

# Sparse Approximation Through Boosting for Learning Large Scale Kernel Machines

Ping Sun and Xin Yao, *Fellow, IEEE*

**Abstract**—Recently, sparse approximation has become a preferred method for learning large scale kernel machines. This technique attempts to represent the solution with only a subset of original data points also known as basis vectors, which are usually chosen one by one with a forward selection procedure based on some selection criteria. The computational complexity of several resultant algorithms scales as  $\mathcal{O}(NM^2)$  in time and  $\mathcal{O}(NM)$  in memory, where  $N$  is the number of training points and  $M$  is the number of basis vectors as well as the steps of forward selection. For some large scale data sets, to obtain a better solution, we are sometimes required to include more basis vectors, which means that  $M$  is not trivial in this situation. However, the limited computational resource (e.g., memory) prevents us from including too many vectors. To handle this dilemma, we propose to add an ensemble of basis vectors instead of only one at each forward step. The proposed method, closely related to gradient boosting, could decrease the required number  $M$  of forward steps significantly and thus a large fraction of computational cost is saved. Numerical experiments on three large scale regression tasks and a classification problem demonstrate the effectiveness of the proposed approach.

**Index Terms**—Boosting, forward selection, kernel machines, large scale data mining, large scale problems, sparsification.

## I. INTRODUCTION

**I**N the last decade, kernel-based learning machines have drawn much attention due to their computational simplicity and excellent generalization performance in traditional machine learning problems such as classification, regression, and dimension reduction tasks [37]. Some well-known examples are regularized least squares (RLSs) and support vector machines (SVMs) [6] for supervised learning and kernel principal component analysis (KPCA) [48] for unsupervised learning. There are several different names for the RLS model [45] such as regularization networks (RNs) [16], kernel ridge regression

(KRR) [46], least squares support vector machines (LS-SVMs) [60], proximal support vector machine (PSVM) [21], kernel Fisher discriminant (KFD) [35], and more. We used the most straightforward one in the literature, i.e., RLS. A comprehensive introduction to kernel machines was described in [49] and [51]. Although kernel methods often lead to solving a simpler optimization problem (e.g., convex optimization) than, for example, multilayer neural networks [49], the computational complexity is still a main limitation in large scale applications. For example, a direct implementation of RLS requires inverting an  $N \times N$  kernel matrix with a cost of  $\mathcal{O}(N^3)$ , where  $N$  is the size of the training set. There are mainly two kinds of state-of-the-art techniques to address large scale kernel learning problem in the literature and they are: iterative optimization, and sparse approximations by representing the solution with only a subset of the original data points, known as basis vectors. The examples of iterative optimization include conjugate gradient (CG) [22], [43] and more efficient decomposition-based methods [27]–[29], [42]. The computational complexity of some iterative optimizations can be further reduced through approximating the matrix–vector product with  $N$ -body approaches [52], [64], but this technique is limited in practice due to the following: 1) it does not work well for high-dimensional data; and 2) a couple of critical parameters are required to be determined. In contrast, sparse approximation, which is easy to implement, flexible, and working well for any dimensionality, has recently become a preferred technique for learning kernel machines [25], [26], [62], [63], [66]. Other feasible approaches dealing with large scale kernel machines are parallel implementations [11] and distributed solutions [65] on a cluster of computers.

Numerous authors proposed a wealth of efficient sparse approximation algorithms especially for the RLS model [2], [7], [12], [23], [26], [53], [55], [59] under different names in the past. Most of those algorithms are based on a forward selection procedure, which iteratively picks up basis vectors one by one based on various selection criteria. The computational cost of several resultant algorithms could be linear in the number of training examples  $N$  and quadratic in the number of chosen basis vectors  $M$ , i.e.,  $\mathcal{O}(NM^2)$  and the corresponding storage scales as  $\mathcal{O}(NM)$ . The linearity in  $N$  is very desirable for large scale data sets while  $M$  was often thought as far less than the size of training data in the learning process and would not affect the total complexity too much. In fact, this is not true for large scale data sets as we will see in Section IV. We did observe that the more basis vectors we select, the better generalization results we could achieve in many applications. However, due to the limits of available computational resources, we cannot

Manuscript received March 11, 2009; revised January 05, 2010 and February 18, 2010; accepted February 18, 2010. Date of publication April 19, 2010; date of current version June 03, 2010. The work of P. Sun was supported by an ORS Award. The work of X. Yao was supported by the EPSRC grant GR/T10671/01.

P. Sun was with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. He is now with Birmingham Health and Wellbeing Partnership, Edgbaston, Birmingham B16 9NX, U.K. (e-mail: ping.sun@bhwp.nhs.uk).

X. Yao is with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. (e-mail: xin.yao@cs.bham.ac.uk).

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.2010.2044244

include many basis vectors in an ideal solution for those traditional forward selection algorithms. To address this dilemma, this paper investigates to pick up basis vectors group by group, i.e., selecting multiple basis vectors (not just one) at one forward step, thereby reducing the required number of forward steps to achieve a lower computational cost. The basic idea is motivated by the concept of boosting [20] explained below.

It is known that forward selection is closely related to a statistical view of boosting which is a well-known ensemble learning technique [19], [20]. The motivation for ensemble learning is that, rather than using just one single strong model for prediction, a linear combination of many individual weak models is used instead. Boosting provides a simple but effective way to construct the state-of-the-art ensemble models [5]. The most popular boosting algorithm is AdaBoost [18]. The idea is to sequentially apply some weak learning algorithm repeatedly to reweighted versions of the original data thereby giving a sequence of weaker learners (or base learners), denoted as  $T_1(\mathbf{x}), T_2(\mathbf{x}), \dots, T_M(\mathbf{x})$ . Here  $\mathbf{x}$  is the input and  $M$  represents the boosting steps. The final prediction is produced by linearly combining all of them, i.e.,  $F_M(\mathbf{x}) = \sum_{m=1}^M c_m T_m(\mathbf{x})$ , where the coefficients  $c_1, \dots, c_M$  are computed by the boosting procedure. The surprising success of AdaBoost attracted many researchers to study why and how the algorithm works from different perspectives such as probably approximately correct (PAC) learning, game theory, Vapnik–Chervonenkis (VC) theory, and statistical learning (see [20] and references therein). Of those, a statistical view has arguably become an intuitive and practical explanation on the mystery of AdaBoost<sup>1</sup> [34]. According to this perspective, AdaBoost can be seen as a forward stagewise modeling procedure optimizing an exponential loss function, shown in Algorithm 1.

---

**Algorithm 1** A basic boosting procedure

---

```

1:  $F_0(\mathbf{x}) = 0$ 
2: for  $m = 1$  to  $M$ 
3:    $(c_m, T_m(\mathbf{x})) = \arg \min_{c, T(\mathbf{x})} \sum_{n=1}^N L(y_n, F_{m-1}(\mathbf{x}_n) + cT(\mathbf{x}_n))$ 
4:    $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + c_m T_m(\mathbf{x})$ 
5: end for  $F(\mathbf{x}) = F_M(\mathbf{x}) = \sum_{m=1}^M c_m T_m(\mathbf{x})$ 

```

---

Here the loss function  $L(y, F)$  can be any differential function which corresponds to different boosting algorithms [19], [20]. In particular,  $L(y, F) = \exp(-y \cdot F)$  is for AdaBoost algorithm.

If we understand forward selection algorithms for learning kernel machines mentioned earlier as a boosting procedure (i.e., Algorithm 1),<sup>2</sup> we argue that previous work adds a too weak learner (i.e., a function associated with some basis vector) at each iteration, which leads to a very large number of boosting

<sup>1</sup>Discussions on which perspective gives a better explanation on the success of AdaBoost is beyond the scope of this paper. We just use the concept of boosting to introduce the motivation of the proposed work.

<sup>2</sup>The difference between forward selection here and boosting algorithm will be clarified in Section III-E.

steps required and is ultimately responsible for expensive computation. Built upon our previous work [57], we propose to include an ensemble of basis vectors instead of one at each boosting step and thereby reduce the required number  $M$  of boosting steps significantly. A large fraction of computational cost as well as memory requirements can be saved. For convenience, we refer to previous forward selection techniques as selection-based boosting methods and our newly proposed one as the ensemble-based boosting approach. We describe efficient learning algorithms for the case of RLS model in the current publication and the main procedure could be straightforwardly extended to other kernel models including kernel logistic regression (KLR) [66] and SVM.

The rest of this paper is organized as follows. In Section II, we briefly introduce kernel machines and summarize previous selection-based boosting techniques for sparse approximation. We investigate the ensemble-based boosting idea in Section III and efficient learning algorithms are provided. Section IV demonstrates the performance of the proposed approach on some large scale regression and classification data sets. We summarize the paper and present some future research directions in Section V.

## II. BACKGROUND

### A. Kernel Machines and Loss Functions

A large class of important kernel machines can be interpreted as the variational problem of finding the function  $f$  that minimizes the functional [16], [66]

$$\min_{f \in \mathcal{H}} H[f] = \sum_{n=1}^N V(y_n, f(\mathbf{x}_n)) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2 \quad (1)$$

where  $V(\cdot, \cdot)$  is a loss function,  $\mathcal{H}$  is the reproducing kernel Hilbert space (RKHS) associated with the Mercer kernel  $K(\cdot, \cdot)$ , and  $\lambda$  is a parameter that trades off the two terms.  $N$  is the size of the training set  $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ . The first term assesses the quality of the prediction  $f(\mathbf{x}_n)$  for the observed target  $y_n$ . The second one is called the regularization term and represents smoothness assumptions on  $f$ . The solution to the problem (1) was given by the well-known representer theorem [47] which shows that each minimizer  $f \in \mathcal{H}$  of  $H[f]$  has the form of

$$f(\mathbf{x}) = \sum_{n=1}^N a_n K(\mathbf{x}_n, \mathbf{x}). \quad (2)$$

Substituting  $f(\mathbf{x})$  for (1) and using the property of RKHS  $\langle K(\mathbf{x}_n, \cdot), K(\mathbf{x}_m, \cdot) \rangle = K(\mathbf{x}_n, \mathbf{x}_m)$ , we obtain

$$\min_{\mathbf{a}} H(\mathbf{a}) = \sum_{n=1}^N V(y_n, [\mathbf{K}\mathbf{a}]_n) + \frac{\lambda}{2} \mathbf{a}^\top \mathbf{K}\mathbf{a} \quad (3)$$

where  $\mathbf{a} = [a_1, \dots, a_N]^\top$  is the weight vector,  $\mathbf{K}$  is the so-called kernel matrix of size  $N \times N$  with the elements  $K_{nm} = K(\mathbf{x}_n, \mathbf{x}_m)$ ,  $n, m = 1, \dots, N$ , and  $[\cdot]_n$  denotes the  $n$ th component of some vector. By defining different types of loss functions, problem (3) corresponds to different kernel models. For example, soft margin loss function  $V(y_n, f(\mathbf{x}_n)) = \max\{0, 1 - y_n f(\mathbf{x}_n)\}^2$  leads to the primal

formulation of SVM, logistic loss function results in KLR, and RLS could be obtained by replacing  $V(\cdot, \cdot)$  with a simple squared loss, i.e.,  $V(y_n, f(\mathbf{x}_n)) = (1/2)(y_n - f(\mathbf{x}_n))^2$ .

