity of NICE-QKLMS is the same as QKLMS, since the NICE computation is considered constant (with the number of clusters $\ll$ centers). However, in practice, NICE performs much faster, since the worst-case scenario occurs with probability zero. Furthermore, an exponentially decaying term λ can be used to gradually diminish the effects and presence of the copied coefficients.

In the case that the minimum VQ distance is less than the predefined threshold (*Cluster & Merge*), QKLMS assumes the current input is a direct copy of its nearest neighbor, thus only updating the coefficients with the instantaneous error. This is an approximation with practical value. An more appropriate treatment is to update the coefficient using convex optimization or gradient descent, in this case. The exact error gradient with respect to the closest-neighbor coefficient α_{i^*} can be computed as

$$\frac{\partial \varepsilon_i}{\partial \alpha_{i^*}} = \frac{\partial e_i^2}{2 \partial \alpha_{i^*}} = -e_i \frac{\partial y_i}{\partial \alpha_{i^*}} = -e_i \mathcal{K}_a(\mathbf{u}_{i^*}, \mathbf{u}_i) \tag{12}$$

where $\varepsilon_i = e_i^2/2$ is the cost function. Clearly, the instantaneous error e_i needs to be scaled by a kernel evaluation between the current input $\mathbf{u}_i$ and its nearest neighbor $\mathbf{u}_{i^*}$. This formulation is termed the modified or M-QKLMS [22]. This option is also reflected in Algorithm 2. Note that it is outside the scope of this paper to compare the merits of existing sparsification and quantization techniques. The emphasis, here, is to show that NICE is complementary to most existing algorithms and can be applied in integration.

III. NICE-QKLMS Mean-Square-Convergence Analysis

Here, we use the energy conservation relation for adaptive filtering [21], [34], [35] to show the sufficient condition for mean square convergence of the NICE-QKLMS algorithm. The upper and lower steady-state excess-mean-square-error bounds are also established. We have to impose here two simplifying hypothesis: the clustering operation that was discussed above is optimal, i.e., no errors in clustering have been introduced, and that the orthogonalization amongst clusters is exact. First, let a general nonlinear model be defined as

$$d_i = f(\mathbf{u}_i) + v_i \tag{13}$$

where d_i is the noisy measurement or desired value, $f(\cdot)$ is the unknown nonlinear mapping, and v_i denotes measurement noise. In this paper, we focus on the following class of kernel adaptive filtering algorithms defined in (6). The universal approximation property [36] states that there exists a vector $\mathbf{\Omega}^*$ such that $f(\cdot) = \mathbf{\Omega}^{*\mathsf{T}} \varphi(\cdot)$. The prediction error

becomes

$$\begin{aligned} e_i &= d_i - \mathbf{\Omega}_{i-1}^{\mathsf{T}}\varphi(\mathbf{u}_i) \\ &= \mathbf{\Omega}^{*\mathsf{T}}\varphi(\mathbf{u}_i) - \mathbf{\Omega}_{i-1}^{\mathsf{T}}\varphi(\mathbf{u}_i) + v_i \\ &= \tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\varphi(\mathbf{u}_i) + v_i \end{aligned} \tag{14}$$

where $\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}} \triangleq \mathbf{\Omega}^{*\mathsf{T}} - \mathbf{\Omega}_{i-1}^{\mathsf{T}}$ is the weight error vector in the functional space $\mathbb{F}$. The steady-state mean-squared-error (MSE) of an adaptive filter is defined as

$$\mathrm{MSE} \triangleq \lim_{i\to\infty} \mathbf{E}[\|e_i\|^2] = \lim_{i\to\infty} \mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\varphi(\mathbf{u}_i) + v_i\|^2] \tag{15}$$

where $\|\cdot\|$ denotes the Euclidean norm. Under the widely-used and often realistic assumption (e.g., [1]–[8])

A.1: The additive noise v_i is zero-mean with variance σ_v^2, independent and identically distributed (i.i.d.), and statistically independent of the input sequence $\varphi(\mathbf{u}_i)$,

the steady-state MSE in (15) reduces to

$$\mathrm{MSE} = \sigma_v^2 + \lim_{i\to\infty} \mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\varphi(\mathbf{u}_i)\|^2]. \tag{16}$$

If we further assume that

A.2: The input vector $\varphi(\mathbf{u}_i)$ is independent of $\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}$, then the MSE expression becomes

$$\mathrm{MSE} = \sigma_v^2 + \lim_{i\to\infty} \mathrm{Tr}(\mathbf{C}_i\mathbf{R}) \tag{17}$$

where $\mathrm{Tr}(\cdot)$ is the trace of a matrix and $\mathbf{C}_i$ is the weight error covariance matrix, i.e., $\mathbf{C}_i \triangleq \mathbf{E}[\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\tilde{\mathbf{\Omega}}_{i-1}]$ and $\mathbf{R} \triangleq \mathbf{E}[\varphi(\mathbf{u}_i)^{\mathsf{T}}\varphi(\mathbf{u}_i)]$.

A. Conservation of Energy for Kernel Adaptive Filtering

First, let's define the *a priori* and *a posteriori* estimation errors, e_i^- and e_i^+ respectively, as

$$e_i^- \triangleq \tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\varphi(\mathbf{u}_i) \tag{18}$$

$$e_i^+ \triangleq \tilde{\mathbf{\Omega}}_{i}^{\mathsf{T}}\varphi(\mathbf{u}_i). \tag{19}$$

Substituting (18) into (14) yields the following relation between the error terms $\{e_i, e_i^-\}$:

$$e_i = e_i^- + v_i. \tag{20}$$

Subtracting the optimal weight $\mathbf{\Omega}^*$ from both sides of the weight update equation, then multiplying both sides by the feature space input $\varphi(\mathbf{u}_i)$, from the right, gives

$$\begin{aligned} \mathbf{\Omega}_i &= \mathbf{\Omega}_{i-1} + \eta e_i\varphi(\mathbf{u}_i) \\ \mathbf{\Omega}_i - \mathbf{\Omega}^* &= \mathbf{\Omega}_{i-1} - \mathbf{\Omega}^* + \eta e_i\varphi(\mathbf{u}_i) \end{aligned} \tag{21}$$

$$\tilde{\mathbf{\Omega}}_i^{\mathsf{T}}\varphi(\mathbf{u}_i) = \tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\varphi(\mathbf{u}_i) - \eta e_i\varphi(\mathbf{u}_i)^{\mathsf{T}}\varphi(\mathbf{u}_i) \tag{22}$$

$$\begin{aligned} e_i^+ &= e_i^- - \eta e_i \\ \eta e_i &= e_i^- - e_i^+ \end{aligned} \tag{23}$$

since $\mathcal{K}(\mathbf{u}_i, \mathbf{u}_i) = \mathcal{K}(\|\mathbf{u}_i - \mathbf{u}_i\|^2) = 1$. Substituting (23) into (21) yields the following weight-error vector update rule

$$\begin{aligned} \mathbf{\Omega}_i - \mathbf{\Omega}^* &= \mathbf{\Omega}_{i-1} - \mathbf{\Omega}^* + (e_i^- - e_i^+)\varphi(\mathbf{u}_i) \\ \tilde{\mathbf{\Omega}}_i &= \tilde{\mathbf{\Omega}}_{i-1} - (e_i^- - e_i^+)\varphi(\mathbf{u}_i). \end{aligned} \tag{24}$$

To evaluate the energy conservation of (24), we square both sides, yielding

$$\begin{aligned} \tilde{\mathbf{\Omega}}_i^{\mathsf{T}}\tilde{\mathbf{\Omega}}_i &= \tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\tilde{\mathbf{\Omega}}_{i-1} + (e_i^- - e_i^+)^2\varphi(\mathbf{u}_i)^{\mathsf{T}}\varphi(\mathbf{u}_i) \\ &\quad - 2(e_i^- - e_i^+)\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\varphi(\mathbf{u}_i) \\ &= \tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\tilde{\mathbf{\Omega}}_{i-1} + \left((e_i^-)^2 - 2e_i^-e_i^+ + (e_i^+)^2\right)\mathcal{K}(\mathbf{u}_i, \mathbf{u}_i) \\ &\quad - 2(e_i^- - e_i^+)e_i^- \\ &= \tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\tilde{\mathbf{\Omega}}_{i-1} - (e_i^-)^2 + (e_i^+)^2 \end{aligned} \tag{25}$$

or, in shorthand notation:

$$\boxed{\|\tilde{\mathbf{\Omega}}_i\|_{\mathbb{F}}^2 + (e_i^-)^2 = \|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2 + (e_i^+)^2} \tag{26}$$

which describes how the energies of the weight-error vectors for two successive time instants $i-1$ and i are related to the energies of the *a priori* and *a posteriori* estimation errors, where $\|\cdot\|_{\mathbb{F}}$ denotes the norm in the feature space $\mathbb{F}$, i.e., $\forall \mathbf{\Omega} \in \mathbb{F}$, $\|\mathbf{\Omega}\|_{\mathbb{F}} \triangleq \sqrt{\mathbf{\Omega}^{\mathsf{T}}\mathbf{\Omega}}$.

