An Information Theoretic Approach of Designing Sparse Kernel Adaptive Filters

Weifeng Liu, *Member, IEEE*, Il Park, and José C. Príncipe, *Fellow, IEEE*

***Abstract*—This paper discusses an information theoretic approach of designing sparse kernel adaptive filters. To determine useful data to be learned and remove redundant ones, a subjective information measure called surprise is introduced. Surprise captures the amount of information a datum contains which is transferable to a learning system. Based on this concept, we propose a systematic sparsification scheme, which can drastically reduce the time and space complexity without harming the performance of kernel adaptive filters. Nonlinear regression, short term chaotic time-series prediction, and long term time-series forecasting examples are presented.**

***Index Terms*—Information measure, kernel adaptive filters, online Gaussian processes, online kernel learning, sparsification, surprise.**

I. Introduction

In machine learning, kernel methods and Gaussian processes (GPs) are very popular nonparametric modeling tools. However, they are computationally expensive, scaling with $O(N^3)$ in time and up to $O(N^2)$ in space where N is the number of training examples. There are many approaches to reduce the complexity, such as subspace approximation [1]–[3] and subset approximation [4]–[7]. Subset approximation achieves efficiency by reducing the effective number of training data. The complexity of these algorithms ranges from $O(M^2N)$ to $O(MN^2)$ where $M(\ll N)$ is the size of the effective data set (or the number of basis functions). It usually requires multiple passes of the whole training data. If computer memory cannot hold the whole training data, disk read operation would slow down the learning speed significantly. Online kernel learning provides more efficient alternatives where the selection of basis functions can be accomplished during sample-by-sample training [8]–[11]. They require only one pass over the training data and can be extremely efficient and effective. The complexity of these algorithms ranges from $O(MN)$ to $O(M^2N)$.

Kernel adaptive filtering is an emerging subfield of online kernel learning. It includes kernel least mean squares (KLMS) [12], kernel recursive least squares (KRLS) [13], kernel affine projection algorithms (KAPA) [14]–[16], and extended kernel recursive least squares (EX-KRLS) [17]. These methods generalize naturally the classical linear adaptive filters [18] in reproducing kernel Hilbert spaces (RKHS) and are very appealing for nonlinear adaptive signal processing. The principal bottleneck of this class of algorithms is its growing structure with each new sample. In other words, time and space complexities grow linearly with the number of training data, which poses a big problem for continuous adaptation. Intuitively, one can expect that after processing sufficient samples from the same source, there is little need to grow the structure of the filters more, because of redundancy. The difficulty is exactly how to select important data and to remove redundancy. Distance- and prediction-error-based criteria are discussed in [19] and [13]. Prediction variance is also used in [20]. A newly proposed coherence criterion is discussed in [16]. Though these methods are quite effective in practical applications, they are heuristic in nature. Therefore, how to mathematically establish the framework to test if a given sample is needed or not is of great significance.

In this regard, a new criterion is proposed based on the concept of surprise. Surprise is a subjective information measure, quantifying how much information a datum point contains relative to the "knowledge of the learning system." Little surprise is expected from redundancy. For example, one watches a weather forecast to learn what kind of weather to expect, but watching the same program for the second time does not add anything to our knowledge. Obviously this concept is very important for learning and adaptation but there is no widely accepted mathematical definition in the literature yet. First, Palm gave a definition of surprise based on the idea of template [21], [22]. Then, Itti and Baldi studied it in a Bayesian framework using the Kullback–Leibler divergence [23]. Most recently, Ranasinghe and Shen investigated its use in robotics using symbolic schemes [24]. In this paper, surprise is defined as the negative log likelihood (NLL) of an observation given the learning machine's hypothesis. The NLL concept is widely used in parameter estimation and hypothesis test [25], [26]. However, its application to active data selection is quite novel. The work of Dima and Hebert [27] is similar to ours but they used kernel density estimation while we use GP theory to derive an analytical solution. Defining the instantaneous learnable information contained on a data sample allows us to discard or include new exemplars systematically and curb the complexity of online kernel algorithms. This information criterion provides a unifying view on existing sparsification methods discussed in [19], [13], [20], and [16], allows new insights to be gained, highlights the relationship between existing methods, and also provides a general framework for redundancy removal, abnormality detection, and knowledge discovery.

Manuscript received April 16, 2009; revised July 13, 2009; accepted September 24, 2009. First published November 17, 2009; current version published December 04, 2009. This work was supported in part by the National Science Foundation (NSF) under Grants ECS-0601271 and ECS-0856441.

W. Liu is with the Forecasting Team, Amazon.com, Seattle, WA 98104 USA (e-mail: weifeng@amazon.com).

I. Park is with the Department of Biomedical Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: memming@cnel.ufl.edu).

J. C. Príncipe is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA (email: 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/TNN.2009.2033676

A family of algorithms is proposed by applying the new concept to different kernel adaptive filters including KLMS, KAPA, and KRLS. The surprise measure is computed based on the theory of Gaussian process regression (GPR). Since KRLS is mathematically equivalent to online GPR [20], the surprise measure can be seamlessly integrated into KRLS. Then, some approximation is used to derive more efficient surprise-based KLMS and KAPA.

The organization of this paper is as follows. In Section II, kernel adaptive filters including KLMS, KAPA, and KRLS are briefly reviewed. Next, in Section III, we introduce the mathematical definition of surprise and derive an analytical formula based on GPR. Then, we focus on how to utilize the concept to design sparse kernel adaptive filters in Section IV. Three experiments are studied in Section V to support our theory. Finally, Section VI summarizes the conclusions and future lines of research.

II. Kernel Adaptive Filters

Suppose the goal is to learn a continuous input–output mapping $f : \mathbb{U} \rightarrow \mathbb{R}$ based on a sequence of input–output examples $\{\mathbf{u}(1), d(1)\}, \{\mathbf{u}(2), d(2)\}, \ldots, \{\mathbf{u}(i), d(i)\}, \ldots$ where $\mathbb{U} \subseteq \mathbb{R}^L$ is the input domain. The output is assumed to be 1-D but it is straightforward to generalize the discussion to multidimensions. Kernel adaptive filters are a framework to estimate f sequentially such that f_i (the estimate at iteration i) is updated based on the last estimate f_{i-1} and the current example $\{\mathbf{u}(i), d(i)\}$.

A kernel is a continuous, symmetric, positive–definite function $\kappa : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}$ [28], [29]. The commonly used kernel is the Gaussian kernel

$$\kappa(\mathbf{u}, \mathbf{u}') = \exp(-a\|\mathbf{u} - \mathbf{u}'\|^2). \tag{1}$$

According to Mercer's theorem, any kernel $\kappa(\mathbf{u}, \mathbf{u}')$ induces a mapping $\boldsymbol{\varphi}$ from the input space $\mathbb{U}$ to a feature space $\mathbb{F}$ such that

$$\boldsymbol{\varphi}(\mathbf{u})^T \boldsymbol{\varphi}(\mathbf{u}') = \kappa(\mathbf{u}, \mathbf{u}'). \tag{2}$$

$\mathbb{F}$ is isometric–isomorphic to the RKHS induced by the kernel and we do not distinguish these two spaces in this paper. Equation (2) is usually called the kernel trick.

A. Kernel Least Mean Square

To derive KLMS [12], the input $\mathbf{u}(i)$ is transformed into $\mathbb{F}$ as $\boldsymbol{\varphi}(\mathbf{u}(i))$. Denote $\boldsymbol{\varphi}(i) = \boldsymbol{\varphi}(\mathbf{u}(i))$ for simplicity. Using the least mean square (LMS) [30] algorithm on the new example sequence $\{\boldsymbol{\varphi}(i), d(i)\}$ yields

$$\begin{aligned} \boldsymbol{\omega}(0) &= 0 \\ e(i) &= d(i) - \boldsymbol{\omega}(i-1)^T \boldsymbol{\varphi}(i) \\ \boldsymbol{\omega}(i) &= \boldsymbol{\omega}(i-1) + \eta e(i) \boldsymbol{\varphi}(i) \end{aligned} \tag{3}$$

where $e(i)$ is called the prediction error, η is the step size, and $\boldsymbol{\omega}(i)$ denotes the estimate (at iteration i) of the weight vector in $\mathbb{F}$. f_i is the composition of $\boldsymbol{\omega}(i)$ and $\boldsymbol{\varphi}$, i.e., $f_i(\cdot) = \boldsymbol{\omega}(i)^T \boldsymbol{\varphi}(\cdot)$, so the learning rule for KLMS is

$$\begin{aligned} f_0 &= 0 \\ e(i) &= d(i) - f_{i-1}(\mathbf{u}(i)) \\ f_i &= f_{i-1} + \eta e(i) \kappa(\mathbf{u}(i), \cdot). \end{aligned} \tag{4}$$

KLMS allocates a new kernel unit for every new example with input $\mathbf{u}(i)$ as the center and $\eta e(i)$ as the coefficient. The algorithm ends up as a growing radial basis function (RBF) network

$$f_i = \sum_{j=1}^{i} \mathbf{a}_j(i) \kappa(\mathbf{u}(j), \cdot).$$

The coefficients $\mathbf{a}(i)$ and the centers $\mathcal{C}(i) = \{\mathbf{u}(j)\}_{j=1}^{i}$ are stored in memory during training. The complexity at iteration i is $O(i)$, linearly increasing with the number of training data.