The advantage of formulating different kernel machines as a unifying optimization problem (3) is that some learning algorithms originally proposed for one of them can be straightforwardly generalized to training other kernel machines. For example, CG methods were first suggested to solve the RLS model in the context of Gaussian process regression [22]. Very recently, Chapelle [8] extended it to deal with the SVM model based on this observation. The new algorithm does not require any quadratic programming (QP) optimization libraries as before. In the same way, the algorithms proposed for the RLS model in this paper apply to KLR and SVM models as well.

### B. Sparse Approximation

For RLS model, the resultant optimization (3) could be solved by directly inverting the matrix  $(\mathbf{K} + \lambda \mathbf{I})$ , which would result in  $\mathcal{O}(N^3)$  time and  $\mathcal{O}(N^2)$  memory. Instead of getting an exact optimal solution, we can employ CG method to obtain an approximate one in  $\mathcal{O}(N^2 \cdot \#iters)$  time, where  $\#iters$  is the number of CG iterations. By using an early stopping strategy,  $\#iters$  could be set to a much smaller value than  $N$  for large data sets [22], and therefore, the computational cost is saved to some extent. A very popular technique to deal with the optimization (3) is sparse approximation, where “sparse” means that many of the entries in the estimate  $\mathbf{a}$  to (3) are zeros. It implies that the function  $f(\mathbf{x})$  takes a more compact representation

$$f_M(\mathbf{x}) = \sum_{m=1}^M a_m K(\tilde{\mathbf{x}}_m, \mathbf{x}) \quad (4)$$

where  $M < N$  is the number of nonzero entries of the estimate  $\mathbf{a}$ , and  $\{\tilde{\mathbf{x}}_m, j = 1, \dots, M\}$  are corresponding data points, also referred to as basis vectors and indexed by the set  $I_m = \{i_1, \dots, i_m\}$ . By this sparse representation, the optimization (3) can be rewritten as (we take RLS as an example)

$$\begin{aligned} \min_{\mathbf{a}_M} H(f_M) &= \frac{1}{2} \sum_{n=1}^N (y_n - f_M(\mathbf{x}_n))^2 + \frac{\lambda}{2} \|f_M\|_{\mathcal{H}}^2 \\ &= \frac{1}{2} \|\mathbf{y} - \mathbf{K}_M \mathbf{a}_M\|^2 + \frac{\lambda}{2} \mathbf{a}_M^\top \mathbf{Q}_M \mathbf{a}_M \end{aligned} \quad (5)$$

where  $[\mathbf{K}_M]_{nm} = K(\mathbf{x}_n, \tilde{\mathbf{x}}_m)$ ,  $[\mathbf{Q}_M]_{mm'} = K(\tilde{\mathbf{x}}_m, \tilde{\mathbf{x}}_{m'})$ ,  $n = 1, \dots, N$ ;  $m, m' = 1, \dots, M$ , and  $\mathbf{a}_M = [a_1, \dots, a_M]^\top$ . The optimal solution to (5) is

$$\mathbf{a}_M = (\mathbf{K}_M^\top \mathbf{K}_M + \lambda \mathbf{Q}_M)^{-1} \mathbf{K}_M^\top \mathbf{y}. \quad (6)$$

If the index set  $I_m$  is known, it is clear to see that sparse formulation (5) just leads to the cost of  $\mathcal{O}(NM^2)$  time and  $\mathcal{O}(NM)$  memory.

Much research has gone into the problem of choosing an appropriate subset  $I_m$  which has a crucial effect on the overall computational and generalization performance. Often this problem is formulated as a forward selection process and at each step one basis vector is chosen from all the unselected

training points based on some selection criteria. A variety of approaches have been developed.<sup>3</sup> For instance, matching pursuit (MP) techniques, which were originally addressed for an unregularized least squares objective [9], [32], [39], [44], have been widely applied to RLS [26], [58] and other general loss functions [23], [62], [66]. By using the property that the kernel matrix  $\mathbf{K}$  is strictly positive definite, Nair *et al.* [38] developed a fast forward selection criterion MAX-RES by simply choosing the basis vector with the largest absolute value in the current residual, which only leads to a very minor selection cost [1], [38]. Another very fast method was presented in [50] by optimizing an INFO-GAIN criterion but it sometimes leads to poor predictive results [26]. Several forward-selection-based techniques were also proposed based on adaptively approximating the kernel matrix  $\mathbf{K}$  with low-rank ones in the measure of matrix trace as done in [2], [3], and [17]. Engel *et al.* [15] suggested an online forward sparsification scheme which includes basis vectors one by one by checking the violation of a so-called approximate-linear-dependent (ALD) condition. Simply speaking, if the candidate vector cannot be linearly approximated by previous selected basis vectors in the kernel induced feature space, it will be added as a basis.<sup>4</sup> The ALD method does not exploit the label information in the selection of basis vectors. Jung and Polani [24] proposed a two-part criterion which extends ALD method by taking into account the contribution of basis vector to the reduction of the cost function [e.g., (5)], similar to the one used in resource-allocating networks (RANs) [41].<sup>5</sup>

Both ALD and RAN methods can be viewed as special cases of so-called kernel affine projection algorithms (KAPA) [30]. Finally, a general probabilistic speedup strategy was proposed in [54], where the authors showed that we do not necessarily choose the best from all the remaining training points at each iteration and it is sufficient to pick up the best from evaluating a much smaller subset of candidate vectors.

Of those approaches, many of them have a favorable linearly scaling computational property in the number of training points, such as the work of [26], [38], and [58]. Very impressive predictive performances have been reported as well. However, the computational cost of all these algorithms is still quadratic in the number of basis vectors and it is not well suited for the situation of requiring more basis vectors in large applications.<sup>6</sup> The reason for this could be explicitly interpreted from the viewpoint of ensemble learning [13], where forward selection can be

<sup>3</sup>Although some approaches (e.g., [15] and [38]) are not exactly proposed for the kernel machine framework (3), the main ideas still apply here.

<sup>4</sup>Let  $\Delta\delta$  be approximation error. An ALD qualified basis should be satisfying with  $\Delta\delta > \tau_1$ , where  $\tau_1 > 0$  is the approximation level parameter.

<sup>5</sup>Let  $\Delta H$  be the reduction in the cost function if supposing that the candidate vector is chosen as a basis. The RAN method will pick it up only if  $\Delta\delta \cdot \Delta H > \tau_2$ , where  $\tau_2 > 0$  is the parameter.

<sup>6</sup>Together with some sequential weight updating algorithms (e.g., least mean square), the computational complexity of the ALD- and RAN-based methods could be linear in both the size of training set and the number of basis vectors, i.e.,  $\mathcal{O}(NM)$  [30]. This paper focuses on the batch-like weight updating scheme and is mainly concerned with the efficiency of basis selections. In the future, it is worthwhile to investigate combining the presented work with a sequential weight updating scheme to tackle very large applications where it is impossible to load all the training points in memory.

seen as a special case of generic boosting procedures [19], [20]. We regard the sparse model (4) as an ensemble model and each kernel term associated with some basis vector is the so-called base learner. As the base learner only includes one kernel term and it is too weak for large complex problems, we have to include many base learners to achieve the desired generalization performance by running the forward-selection-based boosting procedure many steps. An intuitive question here is: Can we add a “stronger” base learner at each single step so that the number of forward boosting steps can be reduced? One way to make the base learner stronger for kernel machines is to add a combination of multiple kernel terms into the current model at each learning stage. This new approach will be detailed in the next section.

### III. A BOOSTING APPROACH

As an extension to (4), the sparse model here we considered has a general form of

$$F_M(\mathbf{x}) = \sum_{m=1}^M c_m f_m(\mathbf{x}) \quad (7)$$

where  $c_m$  is the weight and  $f_m(\mathbf{x})$  denotes the  $m$ th base learner which could be flexibly consisting of multiple kernel terms

$$f_m(\mathbf{x}) = \sum_{s=1}^S a_s^m K(\tilde{\mathbf{x}}_s^m, \mathbf{x}) \quad (8)$$

where  $S$  is the number of kernel terms<sup>7</sup> for each base learner, and  $\tilde{\mathbf{x}}_s^m$  and  $a_s^m$  are corresponding basis vectors and weights, respectively. Note that if  $S = 1$ , the model representation (7) is the same to that of (4).

In order to construct such a model (7) in a forward boosting procedure, we have to answer three questions which are involved in each forward boosting step.

- 1) Given the  $m$ th base learner  $f_m(\cdot)$ , how do we calculate the weight vector  $\mathbf{c}_m = [c_1, \dots, c_m]$  defined in (7)?
- 2) Given  $S$  basis vectors, how do we combine them into the  $m$ th learner (8), i.e., computing the vector  $\mathbf{a}^m = [a_1^m, \dots, a_S^m]$ ?
- 3) How do we choose the set of basis vectors  $\{\tilde{\mathbf{x}}_1^m, \dots, \tilde{\mathbf{x}}_S^m\}$  for the  $m$ th learner to be included?