B. Steady-State MSE Performance Analysis

In the steady state, the following assumption holds

$$\lim_{i\to\infty} \mathbf{E}[\|\tilde{\mathbf{\Omega}}_i\|_{\mathbb{F}}^2] = \mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2] \tag{27}$$

i.e., the mean square deviation converges to a steady-state value. In the steady state, the effect of the weight-error vector cancels out. Taking the expectation on both sides of (26) yields

$$\mathbf{E}[\|\tilde{\mathbf{\Omega}}_i\|_{\mathbb{F}}^2] + \mathbf{E}[(e_i^-)^2] = \mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2] + \mathbf{E}[(e_i^+)^2]. \tag{28}$$

Substituting the expression for the *a posteriori* estimation error e_i^+ in (23) into the right-hand side of (28) gives

$$\begin{aligned} \mathbf{E}[\|\tilde{\mathbf{\Omega}}_i\|_{\mathbb{F}}^2] + \mathbf{E}[(e_i^-)^2] &= \mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2] + \mathbf{E}[(e_i^- - \eta e_i)^2] \\ \mathbf{E}[\|\tilde{\mathbf{\Omega}}_i\|_{\mathbb{F}}^2] &= \mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2] \\ &\quad - 2\eta\mathbf{E}[e_ie_i^-] + \eta^2\mathbf{E}[e_i^2]. \end{aligned} \tag{29}$$

Clearly, a sufficient condition for mean square convergence is to ensure a monotonic decrease of the weight-error power $\mathbf{E}[\|\tilde{\mathbf{\Omega}}_i\|_{\mathbb{F}}^2]$, i.e.,

$$-2\eta\mathbf{E}[e_ie_i^-] + \eta^2\mathbf{E}[e_i^2] \le 0. \tag{30}$$

Since the step size is lower bounded by 0, from (30) and (20), we have

$$\begin{aligned} 0 < \eta &\leq \frac{2\mathbf{E}[e_i e_i^-]}{\mathbf{E}[e_i^2]} \\ &= \frac{2\mathbf{E}[(e_i^- + v_i)e_i^-]}{\mathbf{E}[(e_i^- + v_i)^2]} \\ &\overset{(a)}{=} \frac{2\mathbf{E}[(e_i^-)^2]}{\mathbf{E}[(e_i^-)^2] + \sigma_v^2} \end{aligned} \tag{31}$$

where equality (a) follows from *A.1*, i.e., the cross-term $\mathbf{E}[v_i e_i^-] = \mathbf{E}[v_i]\mathbf{E}[e_i^-] = 0$. From (31), we obtain the following sufficient condition:

$$\begin{aligned} 0 &< \mathbf{E}[(e_i^-)^2] \\ &= \mathbf{E}[(\tilde{\mathbf{\Omega}}_{i-1}^\intercal \varphi(\mathbf{u}_i))^2] \\ &= \mathbf{E}[\tilde{\mathbf{\Omega}}_{i-1}^\intercal \varphi(\mathbf{u}_i)\varphi^\intercal(\mathbf{u}_i)\tilde{\mathbf{\Omega}}_{i-1}] \\ &\overset{(b)}{=} \mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2] \end{aligned} \tag{32}$$

where (b) follows from the kernel trick. Summarizing the sufficient conditions below,

$$\boxed{\begin{aligned} &\mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2] > 0 \\ &0 < \eta \leq \frac{2\mathbf{E}[(e_i^-)^2]}{\mathbf{E}[(e_i^-)^2] + \sigma_v^2} \end{aligned}} \tag{33}$$

we see that for weight adaptation in $\mathbb{F}$ using the current feature space input $\varphi(\mathbf{u}_i)$, as long as the step size η is appropriately selected according to (31), the NICE-KLMS algorithm converges in identical fashion as KLMS [37].

In practice, the step size can be selected using the framework of the FIR adaptive filtering small-step-size theory [38] applied to KAF (KLMS) [5]:

$$\eta < \frac{1}{\lambda_{\max}} \tag{34}$$

where $\lambda_{\max}$ is the largest eigenvalue of the autocorrelation matrix $\mathbf{R}$ in (17), which can be obtained from the trace of the Gram matrix $\mathbf{G}$, yielding a conservative upper-bound

$$\eta < \frac{N}{\mathrm{Tr}[\mathbf{G}]} = \frac{N}{\sum_{i=1}^N \mathcal{K}(\mathbf{u}_i, \mathbf{u}_i)}. \tag{35}$$

For shift-invariant kernels, i.e., $\mathcal{K}(\mathbf{u}_i, \mathbf{u}_i) = g_0$, the upper-bound becomes $\frac{1}{g_0}$, which is data-independent and can be used as a default value [5].

At steady state, the excess mean-squared error (EMSE) is equivalent to setting the step size η equal to its upper-bound in (31), yielding

$$\lim_{i\to\infty} \mathbf{E}[(e_i^-)^2] = \frac{\eta\sigma_v^2}{(2-\eta)}. \tag{36}$$

For the three operations of NICE-QKLMS, we show how the mean square convergence and steady-state EMSE are affected.

1) Cluster & Retain: Let the *a priori* weight vector for cluster $\mathcal{C}_c$ be denoted by $\mathbf{\Omega}_{i-1}^{(c)}$. Updating the weight vector using the current input $\varphi(\mathbf{u}_i)$ does not change the behavior of the mean square convergence or EMSE.

2) Cluster & Merge: Instead of using the current input $\varphi(\mathbf{u}_i)$ to update the weight vector for cluster c, its nearest within-cluster neighbor $\varphi(\mathbf{u}_q^{(c)})$ is used, i.e., $\mathbf{\Omega}_i = \mathbf{\Omega}_{i-1} + \eta e_i \varphi(\mathbf{u}_q^{(c)})$, where $\mathbf{u}_q^{(c)} = \arg\min \|\mathbf{u}_i - U^{(c)}\|$. This affects the kernel trick used to simplify the expressions throughout the steady-state MSE performance analysis. The simple identity $\mathcal{K}(\varphi(\mathbf{u}_i), \varphi(\mathbf{u}_i)) = 1$ is no longer valid, but rather, the value is bounded by a factor $q > 0$, i.e., $\mathcal{K}(\|\mathbf{u}_i - \mathbf{u}_q^{(c)}\|^2 \leq q) \geq \exp(-aq)$.