B. Kernel Affine Projection Algorithms

The KLMS algorithm can be viewed as a stochastic gradient–descent algorithm using the instantaneous values to approximate the covariance matrix and cross-covariance vector [18]. KAPA [14] employs better approximations based on the K most recent inputs and observations

$$\boldsymbol{\Phi}_K(i) = [\boldsymbol{\varphi}(i-K+1), \ldots, \boldsymbol{\varphi}(i)]$$

and

$$\mathbf{d}_K(i) = [d(i-K+1), \ldots, d(i)]^T.$$

It yields the following stochastic gradient descent:

$$\boldsymbol{\omega}(i) = \boldsymbol{\omega}(i-1) + \eta \boldsymbol{\Phi}_K(i)[\mathbf{d}_K(i) - \boldsymbol{\Phi}_K(i)^T \boldsymbol{\omega}(i-1)]. \tag{5}$$

The updating rule for KAPA is

$$\begin{aligned} f_0 &= 0 \\ e(i;k) &= d(k) - f_{i-1}(\mathbf{u}(k)), \qquad \text{for } i-K+1 \le k \le i \\ f_i &= f_{i-1} + \eta \sum_{j=i-K+1}^{i} e(i;j) \kappa(\mathbf{u}(j), \cdot). \end{aligned} \tag{6}$$

Like KLMS, KAPA is also a growing RBF-like network, allocating a new unit with $\mathbf{u}(i)$ as the center and $\eta e(i;i)$ as the coefficient. Unlike KLMS, it also updates the coefficients for the other $K-1$ most recent units by $\eta e(i;k)$. The computational complexity of KAPA is $O(i + K^2)$ at iteration i.

There are several variants of KAPA in [14] and we only discuss the basic gradient–descent algorithm here for simplicity. In addition, similar algorithms are discussed in [15] and [16]. The algorithm presented in [16] is very close to the type 2 KAPA algorithm discussed in [14], but it is more elegant in the sense that all data points are treated in the same affine projection framework regardless of being novel or not. By contrast, all the algorithms discussed in this paper distinguish novel data from redundant data and simply throw away all the redundant data.

C. Kernel Recursive Least Squares

In KRLS [13], the weight vector $\boldsymbol{\omega}(i)$ is the minimizer of

$$\min_{\boldsymbol{\omega}} \sum_{j=1}^{i} |d(j) - \boldsymbol{\omega}^T \boldsymbol{\varphi}(j)|^2 + \lambda \|\boldsymbol{\omega}\|^2 \tag{7}$$

where λ is the regularization parameter. This cost function is also studied in the context of kernel ridge regression [31] and least squares support vector machines [32]. It is shown that $\boldsymbol{\omega}(i) = \boldsymbol{\Phi}(i)\mathbf{a}(i)$ where $\mathbf{a}(i) = [\lambda \mathbf{I} + \mathbf{G}(i)]^{-1}\mathbf{d}(i), \mathbf{d}(i) = [d(1), \ldots, d(i)]^T, \mathbf{G}(i) = \boldsymbol{\Phi}(i)^T\boldsymbol{\Phi}(i)$ and $\boldsymbol{\Phi}(i) = [\boldsymbol{\varphi}(1), \ldots, \boldsymbol{\varphi}(i)]$. KRLS is also a growing RBF-like network

$$f_i = \sum_{j=1}^{i} \mathbf{a}_j(i)\kappa(\mathbf{u}(j), \cdot).$$

Denoting

$$\begin{aligned} \mathbf{Q}(i-1) &= (\lambda \mathbf{I} + \mathbf{G}(i-1))^{-1} \\ \mathbf{h}(i) &= \boldsymbol{\Phi}(i-1)^T \boldsymbol{\varphi}(i) \\ \mathbf{z}(i) &= \mathbf{Q}(i-1)\mathbf{h}(i) \\ r(i) &= \lambda + \kappa(\mathbf{u}(i), \mathbf{u}(i)) - \mathbf{z}(i)^T \mathbf{h}(i) \end{aligned} \tag{8}$$

we have the following sequential learning rule for KRLS:

$$\begin{aligned} f_0 &= 0 \\ e(i) &= d(i) - f_{i-1}(\mathbf{u}(i)) \\ f_i &= f_{i-1} + r(i)^{-1} e(i) \kappa(\mathbf{u}(i), \cdot) \\ &\quad - \sum_{j=1}^{i-1} r(i)^{-1} e(i) \mathbf{z}_j(i) \kappa(\mathbf{u}(j), \cdot) \end{aligned} \tag{9}$$

where $\mathbf{z}_j(i)$ is the jth component of $\mathbf{z}(i)$. The learning procedure of KRLS is very similar to KLMS and KAPA in the sense that it allocates a new unit with $\mathbf{u}(i)$ as the center and $r(i)^{-1}e(i)$ as the coefficient. At the same time KRLS also updates all the previous coefficients by $-r(i)^{-1}e(i)\mathbf{z}(i)$ whereas KLMS never updates previous coefficients and KAPA only updates the $K-1$ most recent ones. The computational complexity of KRLS is $O(i^2)$ at iteration i. The online GPR presented in [20] is equivalent to KRLS.

D. Existing Sparsification Criteria

As we see, the bottleneck problem of kernel adaptive filters is its growing structure (complexity) with each new datum. Active data selection (retaining important data and removing redundant ones) is a natural approach to tackle this issue. There are mainly two existing criteria in the literature of online kernel learning: the novelty criterion (NC) proposed by Platt [19] in resource-allocating networks and the approximate linear dependency (ALD) test introduced by Engel [13] for KRLS. The prediction variance criterion in [20] is similar to ALD. The coherence criterion proposed in [16] can be viewed as an approximation to ALD as we will show later.

Suppose the present learning system is

$$f_i = \sum_{j=1}^{m_i} \mathbf{a}_j(i)\kappa(\mathbf{c}_j, \cdot) \tag{10}$$

with $\mathbf{c}_j$ as the jth center and $\mathbf{a}_j(i)$ as the jth coefficient. Denote $\mathcal{C}(i) = \{\mathbf{c}_j\}_{j=1}^{m_i}$ as the center set (or dictionary). When a new example $\{\mathbf{u}(i+1), d(i+1)\}$ is presented, the learning system needs to decide if $\mathbf{u}(i+1)$ is accepted as a new center.

1) Novelty Criterion: NC first computes the distance of $\mathbf{u}(i+1)$ to the present dictionary

$$\mathrm{dis}_1 = \min_{\mathbf{c}_j \in \mathcal{C}(i)} \|\mathbf{u}(i+1) - \mathbf{c}_j\|.$$

If $\mathrm{dis}_1 < \delta_1, \mathbf{u}(i+1)$ will not be added into the dictionary. Otherwise, it further computes the prediction error $e(i+1) = d(i+1) - f_i(\mathbf{u}(i+1))$. Only if $|e(i+1)| > \delta_2, \mathbf{u}(i+1)$ will be accepted as a new center. δ_1 and δ_2 are two user-specified parameters.

2) Approximate Linear Dependency: ALD tests the following cost:

$$\mathrm{dis}_2 = \min_{\forall \mathbf{b}} \left\| \boldsymbol{\varphi}(\mathbf{u}(i+1)) - \sum_{\mathbf{c}_j \in \mathcal{C}(i)} \mathbf{b}_j \boldsymbol{\varphi}(\mathbf{c}_j) \right\|$$

which indicates the distance of the new input to the linear span of the present dictionary in the feature space. Similar problem is discussed in the context of finding approximate preimage [33]. By straightforward calculus, it turns out that

$$\mathrm{dis}_2^2 = \kappa(\mathbf{u}(i+1), \mathbf{u}(i+1)) - \mathbf{h}(i+1)^T \mathbf{G}^{-1}(i)\mathbf{h}(i+1) \tag{11}$$

where

$$\mathbf{h}(i+1) = [\kappa(\mathbf{u}(i+1), \mathbf{c}_1), \ldots, \kappa(\mathbf{u}(i+1), \mathbf{c}_{m_i})]^T \tag{12}$$

$$\mathbf{G}(i) = \begin{bmatrix} \kappa(\mathbf{c}_1, \mathbf{c}_1) & \cdots & \kappa(\mathbf{c}_{m_i}, \mathbf{c}_1) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{c}_1, \mathbf{c}_{m_i}) & \cdots & \kappa(\mathbf{c}_{m_i}, \mathbf{c}_{m_i}) \end{bmatrix}. \tag{13}$$

$\mathbf{u}(i+1)$ will be rejected if dis_2 is smaller than some preset threshold δ_3.

If regularization is added, (11) becomes

$$\begin{aligned} r(i+1) = \lambda &+ \kappa(\mathbf{u}(i+1), \mathbf{u}(i+1)) \\ &- \mathbf{h}(i+1)^T (\mathbf{G}(i) + \lambda \mathbf{I})^{-1} \mathbf{h}(i+1) \end{aligned} \tag{14}$$

which is a quantity already defined in KRLS (8). It is also the prediction variance in GPR [20]. This understanding is very important to derive efficient approximations in the following sections.

ALD is quite computationally expensive scaling quadratically with the size of the dictionary. A natural simplification is to use the "nearest" center in the dictionary to estimate the overall distance, i.e.,

$$\mathrm{dis}_3 = \min_{\forall b, \forall \mathbf{c}_j \in \mathcal{C}(i)} \|\boldsymbol{\varphi}(\mathbf{u}(i+1)) - b\boldsymbol{\varphi}(\mathbf{c}_j)\|. \tag{15}$$

By straightforward calculus, one has

$$\mathrm{dis}_3^2 = \min_{\forall \mathbf{c}_j \in \mathcal{C}(i)} \left[\kappa(\mathbf{u}(i+1), \mathbf{u}(i+1)) - \frac{\kappa^2(\mathbf{u}(i+1), \mathbf{c}_j)}{\kappa(\mathbf{c}_j, \mathbf{c}_j)} \right].$$

When κ is a unit-norm kernel, that is, $\kappa(\mathbf{u}, \mathbf{u}) = 1$ for all $\mathbf{u}$, this distance measure is equivalent to the coherence measure. When κ is an RBF, this distance measure is equivalent to dis_1 used in the NC.

Notice that the target $d(i+1)$ is not used in ALD.

III. Surprise

Suppose the learning machine is $y(\mathbf{u}; \mathcal{T}(i))$ after processing $\mathcal{D}(i) = \{\mathbf{u}(j), d(j)\}_{j=1}^{i}$, where $\mathcal{T}(i)$ specifies the state of the learning system at time i. The problem is to measure how much information a new example $\{\mathbf{u}(i+1), d(i+1)\}$ contains which is "transferable to" the current learning system.