In the following, we will answer these three questions one by one and then present the entire structure of the proposed approach.

#### A. Computing the Weight Vector

Similar to the derivation of the sparse formulation (5), we replace the functional  $f(\cdot)$  in (1) with  $F_M(\cdot)$  defined in (7) and then obtain

$$\begin{aligned} H(F_M) &= \frac{1}{2} \sum_{n=1}^N (y_n - F_M(\mathbf{x}_n))^2 + \frac{\lambda}{2} \|F_M\|_{\mathcal{H}}^2 \\ &= \frac{1}{2} \|\mathbf{y} - \mathbf{F}_M \mathbf{c}_M\|^2 + \frac{\lambda}{2} \mathbf{c}_M^\top \mathbf{\Omega}_M \mathbf{c}_M \end{aligned} \quad (9)$$

<sup>7</sup>For convenience, here we assume that all the base learners are composed of the same number of basis vectors. It is straightforwardly extendable for the situation where  $S$  may be different at different boosting steps.

where  $\mathbf{c}_M = [c_1, \dots, c_M]^\top$ , the elements of the matrices  $\mathbf{F}_M$  and  $\mathbf{\Omega}_M$  are defined, respectively, by

$$F_{nm} \triangleq \sum_{s=1}^S a_s^m K(\mathbf{x}_n, \tilde{\mathbf{x}}_s^m), \quad n = 1, \dots, N, \quad m = 1, \dots, M \quad (10)$$

and

$$\Omega_{mm'} \triangleq \langle f_m(\cdot), f_{m'}(\cdot) \rangle_{\mathcal{H}} = \sum_{s=1}^S \sum_{s'=1}^S a_s^m a_{s'}^{m'} K(\tilde{\mathbf{x}}_s^m, \tilde{\mathbf{x}}_{s'}^{m'}) \quad (11)$$

where  $m, m' = 1, \dots, M$  and the operator  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$  denotes the inner product in RKHS  $\mathcal{H}$ .

Note that the property of reproducing kernel has been used in the above calculations. It is easy to see

$$\mathbf{c}_M = (\mathbf{F}_M^\top \mathbf{F}_M + \lambda \mathbf{\Omega}_M)^{-1} \mathbf{F}_M^\top \mathbf{y}. \quad (12)$$

Therefore, it is not surprising that the calculations of  $\mathbf{c}_M$  are the same as those of  $\mathbf{a}_M$  in the traditional sparse formulation (5) just by replacing  $\mathbf{K}_M, \mathbf{Q}_M$  with  $\mathbf{F}_M, \mathbf{\Omega}_M$ , respectively.

#### B. Computing the Combination Vector

For previous forward-selection-based boosting methods, we need some criteria to evaluate the goodness of one basis vector. Similarly, we are also required to have some criteria to evaluate a base learner which is composed of multiple kernel terms in our ensemble-based boosting approach. Assume that we have an ensemble model  $F_{M-1}(\mathbf{x})$  and a base learner candidate  $f_M^*(\mathbf{x})$ , i.e.,

$$f_M^*(\mathbf{x}) = \sum_{s=1}^S a_s^* K(\tilde{\mathbf{x}}_s^M, \mathbf{x}) \quad (13)$$

and let  $\mathbf{a}^* = [a_1^*, \dots, a_S^*]^\top$ .

We could use some of the basis selection criteria (e.g., MP approach [26]), summarized in Section II, to search for some  $\mathbf{a}^*$  for optimally combining  $S$  kernel terms. Here we look for a more principled approach which is motivated by the work of generic gradient-based boosting algorithms [19], [33]. Following Friedman's pioneering work [19], the base learner could be chosen by maximizing the correlation to the functional gradient on the loss function. In our case, the loss function is

$$H(f) = \frac{1}{2} \sum_{n=1}^N (y_n - f(\mathbf{x}_n))^2 + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2 \quad (14)$$

and the gradient of  $H(f)$  with respect to (w.r.t.) the functional  $f$  in the RKHS  $\mathcal{H}$  is calculated as

$$\left. \frac{\partial H(f)}{\partial f} \right|_{f=F_{M-1}} = \sum_{n=1}^N (f(\mathbf{x}_n) - y_n) K(\mathbf{x}_n, \cdot) + \lambda \sum_{m=1}^{M-1} c_m f_m \cdot \quad (15)$$

Then, the candidate  $f_M^*(\mathbf{x})$  could be optimized such that it is most collinear with the direction of steepest descent [62], i.e.,

$$f_M(\cdot) = \arg \max_{\mathbf{a}^*} \left\{ \frac{\left\langle f_M^*(\cdot), -\frac{\partial H(f)}{\partial f} \right\rangle_{\mathcal{H}}^2}{\left\langle f_M^*(\cdot), f_M^*(\cdot) \right\rangle_{\mathcal{H}}} \right\}. \quad (16)$$

Combining with (15), the criterion (16) can be simplified as

$$f_M(\cdot) = \arg \max_{\mathbf{a}^*} \left\{ \frac{\left[ \mathbf{f}_M(\mathbf{a}^*)^\top \mathbf{r}^{M-1} - \lambda \boldsymbol{\omega}_M(\mathbf{a}^*)^\top \mathbf{c}_{M-1} \right]^2}{\omega_M^*(\mathbf{a}^*)} \right\} \quad (17)$$

where

$$\begin{aligned} \mathbf{f}_M(\mathbf{a}^*) &= \mathbf{K}_{N,S} \mathbf{a}^* \\ \boldsymbol{\omega}_M(\mathbf{a}^*) &= \mathbf{Q}_{M-1,S} \mathbf{a}^* \\ \omega_M^*(\mathbf{a}^*) &= (\mathbf{a}^*)^\top \mathbf{Q}_{S,S} \mathbf{a}^* \end{aligned} \quad (18)$$

$\mathbf{r}^{M-1}$  is the residual, and  $\mathbf{c}_{M-1}$  is the weight vector after  $(M-1)$  forward boosting steps. Note that in (17) only the quantities  $\mathbf{f}_M, \boldsymbol{\omega}_M, \omega_M^*$  are dependent on the combination vector  $\mathbf{a}^*$ . In the above expressions, we have defined the matrix  $\mathbf{K}_{N,S}$ ,  $\mathbf{Q}_{M-1,S}$ , and  $\mathbf{Q}_{S,S}$  with the elements, respectively

$$[\mathbf{K}_{N,S}]_{ns} = K(\mathbf{x}_n, \tilde{\mathbf{x}}_s^M), \quad n = 1, \dots, N, \quad s = 1, \dots, S \quad (19)$$

$$\begin{aligned} [\mathbf{Q}_{M-1,S}]_{ms} &= \sum_{s'=1}^S a_{s'}^m K(\tilde{\mathbf{x}}_{s'}^m, \tilde{\mathbf{x}}_s^M), \\ m &= 1, \dots, M-1, \quad s = 1, \dots, S \end{aligned} \quad (20)$$

and

$$[\mathbf{Q}_{S,S}]_{ss'} = K(\tilde{\mathbf{x}}_s^M, \tilde{\mathbf{x}}_{s'}^M), \quad s, s' = 1, \dots, S. \quad (21)$$

By now, we have obtained a new evaluation criterion (17) for optimizing base learners at each ensemble-based boosting step. Now let us have a closer look at this criterion (17), that is

$$\begin{aligned} & \frac{[\mathbf{f}_M^\top \mathbf{r}^{M-1} - \lambda \boldsymbol{\omega}_M^\top \mathbf{c}_{M-1}]^2}{\omega_M^*} \\ &= \frac{\left[ (\mathbf{a}^*)^\top (\mathbf{K}_{N,S}^\top \mathbf{r}^{M-1} - \lambda \mathbf{Q}_{M-1,S}^\top \mathbf{c}_{M-1}) \right]^2}{(\mathbf{a}^*)^\top \mathbf{Q}_{S,S} \mathbf{a}^*}. \end{aligned} \quad (22)$$

It is obvious that the expression in (22) is the maximization of the so-called Rayleigh quotient [36]. The optimal solution  $\mathbf{a}^*$  corresponds to the eigenvector with the largest eigenvalue of the matrix

$$\mathbf{Q}_{S,S}^{-1} (\mathbf{u} \mathbf{u}^\top) \quad (23)$$

where

$$\mathbf{u} = \mathbf{K}_{N,S}^\top \mathbf{r}^{M-1} - \lambda \mathbf{Q}_{M-1,S}^\top \mathbf{c}_{M-1}. \quad (24)$$

Instead of solving an eigenvalue problem, we can simplify this computation and obtain

$$\mathbf{a}^* = a_0 \mathbf{Q}_{S,S}^{-1} \mathbf{u} = a_0 \mathbf{v} \quad (25)$$

where  $\mathbf{v} = \mathbf{Q}_{S,S}^{-1} \mathbf{u}$  and  $a_0$  is an arbitrary constant. Even though the value of  $a_0$  has no effect on the final prediction results, we still want the resultant base learner  $f_M(\cdot)$  having a good prediction ability. One way of achieving this is to make the output of  $M$ th base learner  $\mathbf{f}_M = \mathbf{K}_{N,S} \mathbf{a}^*$  and the current residual  $\mathbf{r}^{M-1}$  as close as possible, i.e., minimizing the squared error

$$\min_{a_0} \|\mathbf{f}_M - \mathbf{r}^{M-1}\|^2 = \|a_0 \mathbf{K}_{N,S} \mathbf{v} - \mathbf{r}^{M-1}\|^2 \quad (26)$$

which gives

$$a_0 = \frac{(\mathbf{K}_{N,S} \mathbf{v})^\top \mathbf{r}^{M-1}}{(\mathbf{K}_{N,S} \mathbf{v})^\top (\mathbf{K}_{N,S} \mathbf{v})}.$$