This introduces a new energy conservation relation. Substituting the current input with its nearest within-cluster neighbor in (22) gives

$$\begin{aligned} (-\tilde{\mathbf{\Omega}}_i^\intercal)\varphi(\mathbf{u}_i) &= (-\tilde{\mathbf{\Omega}}_{i-1}^\intercal)\varphi(\mathbf{u}_i) + \eta e_i \varphi(\mathbf{u}_q^{(c)})^\intercal \varphi(\mathbf{u}_i) \\ e_i^+ &= e_i^- - \eta e_i \mathcal{K}(\mathbf{u}_q^{(c)}, \mathbf{u}_i) \\ \eta e_i &= \frac{e_i^- - e_i^+}{\mathcal{K}(\mathbf{u}_q^{(c)}, \mathbf{u}_i)}. \end{aligned} \tag{37}$$

Substituting this new expression that relates the three error terms $\{e_i^+, e_i^-, e_i\}$ during the merge update for (23) in (21), the energy conservation relation in (26) becomes

$$\begin{aligned} \|\tilde{\mathbf{\Omega}}_i\|_{\mathbb{F}}^2 &= \|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2 - 2\frac{e_i^- - e_i^+}{\mathcal{K}(\mathbf{u}_q^{(c)}, \mathbf{u}_i)} \tilde{\mathbf{\Omega}}_{i-1}^\intercal \varphi(\mathbf{u}_q^{(c)}) \\ &\quad + \frac{(e_i^- - e_i^+)^2}{\mathcal{K}^2(\mathbf{u}_q^{(c)}, \mathbf{u}_i)} \varphi(\mathbf{u}_q^{(c)})^\intercal \varphi(\mathbf{u}_q^{(c)}) \\ &= \|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2 + \frac{(e_i^+)^2 - (e_i^-)^2}{\mathcal{K}^2(\mathbf{u}_q^{(c)}, \mathbf{u}_i)} \\ &\quad + \underbrace{\frac{2(e_i^+ - e_i^-)\left(\tilde{\mathbf{\Omega}}_{i-1}^\intercal \varphi(\mathbf{u}_q^{(c)})\mathcal{K}(\mathbf{u}_q^{(c)}, \mathbf{u}_i) - e_i^-\right)}{\mathcal{K}^2(\mathbf{u}_q^{(c)}, \mathbf{u}_i)}}_{J_q} \end{aligned} \tag{38}$$

where $J_q \triangleq \frac{2(e_i^+ - e_i^-)\left(\tilde{\mathbf{\Omega}}_{i-1}^\intercal \varphi(\mathbf{u}_q^{(c)})\mathcal{K}(\mathbf{u}_q^{(c)}, \mathbf{u}_i) - e_i^-\right)}{\mathcal{K}^2(\mathbf{u}_q^{(c)}, \mathbf{u}_i)}$ denotes the quantization energy due to the merge operation. It follows that

$$\boxed{\|\tilde{\mathbf{\Omega}}_i\|_{\mathbb{F}}^2 + \frac{(e_i^-)^2}{\mathcal{K}^2(\mathbf{u}_q^{(c)}, \mathbf{u}_i)} = \|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2 + \frac{(e_i^+)^2}{\mathcal{K}^2(\mathbf{u}_q^{(c)}, \mathbf{u}_i)} + J_q.} \tag{39}$$

In the limit as the quantization factor $q \to 0$, i.e., $\lim_{q\to 0} \mathbf{u}_q^{(c)} = \mathbf{u}_i$, the quantization energy $J_q \to 0$ and (39) reduces to (26).

Again, using (37), the sufficient conditions for mean square convergence in (33) become

$$\boxed{\begin{aligned}&\mathbf{E}[\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\mathcal{K}(\mathbf{u}_q^{(c)},\mathbf{u}_i)\tilde{\mathbf{\Omega}}_{i-1}]>0\\&0<\eta\leq\frac{2\mathbf{E}[e_i^-\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\varphi(\mathbf{u}_q^{(c)})]}{\mathbf{E}[(e_i^-)^2]+\sigma_v^2}\end{aligned}} \tag{40}$$

which is satisfied with an appropriately selected step size and a sufficiently small quantization factor q such that $\mathcal{K}(\|\mathbf{u}_q^{(c)}-\mathbf{u}_i\|^2)>0$. The rule-of-thumb in (35) also applies here. It follows that the steady-state EMSE is

$$\begin{aligned}&\eta\left(\lim_{i\to\infty}\mathbf{E}[(e_i^-)^2+\sigma_v^2]\right)\\&=2\lim_{i\to\infty}\mathbf{E}[e_i^-\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\varphi(\mathbf{u}_q^{(c)})]\\&=2\lim_{i\to\infty}\mathbf{E}[e_i^-\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}(\varphi(\mathbf{u}_i)-\varphi(\mathbf{u}_i)+\varphi(\mathbf{u}_q^{(c)}))]\\&=2\lim_{i\to\infty}\left(\mathbf{E}[(e_i^-)^2]+\mathbf{E}[e_i^-\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}(\varphi(\mathbf{u}_q^{(c)})-\varphi(\mathbf{u}_i))]\right)\\&\lim_{i\to\infty}\mathbf{E}[(e_i^-)^2]=\frac{\eta\sigma_v^2-2\lim_{i\to\infty}\mathbf{E}[e_i^-\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}(\varphi(\mathbf{u}_q^{(c)})-\varphi(\mathbf{u}_i))]}{2-\eta}\end{aligned} \tag{41}$$

The expected value in the numerator, on the right-hand side of (41), can be expanded as

$$\begin{aligned}&\mathbf{E}[e_i^-\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}(\varphi(\mathbf{u}_q^{(c)})-\varphi(\mathbf{u}_i))]\\&\quad=\mathbf{E}[\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\varphi(\mathbf{u}_i)\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}(\varphi(\mathbf{u}_q^{(c)})-\varphi(\mathbf{u}_i))]\\&\quad=\mathbf{E}[\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\varphi(\mathbf{u}_i)(\varphi(\mathbf{u}_q^{(c)})-\varphi(\mathbf{u}_i))^{\mathsf{T}}\tilde{\mathbf{\Omega}}_{i-1}]\\&\quad=\mathbf{E}[\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\langle\varphi(\mathbf{u}_i),\varphi(\mathbf{u}_q^{(c)})-\varphi(\mathbf{u}_i)\rangle\tilde{\mathbf{\Omega}}_{i-1}]\\&\quad\overset{(c)}{=}\mathbf{E}[\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\langle\varphi(\mathbf{u}_q^{(c)})-\varphi(\mathbf{u}_i),\varphi(\mathbf{u}_i)\rangle\tilde{\mathbf{\Omega}}_{i-1}]\\&\quad\overset{(d)}{=}\mathbf{E}[\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\left(\langle\varphi(\mathbf{u}_q^{(c)}),\varphi(\mathbf{u}_i)\rangle-\langle\varphi(\mathbf{u}_i),\varphi(\mathbf{u}_i)\rangle\right)\tilde{\mathbf{\Omega}}_{i-1}]\\&\quad\overset{(e)}{=}\mathbf{E}\left[\left(\mathcal{K}(\mathbf{u}_q^{(c)},\mathbf{u}_i)-\mathcal{K}(\mathbf{u}_i,\mathbf{u}_i)\right)\tilde{\mathbf{\Omega}}_{i-1}^{\mathsf{T}}\tilde{\mathbf{\Omega}}_{i-1}\right]\\&\quad\overset{(f)}{=}\left(\mathbf{E}\left[\exp(-a\|\mathbf{u}_i-\mathbf{u}_q^{(c)}\|^2)\right]-1\right)\mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2]\end{aligned} \tag{42}$$

where equalities (c) and (d) follow from the symmetry property and the scaling-and-distributive property of RKHS, respectively, (e) holds because inner products are scalars, and (f) results from *A.2*.

Since the maximum squared distance for the merge operation is determined by the quantization factor q, it follows that (42) is bounded as

$$\begin{aligned}&(\exp(-aq)-1)\mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2]\\&\qquad\leq\left(\mathbf{E}\left[\exp(-a\|\mathbf{u}_i-\mathbf{u}_q^{(c)}\|^2)\right]-1\right)\mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2]\leq 0\end{aligned} \tag{43}$$

where the upper-bound is achieved when the current input is an existing support, i.e., $\mathbf{u}_q^{(c)}=\mathbf{u}_i$.

Substituting (42) and (43) into (41) yields the following bounds for the NICE-QKLMS EMSE:

$$\boxed{\begin{aligned}\frac{\eta\sigma_v^2}{(2-\eta)}&\leq\lim_{i\to\infty}\mathbf{E}[(e_i^-)^2]\\&\leq\frac{\eta\sigma_v^2+2(1-\exp(-aq))\lim_{i\to\infty}\mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2]}{(2-\eta)}.\end{aligned}} \tag{44}$$

Compared to (36), the NICE-KLMS is a special case of NICE-QKLMS. The universal approximation property and the mean square convergence (40) tells us that $\lim_{i\to\infty}\mathbf{E}[\|\tilde{\mathbf{\Omega}}_{i-1}\|_{\mathbb{F}}^2]=0$ when i approaches infinity and the quantization factor is zero, i.e., given infinite training data and no quantization. Note that this is the average asymptotic behavior for the assemble of KLMS filters; individual performance using finite training data may vary.