First, let us see why the definition from the classic information theory does not apply here. By [34], the information contained on an exemplar $\{\mathbf{u}(i+1), d(i+1)\}$ is defined as

$$I(i+1) = -\ln p(\mathbf{u}(i+1), d(i+1)) \tag{16}$$

where $p(\mathbf{u}(i+1), d(i+1))$ is the true joint probability density. This definition is widely used and achieves huge successes in digital communications, game theory, and other fields [35]. However, it has at least two problems in the learning setting. First, the learning machine never knows the true joint probability density, which is the ultimate knowledge meant to be learned by the machine [36]. Second, it is observer independent. This is certainly undesirable in the context of learning because it is unable to distinguish novelty and redundancy.

A. Definition of Surprise

Since the true joint distribution is unknown and an observer-dependent information measure is sought, a natural idea is to define the information measure based on the posterior distribution hypothesized by the learning system.

Definition 1: Surprise is a subjective information measure of an exemplar $\{\mathbf{u}, d\}$ with respect to a learning system $\mathcal{T}$. Denoted by $S_{\mathcal{T}}(\mathbf{u}, d)$, it is defined as the NLL of the exemplar given the learning system's hypothesis on the data distribution

$$S_{\mathcal{T}}(\mathbf{u}, d) = -\ln p(\mathbf{u}, d|\mathcal{T}) \tag{17}$$

where $p(\mathbf{u}, d|\mathcal{T})$ is the subjective probability of $\{\mathbf{u}, d\}$ hypothesized by $\mathcal{T}$.

$S_{\mathcal{T}}(\mathbf{u}, d)$ measures how "surprising" the exemplar is to the learning system. Applying this definition directly to the active online learning problem, we have the surprise of $\{\mathbf{u}(i+1), d(i+1)\}$ to the current learning system $\mathcal{T}(i)$ simply as

$$S_{\mathcal{T}(i)}(\mathbf{u}(i+1), d(i+1)) = -\ln p(\mathbf{u}(i+1), d(i+1)|\mathcal{T}(i)) \tag{18}$$

where $p(\mathbf{u}(i+1), d(i+1)|\mathcal{T}(i))$ is the posterior distribution of $\{\mathbf{u}(i+1), d(i+1)\}$ hypothesized by $\mathcal{T}(i)$. Denote $S(i+1) = S_{\mathcal{T}(i)}(\mathbf{u}(i+1), d(i+1))$ for simplicity in the following discussion.

Intuitively, if $p(\mathbf{u}(i+1), d(i+1)|\mathcal{T}(i))$ is very large, the new datum $\{\mathbf{u}(i+1), d(i+1)\}$ is well expected by the learning system $\mathcal{T}(i)$ and thus contains a small amount of information to be learned. On the other hand, if $p(\mathbf{u}(i+1), d(i+1)|\mathcal{T}(i))$ is small, the new datum "surprises" the learning system, which means either the data contains something new for the system to discover or it is suspicious.

According to this measure, we can classify the new exemplar into three categories:

- *abnormal*: $S(i+1) > T_1$;
- *learnable*: $T_1 \geq S(i+1) \geq T_2$;
- *redundant*: $S(i+1) < T_2$.

T_1 and T_2 are problem-dependent parameters. The choice of the thresholds and learning strategies defines the characteristics of the learning system.

B. Evaluation of Surprise

It is a difficult problem in general to estimate the posterior distribution. One way is to use kernel density estimation as in [27] but this is problematic when the dimensionality of $\mathbf{u}$ is high. Another way is to convert it to a parameter estimation problem by assuming a parametric distribution family. In [23], Itti studied neural activities by assuming a Poisson distribution across the models. In this paper, we use the GP theory.

1) Gaussian Processes Regression: In the GPR, the prior distribution of system outputs is assumed to be jointly Gaussian [37], i.e.,

$$[y(\mathbf{u}(1)), \ldots, y(\mathbf{u}(i))]^T \sim \mathcal{N}(0, \sigma_n^2 \mathbf{I} + \mathbf{G}(i))$$

where

$$\mathbf{G}(i) = \begin{bmatrix} \kappa(\mathbf{u}(1), \mathbf{u}(1)) & \cdots & \kappa(\mathbf{u}(i), \mathbf{u}(1)) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{u}(1), \mathbf{u}(i)) & \cdots & \kappa(\mathbf{u}(i), \mathbf{u}(i)) \end{bmatrix}$$

for any i. σ_n^2 is the variance of the noise contained in the observation. κ is the covariance function. Covariance functions are equivalent to reproducing kernels in the sense that any covariance function can be a reproducing kernel and *vice versa*. With this prior assumption, the posterior distribution of the output given the input $\mathbf{u}(i+1)$ and all past observations $\mathcal{D}(i)$ can be derived as

$$p(y(\mathbf{u}(i+1))|\mathbf{u}(i+1), \mathcal{D}(i)) \sim \mathcal{N}(\bar{d}(i+1), \sigma^2(i+1)) \tag{19}$$

which is again normally distributed, with

$$\bar{d}(i+1) = \mathbf{h}(i+1)^T [\sigma_n^2 \mathbf{I} + \mathbf{G}(i)]^{-1} \mathbf{d}(i) \tag{20}$$

$$\sigma^2(i+1) = \sigma_n^2 + \kappa(\mathbf{u}(i+1), \mathbf{u}(i+1)) - \mathbf{h}(i+1)^T [\sigma_n^2 \mathbf{I} + \mathbf{G}(i)]^{-1} \mathbf{h}(i+1) \tag{21}$$

where $\mathbf{h}(i+1) = [\kappa(\mathbf{u}(i+1), \mathbf{u}(1)), \ldots, \kappa(\mathbf{u}(i+1), \mathbf{u}(i))]^T$ and $\mathbf{d}(i) = [d(1), \ldots, d(i)]^T$. It is clear that GPR is equivalent to KRLS.

2) Evaluation of Surprise: Let us assume $\mathcal{T}(i) = \mathcal{D}(i)$ for now, i.e., the learning system memorizes all the past

input–output pairs. By (19), the posterior joint probability density $p(\mathbf{u}(i+1), d(i+1)|\mathcal{T}(i))$ becomes

$$\begin{aligned} &p(\mathbf{u}(i+1), d(i+1)|\mathcal{T}(i)) \\ &\quad = p(d(i+1)|\mathbf{u}(i+1), \mathcal{T}(i))\, p(\mathbf{u}(i+1)|\mathcal{T}(i)) \\ &\quad = \frac{1}{\sqrt{2\pi}\sigma(i+1)} \exp\left(-\frac{(d(i+1)-\bar{d}(i+1))^2}{2\sigma^2(i+1)}\right) \\ &\qquad \times p(\mathbf{u}(i+1)|\mathcal{T}(i)) \end{aligned}$$

and therefore, the surprise measure is

$$\begin{aligned} S(i+1) &= -\ln[p(\mathbf{u}(i+1), d(i+1)|\mathcal{T}(i))] \qquad (22) \\ &= \ln\sigma(i+1) + \frac{(d(i+1)-\bar{d}(i+1))^2}{2\sigma^2(i+1)} \\ &\quad - \ln[p(\mathbf{u}(i+1)|\mathcal{T}(i))] + \ln\sqrt{2\pi}. \qquad (23) \end{aligned}$$

Equation (22) gives a whole picture of what factors and how these factors affect the surprise measure of the new datum. Some observations are as follows.

1) The surprise measure is proportional to the magnitude of the prediction error. $e(i+1) = d(i+1) - \bar{d}(i+1)$ is the prediction error since $\bar{d}(i+1)$ is the maximum *a posteriori* (MAP) estimation of $d(i+1)$ by the current learning system $\mathcal{T}(i)$. If $e^2(i+1)$ is very small, which means the learning system predicts well near $\mathbf{u}(i+1)$, the corresponding $S(i+1)$ is small.
2) The surprise measure is proportional to the prediction variance if the prediction error is very small. In the case of a very small prediction error, say $e(i+1) \approx 0$, the second term $(d(i+1)-\bar{d}(i+1))^2/2\sigma^2(i+1)$ is ineffective, and $S(i+1)$ is directly proportional to $\sigma(i+1)$. A large variance indicates that the learning system is uncertain about its guess though the guess happens to be right. Incorporating the new datum will boost the prediction confidence near the neighborhood of the new datum in the future inference.
3) The surprise measure is huge with a small variance and a large prediction error. With a large prediction error and a small variance, the second term $(d(i+1)-\bar{d}(i+1))^2/2\sigma^2(i+1)$ dominates the measure. This is a strong indication of abnormality: the machine is sure about its prediction but its prediction is far away from the observation. It means exactly that the current observation is contradictory to what the machine had learned before.
4) A rare occurrence means more surprise. A smaller $p(\mathbf{u}(i+1)|\mathcal{T}(i))$ leads to a larger $S(i+1)$.