Note that, by setting  $S = 1$ , the expression (22) can also be used as an alternative selection criterion, referred to as BOOST, for scoring basis vectors in selection-based boosting procedures and it can be simplified as

$$\frac{1}{2} \frac{(\mathbf{k}_j^\top \mathbf{r}^{M-1} - \lambda \mathbf{q}_j^\top \mathbf{a}_{M-1})^2}{q_j^*}, \quad j \notin I_{M-1} \quad (27)$$

where  $\mathbf{x}_j$  is a basis candidate and  $[\mathbf{k}_j]_n = K(\mathbf{x}_n, \mathbf{x}_j)$ ,  $n = 1, \dots, N$ ,  $[\mathbf{q}_j]_m = K(\tilde{\mathbf{x}}_m, \mathbf{x}_j)$ ,  $m = 1, \dots, M-1$ ,  $q_j^* = K(\mathbf{x}_j, \mathbf{x}_j)$ . This alternative is closely related to the MP approach [26], which suggests that by solving a sequence of 1-D minimization problems over the unselected basis candidates, the point which leads to the greatest reduction in the objective (5) can be chosen as a new basis vector. The resulting selection criterion is equivalent to evaluate

$$\frac{1}{2} \frac{(\mathbf{k}_j^\top \mathbf{r}^{M-1} - \lambda \mathbf{q}_j^\top \mathbf{a}_{M-1})^2}{\lambda q_j^* + \mathbf{k}_j^\top \mathbf{k}_j}, \quad j \notin I_{M-1}. \quad (28)$$

The only difference between BOOST and MP criteria lies in the denominator part and BOOST should be faster than MP since an extra computation of  $\mathbf{k}_j^\top \mathbf{k}_j$  is required for evaluating (28). Empirically, MP slightly outperforms BOOST in terms of predictive accuracy. However, if extending MP to be the evaluation criterion of base learners in ensemble-based boosting, the optimal  $\mathbf{a}^*$  has a form of

$$\mathbf{a}^* = a_0 (\lambda \mathbf{Q}_{S,S} + \mathbf{K}_{N,S}^\top \mathbf{K}_{N,S})^{-1} \mathbf{u} \quad (29)$$

which is much more computationally expensive than that of (25). That is the reason why we did not use the MP criterion in our proposed work.

### C. Initializing the Base Learner

As we have mentioned above, the weakness of selection-based boosting methods is that only one basis vector is selected at each boosting step. This leads to a large number of boosting steps required as well as heavy computational burden. Hence we propose to add a much stronger base learner at each boosting step, i.e., setting  $S$  in (8) to a larger value. Now an important issue is how to initialize the base learner (i.e., basis vectors) in the course of boosting procedure. We cannot use the strategy of ranking a  $\kappa$ -size ( $\kappa > S$ ) set of basis candidates at each step [53] and picking up the best  $S$  ones since  $\kappa$  is very large for a large  $S$ . Scoring a large number of basis candidates

TABLE I

A SUMMARY OF COMPUTATIONAL COMPLEXITY FOR BOTH SELECTION-BASED AND ENSEMBLE-BASED BOOSTING ALGORITHMS. NOTE THAT  $M$  DENOTES THE NUMBER OF BASIS VECTORS FOR SELECTION-BASED ALGORITHM AS WELL AS THE NUMBER OF BOOSTING STEPS. HOWEVER, IN THE CASE OF ENSEMBLE-BASED ALGORITHM,  $M$  JUST REPRESENTS THE NUMBER OF BOOSTING STEPS AND THE NUMBER OF BASIS VECTORS IS  $S \cdot M$ . TO ARRIVE AT A SIMILAR TEST ACCURACY, THE  $M$  FOR ENSEMBLE-BASED ALGORITHMS IS USUALLY MUCH SMALLER THAN THAT ONE FOR SELECTION-BASED APPROACHES; SEE THE TEXT FOR DETAILS

Algorithm	TRAINING		TESTING	
	Time complexity	Memory	Time complexity	Memory
Selection-based boosting	$\mathcal{O}(NM^2)$	$\mathcal{O}(NM)$	$\mathcal{O}(M)$	$\mathcal{O}(M)$
Ensemble-based boosting	$\mathcal{O}(TS^2M + NSM + NM^2)$	$\mathcal{O}(N \cdot \max\{S, M\})$	$\mathcal{O}(SM)$	$\mathcal{O}(SM)$
	$\mathcal{O}(NM^2)$	$\mathcal{O}(NM)$		

on the whole training data is computationally prohibitive for large scale data sets. We need to pursue a cheaper method.

Motivated by the idea of divide-and-conquer strategy, we can initialize the base learner, at each boosting step, by running a fast forward-selection-based boosting algorithm to choose  $S$  basis vectors on a random subset of the entire training data set while retaining the current residual as the target vector instead of original  $\mathbf{y}$ . Let  $T$  be the size of this random subset. According to the empirical results conducted in our previous work [56], the method MAX-RES [1], [38] is an excellent one among those fastest selection-based boosting algorithms and becomes a natural choice of accomplishing this initialization task. Certainly, we could also employ other computationally slower selection criteria [26], [58] to seek a better generalization.

#### D. Structure of the Proposed Approach

Combining the answers to the three questions raised at the beginning of this section, we summarize the steps of our ensemble-based boosting method as follows.

- 1) Parameters setting
  - a) size  $T$  of random training subset (e.g.,  $T = 500$ );
  - b) number  $S$  of chosen basis vectors for each base learner (e.g.,  $S = 50$ );
  - c) maximal number  $M$  of boosted base learners (e.g.,  $M = 500$ ).
- 2)  $F_0(\mathbf{x}) = 0$ ,  $\mathbf{r}^0 = \mathbf{y}$ , and  $m = 0$ .
- 3) While  $m < M$ 
  - a) randomly generate a  $T$ -size subset  $\mathcal{D}^m$  from the whole training data and replace the target vector with the corresponding residual  $\mathbf{r}^{m-1}$ ;
  - b) select  $S$  basis vectors from  $\mathcal{D}^m$  by running some selection-based boosting algorithm (e.g., MAX-RES [38]) on  $\mathcal{D}^m$ ;
  - c) compute the optimal vector  $\mathbf{a}^m$  to (17) and evaluate the optimized base learner  $f_m(\mathbf{x}) = \sum_{s=1}^S a_s^m K(\tilde{\mathbf{x}}_s^m, \mathbf{x})$  by (25);
  - d) absorb  $f_m(\mathbf{x})$  into the ensemble model  $F_m(\mathbf{x})$  by recursively upgrading  $\mathbf{c}_{m-1}, \mathbf{r}_{m-1}$  to  $\mathbf{c}_m, \mathbf{r}_m$ , respectively, and  $m = m + 1$ .
- 4) Output the ensemble model  $F(\mathbf{x}) = F_M(\mathbf{x}) = \sum_{m=1}^M c_m f_m(\mathbf{x})$ .

Note that the implementation of step 3d) is exactly the same as the updates of  $\mathbf{a}_m$  to (5) involved in all the selection-based boosting algorithms. A similar computational complexity incurs, i.e.,  $\mathcal{O}(NM^2)$  time and  $\mathcal{O}(NM)$  memory, for  $M$  forward boosting steps. Another major computational cost caused by the

TABLE II  
PROPERTIES OF FOUR DATA SETS

Data set	#training	#testing	#dimension
OUTAOUAIS	20,000	9,000	37
KIN40K	36,000	4,000	8
SARCOS	44,484	4,449	21

presented approach is the step of optimizing the base learner [i.e. step 3c)], which leads to a total complexity of  $\mathcal{O}(N \cdot S \cdot M)$  time and  $\mathcal{O}(N \cdot S)$  memory.<sup>8</sup> For the step 3b), all the  $M$  boosting steps result in a total  $\mathcal{O}(M \cdot T \cdot S^2)$  time complexity.

In practical implementation, the number  $S$  is a fraction of  $T$  (e.g.,  $\leq 10\%$ ), normally set to be less than  $M$  and the product  $(M \cdot T)$  scales as  $\mathcal{O}(N)$ . Therefore, the overall complexity of the proposed approach is still dominated by  $\mathcal{O}(NM^2)$  time and  $\mathcal{O}(NM)$  memory. For the ensemble-based boosting approach, the value of  $M$  is very trivial in contrast to the number of training examples. Although many selection-based boosting algorithms appear to have a similar computational complexity, the value of  $M$  for selection-based techniques is significantly larger than the setting for the ensemble-based boosting method especially on large data sets. According to this analysis, we expect our ensemble-based boosting could dramatically reduce the computational cost as well as memory requirement while maintaining a similar test accuracy when compared to those forward-selection-based boosting algorithms on large scale problems.

Note that, for selection-based methods,  $M$  is the number of basis vectors as well as the steps of forward selection. But for the ensemble-based boosting algorithm,  $M$  is only equal to the boosting steps and the number of basis vectors is  $(M \cdot S)$ . As selection-based methods optimize the weights of basis vectors “one-by-one” as in (6) whereas ensemble-based methods do this “group-by-group” as in (12), it is not surprising that the latter requires more basis vectors than the former for achieving the same predictive performance. Therefore, ensemble-based boosting needs a longer time to evaluate the same test set than that of selection-based techniques. For a fair numerical comparison, in the experiments section, the central processing unit (CPU) time reported consists of both training and testing time. Table I summarizes the computational complexity in terms of time and memory for both selection-based and ensemble-based algorithms.

<sup>8</sup>This cost consists of calculating both  $\mathbf{u}$  and  $\mathbf{Q}_{S,S}^{-1}\mathbf{u}$  for  $M$  boosting steps. It is clear that two parts lead to the computational cost  $\mathcal{O}(N \cdot S \cdot M)$  and  $\mathcal{O}(M \cdot S^3)$ , respectively. Since  $M \cdot S$  is the number of basis vectors and it should be less than  $N$ , the later one becomes  $\mathcal{O}(N \cdot S^2)$  which will be absorbed into the total complexity  $\mathcal{O}(N \cdot S \cdot M)$  by considering  $S < M$ .