3) Split & Retain: Creating a new cluster c' and updating the new weight vector $\mathbf{\Omega}_{i-1}^{(c')}$ using the current input $\varphi(\mathbf{u}_i)$ does not change the behavior of the mean square convergence or EMSE.

As long as we maintain these operations, which are essentially the same building blocks of QKLMS, the mean square convergence is not changed from the QKLMS analysis in [21].

IV. Simulation Results

Here, we evaluated the performance of the proposed NICE-KLMS algorithm and the generalized NICE-QKLMS algorithm, for the task of short-term chaotic time series prediction and transfer learning. Since the QKLMS algorithm was studied extensively and established as the state-of-the-art performer for curbing the growth of the RBF structure in kernel adaptive filtering [21], we focused our comparisons with the QKLMS algorithm. Specifically, we showed that the NICE-QKLMS algorithm can outperform the QKLMS algorithm, using finite training data, with fewer data centers per evaluation. And under the framework of transfer learning, NICE-QKLMS can leverage previously learned knowledge, i.e., filter parameters, to related task or domain.

A. Mackey-Glass Time Series Prediction

First, we tested NICE-QKLMS on the Mackey-Glass (MG) chaotic time series [39]. It was generated using the following time-delay ordinary differential equation

$$\frac{dx(t)}{dt}=\frac{\beta x(t-\tau)}{1+x(t-\tau)^n}-\gamma x(t),\ \text{where}\ \beta,\gamma,n>0 \tag{45}$$

with $\beta=0.2$, $\gamma=0.1$, $\tau=30$, and discretized at a sampling period of 6 seconds [5]. Chaotic dynamics are highly sensitive to initial conditions: small differences in initial conditions produce widely diverging outcomes, rendering long-term prediction intractable in general. Additive Gaussian noise with zero-mean and standard deviation $\sigma_n=0.04$, i.e., $V\sim\mathcal{N}(0,1.6\times10^{-3})$, were introduced. The time-delay embedding length or filter length was set at $L=12$; learning rate for all algorithms at $\eta=0.1$; kernel parameter for all three KAF algorithms at $a=1$; the quantization threshold at $d_q=0.1$ for QKLMS and

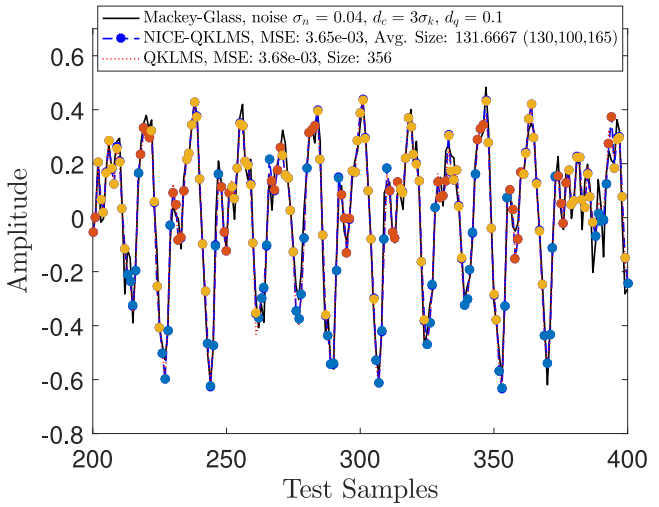


Fig. 3. Snapshot of a single trial Mackey-Glass chaotic time series prediction performance-comparison between QKLMS and NICE-QKLMS, on the testing set with additive zero-mean, $\sigma_n = 0.04$, Gaussian noise. The color-coding shows the "spatial-band" or subfilter used for each NICE-KLMS prediction.

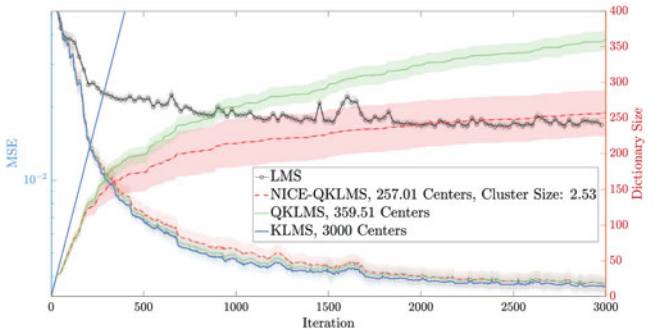


Fig. 4. Average NICE-KLMS performance over 100 independent trials, with ± 1 standard deviation. The left y-axis measures the MSE on the learning curves, which starts from the upper-left corner. The right y-axis measures the network growth.

NICE-QKLMS; and the centroid distance threshold for NICE-QKLMS at $d_c = 3\sigma_k = 2.1213$. The training set consists of 3000 consecutive samples. Testing is comprised of 400 independent samples.

Fig. 3 shows a single trial of the Mackey-Glass chaotic time series prediction performance-comparison between QKLMS and NICE-QKLMS, on the testing set with additive zero-mean, $\sigma_n = 0.04$, Gaussian noise. The NICE-QKLMS predictions are color-coded using dot marker symbols, with each color corresponding to a different cluster. Both algorithms use the same quantization threshold of $d_q = 0.1$. NICE-QKLMS has an additional parameter to partition training samples into clusters, using the centroid distance thrshold of $3\sigma_k$, with respect to the Gaussian kernel width. We see that the NICE-QKLMS predictions form three distinct spatial regions or clusters (some distorted by noise), each corresponding to a different cluster-filter. Only one cluster-filter was used per input sample, and the selection process took only three distance comparisons with the respectively cluster centroids. Compared to the 349 centers used for QKLMS, NICE-QKLMS uses roughly a third of the centers (average number of centers for the three filters is 130). This is expected since NICE-QKLMS segments the data into three sets, with overlaps due to transient-smoothing or knowledge transfer during training. In the training process, the earlier the clusters are determined, the smaller the overlaps.

We ran 100 Monte Carlo simulations, in which training consists of the same 3000 consecutive samples but with noise re-sampled from the same distribution, and testing consists of 400 independent consecutive samples with re-sampled noise and random starting index. Fig. 4 shows the averaged test performance with ± 1 standard deviation shaded on a two y-axes plot. The left, blue, y-axis measures the average MSE of the learning curves for the LMS, KLMS, QKLMS, and NICE-QKLMS algorithms, which starts from the upper-left corner. The right, red, y-axis measures the average network or dictionary size for the four algorithms, which starts from the lower-left corner or origin. In the case of the NICE-QKLMS algorithm, the average dictionary size across clusters is used. As expected, the KAF algorithms outperformed LMS, converging to much smaller values of MSE, due to their nonlinear nature. We see that NICE-QKLMS produced the same testing MSE at the final stage of the adaptation as QKLMS, but used a more parsimonious structure. For this particular experimental setup, NICE-QKLMS network used more than 100 fewer centers (257.01 vs 359.51) than QKLMS after 3000 iterations. Left uncurbed, the KLMS grew linearly, with 3000 centers, after the same number of updates. The vector quantization algorithm in QKLMS bounds the center-to-center distances from below, sequentially merging nearby centers into existing centers. However, it lacks a mechanism to bound center-to-center distances from above. For a given input sample, many of the centers in the QKLMS dictionary are sufficiently far away for the output of the Gaussian reproducing kernel to produce significant contributions. On the other hand, by partitioning the input/feature space into distinct spatial regions, NICE-QKLMS is able to specialize and provide better or similar performances using fewer samples per operation. The average number of clusters at the end of the adaptation is 2.53. On average, to evaluate the function approximate at each input, NICE-QKLMS automatically selects one of the 2.53 filters (with an average of 257 centers per filter) based on the minimum input-to-centroid distance threshold and performs KAF. For the same performance, the computational savings of NICE-QKLMS vs QKLMS is approximately 100 kernel evaluations, taking into account the 2.53 centroid distance computations used for filter selection.