C. Input Distribution

The distribution $p(\mathbf{u}(i+1)|\mathcal{T}(i))$ is problem dependent. In the regression model, it is reasonable to assume

$$p(\mathbf{u}(i+1)|\mathcal{T}(i)) = p(\mathbf{u}(i+1)) \qquad (24)$$

that is, the distribution of $\mathbf{u}(i+1)$ is independent of the previous observations, or memoryless. If the input has a normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we have

$$\begin{aligned} S(i+1) = \ln\sigma(i+1) &+ \frac{(d(i+1)-\bar{d}(i+1))^2}{2\sigma^2(i+1)} \\ &+ \frac{(\mathbf{u}(i+1)-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{u}(i+1)-\boldsymbol{\mu})}{2}. \qquad (25) \end{aligned}$$

In general, we can assume the distribution $p(\mathbf{u}(i+1))$ is uniform if no *a priori* information is available. Therefore, by discarding the constant terms, the surprise measure (22) is simplified as

$$S(i+1) = \ln\sigma(i+1) + \frac{(d(i+1)-\bar{d}(i+1))^2}{2\sigma^2(i+1)}. \qquad (26)$$

D. Unknown Desired Signal

The surprise measure depends on the knowledge of the desired signal $d(i+1)$. In the case of unknown desired signal, we simply average $S(i+1)$ over the posterior distribution of $y(\mathbf{u}(i+1))$. By using (22), one has

$$\begin{aligned} \bar{S}(i+1) &= \int S(i+1)p(y|\mathbf{u}(i+1), \mathcal{T}(i))dy \\ &= \ln\sqrt{2\pi} + \ln\sigma(i+1) + \frac{1}{2} - \ln[p(\mathbf{u}(i+1)|\mathcal{T}(i))]. \end{aligned}$$

Neglecting the constant terms yields

$$\bar{S}(i+1) = \ln\sigma(i+1) - \ln[p(\mathbf{u}(i+1)|\mathcal{T}(i))]. \qquad (27)$$

Furthermore, under a memoryless uniform input assumption, it is simplified as

$$\bar{S}(i+1) = \ln\sigma(i+1). \qquad (28)$$

Therefore, ALD and variance criterion are a special case of the surprise criterion.

E. The Probability of Novelty

Assume that the probability density function of the prediction error is $p(e)$, which does not contain any delta function at 0. According to the surprise criterion, the probability of accepting $\mathbf{u}(i+1)$ into the dictionary is

$$\begin{aligned} &P(\mathbf{u}(i+1) \text{ is accepted into the dictionary}) \\ &\quad = P\left(T_1 \geq -\ln\left(\frac{1}{\sqrt{2\pi}\sigma(i+1)}\exp\left(-\frac{e^2(i+1)}{2\sigma^2(i+1)}\right)\right) \geq T_2\right) \\ &\quad = P(2\sigma^2(i+1)(T_1 - \ln\sigma(i+1) - \ln\sqrt{2\pi}) \\ &\quad \geq e^2(i+1) \geq 2\sigma^2(i+1)(T_2 - \ln\sigma(i+1) - \ln\sqrt{2\pi})) \\ &\quad \leq \int_{-\sqrt{2\sigma^2(i+1)(T_1-\ln\sigma(i+1)-\ln\sqrt{2\pi})}}^{\sqrt{2\sigma^2(i+1)(T_1-\ln\sigma(i+1)-\ln\sqrt{2\pi})}} p(e)de \qquad (29) \end{aligned}$$

when $T_1 - \ln\sigma(i+1) - \ln\sqrt{2\pi} > 0$, or equivalently, $\sigma(i+1) < (\exp(T_1))/\sqrt{2\pi}$. Otherwise, the probability is 0.

The key point here is

$$\lim_{\sigma(i+1)\to 0} 2\sigma^2(i+1)(T_1 - \ln\sigma(i+1) - \ln\sqrt{2\pi}) = 0.$$

In other words

$$\lim_{\sigma(i+1)\to 0} P(\mathbf{u}(i+1) \text{ is accepted into the dictionary}) = 0.$$

Therefore, the probability of being accepted into the dictionary is very small if $\sigma(i+1)$ is small. In this sense, we say the prediction variance $\sigma(i+1)$ is probabilistically lower bounded in online learning algorithms with a surprise criterion. By using the arguments presented in [13] and [16], we can conclude that the number of centers added into the dictionary is essentially bounded even as the number of training examples increases without bound.

IV. Surprise Criterion for Sparse Kernel Adaptive Filters

A. Surprise Criterion Kernel Recursive Least Squares

It is easy to verify that $\bar{d}(i+1)$ in (20) and $\sigma^2(i+1)$ in (21) equal $f_i(\mathbf{u}(i+1))$ and $r(i+1)$, respectively, in KRLS with $\sigma_n^2 = \lambda$. Therefore, the surprise criterion can be integrated into KRLS seamlessly. We call it SC-KRLS. The system starts with $f_1 = \mathbf{a}(1)\kappa(\mathbf{u}(1),\cdot)$ with $\mathbf{a}(1) = \mathbf{Q}(1)d(1), \mathbf{Q}(1) = [\lambda + \kappa(\mathbf{u}(1),\mathbf{u}(1))]^{-1}$ and $\mathcal{C}(1) = \{\mathbf{c}_1 = \mathbf{u}(1)\}$. Then, it iterates the following procedure for $i \geq 1$.

For a new datum $\{\mathbf{u}(i+1), d(i+1)\}$, it computes the following quantities:

$$\begin{aligned}
\mathbf{h}(i+1) &= [\kappa(\mathbf{u}(i+1),\mathbf{c}_1),\ldots,\kappa(\mathbf{u}(i+1),\mathbf{c}_{m_i})]^T \\
f_i(\mathbf{u}(i+1)) &= \mathbf{h}(i+1)^T\mathbf{a}(i) \\
e(i+1) &= d(i+1) - f_i(\mathbf{u}(i+1)) \\
r(i+1) &= \lambda + \kappa(\mathbf{u}(i+1),\mathbf{u}(i+1)) - \mathbf{h}(i+1)^T\mathbf{Q}(i)\mathbf{h}(i+1)
\end{aligned}$$

and the surprise measure becomes

$$S(i+1) = \frac{1}{2}\ln r(i+1) + \frac{e^2(i+1)}{2r(i+1)} - \ln[p(\mathbf{u}(i+1)|\mathcal{T}(i))]$$

where $p(\mathbf{u}(i+1)|\mathcal{T}(i))$ can be assumed to be constant if no *a priori* information is available. As we see, the computation of the surprise criterion comes with no additional complexity.

Based on this surprise measure, we can decide if the example is abnormal, learnable, or redundant. If it is abnormal or redundant, it can be simply thrown away. If it is learnable, the system is updated by the standard KRLS algorithm as shown in (30)–(33) at the bottom of the page.

The updating rule is consistent with the observations of (22). If the prediction error is small, the modifying quantities are small. In the extreme case, a redundant datum leads to negligible update. On the other hand, a large prediction error and a small prediction variance result in a large modification to the coefficients. In the extreme case, an abnormal datum causes instability. The overall complexity of SC-KRLS is $O(m_i^2)$.

B. Surprise Criterion Kernel Least Mean Square

The complexity of computing the exact surprise measure for KLMS is $O(m_i^2)$ at iteration i, which may offset the advantage of KLMS over other kernel adaptive filters in terms of simplicity. As we know the surprise involves two basic concepts: the prediction error and the prediction variance. It is relatively easy to get the prediction error and the question is how we simplify the computation of the prediction variance. The approximation is based on the relationship between the prediction variance $\sigma^2(i+1)$ and the distance measure dis_2. To simplify the computation of dis_2, we use the following distance measure as an approximation:

$$\mathrm{dis}_3 = \min_{\forall b, \forall \mathbf{c}_j \in \mathcal{C}(i)} ||\boldsymbol{\varphi}(\mathbf{u}(i+1)) - b\boldsymbol{\varphi}(\mathbf{c}_j)|| \tag{34}$$

i.e., selecting the "nearest" center in the dictionary to estimate the overall distance just like in the NC. By straightforward calculus, one has

$$\mathrm{dis}_3^2 = \min_{\forall \mathbf{c}_j \in \mathcal{C}(i)} \left[\kappa(\mathbf{u}(i+1),\mathbf{u}(i+1)) - \frac{\kappa^2(\mathbf{u}(i+1),\mathbf{c}_j)}{\kappa(\mathbf{c}_j,\mathbf{c}_j)}\right].$$

When κ is an RBF, this distance measure is equivalent to dis_1 used in the NC, and we can add a regularization term simply by including λ. Using the same notation from SC-KRLS, we have

$$r(i+1) = \lambda + \kappa(\mathbf{u}(i+1),\mathbf{u}(i+1)) - \max_{\forall \mathbf{c}_j \in \mathcal{C}(i)} \frac{\kappa^2(\mathbf{u}(i+1),\mathbf{c}_j)}{\kappa(\mathbf{c}_j,\mathbf{c}_j)}. \tag{35}$$

Its complexity is $O(m_i)$ which is acceptable to KLMS. Therefore, we have the following surprise criterion KLMS (SC-KLMS). The system starts with $f_1 = \mathbf{a}(1)\kappa(\mathbf{u}(1),\cdot)$ with $\mathbf{a}(1) = \eta d(1)$ and $\mathcal{C}(1) = \{\mathbf{c}_1 = \mathbf{u}(1)\}$. Then, it iterates the following procedure for $i \geq 1$.