TABLE III  
PARAMETER SETTINGS FOR ALL SIX METHODS ON THREE REGRESSION DATA SETS

Data set	Random	MP	Max-Res	ALD	RAN	Ensemble
OUTAOUAIS	$M = 1500, \tau_1 = 10^{-2}$ (ALD), $\tau_2 = 10^{-4}$ (RAN)					$T = 500, S = 50, M = 500$
KIN40K	$M = 1000, \tau_1 = 10^{-2}$ (ALD), $\tau_2 = 10^{-4}$ (RAN)					$T = 500, S = 50, M = 500$
SARCOS	$M = 1000, \tau_1 = 10^{-2}$ (ALD), $\tau_2 = 10^{-4}$ (RAN)					$T = 500, S = 50, M = 500$

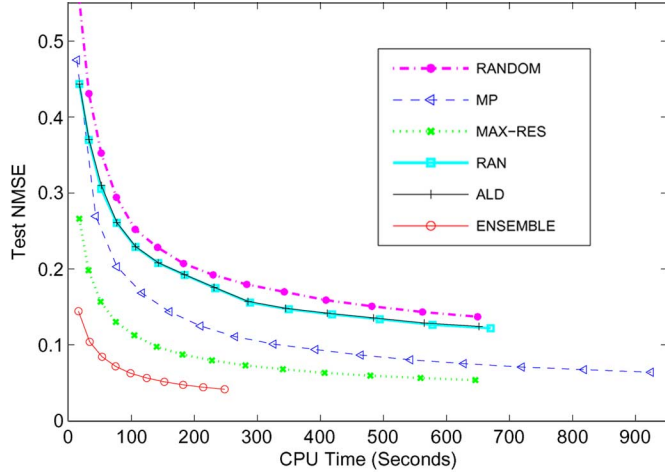


Fig. 1. Test NMSE comparisons of selection- and ensemble-based boosting on the OUTAOUAIS data set as a function of running time in seconds.

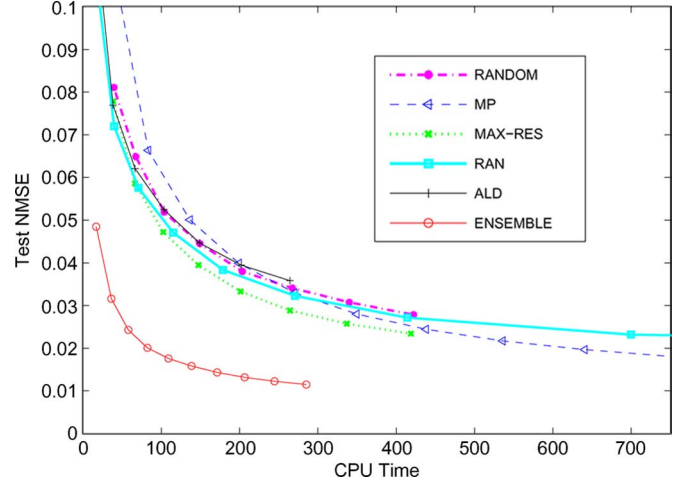


Fig. 2. Test NMSE comparisons of selection- and ensemble-based boosting on the KIN40K data set as a function of running time in seconds.

E. Relations to Gradient Boosting

In general, the proposed ensemble-based boosting approach can be interpreted as a gradient boosting procedure [19] by using sparse kernel models (8) instead of more traditional decision trees as the base learners. However, there are at least two obvious differences between them.

First, gradient boosting does not include a regularization term in the objective and utilizes an empirical method, i.e., the so-called “shrinkage” in statistics to avoid overfitting. The idea is not to include the base learner fully into the ensemble model  $F(\mathbf{x})$  and just absorb it partially, that is

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + (\nu \cdot c_m) f_m(\mathbf{x}) \quad (30)$$

where  $0 < \nu \leq 1$  is a shrinkage factor. It was found that this strategy greatly improved the generalization performance [19]. The cost paid for better performance is increased computation since a huge number of base learners are required. This conflicts with our expected objective of reducing computational cost for large scale kernel machines. For our ensemble-based boosting approach, the objective function (9) includes a regularization term  $(\lambda/2) \mathbf{c}_M^T \mathbf{\Omega}_M \mathbf{c}_M$ , which could fulfill the task of controlling the model complexity effectively while not increasing the required number of base learners.

Second, gradient boosting fixes the weights of the previously entered base learners while ensemble-based boosting proposed in this work adapts them fully via (12) when a new base learner is added. We refer to two different schemes as ONE-UPDATE and FULL-UPDATE, respectively. In detail, the ONE-UPDATE scheme computes the  $M$ th weight  $c_M$  by optimizing (9) while fixing  $c_{M-1}$  and it turns out to be

$$c_M = \frac{\mathbf{f}_M^T \mathbf{r}^{M-1} - \lambda \mathbf{\omega}_M^T \mathbf{c}_{M-1}}{\lambda \mathbf{\omega}_M^* + \mathbf{f}_M^T \mathbf{f}_M} \quad (31)$$

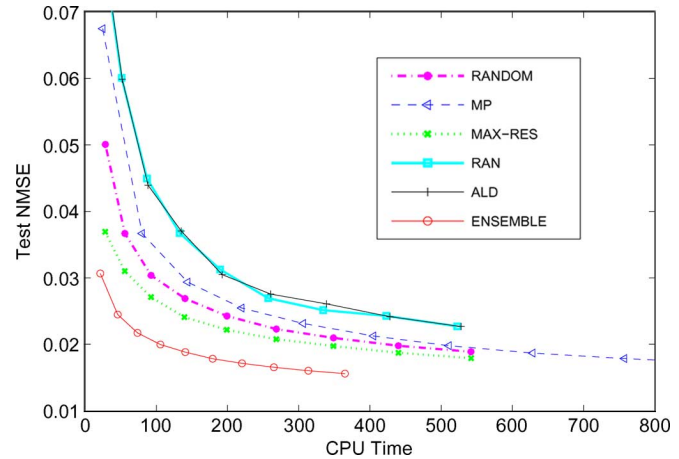


Fig. 3. Test NMSE comparisons of selection- and ensemble-based boosting on the SARCOS data set as a function of running time in seconds.

TABLE IV  
COMPARISONS OF THE TEST NMSE (IN PERCENT) OBTAINED BY SELECTION- AND ENSEMBLE-BASED BOOSTING ALGORITHMS AT THE SAME COMPUTATIONAL BUDGET WITH 200 s

Data set	RANDOM	MP	MAX-RES	ALD	RAN	Ensemble
OUTAOUAIS	20.16	12.85	8.46	18.71	18.71	4.57
KIN40K	3.84	3.97	3.35	3.97	3.69	1.34
SARCOS	2.42	2.65	2.21	3.01	3.05	1.75

where  $\mathbf{f}_M$ ,  $\mathbf{\omega}_M$ , and  $\mathbf{\omega}_M^*$  were defined in (18). For the FULL-UPDATE scheme, it computes the weight  $c_M$  by recursively maintaining the expression [see also (12)]

$$\mathbf{c}_M = (\mathbf{F}_M^T \mathbf{F}_M + \lambda \mathbf{\Omega}_M)^{-1} \mathbf{F}_M^T \mathbf{y}. \quad (32)$$

It is obvious that the computations of ONE-UPDATE are much cheaper than those of FULL-UPDATE. For each boosting step, the former leads to  $\mathcal{O}(N)$  complexity whereas the later incurs

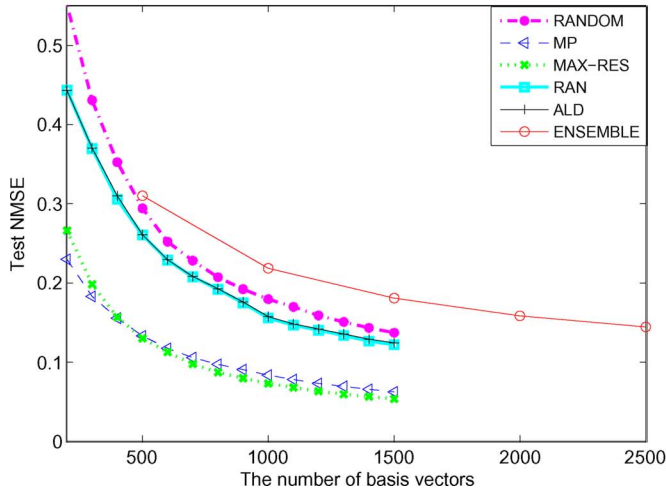


Fig. 4. Comparisons of test NMSE and CPU time as the function of the number of basis vectors used for different learning algorithms on the OUTAOUAIS data set.

$\mathcal{O}(NM)$  if recursive equations are used for computing (32). However, the disadvantage of ONE-UPDATE scheme is that it requires to append too many base learners to remedy the suboptimality in the sense of minimizing the objective (9).

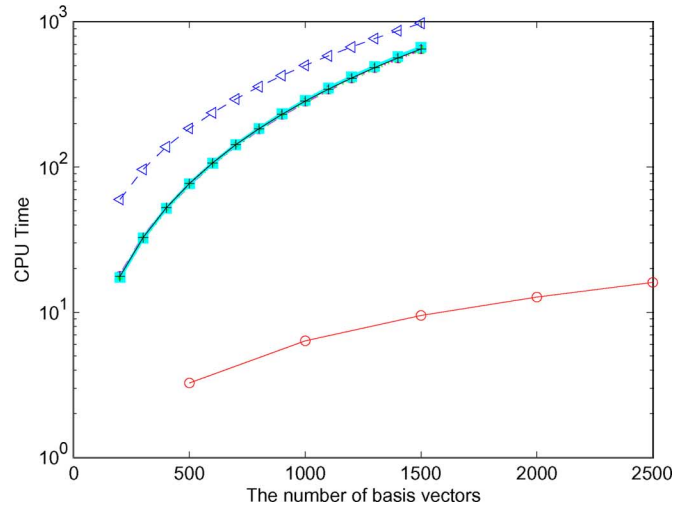
The difference between ONE-UPDATE and FULL-UPDATE schemes motivates us to propose a compromise one, which enables us to apply successfully our ensemble-based boosting approach to the COVTYPE data set, which is composed of 522 910 training points, on a personal computer. This will be demonstrated in Section IV. The idea is to construct a series of ensemble models  $F^1(\cdot), \dots, F^k(\cdot)$  instead of one large ensemble model  $F(\cdot)$ . For each component, it consists of  $L$  base learners where  $1 < L < M$  and is learned as usual by the ensemble-based boosting algorithm provided in Section III-D on the entire training inputs paired with the current residual as the target. The final model is simply summing up from  $F^1(\cdot)$  to  $F^k(\cdot)$  and note that the equality  $M = k \cdot L$ . This compromise scheme results in a total complexity of  $\mathcal{O}(N \cdot L \cdot M)$  time and  $\mathcal{O}(N \cdot L)$  memory, which is between the cost of ONE-UPDATE and that of FULL-UPDATE balanced by the setting of  $L$ . Note that we only use this compromise scheme for relatively large data sets, for example, COVTYPE mentioned above. Otherwise, the proposed ensemble-based boosting is sufficient for dealing with large applications.