B. Lorenz Time Series Prediction

Next, we considered the Lorenz chaotic system [40] described by the following three ordinary differential equations:

$$\begin{cases} \frac{dx}{dt} = -\sigma(y-x) \\ \frac{dy}{dt} = x(\rho - z) - y \\ \frac{dz}{dt} = -xy - \beta z \end{cases} \tag{46}$$

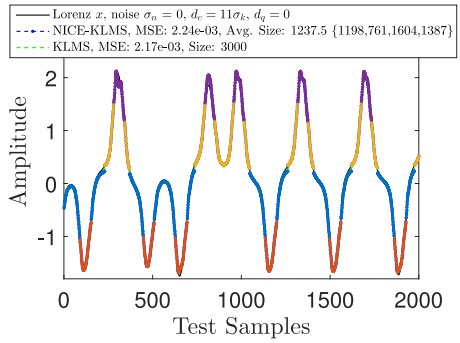


Fig. 5. Lorenz chaotic time series prediction. Data samples are self-organized into color-coded spatial regions by the NICE-KLMS algorithm.

where $\sigma = 10$, $\beta = \frac{8}{3}$, and $\rho = 28$ are the parameters which the system exhibits chaotic behavior. The Lorenz system is nonlinear and aperiodic. The x-component is used in the following short-term prediction task. The signal is normalized to be zero-mean and unit-variance.

Fig. 5 compares NICE-KLMS with KLMS on the clean x-component of the Lorenz chaotic time series. The filter length was set at $L = 8$; learning rate for both algorithms at $\eta = 0.1$; kernel parameter at $a = 1$; the centroid-distance threshold for NICE-QKLMS at $d_c = 11\sigma_k$ with respect to the Gaussian kernel width. Training consisted of 3000 consecutive sample; and testing, 2000 independent consecutive samples. Without noise distortion, we can clearly see the spatial regions self-organized by NICE-KLMS, in four color-coded clusters. The NICE-KLMS achieved comparable testing performance as KLMS (MSE of 2.24×10^{-3} vs 2.17×10^{-3}), but used significantly fewer centers per evaluation (average final cluster size of 1237.5 vs 3000).

For a more comprehensive comparison between the NICE-(Q)KLMS and (Q)KLMS algorithms, we visualized their performances using 3D surface plots in Fig. 6. We used the prediction gain [41] as the quantitative predictive performance measure, defined by

$$R_p \triangleq 10 \log_{10} \frac{\sigma_u^2}{\sigma_e^2} (\text{dB}) \tag{47}$$

where σ_u^2 is the signal power and σ_e^2 is the MSE. Each of the six subplots corresponds to the KAF performances using a different vector quantization threshold. The learning rate is fixed at $\eta = 0.1$, and the kernel parameter at $a = 1$. Within each subplot, there are three mini-graphs: the first is the prediction gain surface plot of the NICE-(Q)KLMS algorithm with its (Q)KLMS counterpart. The x-axis denotes the filter length, ranging from 1 to 15 in unit increments. The y-axis denotes the centroid-distance threshold in multiple of σ_k with respect to the Gaussian kernel width, ranging from 1 to 15 in unit-σ_k increments. The z-axis measures the prediction gain in dB and color-mapped with respect to magnitude. The performance surfaces of the NICE family of KAF algorithms are rendered using opaque colors, while the surfaces of the original KAF algorithms are rendered using translucent mesh. Since the (Q)KLMS performance is invariant with respect to the centroid-distance threshold, a side-profile of the performance surface is shown in the middle mini-graph of each subplot. The (Q)KLMS performance is shown using a solid color-mapped line, against the semitransparent surface-background, corresponding to all the achievable performance-regions of their NICE counterparts superimposed or layered for each centroid-distance threshold. The third or right mini-graph of each subplot shows the final per-evaluation dictionary sizes used by each algorithm to achieve the corresponding prediction gains. Again, a side-profile is used for clarity. Since the (Q)KLMS algorithms are invariant with respect to the centroid-distance thresholds, their RBF structure sizes appears as a single line, while the NICE average subfilter sizes are layered surfaces with respect to the centroid distances. The NICE subfilter size is proportional to the centroid-distance threshold: The upper-limit on the layered surface corresponds to the largest threshold $(15 \times \sigma_k)$, i.e., more inclusive; and the lower-limit, the smallest $(1 \times \sigma_k)$.

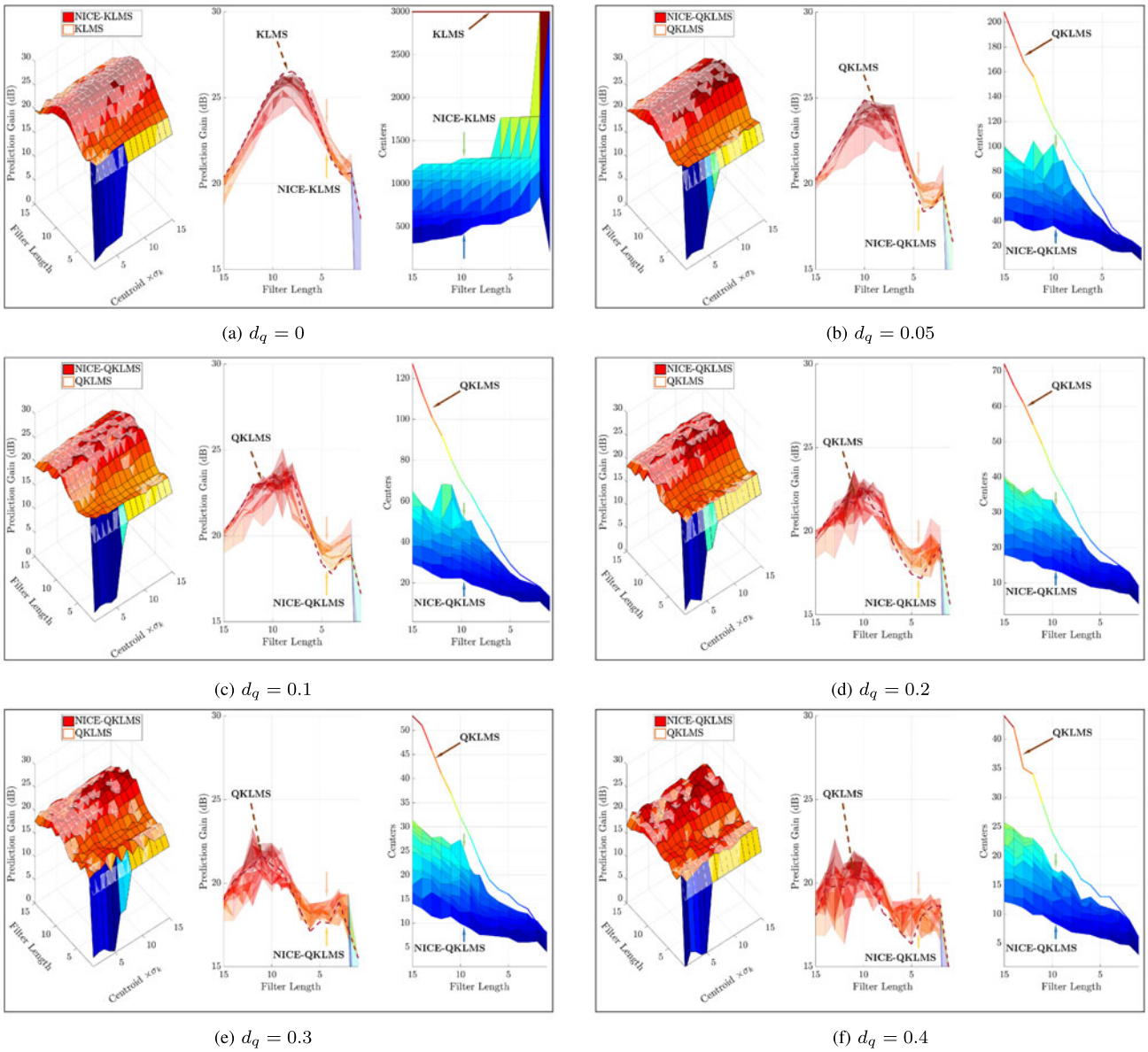
TABLE I
SUMMARY OF THE BEST PREDICTION-GAIN PERFORMANCES

Quantization	Filter	(Q)KLMS		NICE-(Q)KLMS		
d_q	L	R_p (dB)	Size	R_p (dB)	$d_c(\sigma_k)$	Size
0	8	**26.54**	3000	26.41	11	1237.50
0.05	10	24.88	116	**24.89**	14	102.25
0.1	9	23.12	63	**25.09**	8	37.75
0.2	11	21.67	49	**23.68**	7	26.20
0.3	12	21.0473	41	**23.29**	12	24.50
0.4	13	19.8217	35	**22.62**	13	22.25

As expected, the best performance is achieve when the quantization threshold is at zero, in Fig. 6(a), with a best filter length of $L = 8$. The KLMS performance serves as an upper-bound for most of the design parameter pairs, although the NICE-KLMS performance is very competitive (26.41 dB vs 26.54 dB) using roughly a third of the centers. For non-zero quantization thresholds, we see that there exists an operating point for which the NICE-QKLMS outperforms the QKLMS algorithm using fewer centers per evaluation. We tabulate the results in Table I.