For a new datum $\{\mathbf{u}(i+1), d(i+1)\}$, it computes the following quantities:

$$\begin{aligned}
\mathbf{h}(i+1) &= [\kappa(\mathbf{u}(i+1),\mathbf{c}_1),\ldots,\kappa(\mathbf{u}(i+1),\mathbf{c}_{m_i})]^T \\
f_i(\mathbf{u}(i+1)) &= \mathbf{h}(i+1)^T\mathbf{a}(i) \\
e(i+1) &= d(i+1) - f_i(\mathbf{u}(i+1)) \\
r(i+1) &= \lambda + \kappa(\mathbf{u}(i+1),\mathbf{u}(i+1)) - \max_{\forall \mathbf{c}_j \in \mathcal{C}(i)} \frac{\kappa^2(\mathbf{u}(i+1),\mathbf{c}_j)}{\kappa(\mathbf{c}_j,\mathbf{c}_j)}.
\end{aligned}$$

$$\mathcal{C}(i+1) = \{\mathcal{C}(i), \mathbf{u}(i+1)\} \tag{30}$$

$$\mathbf{z}(i+1) = \mathbf{Q}(i)\mathbf{h}(i+1) \tag{31}$$

$$\mathbf{Q}(i+1) = \begin{bmatrix} \mathbf{Q}(i) + \mathbf{z}(i+1)\mathbf{z}(i+1)^T/r(i+1) & -\mathbf{z}(i+1)/r(i+1) \\ -\mathbf{z}(i+1)^T/r(i+1) & 1/r(i+1) \end{bmatrix} \tag{32}$$

$$\mathbf{a}(i+1) = \begin{bmatrix} \mathbf{a}(i) - \mathbf{z}(i+1)e(i+1)/r(i+1) \\ e(i+1)/r(i+1) \end{bmatrix} \tag{33}$$

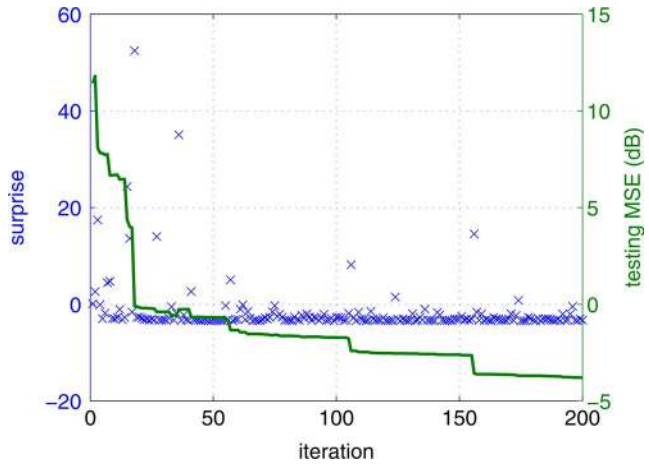

Fig. 1. Surprise measure of training data and corresponding learning curve in nonlinear regression.

The surprise measure is

$$S(i+1) = \frac{1}{2}\ln r(i+1) + \frac{e^2(i+1)}{2r(i+1)} - \ln[p(\mathbf{u}(i+1)|\mathcal{T}(i))]$$

where $p(\mathbf{u}(i+1)|\mathcal{T}(i))$ can be assumed to be constant if no *a priori* information is available. If the example is learnable, the system is updated by the standard KLMS algorithm

$$\mathcal{C}(i+1) = \{\mathcal{C}(i), \mathbf{u}(i+1)\} \tag{36}$$

$$\mathbf{a}_{i+1}(i+1) = \eta e(i+1). \tag{37}$$

The overall complexity of SC-KLMS is $O(m_i)$.

C. Surprise Criterion Kernel Affine Projection Algorithms

As we see, SC-KLMS uses only the "nearest" sample in the dictionary to approximate the distance. A natural idea is to use multiple (say K) centers to improve the approximation. Putting it into an optimization equation, we have

$$\text{dis}_4 = \min_{\forall \mathbf{b}, \forall \mathbf{n}_j \in \mathcal{C}(i)} \left\| \boldsymbol{\varphi}(\mathbf{u}(i+1)) - \sum_{j=1}^{K} \mathbf{b}_j \boldsymbol{\varphi}(\mathbf{n}_j) \right\| \tag{38}$$

i.e., selecting K centers in the dictionary to estimate the overall distance. This is a combinatorial optimization scaling with $O(C_{m_i}^K K^3)$, where $C_{m_i}^K$ denotes the number of K-combinations from the dictionary. A more feasible alternative is to select the K "nearest" neighbors one-by-one simply based on (34), whose complexity is $O(Km_i)$. After selecting the neighbors, the surprise measure can be computed and the learning system can be updated by the KAPA algorithms based on the K selected neighbors. To summarize, the surprise criterion KAPA (SC-KAPA) is

$$\begin{aligned}
\mathbf{h}(i+1) &= [\kappa(\mathbf{u}(i+1), \mathbf{c}_1), \ldots, \kappa(\mathbf{u}(i+1), \mathbf{c}_{m_i})]^T \\
f_i(\mathbf{u}(i+1)) &= \mathbf{h}(i+1)^T \mathbf{a}(i) \\
e(i+1) &= d(i+1) - f_i(\mathbf{u}(i+1)) \\
r(i+1) &= \lambda + \kappa(\mathbf{u}(i+1), \mathbf{u}(i+1)) - \mathbf{h}_{\mathbf{u}}^T [\mathbf{G}_{\mathbf{n}} + \lambda \mathbf{I}]^{-1} \mathbf{h}_{\mathbf{u}} \\
S(i+1) &= \frac{1}{2}\ln r(i+1) + \frac{e^2(i+1)}{2r(i+1)} - \ln[p(\mathbf{u}(i+1)|\mathcal{T}(i))]
\end{aligned}$$

where

$$\mathbf{h}_{\mathbf{u}} = [\kappa(\mathbf{u}(i+1), \mathbf{n}_1), \ldots, \kappa(\mathbf{u}(i+1), \mathbf{n}_K)]^T$$

$$\mathbf{G}_{\mathbf{n}} = \begin{bmatrix} \kappa(\mathbf{n}_1, \mathbf{n}_1) & \cdots & \kappa(\mathbf{n}_1, \mathbf{n}_K) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{n}_K, \mathbf{n}_1) & \cdots & \kappa(\mathbf{n}_K, \mathbf{n}_K) \end{bmatrix}.$$

If the datum is determined to be learnable, the system is updated in the following way:

$$\begin{aligned}
e(i+1;k) &= d(\mathbf{n}_k) - f_i(\mathbf{n}_k), \qquad \text{for } 1 \le k \le K \\
f_{i+1} &= f_i + \eta e(i+1)\kappa(\mathbf{u}(i+1), \cdot) \\
&\quad + \eta \sum_{k=1}^{K} e(i+1;k)\kappa(\mathbf{n}_k, \cdot)
\end{aligned}$$

where $d(\mathbf{n}_k)$ is the output associated with $\mathbf{n}_k$. The above SC-KAPA is significantly different from KAPA and demands more computational resources ($O(Km_i + K^3)$). KAPA uses the most K recent data points to approximate the gradient vector whereas SC-KAPA uses the K nearest neighbors. It is argued that using the nearest neighbors is more effective in terms of modeling while using the most recent ones is better at tracking. In principle, approximating the prediction variance and approximating the gradient direction are independent and can be dealt with differently. For example, we can simply use the approximation in SC-KLMS to compute the surprise in SC-KAPA without modification. As we will emphasize here, the real advantage of SC-KAPA is its flexibility in design and users should tailor the algorithm accordingly in practice.

V. Simulations

A. Nonlinear Regression

In the example, we use a simple nonlinear regression problem to illustrate the main idea of surprise and its effectiveness in learning. The input–output mapping is $y = -x + 2x^2 + \sin x$ and the input is Gaussian distributed with zero mean and unit variance.

In the first simulation, we use SC-KRLS and assume all data are learnable. The surprise is computed for every point during training. In general, the surprise measure of each datum depends on the relative order by which it is presented to the learning system. There are 200 points for training and 100 for testing. Fig. 1 is a typical learning curve with mean square error (MSE) calculated at each iteration on the testing data. It clearly shows that decreases in testing MSE (solid) are directly resulted from learning informative data (cross). The Gaussian kernel (1) is used with $a = 0.2$. The regularization parameter $\lambda = 0.001$. The parameters are selected through cross validation.

We also plot the 5% most "informative" data in Fig. 2. It exhibits the characteristic of active sampling with few data in the well-defined region and more data near the boundary. The surprise measure is very effective to distinguish novelty and redundancy.

In the second simulation, we show how effective the method is to remove redundancy. We compare the surprise criterion (SC) [see (25)] with the ALD test [see (28)] in KRLS. We test

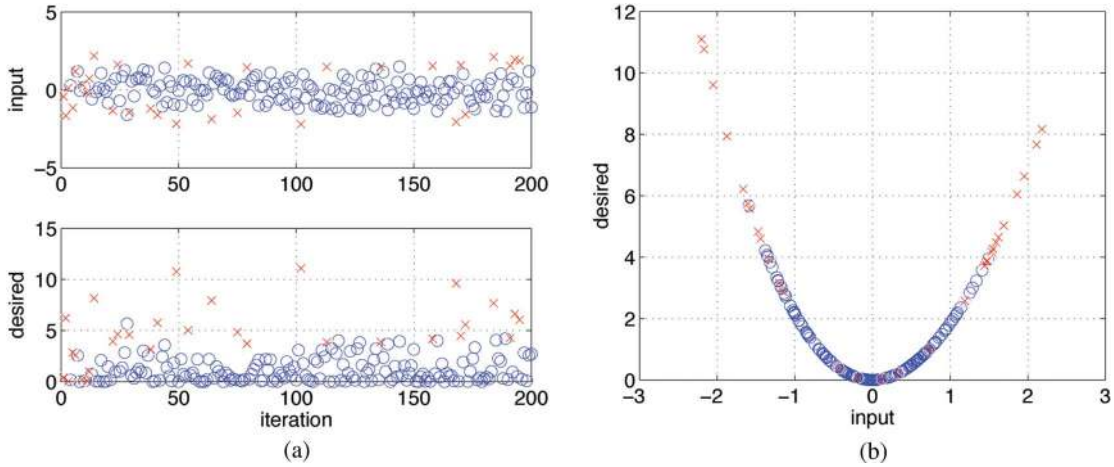

Fig. 2. Informative training data according to the surprise measure in nonlinear regression. (a) The 5% most informative training data (cross) along training iteration. (b) The 5% most informative training data (cross) in 2-D space.

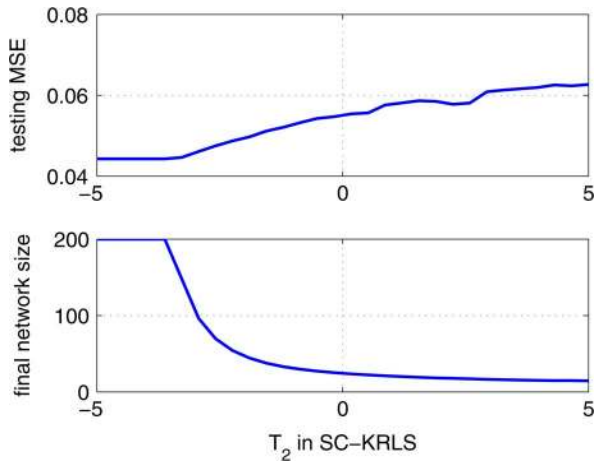

Fig. 3. Final network size versus testing MSE for SC-KRLS in nonlinear regression.