#### IV. NUMERICAL EXPERIMENTS

In this section, we empirically demonstrate that the proposed ensemble-based boosting approach can achieve better generalization performance and lower computational requirements simultaneously when compared to previous state-of-the-art forward selection algorithms. We use the following regression data sets in the experimental studies and they are some of the largest tasks we can find in the regression literature.

- 1) OUTAOUAIS: The data was used in the ‘‘Evaluating Predictive Uncertainty Challenge’’ and its background information is not available.<sup>9</sup>
- 2) KIN40K: This data set represents the forward dynamics of an eight-link all-revolute robot arm; the task is to predict

<sup>9</sup><http://predict.kyb.tuebingen.mpg.de/datasets/>



the distance of the end-effector from a target, given the twist angles of the eight links as features.<sup>10</sup>

- 3) SARCOS: The task is related to learning of the inverse dynamics of a seven degrees-of-freedom SARCOS anthropomorphic robot arm.<sup>11</sup>

Some properties of these data sets are shown in Table II.

To further show the advantage of ensemble-based boosting for dealing with relatively large data set, we also report the result on the COVTYPE data, which is binary classification problem including 522 910/58 102 training/testing examples.<sup>12</sup> These data were previously used for evaluating several large scale SVM learning algorithms [11], [14], [61].

Note that the algorithms presented in this section are coded in Matlab 7.0<sup>13</sup> and all the numerical experiments are conducted on a 2.6-GHz Dual-Core AMD Opteron machine with 2-GB RAM, running Windows XP.

##### A. Regression Data

To evaluate prediction performance, we utilize normalized mean square error (NMSE) given by  $(1/N_{\text{test}}) \sum_i ([y_i - f(\mathbf{x}_i)]^2 / \text{var}(y))$ , where  $y_i$  is the true target of test input  $\mathbf{x}_i$ ,  $N_{\text{test}}$  is the number of test examples, and  $\text{var}(y)$  is the empirical variance of training targets. Two state-of-the-art selection-based boosting algorithms will be compared to our ensemble-based boosting method and they are MP [26] and MAX-RES [38]<sup>14</sup> criteria, respectively. To reduce the selection cost of MP approach, we evaluate the criterion (28) at each forward step only on a  $\kappa$ -size (we use  $\kappa = 60$  suggested in [54]) subset of all the unselected basis candidates. MAX-RES is a much simpler selection criterion and it just selects the basis vector corresponding to the maximal absolute value of the residual  $\mathbf{r}_{M-1}$ . Obviously, the MAX-RES method takes a

<sup>10</sup><http://ida.first.fraunhofer.de/~anton/data/>

<sup>11</sup><http://www.gaussianprocess.org/gpml/data/>

<sup>12</sup><http://www.cs.ust.hk/~jamesk/data/forest.zip>

<sup>13</sup>The source code can be downloaded from [http://www.cs.bham.ac.uk/~xin/journal\\_papers.html](http://www.cs.bham.ac.uk/~xin/journal_papers.html)

<sup>14</sup>Although this criterion is not proposed for RLS model, it could be easily adapted to deal with it, as done in [59].

TABLE V  
COMPARISONS OF RUNNING TIME IN SECONDS ELAPSED WHEN SELECTION AND ENSEMBLE BOOSTING TECHNIQUES REACH THE SAME TEST ACCURACY.  
THE VALUES IN PARENTHESES ARE THE (APPROXIMATE) NUMBER OF BASIS VECTORS USED. THE NOTATION “-” MEANS  
THAT THIS METHOD CANNOT ACHIEVE THE SPECIFIED TEST ERROR

Data set	NMSE (%)	RANDOM	MP	MAX-RES	ALD	RAN	Ensemble
OUTAOUAIS	10.00	-	335 (800)	135 (700)	-	-	40 (5,000)
KIN40K	3.00	360 (900)	320 (600)	250 (800)	-	335 (750)	40 (5,000)
SARCOS	2.00	425 (900)	495 (700)	330 (800)	-	-	110 (10,000)

very minor cost (almost the same as a random selection) for the subset selection compared to the MP approach, even though both of them lead to the same overall complexity of  $\mathcal{O}(NM^2)$  by combining the cost of weight updating together. Both ALD [15] and RAN [24] (two online forward sparsification schemes) can be easily adapted to learning RLS model (5) in a batch manner and we report their results on these data sets as well for reference.<sup>15</sup> Similar to MP and MAX-RES approaches, these two newly adapted algorithms have a similar computational requirement when the same number of basis vectors is selected. For each of them, there is a parameter determining the sparsity level of the model, denoted as  $\tau_1$  (ALD) and  $\tau_2$  (RAN), respectively [24]. As a baseline, we also provide the running results by a random selection of basis vectors.

For all the regression data sets, a squared-exponential kernel function was used

$$k(x_i, x_j; \theta) = \theta_0 \exp\left(-\frac{1}{2} \sum_{d=1}^D \theta_l (x_{i,d} - x_{j,d})^2\right) + \theta_b \quad (33)$$

where  $\theta_0, \theta_l, \theta_b > 0$  are hyperparameters,  $D$  is the dimension, and  $x_{i,d}$  is the  $d$ th entry of the input  $x_i$ . Since the RLS model corresponds to a MAP estimate of a Bayesian Gaussian process regression (GPR) [26], we can maximize the marginal likelihood of GPR on a random training subset (we use 1000 examples) for optimal settings of the hyperparameters  $\theta_0, \theta_l, \theta_b$  as well as regularization parameter  $\lambda$ , which are shared by all the boosting algorithms. We utilize some routines of the well-known NETLAB software<sup>16</sup> for doing this task.

For the proposed ensemble-based boosting approach, we fix the size of the training subset to  $T = 500$ , the size of each learner  $S = 10\% \times T = 50$ , and the upper limit of the ensemble model size  $M = 500$  for all data sets considered. For all the selection-based algorithms, the upper limit is set as  $M = 1500$  for OUTAOUAIS and  $M = 1000$  for other two larger sets by considering our PC’s configurations. For ALD and RAN methods, we use the same settings as those of [24] for the parameters:  $\tau_1 = 10^{-2}$  and  $\tau_2 = 10^{-4}$  on all the data sets.<sup>17</sup> Table III summarizes the parameter settings for all six methods on three regression data sets.

<sup>15</sup>We did not report the results of online versions of both algorithms. First, if going through the whole training set, online versions have a similar computational complexity to those of batch ones, i.e.,  $\mathcal{O}(NM^2)$  in time. Second, there is no public source codes available for two online algorithms and our implementations in Matlab are much slower than those of batch versions due to the inefficiency of Matlab dealing with a large number of iterations (going through the whole training set one by one).

<sup>16</sup>It is available at <http://www.ncrg.aston.ac.uk/netlab/index.php>

<sup>17</sup>We found that the numerical results are quite sensitive to the setting of both parameters, but the original work did not give a principled way to determine the optimal ones.

To remove the randomness involved in the algorithms, the results reported below are averaged on 30 independent runs. Figs. 1–3 illustrate the results of test NMSE as a function of CPU time by learning a single sparse model and boosted ensembles on three data sets, respectively. As we mentioned earlier, the CPU time reported here includes both training and testing time for a fair comparison. It is clear that ensemble-based boosting consistently achieved much better generalization performance than classical state-of-the-art forward selection methods given the same CPU time. Among those selection-based techniques, the simple MAX-RES method performed (significantly) better than the other approaches in terms of the computational efficiency on all three data sets. Notice that the baseline RANDOM selection method should not be simply ignored in comparison to other elaborate selection-based algorithms since it gives quite competitive performance for the two out of three tasks. For both ALD and RAN selection methods, we found that the settings of  $\tau_1$  and  $\tau_2$  have a great effect on the algorithm performance of both computational time and predictive capability. It seems not wise to set the same parameters for all the data sets as suggested in [24]. Some model selection procedures might be used to identify a good value but it will lead to more extra computational cost compared to other selection-based techniques free of such kind of parameters.

Table IV summarizes the test NMSE obtained by six algorithms on three data sets at the same computational budget of 200 s.<sup>18</sup> Table V reports the running time elapsed and the (approximate) number of basis vectors used when selection and ensemble techniques reach the same test accuracy. Obviously, ensemble-based boosting is much more computationally efficient than the selection-based methods. The shortcoming of ensemble-based boosting is that the obtained model is not as parsimonious as that of selection-based methods, which is reflected from the number of basis vectors used. Fig. 4 illustrates the comparison results of test NMSE and CPU time as the same number of basis vectors is used for different learning algorithms on the OUTAOUAIS data set. It is clear that ensemble-based boosting is not a good choice if the goal is to achieve the best generalization performance with the same number of basis vectors. It is particularly very useful when dealing with relatively large data sets where selection-based methods are not working properly, or the number of selected basis vectors is not a major concern.

Finally, to see whether the computational efficiency is sensitive to the setting of  $T$ ,<sup>19</sup> we also tried other  $T$  values during the experiments. Fig. 5 illustrates some results on the OUTAOUAIS

<sup>18</sup>The reported results are estimated from Figs. 1–3 by fitting the curves using linear splines in MATLAB; similarly, for Table III.

<sup>19</sup>Note that  $S$  is specified once  $T$  is decided because we fix the parameter  $S$  to be 10% of  $T$ .