C. Transfer Learning Using Content Addressable Filter Bank

Under the NICE framework, partial functionals comprising the adaptive filter are quickly stored and retrieved based on the input pattern. Instead of frequency bands, the subfilters are organized by amplitude or spatial bands or patterns. Since each cluster or distinct quasi-orthogonal region corresponds to a specialized "spatial-band" subfilter, the filter evaluation becomes the update of one of the partial filters, creating a content addressable filter bank or associative filter storage. This CAFB can be incrementally updated for new signal applications with mild constraints (amplitude normalization and same embedding dimension), opening the door for transfer learning and significantly more efficient training for new data scenarios, avoiding large initial errors produced by training from scratch, as have been done since the invention of adaptive filtering, and leverage previously learned knowledge to enhance prediction on limited data.



(a) $d_q = 0$ (b) $d_q = 0.05$

(c) $d_q = 0.1$ (d) $d_q = 0.2$

(e) $d_q = 0.3$ (f) $d_q = 0.4$

Fig. 6. Prediction gain and per-evaluation network size for the NICE-(Q)KLMS and corresponding (Q)KLMS algorithms with quantization threshold values of $d_q = 0, 0.05, 0.1, 0.2, 0.3, 0.4$. Each subplot consists of three mini graphs: the prediction gain plot for NICE-(Q)KLMS (full-color surface) and (Q)KLMS (translucent mesh), side-profile of the performance surfaces, and the RBF network size.

Here, we demonstrate the multipurpose capability of the NICE algorithm by showing that each subfilter can be shared across different signals. Specifically, we show that a NICE CAFB trained on one chaotic time series (Mackey-Glass) can be quickly repurposed for another time series (Lorenz), and one trained on the Lorenz time series can be transfered to enhance the performance of the real-world sunspot one-step-ahead prediction task. We expect that this will be the case for other applications where a model for the time series is required but the number of labeled data is limited.

1) Chaotic Time Series Prediction: Fig. 7 shows the data (zero-mean, unit variance) used for training and testing purposes. The first 6000 samples of the MG chaotic time series ($\beta = 0.2$, $\gamma = 0.1$, $\tau = 17$, initial condition $x_0 = 1.2$, and time step $\Delta_t = 0.1$) were used to train a NICE-QKLMS filter (filter length $L = 12$, learning rate $\eta = 0.1$, kernel parameter $a = 1$, quantization threshold $d_q = 0.01$, and centroid distance threshold $d_c = 6\sigma_k = 4.2426$). Testing consisted of 2000 consecutive samples, 1000 samples in the future, as shown in Fig. 7. We tested the trained NICE-QKLMS filter on both Mackey-Glass and a totally different chaotic time series, the Lorenz x-component. The Lorenz chaotic time series was generated using MATLAB's ordinary differential equation (ODE) solver ode45 in the time interval $T = [0, 50]$ with variable step size, $\sigma = 10$, $\beta = \frac{8}{3}$, $\rho = 28$, initial condition $(x_0, y_0, z_0) = (0, 1, 1.05)$.

The top plot of Fig. 8 shows the test performance of the MG-trained NICE-QKLMS filter on the MG test set. We see that NICE organized the data points into six distinct color-coded regions (only five were automatically selected for this

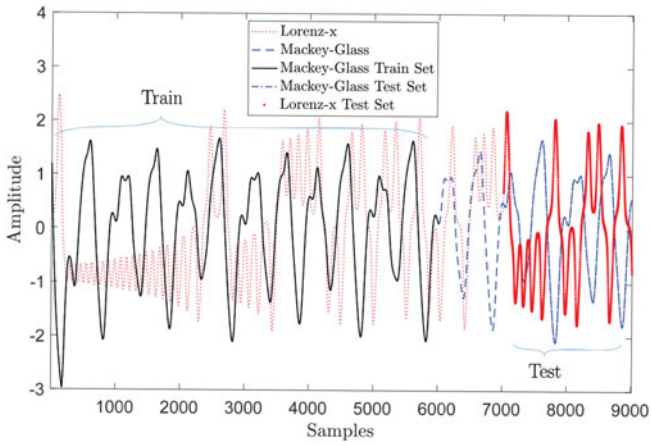


Fig. 7. The data used for the CAFB demonstration consist of two distinct chaotic time series: Mackey-Glass and Lorenz x-component.

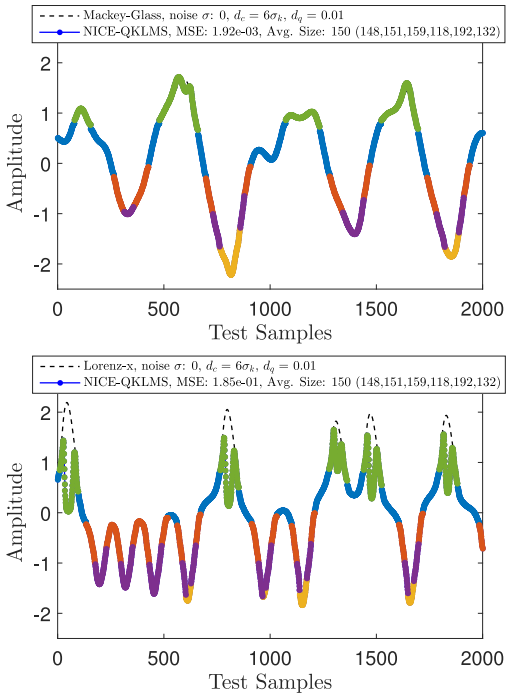


Fig. 8. Lorenz chaotic time series prediction using NICE filter trained on Mackey-Glass. Top plot shows the MG test set performance. The bottom plot shows the Lorenz test set performance using the exact same filters. The color-coding shows the "spatial-band" or subfilter used for each NICE-QKLMS prediction.

particular test set), with an average of 150 centers representing each region (compared to 6000 centers for normal KLMS operation). The bottom graph shows the MG-trained NICE-QKLMS performance on the Lorenz x-component test set. Although the filter has never seen this chaotic time series before, and they have different time structure, it was able to represent shared local structures using the same color-coded clusters. The performance degradation, from an MSE of $1.92\text{e}{-3}$ to $1.85\text{e}{-1}$, is primarily due to the filter's inability to represent the high peaks of the Lorenz time series. Although both time series are zero-mean and unit-variance, we see from Fig. 7 that rise/fall times of

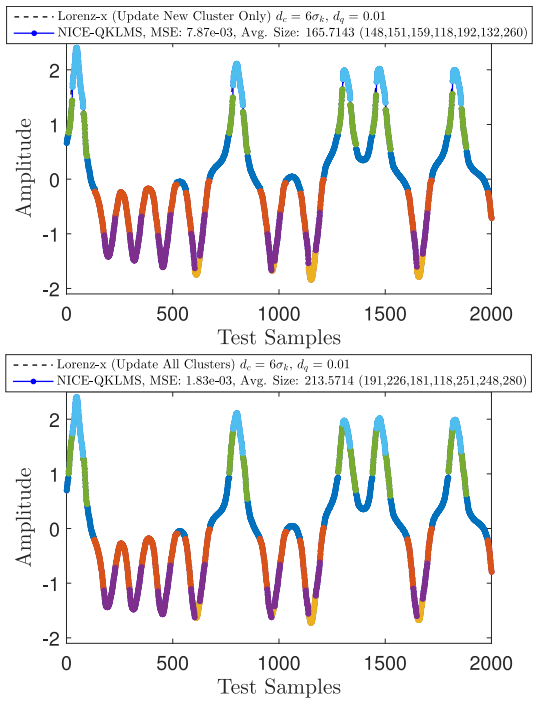


Fig. 9. Lorenz chaotic time series prediction using updated NICE filters initially trained on MG. Top plot shows the test set performance by updating new clusters only (*Split & Retain* followed by update procedures on the new clusters only). The bottom plot shows the test set performance by updating all existing clusters (all three operations were allowed for any given point).

the two are slightly shifted: the Lorenz series has higher peaks than the MG, while a typical MG peak dips in the middle and has lower troughs.