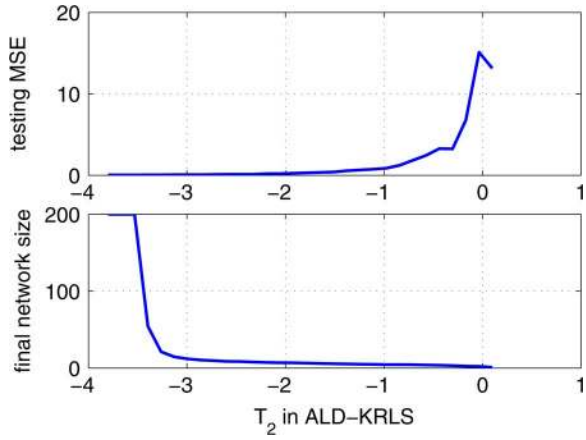

Fig. 4. Final network size versus testing MSE for ALD-KRLS in nonlinear regression.

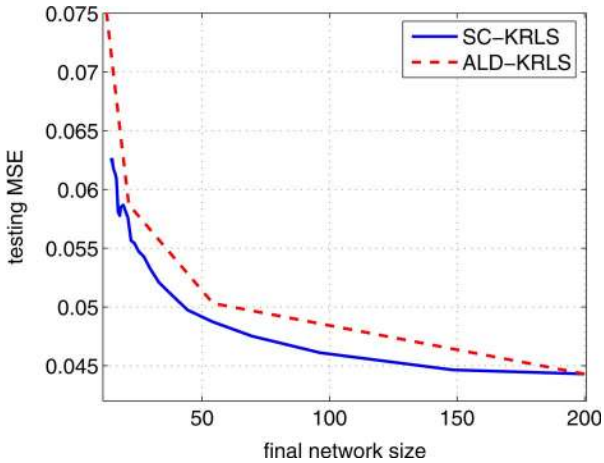

Fig. 5. Comparison of SC-KRLS and ALD-KRLS in redundancy removal in nonlinear regression.

both SC-KRLS and ALD-KRLS with 30 different thresholds of T_2. A large T_1 is used to disable the abnormality detection. For each T_2, 100 Monte Carlo simulations are conducted with independent inputs to calculate the average number of centers and the corresponding average testing MSE. For each Monte Carlo simulation, 200 training points and 100 testing points are used. The results are illustrated in Figs. 3–5. It is clear that SC is very effective with T_2 in a wide range of $[-3, 5]$. Though ALD is equally effective with T_2 in the range of $[-3, -2]$, a larger T_2 leads to catastrophic results (almost all points are excluded except the first one). By contrast, SC provides a balance by checking the prediction error. As shown in Fig. 5, SC-KRLS is superior over ALD-KRLS. For the same MSE, SC-KRLS requires 10–20 less data points on average.

In the third simulation, we show how SC-KRLS can be used to detect outliers while ALD-KRLS *cannot*. Two hundred training data are generated as before but 15 outliers are manually added at time indices 50, 60, ..., 190 (by flipping their signs). We choose $T_2 = -3.14, T_1 = 200$ in SC and $T_2 = -3.37$ in ALD based on the result of the second simulation. There are actually 12 effective outliers as shown in Fig. 6 since another three points are very close to the origin, and SC-KRLS correctly detects all the outliers as shown in Fig. 7. The outliers seriously compromise the performance of ALD-KRLS as shown in Fig. 8. This example clearly demonstrates the ability of SC to detect and reject outliers.

B. Mackey–Glass Time Series Prediction

The Mackey–Glass (MG) chaotic time series [38], [39] is widely used as a benchmark data set for nonlinear learning

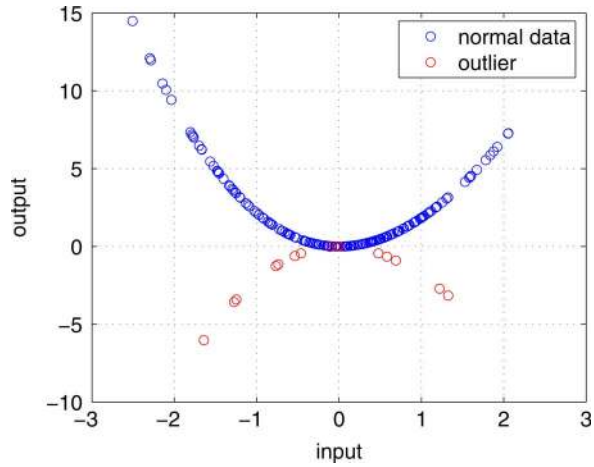

Fig. 6. Training data with outliers in nonlinear regression.

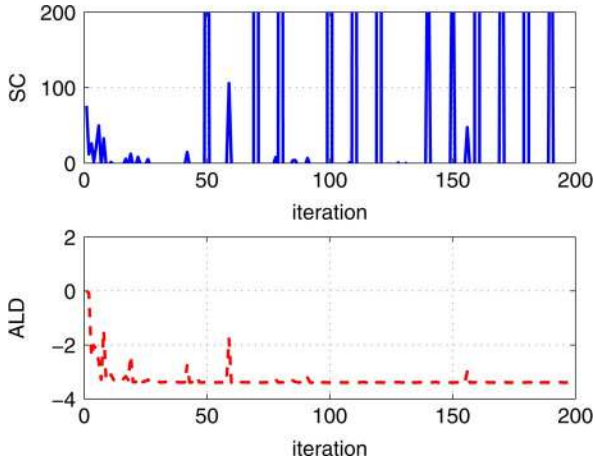

Fig. 7. Comparison of surprise measure in SC-KRLS and ALD measure in ALD-KRLS in nonlinear regression with outliers.

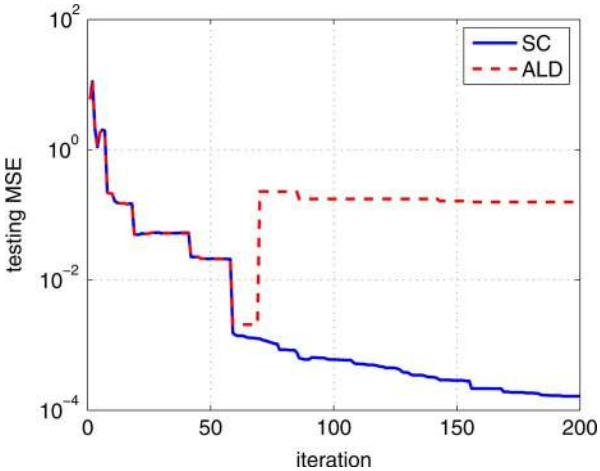

Fig. 8. Learning curves of SC-KRLS and ALD-KRLS in nonlinear regression with outliers.

methods. The time series is generated from the following time delay ordinary differential equation:

$$\frac{dx(t)}{dt} = -bx(t) + \frac{ax(t-\tau)}{1+x(t-\tau)^{10}} \tag{39}$$

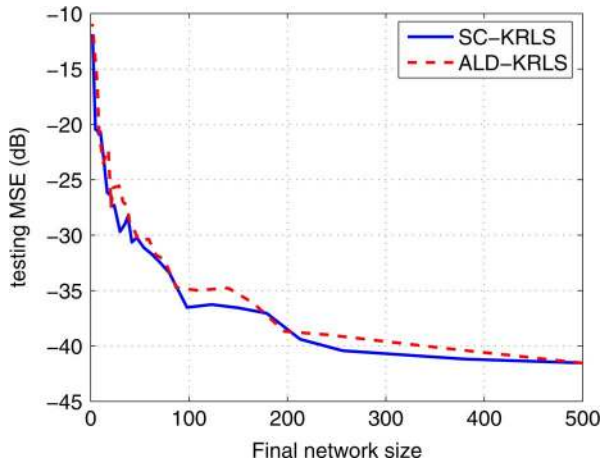

Fig. 9. Final network size versus testing MSE of SC-KRLS and ALD-KRLS with different T_2 in Mackey–Glass time series prediction.

with $b = 0.1, a = 0.2$, and $\tau = 30$. The time series is discretized at a sampling period of 6 s. The problem setting for short term prediction is as follows: the previous seven points $\mathbf{u}(i) = [x(i-7), x(i-6), \ldots, x(i-1)]^T$ are used to predict the present one $x(i)$. The number of previous data points used as input is called time embedding and can be determined by Takens theorem [40]. A segment of 500 samples is used as the training data and another 100 points as the test data (in the testing phase, the filter is fixed).

First, we compare the performance of SC-KRLS with ALD-KRLS. A Gaussian kernel with kernel parameter $a = 1$ is chosen. A large T_1 is used to disable the abnormality detection. We test both algorithms with 30 different T_2. The result is illustrated in Fig. 9. It is seen that the overall performance of SC-KRLS is better than ALD-KRLS though comparable. The regularization parameter $\lambda = 0.001$ is selected by cross validation in both algorithms and memoryless uniform input distribution is assumed in the computation of a surprise.

Second, we compare the performance of SC-KLMS with the coherence criterion KLMS (CC-KLMS). A Gaussian kernel with kernel parameter $a = 1$ is chosen. A large T_1 is used to disable the abnormality detection in SC-KLMS. We test both algorithms with 50 different thresholds. The result is illustrated in Fig. 10. It is seen that SC-KLMS outperforms CC-KLMS. The regularization parameter in SC-KLMS is $\lambda = 0.001$. The step size is set as 0.4 for both algorithm. All the free parameters are selected by cross validation. Memoryless uniform input distribution is assumed in the computation of a surprise.