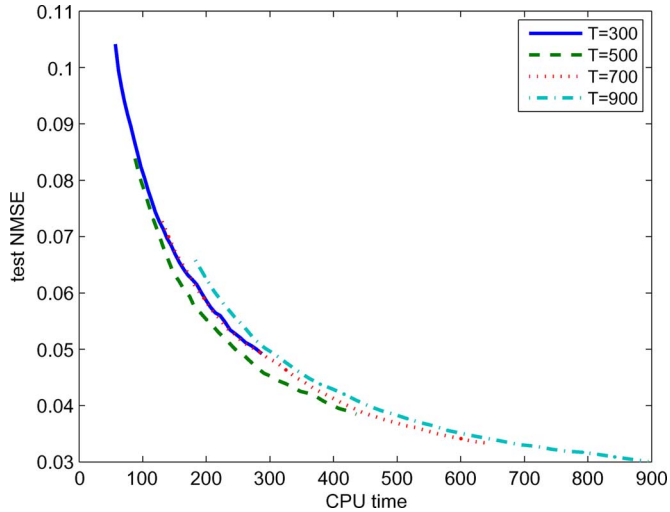


Fig. 5. Comparisons of the computational efficiency of the ensemble-based boosting by varying the values of  $T$  (and  $S$ ) with the same  $M = 500$  on the OUTAOUAIS data set.

data set and we can see that the performance of the proposed ensemble-based boosting method is relatively robust to the setting of  $T$ , where we tried four different values of  $T$  with the same maximally allowed number of base learners  $M = 500$ .

### B. COVTYPE: Relatively Large Scale Application

To further evaluate the ability of the proposed algorithm for dealing with relatively large data sets, we implement the ensemble-based boosting method on the COVTYPE data, which contains a total of 581 012 instances with 54 input features and is currently one of the largest databases available on the UCI website. Following [11], we aim at separating class 2 from other classes. Ninety percent of the whole data is used for training (i.e., 522 910 examples) while the remaining are used for testing (i.e., 58 102 examples). We use the Gaussian kernel  $K(x_i, x_j) = \exp\{(x_i - x_j)^2/\beta\}$  with  $\beta = 10\,000$  and the regularization parameter  $\lambda = 0.0001$ , which is taken from the SVM setting of [61].

The training data set is so large that the selection-based boosting algorithm cannot be run after just 50 basis vectors are chosen due to an error of *out of memory* in Matlab. For such an application, the ensemble-based boosting algorithm shows its distinct advantage by incorporating the compromise scheme mentioned earlier in Section III-E. Instead of building up a big ensemble  $F(\cdot)$ , we learn a series of ensembles  $F^1(\cdot), \dots, F^k(\cdot)$  and each of them consists of  $L$  base learners and we have the number of total base learners  $M = k \cdot L$ . By setting  $L$  to a small number, we get a total complexity of  $\mathcal{O}(N \cdot L \cdot M)$  in time and  $\mathcal{O}(N \cdot L)$  in memory. This compromise scheme reduces the memory requirement from  $\mathcal{O}(N \cdot M)$  to  $\mathcal{O}(N \cdot L)$ . In our experiment, we set the parameter value of  $L = 50$  and the maximal allowed number of ensembles  $k = 100$ . Thus, the maximal number of base learners is  $M = k \cdot L = 100 \times 50 = 5000$ . For each base learner, the number of basis vectors is set to  $S = 50$ , which is chosen from a random subset of  $T = 2000$  by some selection-based boosting techniques. Here, we employ the MP selection-based approach [26] instead of MAX-RES [38] to

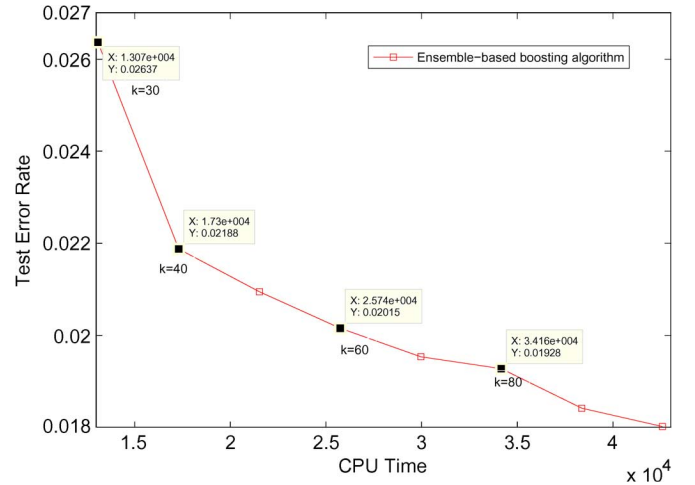


Fig. 6. Test error rate on the COVTYPE data set as a function of running time in seconds.

initialize basis vectors and increase the value of  $T$  from the default 500 to 2000 for seeking a better generalization.

In order to further enhance the computational performance of the compromise scheme, we propose to learn each ensemble  $F^k(\cdot)$  only on the training examples which correspond to  $\{\mathbf{x}_n | y_n F(\mathbf{x}_n) < 0.5, n = 1, \dots, N\}$ , i.e., mainly focusing on those incorrectly or least correctly classified instances,<sup>20</sup> where  $F(\cdot)$  is the sum of previously constructed ensembles, i.e.,  $F(\cdot) = F^1(\cdot) + \dots + F^{k-1}(\cdot)$ . If the size of this shrank training set is still too large, we choose the first 25 000 examples (i.e., around half of the entire training set) by sorting the values of  $\{y_n F(\mathbf{x}_n), n = 1, \dots, N\}$  in an ascending order. We observed that this speedup strategy not only significantly reduces the computational demands but also slightly improves the generalization performance of the final model  $F(\cdot)$ .

The classification error rate as a function of CPU time (both training and testing) is illustrated in Fig. 6. It can be seen that we could achieve an error rate of 2.19% with a running time around 17 000 s after  $k = 40$  ensembles are constructed. The number of basis vectors used is  $k \cdot L \cdot S = 100\,000$  at this point. This performance is very competitive with the state-of-the-art technique, i.e., core vector machine (CVM) [61], which is designed to learn large scale SVM models. We use the available package<sup>21</sup> to learn the same data set and get the error rate 2.27% with a running time around 25 000 s. Within this time, our proposed approach could reach an even better predictive result, i.e., 2.02%; see Fig. 6. Compared to the technique of CVM, the presented approach has at least two advantages.

- 1) For ensemble-based boosting, we can directly control the computational cost by adjusting number  $M$ ; see Fig. 6. It was found that the convergence of CVM as well as the computational cost is sometimes sensitive to the setting of hyperparameters [31] and also cannot be controlled as conveniently as our proposed method.
- 2) CVM involves a series of smaller QP problems which are much more difficult to implement than simple linear algebra operations used in ensemble-based boosting.

<sup>20</sup>Note that the instance  $\mathbf{x}_n$  is correctly classified if  $y_n F(\mathbf{x}_n) \geq 0$ .

<sup>21</sup><http://www.cse.ust.hk/~ivor/cvm.html>

It is straightforward to extend the current ensemble-based boosting algorithm to SVM models due to the shared learning framework (3). It would be an interesting work to benchmark two different techniques on a wider range of applications in the future.

## V. CONCLUSION

Some classical forward selection-based sparse approximation techniques (e.g., MP [26] and MAX-RES [38]) have proved to be an effective means for learning large scale kernel machines. However, due to the quadratic computational complexity in the number of selected basis vectors, such techniques are still computationally prohibitive for the situation when we want to include many more basis vectors for a better generalization. In this paper, we have proposed an ensemble-based boosting approach which attempts to include an ensemble of multiple basis vectors at each forward step, instead of only one as in the selection-based methods. The computational complexity of the newly proposed method is quadratic in the number of base learners and will not be dominated by the number of basis vectors used, and it is much more efficient than that of the selection-based methods, demonstrated on three large scale regression data sets. The advantage of the proposed work was further highlighted by applying it to a large classification task where the traditional selection-based techniques cannot work properly due to large time and space requirements.

In the future, we will investigate the performance of our proposed algorithm in comparison with other kernel models (e.g., KLR and SVM) and other kernel-based learning tasks such as semisupervised kernel machines [4] and large scale density estimation problems [10], [40]. It is also interesting to investigate the ensemble-based boosting methods where  $S$  may be different at different forward steps.

## ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and all reviewers for their constructive comments that have helped to improve significantly the quality of this paper.