NICE self-organizes the data into interchangeable local components that can be used for different signals. To further illustrate its multipurpose capability, we adapted the MG-trained filter using the Lorenz data and show that it's faster to train than from scratch.

NICE provides fast, native support for automatically isolating and identifying a problem region. In our example, the Lorenz data contains new sample points (higher narrower peaks) that are not represented in the MG training data. Rather than updating the entire filter, we only allowed NICE to automatically create/split and update a single new cluster, using the exact same centroid parameters as before. The top plot of Fig. 9 shows the performance of this updated NICE-QKLMS algorithm. We see that a new color-coded cluster has been automatically introduced to accommodate the peculiar time structure of Lorenz. The coefficients and centers of the existing clusters are completely unchanged. By running through the Lorenz training data and only updating the centers and coefficients of new clusters, i.e., only the *Split & Retain* operation was allowed, followed by update procedures on just the new clusters, we improved the test-set performance from an MSE of $1.85\text{e}{-1}$ to $7.87\text{e}{-3}$ with an average increase of 15.7 centers per cluster. When we allowed all clusters (old and new) to adapt to the new data, i.e., all three operations can be performed at any given point, we obtained an MSE of $1.83\text{e}{-3}$ with 63.6 additional centers per

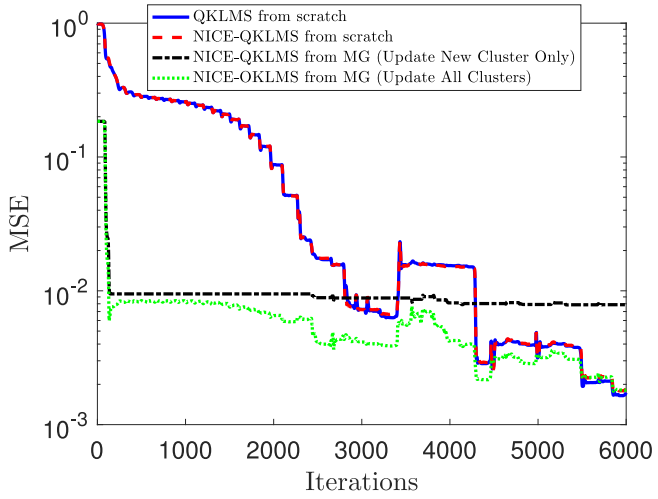


Fig. 10. Learning curve comparisons of Lorenz chaotic time series prediction using updated NICE filters initially trained on MG.

cluster, as shown in the bottom plot of Fig. 9. To gain more individual control, we can judiciously select the clusters to adapt, by computing sensitivities and allowing adaptation on clusters with certain high values.

Finally, we show the learning curves of the updated filters compared to the learning curves of filters learned from scratch, in Fig. 10. We see that by starting from an existing filter trained on a completely different chaotic time series, NICE-QKLMS adapted extremely rapidly in the new data environment and generated a head-start of more than 5000 iterations. This demonstrates the multipurpose capability of the NICE CAFB formulation.

2) Sunspot Prediction: The sunspot time series consists of the 13-month smoothed monthly total sunspot numbers from 1749 to 2016 [42]. The first 6000 samples of the Lorenz chaotic time series in Fig. 7 were used to train a NICE-QKLMS filter (filter length $L = 5$, learning rate $\eta = 0.05$, kernel parameter $a = 1$, quantization threshold $d_q = 0.01$, and centroid distance threshold $d_c = 6\sigma_k = 4.2426$). We normalized the time series to zero-mean with unit variance. Furthermore, we made the dynamic range of the sunspot time series the same as the Lorenz x-component's, using the ratio of their maximum absolute values. Sunspot numbers from July 1749 to February 1966 were used for training, and testing consists of sunspots from August 1966 to November 2016.

The top subplot of Fig. 11 shows the sunspot test set performance of the NICE-QKLMS CAFB trained using only the Lorenz time series, and compared with a QKLMS filter trained from scratch on the sunspot time series. The CAFB produced comparable performance even though it was trained on a completely different time series and, on average, used one third of the centers in a QKLMS implementation. If we allow the CAFB to adapt using the new training data, we obtain a superior test set performance than QKLMS (MSE of $3.20\text{e}{-3}$ vs. $4.17\text{e}{-3}$) with approximately a 50% saving in computation (average center size of 159.5 vs 332), as shown in the bottom subplot of Fig. 11. The KLMS, QKLMS, and NICE-QKLMS learning curves are

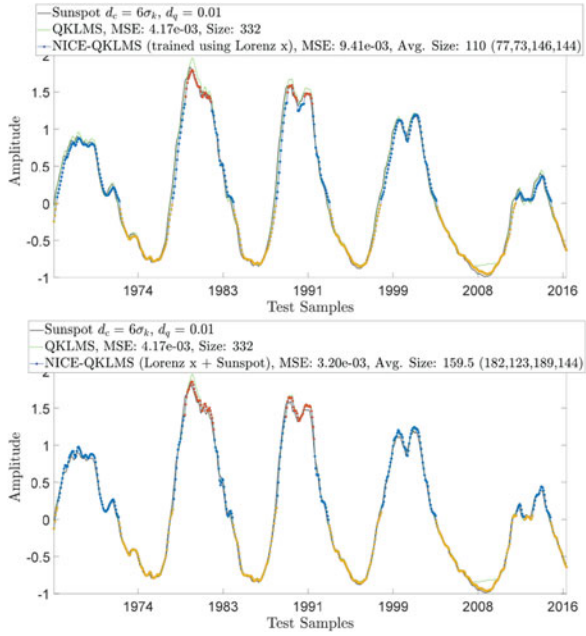


Fig. 11. Test set performances of the 13-month smoothed monthly total sunspot numbers time series. The top plot shows the CAFB performance trained completely on the Lorenz time series. The bottom plot shows the updated test performance after adaptation on the sunspot training set. The color-coding shows the "spatial-band" or subfilter used for each NICE-QKLMS prediction.

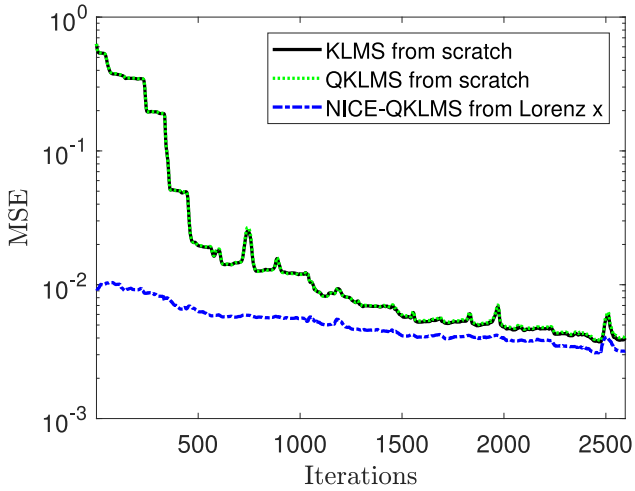


Fig. 12. Learning curve comparisons of sunspot prediction using updated NICE CAFB initially trained on the Lorenz time series.

plotted in Fig. 12. We see that CAFB is able to leverage previously learned local structures to outperform filters trained from scratch with a greater efficiency. This opens the door for similar problems where we can transfer the knowledge learned from an abundantly available synthetic data to a target task with limited measurements.