Finally, we compare the performance of a linear filter trained with least mean square (LMS), novelty criterion kernel least mean square (NC-KLMS), surprise criterion kernel least mean square (SC-KLMS), resource-allocating network (RAN) [19], and SC-KRLS. A Gaussian kernel (1) with kernel parameter $a = 1$ is chosen for all the kernel-based algorithms. One hundred Monte Carlo simulations are run with different realizations of noise. The noise is additive white Gaussian noise with zero mean and 0.004 variance. The step size for LMS is 0.01. The step size is 0.5 for NC-KLMS, and $\delta_1 = 0.1$ and $\delta_2 = 0.1$ are used in the NC. SC-KLMS uses step size 0.5, $T_2 = -1$,

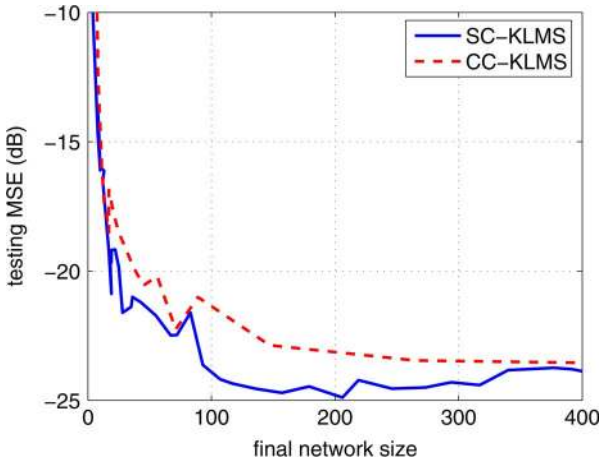

Fig. 10. Final network size versus testing MSE of SC-KLMS and CC-KLMS with different criterion thresholds in Mackey–Glass time series prediction.

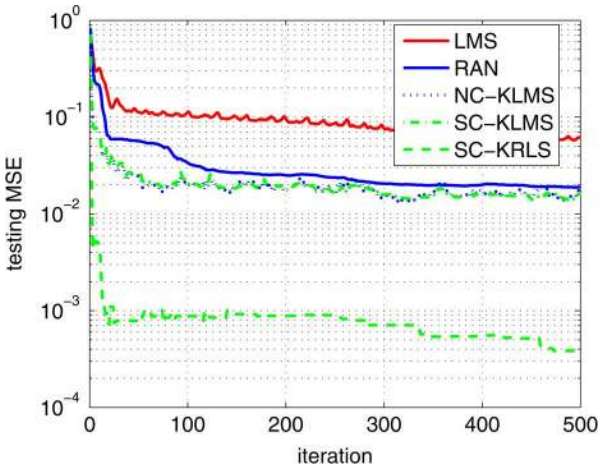

Fig. 11. Learning curves of LMS, RAN, NC-KLMS, SC-KLMS, and SC-KRLS in Mackey–Glass time-series prediction.

and $\lambda = 0.01$. For RAN, the step size is 0.05 and the tolerance for prediction error is 0.05. The distance resolution parameters are $\delta_{\max} = 0.5, \delta_{\min} = 0.05$, and $\tau_{\text{ran}} = 45$. The overlap factor is 0.87. Refer to [19] for the parameter settings of RAN. SC-KRLS uses $T_2 = -1$ and $\lambda = 0.01$. The parameters are set by cross validation. Fig. 11 is the ensemble learning curves for LMS, NC-KLMS, SC-KLMS, RAN, and SC-KRLS, respectively. Performances of RAN, NC-KLMS, and SC-KLMS are comparable and SC-KRLS outperforms all significantly. The network sizes are listed in Table I. It can be seen that SC-KLMS has a much smaller network size than NC-KLMS and RAN, which shows the superiority of the surprise criterion over the heuristic NC. In addition, the surprise criterion is simpler than the NC in the sense that it needs one threshold to determine redundancy whereas the NC requires two.

C. CO_2 Concentration Forecasting

The data consist of monthly average atmospheric CO_2 concentrations [in parts per million by volume (ppmv)] collected at Mauna Loa Observatory, HI, between 1958 and 2008 with a total of 603 observations [41]. The first 423 points are used for training and the last 180 points are for testing. The data are

TABLE I
NETWORK SIZES OF RAN, NC-KLMS, SC-KLMS, AND SC-KRLS

Algorithm	network size
RAN	361 ± 11
NC-KLMS	201 ± 11
SC-KLMS	109 ± 8
SC-KRLS	70 ± 9

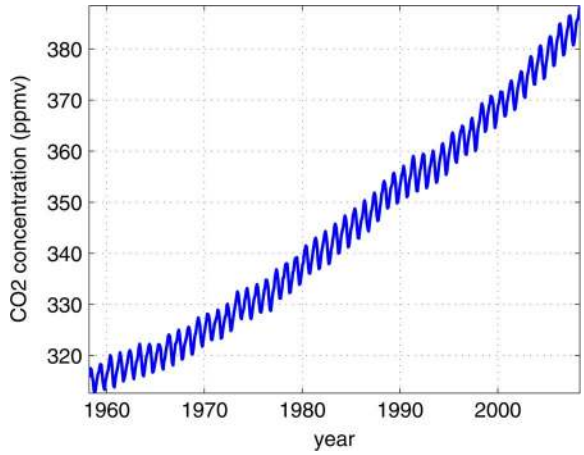

Fig. 12. CO_2 concentration trend from 1958 to 2008.

shown in Fig. 12. We try to model the CO_2 concentration as a function of time. Several features are immediately apparent: a long term rising trend, a pronounced seasonal variation, and some smaller irregularities. The problem of kernel design for this specific task is thoroughly discussed in [37]. We use the same kernel in this example. Our goal is to test how effective SC-KRLS is to model this nonlinear time series.

First, we simply assume all data are learnable and calculate the surprise of every point during training. The learning curve is the MSE calculated on the testing data. Fig. 13 shows the correspondence between the additions of informative data (cross) and drops in testing MSE (solid).

Next we show how effective SC-KRLS is to remove redundancy. A large T_1 is used to disable the abnormality detection. Fifty different T_2 are chosen from $[-1.5, 3]$. The result is illustrated in Fig. 14. The number of centers can be safely reduced from 423 to 77 with equivalent accuracy. By setting $T_2 = 0.03061$, we have the corresponding learning curves with the effective training data highlighted (circle) in Fig. 15. The two learning curves in Figs. 13 and 15 are almost the same even though the latter only uses 77 out of 423 total training data. This shows the feasibility and necessity of removing redundancy in learning. The long term prediction result from the last training is plotted in Fig. 16. As is clear, the prediction is very accurate at the beginning but deviates in the far future. Actually it is clear from Fig. 16 that the increase of the CO_2 concentration accelerates at an unforeseen speed.

VI. DISCUSSION

This paper presents an information theoretic criterion for online active learning. Active learning is crucial in many machine

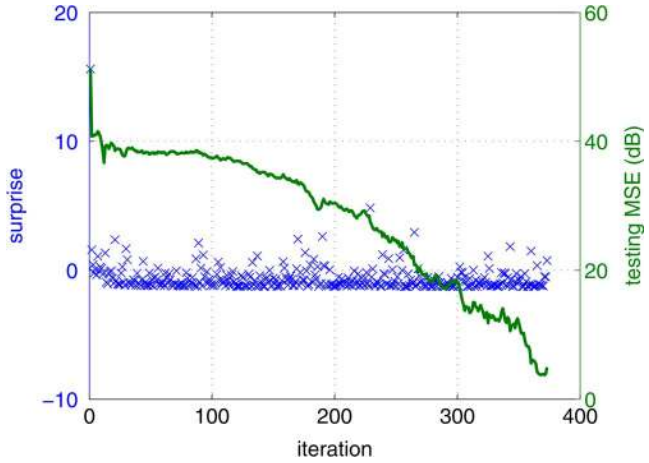

Fig. 13. Learning curve of SC-KRLS and a surprise measure of training data along iteration assuming all data learnable in CO_2 concentration forecasting.

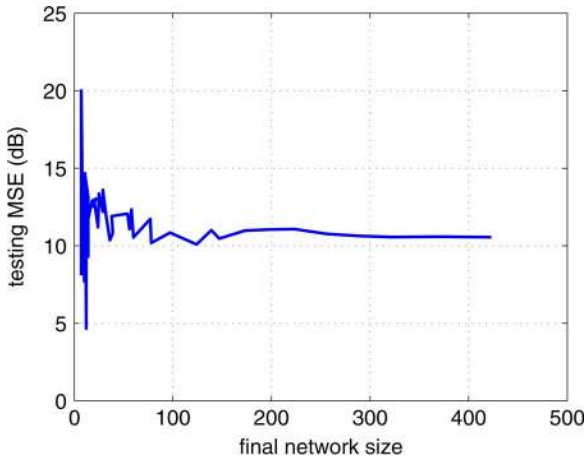

Fig. 14. Final network size versus testing MSE of SC-KRLS for different T_2 in CO_2 concentration forecasting.