## REFERENCES

- [1] E. Andelic, M. Schaffner, M. Katz, S. E. Kruger, and A. Wendemuth, "Kernel least-squares models using updates of the pseudoinverse," *Neural Comput.*, vol. 18, pp. 2928–2935, 2006.
- [2] F. R. Bach and M. I. Jordan, "Predictive low-rank decomposition for kernel methods," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 33–40.
- [3] G. Baudat and F. Anouar, "Kernel-based methods and function approximation," in *Proc. Int. Joint Conf. Neural Netw.*, 2001, pp. 1244–1249.
- [4] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, 2006.
- [5] L. Breiman, "Arcing classifiers (with discussion)," *Ann. Stat.*, vol. 26, pp. 801–849, 1998.
- [6] 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.
- [7] G. C. Cawley and N. L. C. Talbot, "Reduced rank kernel ridge regression," *Neural Process. Lett.*, vol. 16, no. 3, pp. 293–302, 2002.
- [8] O. Chapelle, "Training a support vector machine in the primal," *Neural Comput.*, vol. 19, pp. 1155–1178, 2007.
- [9] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, 1991.
- [10] S. Chen, X. Hong, and C. J. Harris, "An orthogonal forward regression technique for sparse kernel density estimation," *Neurocomputing*, vol. 71, no. 4–6, pp. 925–943, 2008.
- [11] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixture of SVMs for very large scale problems," *Neural Comput.*, vol. 14, no. 5, pp. 1105–1114, 2002.
- [12] L. Csato and M. Opper, "Sparse on-line Gaussian processes," *Neural Comput.*, vol. 14, no. 3, pp. 641–668, 2002.
- [13] T. G. Dietterich, "Ensemble methods in machine learning," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2000, vol. 1857, pp. 1–15.
- [14] J.-X. Dong, C. Y. Suen, and A. Krzyzak, "Fast SVM training algorithm with decomposition on very large data sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 4, pp. 603–618, Apr. 2005.
- [15] 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.
- [16] T. Evgeniou, M. Pontil, and T. Poggio, "Regularization networks and support vector machines," *Adv. Comput. Math.*, vol. 13, no. 1, pp. 1–50, 2004.
- [17] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *J. Mach. Learn. Res.*, vol. 2, pp. 243–264, 2002.
- [18] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 148–156.
- [19] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [20] J. H. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Ann. Stat.*, vol. 28, no. 2, pp. 337–407, 2000.
- [21] G. Fung and O. L. Mangasarian, "Proximal support vector machine classifiers," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining*, San Francisco, CA, 2001, pp. 77–86.
- [22] M. N. Gibbs and D. J. C. MacKay, "Efficient implementation of Gaussian processes," Dept. Physics, Cavendish Lab., Cambridge Univ., Cambridge, U.K., Tech. Rep., 1997.
- [23] L. Jiao, L. Bo, and L. Wang, "Fast sparse approximation for least square support vector machine," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 685–697, May 2007.
- [24] T. Jung and D. Polani, "Sequential learning with LS-SVM for large-scale data sets," in *Proc. 16th Int. Conf. Artif. Neural Netw.*, 2006, pp. 381–390.
- [25] S. S. Keerthi, O. Chapelle, and D. DeCoste, "Building support vector machines with reduced classifier complexity," *J. Mach. Learn. Res.*, vol. 7, pp. 1493–1515, 2006.
- [26] S. S. Keerthi and W. Chu, "A matching pursuit approach to sparse Gaussian process regression," in *Advances in Neural Information Processing Systems 18*. Cambridge, MA: MIT Press, 2006, pp. 643–650.
- [27] S. S. Keerthi, K. Duan, S. Shevade, and A. Poo, "A fast dual algorithm for kernel logistic regression," *Mach. Learn.*, vol. 61, no. 1–3, pp. 151–165, 2005.
- [28] S. S. Keerthi and S. K. Shevade, "SMO algorithm for least-squares SVM formulations," *Neural Comput.*, vol. 15, pp. 487–507, 2003.
- [29] W. Li, K.-H. Lee, and K.-S. Leung, "Large-scale RLSC learning without agony," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 529–536.
- [30] W. Liu and J. Principe, "Kernel affine projection algorithms," *EURASIP J. Adv. Signal Process.*, vol. 2008, 2008, DOI: 10.1155/2008/784292.
- [31] G. Loosli and S. Canu, "Comments on the 'core vector machines: Fast SVM training on very large data sets'," *J. Mach. Learn. Res.*, vol. 8, pp. 291–301, 2007.
- [32] S. Mallat and Z. Zhang, "Matching pursuit with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.
- [33] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent," in *Advances in Neural Information Processing Systems 12*. Cambridge, MA: MIT Press, 2000, pp. 512–518.
- [34] D. Mease and A. Wyner, "Evidence contrary to the statistical view of boosting (with discussions)," *J. Mach. Learn. Res.*, vol. 9, pp. 131–194, 2008.
- [35] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels," in *Proc. 6th IEEE Signal Process. Soc. Workshop*, Madison, WI, 1999, pp. 41–48.

- [36] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, A. Smola, and K.-R. Muller, "Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 623–628, May 2003.
- [37] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 1045–9227, Mar. 2001.
- [38] P. B. Nair, A. Choudhury, and A. J. Keane, "Some greedy learning algorithms for sparse regression and classification with mercer kernels," *J. Mach. Learn. Res.*, vol. 3, pp. 781–801, 2002.
- [39] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J. Comput.*, vol. 25, no. 2, pp. 227–234, 1995.
- [40] U. Ozertem, I. Uysal, and D. Erdogmus, "Continuously differentiable sample-spacing entropy estimation," *IEEE Trans. Neural Netw.*, vol. 19, no. 11, pp. 1978–1984, Nov. 2008.
- [41] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, 1991.
- [42] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA: MIT Press, 1999, pp. 185–208.
- [43] N. Ratliff and J. Bagnell, "Kernel conjugate gradient for fast kernel machines," in *Proc. Int. Joint Conf. Artif. Intell.*, 2007, pp. 1017–1022.
- [44] L. Rebollo-Neira and D. Lowe, "Optimized orthogonal matching pursuit approach," *IEEE Signal Process. Lett.*, vol. 9, no. 4, pp. 137–140, Apr. 2002.
- [45] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *J. Mach. Learn. Res.*, vol. 5, pp. 101–141, 2004.
- [46] C. Saunders, A. Gammermann, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *Proc. 15th Int. Conf. Mach. Learn.*, 1998, pp. 515–521.
- [47] B. Scholkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Proc. 14th Annu. Conf. Comput. Learn. Theory*, 2001, pp. 416–426.
- [48] B. Scholkopf, A. Smola, and K.-R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, 1998.
- [49] B. Scholkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2002.
- [50] M. Seeger, C. K. I. Williams, and N. D. Lawrence, "Fast forward selection to speed up sparse Gaussian process regression," in *Proc. 9th Int. Workshop Artif. Intell. Stat.*, Key West, FL, 2003.
- [51] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
- [52] Y. Shen, A. Ng, and M. Seeger, "Fast Gaussian process regression using kd-trees," in *Advances in Neural Information Processing Systems 18*. Cambridge, MA: MIT Press, 2006, pp. 1225–1232.
- [53] A. J. Smola and P. Bartlett, "Sparse greedy Gaussian process regression," in *Advances in Neural Information Processing Systems 14*. Cambridge, MA: MIT Press, 2001, pp. 619–625.
- [54] A. J. Smola and B. Schölkopf, "Sparse greedy matrix approximation for machine learning," in *Proc. 14th Int. Conf. Mach. Learn.*, 2000, pp. 911–918.
- [55] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems 18*. Cambridge, MA: MIT Press, 2006, pp. 1257–1264.
- [56] P. Sun, "Hierarchical boosting of sparse kernel machines," Ph.D. dissertation, Schl. Comput. Sci., Univ. Birmingham, Birmingham, U.K., 2007.
- [57] P. Sun and X. Yao, "Boosting kernel models for regression," in *Proc. 6th IEEE Int. Conf. Data Mining*, Hong Kong, 2006, pp. 583–591.
- [58] P. Sun and X. Yao, "Greedy forward selection algorithms to sparse Gaussian process regression," in *Proc. Int. Joint Conf. Neural Netw.*, 2006, pp. 159–165.
- [59] P. Sun and X. Yao, "A gradient-based forward greedy algorithm for sparse Gaussian process regression," in *Trends in Neural Computation*, K. Chen and L. Wang, Eds. Berlin, Germany: Springer-Verlag, 2007, pp. 241–263.
- [60] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.
- [61] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *J. Mach. Learn. Res.*, vol. 6, pp. 363–392, 2005.
- [62] P. Vincent and Y. Bengio, "Kernel matching pursuit," *Mach. Learn.*, vol. 48, no. 1–3, pp. 165–187, 2002.
- [63] M. Wu, B. Scholkopf, and G. Bakir, "A direct method for building sparse kernel learning algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 603–624, 2006.
- [64] C. Yang, R. Duraiswami, and L. Davis, "Efficient kernel machines using the improved fast Gauss transform," in *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press, 2005, pp. 1561–1568.
- [65] E. Yom-Tov, "A distributed sequential solver for large scale SVMs," in *Large Scale Kernel Machines*, O. Chapelle, D. DeCoste, J. Weston, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2007, pp. 141–156.
- [66] J. Zhu and T. Hastie, "Kernel logistic regression and the import vector machine," *J. Comput. Graph. Stat.*, vol. 14, no. 1, pp. 185–205, 2005.



**Ping Sun** received the B.Sc. degree in computational mathematics and the M.Sc. degree in applied mathematics from Xi'an Jiaotong University, Xi'an, China, in 2000 and 2003, respectively, and the Ph.D. degree in computer science from the University of Birmingham, Birmingham, U.K., in 2007.

He was a Research Associate with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, in 2008. Currently, he is an Information Analyst at the National Health Service (NHS). His major research interests include statistical machine learning and applications of data mining techniques.



**Xin Yao** (M'91–SM'96–F'03) received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, China, in 1982, the M.Sc. degree from the North China Institute of Computing Technology, Beijing, China, in 1985, and the Ph.D. degree from USTC in 1990, all in computer science.

From 1985 to 1990, he was an Associate Lecturer and Lecturer with USTC, while working toward the Ph.D. degree in simulated annealing and evolutionary algorithms. In August 1990, he became a Postdoctoral Fellow at the Computer Sciences Laboratory, Australian National University, Canberra, Australia, where he continued his work on simulated annealing and evolutionary algorithms. In 1991, he became a Postdoctoral Fellow (Level B) at the Knowledge-Based Systems Group, Commonwealth Scientific and Industrial Research Organization, Division of Building, Construction and Engineering, Melbourne, Australia, where he worked primarily on an industrial project on automatic inspection of sewage pipes. In 1992, he returned to Canberra to take up a lectureship in the School of Computer Science, University College, University of New South Wales, Australian Defense Force Academy, Sydney, Australia, where he was later promoted to a Senior Lecturer and Associate Professor. Since April 1999, he has been a Chair of Computer Science at the University of Birmingham, U.K. He is the Director of the Center of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, and a Changjiang (Visiting) Chair Professor (Cheung Kong Scholar) with the Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, USTC. He has given more than 60 invited keynote and plenary speeches at conferences and workshops worldwide, and published over 300 peer-reviewed technical papers. His major research interests include evolutionary computation, neural network ensembles, and their applications. His work has been supported by approximately £1m per annum external funding.

Dr. Yao was the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2003 to 2008, an Associate Editor or an editorial board member of 12 other journals, and the Editor of the World Scientific Book Series on *Advances in Natural Computation*. He was the recipient of the President's Award for Outstanding Thesis by the Chinese Academy of Sciences for his Ph.D. work on simulated annealing and evolutionary algorithms in 1989. He was the recipient of the 2001 IEEE Donald G. Fink Prize Paper Award for his work on evolutionary artificial neural networks.