V. Conclusion

In this paper, we proposed a novel online nearest-neighbors approach to organize and curb the growth of the radial basis function (RBF) structure in kernel adaptive filtering (KAF) algorithms. The nearest-instance-centroid-estimation (NICE)

kernel least-mean-square (KLMS) algorithm is an instance-based learning that provides the appropriate time-space tradeoff with good online performance. Its centers or support vectors in the input/feature space form quasi-orthogonal regions. We trade the need to maintain complex search data structures for a depth-1 forest with the iteratively updated centroid of each cluster at the root. A linear search among the centroids determines the local support or subfilter used to evaluate a given function approximation. Compared to the popular RBF network reduction algorithm used in quantized KLMS, which only bounds the network size or center-to-center distances from below, NICE bounds the network size from above, by relocating centers outside of a certain responsive domain to a different subfilter. Using the energy conservation relation for adaptive filtering, we showed the sufficient condition for mean square convergence of the NICE-KLMS algorithm. The upper and lower steady-state excess-mean-square-error (EMSE) bounds were also established. As a proof-of-concept, we combined vector quantization (VQ) with NICE to formulate the novel KAF algorithm. Simulations on chaotic time-series prediction tasks demonstrated that our proposed method outperforms existing vector quantization method using fewer centers per evaluation. Furthermore, we demonstrated the multipurpose or transfer learning capability of our novel approach by performing regression on different signals using the same content addressable filter bank (CAFB) or associative filter storage. NICE CAFB can leverage previously learned knowledge to related task or domain.

We presented a novel approach for cluster analysis or unsupervised learning within the kernel adaptive filtering framework for regression. There are many issues that need be studied in the future. By self-organizing the data centers into distinct spatial regions, and with NICE's inherent ability to detect changes in data distribution, we open the door for non-stationary learning systems. As a CAFB, it opens the door to universal filtering of different signals. The NICE framework is also closely related to multiple and mixture kernel learning, but formulated within a single fixed RKHS. In the future, we will develop enhanced versions using different kernel parameters, introduce adaptive learning parameters, and apply the associative filter storage to multiple tasks.

REFERENCES

[1] B. Scholkopf and A. J. Smola, *Learning With Kernels, Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

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

[3] B. Scholkopf, A. J. Smola, and K. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[4] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[5] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. Hoboken, NJ, USA: Wiley, 2010.

[6] W. Liu, P. Pokharel, and J. C. Príncipe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 543–554, Feb. 2008.

[7] W. Liu and J. Principe, "Kernel affine projection algorithms," *EURASIP J. Adv. Signal Process.*, vol. 2008, no. 1, pp. 1–13, 2008.

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

[9] W. Liu, I. Park, Y. Wang, and J. C. Príncipe, "Extended kernel recursive least squares algorithm," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 3801–3814, Oct. 2009.

[10] K. Li and J. C. Príncipe, "The kernel adaptive autoregressive-moving-average algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 334–346, Feb. 2016.

[11] P. Honeine, "Analyzing sparse dictionaries for online learning with kernels," *IEEE Trans. Signal Process.*, vol. 63, no. 23, pp. 6343–6353, Dec. 2015.

[12] J. Suykens, J. D. Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: Robustness and sparse approximation," *Neurocomputing*, vol. 48, no. 14, pp. 85–105, 2002.

[13] B. J. de Kruif and T. J. A. de Vries, "Pruning error minimization in least squares support vector machines," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 696–702, May 2003.

[14] J. A. K. Suykens, T. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.

[15] L. Hoegaerts, J. Suykens, J. Vandewalle, and B. D. Moor, "Subset based least squares subspace regression in RKHS," *Neurocomputing*, vol. 63, pp. 293–323, 2005.

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

[17] L. Csato and M. Opper, "Sparse on-line Gaussian processes," *Neural Comput.*, vol. 14, no. 3, pp. 641–668, 2002.

[18] W. Liu, P. Pokharel, and J. C. Príncipe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 1950–1961, Dec. 2009.

[19] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[20] M. A. Takizawa and M. Yukawa, "Efficient dictionary-refining kernel adaptive filter with fundamental insights," *IEEE Trans. Signal Process.*, vol. 64, no. 16, pp. 4337–4350, Aug. 2016.

[21] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, "Quantized kernel least mean square algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 1, pp. 22–32, Jan. 2012.

[22] Y. Zheng, S. Wang, J. Feng, and C. K. Tse, "A modified quantized kernel least mean square algorithm for prediciton of chaotic time series," *Digit. Signal Process.*, vol. 48, pp. 130–136, 2016.

[23] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proc. 24th Int. Conf. Very Large Databases*, New York, NY, USA, Aug. 1998, pp. 194–205.

[24] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *Amer. Statistician*, vol. 46, no. 3, pp. 175–185, Aug. 1992.

[25] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, 1967, pp. 281–297.

[26] M. G. Genton, "Classes of kernels for machine learning: A statistics perspective," *J. Mach. Learn. Res.*, vol. 2, pp. 299–312, 2001.

[27] O. Toda and M. Yukawa, "An efficient kernel normalized least mean square algorithm with compactly supported kernel," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2015, pp. 3367–3371.

[28] S. Sonnenburg, G. Ratsch, C. Schafer, and B. Scholkopf, "Large scale multiple kernel learning," *J. Mach. Learn. Res.*, vol. 7, pp. 1531–1565, 2006.

[29] M. Yukawa, "Multikernel adaptive filtering," *IEEE Trans. Signal Process.*, vol. 60, no. 9, pp. 4672–4682, Sep. 2012.

[30] R. Pokharel, S. Seth, and J. C. Principe, "Mixture kernel least mean square," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Dallas, TX, USA, Aug. 2013, pp. 1–7.

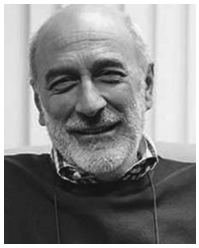
[31] S. Zhao, B. Chen, P. Zhu, and J. C. Príncipe, "Fixed budget quantized kernel least-mean-square algorithm," *Signal Process.*, vol. 93, no. 9, pp. 2759–2770, 2013.

[32] S. Zhao, B. Chen, Z. Cao, P. Zhu, and J. C. Principe, "Self-organizing kernel adaptive filtering," *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 106, pp. 1–12, 2016.

[33] B. Scholkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Proc. 14th Annu. Conf. Comput. Learn. Theory*, 2001, pp. 416–426.

[34] N. R. Yousef and A. H. Sayed, "A unified approach to the steady-state and tracking analyses of adaptive filters," *IEEE Trans. Signal Process.*, vol. 49, no. 2, pp. 314–324, Feb. 2001.
[35] T. Y. Al-Naffouri and A. H. Sayed, "Adaptive filters with error nonlinearities: Mean-square analysis and optimum design," *EURASIP J. Appl. Signal Process.*, vol. 2001, no. 4, pp. 192–205, 2001.
[36] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.
[37] B. Chen, S. Zhao, P. Zhu, and J. C. Prncipe, "Mean square convergence analysis for kernel least mean square algorithm," *Signal Process.*, vol. 92, no. 11, pp. 2624–2632, 2012.
[38] S. Haykin, *Adaptive Filter Theory*, 4th ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.
[39] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, Jul. 1977.
[40] E. N. Lorenz, "Deterministic nonperiodic flow," *J. Atmos. Sci.*, vol. 20, no. 2, pp. 130–141, 1963.
[41] S. Haykin and L. Li, "Nonlinear adaptive prediction of nonstationary signals," *IEEE Trans. Signal Process.*, vol. 43, no. 2, pp. 526–535, Jul. 1995.
[42] World Data Center SILSO, "Sunspot number and long-term solar observations," Royal Observatory of Belgium, Brussels, 2016. [Online]. Available: http://www.sidc.be/silso/

Kan Li (S'08–M'17) received the B.A.Sc. degree in electrical engineering from the University of Toronto, Toronto, ON, USA, in 2007, the M.S. degree in electrical engineering from the University of Hawaii, Honolulu, HI, USA, in 2010, and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2015. He is a Postdoctoral Associate in the Computational NeuroEngineering Laboratory, University of Florida. His research interests include machine learning and signal processing.



José C. Príncipe (M'83–SM'90–F'00) is the Bell-South and Distinguished Professor of electrical and biomedical engineering with the University of Florida, Gainesville, FL, USA and the Founding Director of the Computational NeuroEngineering Laboratory. His primary research interests include advanced signal processing with information theoretic criteria and adaptive models in reproducing kernel Hilbert spaces, with application to brain–machine interfaces. He is a Fellow of ABME and AIBME. He is the past Editor-in-Chief of the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, past Chair of the Technical Committee on Neural Networks of the IEEE Signal Processing Society, Past-President of the International Neural Network Society, and received the IEEE EMBS Career Award and the IEEE Neural Network Pioneer Award.