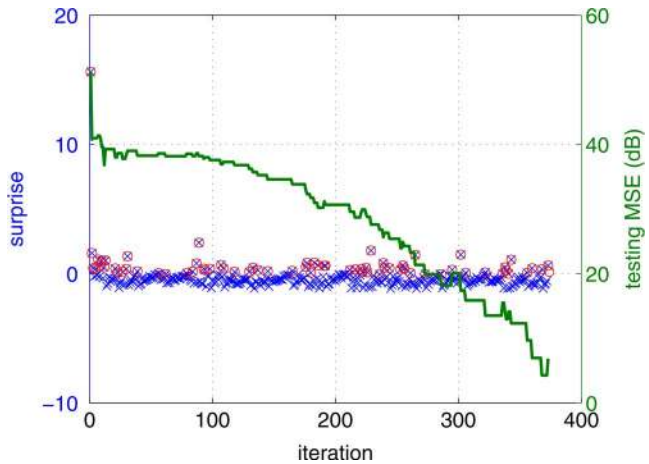

Fig. 15. Learning curve of SC-KRLS and a surprise measure of training data along iteration with effective examples circled in CO_2 concentration forecasting.

learning applications, as many meaningful learning systems interact with learning environments with state models. Therefore, not all new samples encountered contain the same information to update the system state. An information measure of an example which is "transferable" to the learning system is very significant. As we show theoretically and experimentally, the

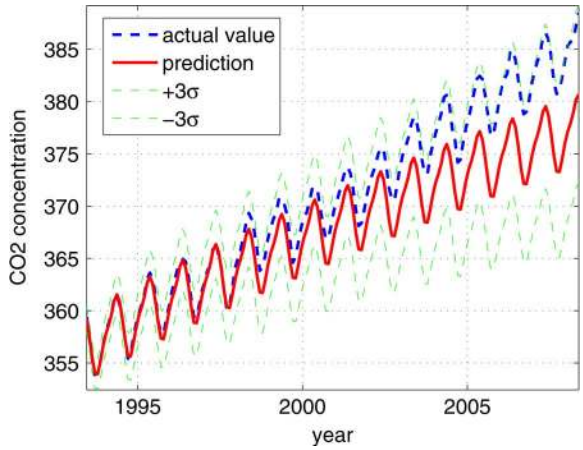

Fig. 16. Forecasting result of SC-KRLS ($T_2 = .0306$) for CO_2 concentration.

introduced surprise measure successfully quantifies the "transferable" information content contained in a datum with respect to the learning system, and the GP theory enables an elegant and still reasonably efficient algorithm to carry on the computation in real time.

We are particularly interested in applying the surprise concept to designing sparse kernel adaptive filters. We systematically study the surprise criterion kernel adaptive filters including SC-KRLS, SC-KLMS, and SC-KAPA. We theoretically highlight the close relationship between the surprise criterion and other existing methods such as NC and ALD. We experimentally show that the surprise criterion is superior or equivalent to existing methods in the simulations of nonlinear regression, short term chaotic time-series prediction, and long term time-series forecasting. Moreover, the surprise criterion is more principled and elegant in comparison with others.

There are many more applications that can benefit from this development. The interesting next question is how to apply the same technique for classification problems. The second interesting question is how to set the thresholds in a surprise criterion automatically. Any criterion requires some threshold picking. Cross validation is usually used as a default method. Since surprise is a more meaningful quantity, it might be easier than others to choose the thresholds theoretically. Another note is about learning with abnormal data. Throwing away abnormal data may not be efficient when the learning environment is nonstationary or subject to sudden changes. A feasible learning strategy to deal with abnormal data in a tracking mode is to make small, controlled adjustments using stochastic gradient descent. We leave all these interesting questions for future work.

REFERENCES

[1] J. Quinonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *J. Mach. Learn. Res.*, vol. 6, pp. 1939–1959, 2005.

[2] C. K. I. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. Cambridge, MA: MIT Press, 2001, vol. 13, pp. 682–688.

[3] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *J. Mach. Learn. Res.*, vol. 2, pp. 242–264, 2001.

[4] D. MacKay, "Information-based objective functions for active data selection," *Neural Comput.*, vol. 4, no. 4, pp. 590–604, 1992.

[5] A. J. Smola and P. L. Bartlett, "Sparse greedy Gaussian process regression," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2001, vol. 13, pp. 619–625.
[6] M. Seeger and C. Williams, "Fast forward selection to speed up sparse Gaussian process regression," in *Proc. Workshop Artif. Intell. Statist.*, 2003, pp. 205–212.
[7] L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. de Moor, "Subset based least squares subspace regression in RKHS," *Neurocomputing*, vol. 63, pp. 293–323, 2005.
[8] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2000, pp. 409–415.
[9] J. Quinonero-Candela and O. Winther, "Incremental Gaussian processes," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2003, vol. 15, pp. 1001–1008.
[10] K. Crammer, J. Kandola, and Y. Singer, "Online classification on a budget," in *Advances in Neural Information Processing Systems*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004, vol. 16, pp. 225–232.
[11] J. Kivinen, A. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.
[12] W. Liu, P. Pokharel, and J. Príncipe, "The kernel least mean square algorithm," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 543–554, Feb. 2008.
[13] 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.
[14] W. Liu and J. C. Príncipe, "The kernel affine projection algorithms," *EURASIP J. Adv. Signal Process.* 2008 [Online]. Available: http://www.hindawi.com/GetArticle.aspx?doi=10.1155/2008/784292
[15] K. Slavakis and S. Theodoridis, "Sliding window generalized kernel affine projection algorithm using projection mappings," *EURASIP J. Adv. Signal Process.* 2008 [Online]. Available: http://www.hindawi.com/GetArticle.aspx?doi=10.1155/2008/735351
[16] 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. 1058–1066, Mar. 2009.
[17] 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.
[18] S. Haykin, *Adaptive Filter Theory*, 4th ed. Englewood Cliffs, NJ: Prentice-Hall, 2002.
[19] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, 1991.
[20] L. Csato and M. Opper, "Sparse online Gaussian processes," *Neural Comput.*, vol. 14, pp. 641–668, 2002.
[21] G. Palm, "Evidence, information, and surprise," *Biol. Cybern.*, vol. 42, no. 1, pp. 57–68, 1981.
[22] E. Pfaffelhuber, "Learning and information theory," *Int. J. Neurosci.*, vol. 3, no. 2, pp. 83–88, 1972.
[23] L. Itti and P. Baldi, "Bayesian surprise attracts human attention," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2006, vol. 19, pp. 1–8.
[24] N. Ranasinghe and W. Shen, "Surprise-based learning for developmental robotics," in *Proc. ECSIS Symp. Learn. Adapt. Behav. Robot. Syst.*, 2008, pp. 65–70.
[25] P. A. N. Bosman and D. Thierens, "Negative log-likelihood and statistical hypothesis testing as the basis of model selection in IDEAs," in *Proc. Genetic Evol. Comput. Conf. (Late Breaking Papers)*, D. Whitley, Ed., Las Vegas, NV, 2000, pp. 51–58.
[26] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: Wiley-Interscience, 2000.
[27] C. Dima and M. Hebert, "Active learning for outdoor obstacle detection," in *Proc. Sci. Syst. I*, Aug. 2005.
[28] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, pp. 337–404, 1950.
[29] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Disc.*, vol. 2, no. 2, pp. 121–167, 1998.
[30] B. Widrow and M. E. Hoff, "Adaptive switching circuits," *IRE WESCON Convention Record*, vol. 4, pp. 96–104, 1960.
[31] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *Proc. 15th Int. Conf. Mach. Learn.*, 1998, pp. 515–521.
[32] J. A. K. Suykens, T. V. Gestel, J. Brabanter, B. D. Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.
[33] B. Schölkopf, B. Mika, C. J. C. Burges, P. Knirsch, K. Müller, G. Rätsch, and A. Smola, "Input space vs. feature space in kernel-based methods," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1000–1017, Sep. 1999.
[34] C. E. Shannon, "A mathematical theory of communication," in *Bell Syst. Tech. J.*, Jul. 1948, pp. 379–423.
[35] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley-Interscience, 1991.
[36] T. Poggio and S. Smale, "The mathematics of learning: Dealing with data," *Trans. Amer. Math. Soc.*, vol. 50, pp. 537–544, Nov. 2003.
[37] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.
[38] L. Glass and M. Mackey, *From Clocks to Chaos: The Rhythms of Life*. Princeton, NJ: Princeton Univ. Press, 1988.
[39] S. Mukherjee, E. Osuna, and F. Girosi, "Nonlinear prediction of chaotic time series using support vector machines," in *Proc. IEEE Workshop Neural Netw. Signal Process. VII*, J. Príncipe, L. Giles, N. Morgan, and E. Wilson, Eds., 1997, pp. 511–519.
[40] F. Takens, "Detecting strange attractors in turbulence," *Dyn. Syst. Turbulence*, pp. 366–381, 1981.
[41] P. Tans, "Trends in atmospheric carbon dioxide—Mauna loa," NOAA/ESRL, 2008 [Online]. Available: www.esrl.noaa.gov/gmd/ccgg/trends

Weifeng Liu (M'06) received the B.S. and M.S. degrees in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2003 and 2005, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, in 2008.

Upon graduation, he joined the Forecasting Team at Amazon.com, Seattle, WA. He has published nine journal papers, nine conference papers, and one book on kernel adaptive filtering (in print). His research interests include adaptive signal processing, machine learning, and data analysis.

Il Park received the B.S. degree in computer science from the Korea Advanced Institute of Science and Technology, Daejeon, China, Korea, in 2004 and the M.S. degree in electrical and computer engineering from University of Florida, Gainesville, in 2007, where he is currently working towards the Ph.D. degree in biomedical engineering.

He has been working at the Computational NeuroEngineering Laboratory under the supervision of Dr. J. C. Príncipe since 2005. His research interests include neural computation, point processes, signal processing, and machine learning.

José C. Príncipe (M'83–SM'90–F'00) he is currently the Distinguished Professor of Electrical and Biomedical Engineering at the University of Florida, Gainesville, where he teaches advanced signal processing and artificial neural networks (ANNs) modeling. He is BellSouth Professor and Founder and Director of the University of Florida Computational NeuroEngineering Laboratory (CNEL). He is involved in biomedical signal processing, in particular, the electroencephalogram (EEG) and the modeling and applications of adaptive systems. He has more than 150 publications in refereed journals, 15 book chapters, and over 400 conference papers. He has directed over 60 Ph.D. dissertations and 61 Master's degree theses.

Dr. Príncipe is the past Editor-in-Chief of the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, past President of the International Neural Network Society, and former Secretary of the Technical Committee on Neural Networks of the IEEE Signal Processing Society. He is an AIMBE Fellow and a recipient of the IEEE Engineering in Medicine and Biology Society Career Service Award. He is also a former member of the Scientific Board of the Food and Drug Administration, and a member of the Advisory Board of the McKnight Brain Institute at the University of Florida.