Column-Generation Boosting Methods for Mixture of Kernels

(KDD-04 464)

Jinbo Bi
Computer-Aided Diagnosis & Therapy Group
Siemens Medical Solutions
Malvern, PA 19355
jinbo.bi@siemens.com

Tong Zhang
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
tzhang@watson.ibm.com

Kristin P. Bennett
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180
bennek@rpi.edu

ABSTRACT

We devise a boosting approach to classification and regression based on column generation using a mixture of kernels. Traditional kernel methods construct models based on a single positive semi-definite kernel with the type of kernel predefined and kernel parameters chosen from a set of possible choices according to cross validation performance. Our approach creates models that are mixtures of a library of kernel models, and our algorithm automatically determines kernels to be used in the final model. The 1-norm and 2-norm regularization methods are employed to restrict the ensemble of kernel models. The proposed method produces more sparse solutions. Hence it can handle larger problems, and significantly reduces the testing time. By extending the column generation (CG) optimization which existed for linear programs with 1-norm regularization to quadratic programs that use 2-norm regularization, we are able to solve many learning formulations by leveraging various algorithms for constructing single kernel models. Computational issues which we have encountered when applying CG boosting are addressed. In particular, by giving different priorities to columns to be generated, we are able to scale CG boosting to large datasets. Experimental results on benchmark problems are included to analyze the performance of the proposed CG approach and to demonstrate its effectiveness.

Keywords

Kernel methods, Boosting, Column generation

1. INTRODUCTION

Kernel-based algorithms have proven to be very effective for solving classification, regression and other inference problems in many applications. By introducing a positive semi-definite kernel K, nonlinear models can be created using linear learning algorithms such as in support vector machines (SVM), kernel ridge regression, kernel logistic regression, etc. The idea is to map data into a feature space, and construct optimal linear functions in the feature space that correspond to nonlinear functions in the original space. The key property is that the resulting model can be expressed as a kernel expansion. For example, the model (or the decision boundary) f obtained by SVM classification is expressed as

$$f(\mathbf{x}) = \sum_j \alpha_j K(\mathbf{x}, \mathbf{x}_j), \qquad (1)$$

where α is the model coefficients and $\mathbf{x}_j$ is the j^{th} training input vector. Here $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_\ell, y_\ell)$ are the training examples drawn i.i.d. from an underlying distribution. Throughout this article, vectors are presumed to be column vectors unless otherwise stated, and denoted using bold-face lower letters. Matrices are denoted by bold-face upper letters.

In such kernel methods, the choice of kernel mapping is of crucial importance. Usually, the choice of kernel is determined by predefining the type of kernel (e.g, RBF, and polynomial kernels), and tuning the kernel parameters using cross-validation performance. Cross-validation is expensive and the resulting kernel is not guaranteed to be an excellent choice. Recent work [12, 8, 4, 10, 7] has attempted to form kernels that adapt to the data of a particular task to be solved. For example, Lanckriet et al. [12] and Hammer et al. [10] proposed the use of a linear combination of kernels $K = \sum_p \mu_p K_p$ from a family of various kernel functions K_p. To ensure the positive semi-definiteness, the combination coefficients μ_p are either simply required to be nonnegative or determined in the way such that the composite kernel is positive semi-definite, for instance, by solving a semi-definite program as in [12] or by boosting as in [7]. The decision boundary f thus becomes

$$f(\mathbf{x}) = \sum_j \alpha_j \left(\sum_p \mu_p K_p(\mathbf{x}, \mathbf{x}_j) \right). \qquad (2)$$

In our approach, we do not make an effort to form a new kernel or a kernel matrix (the Gram matrix induced by a ker-

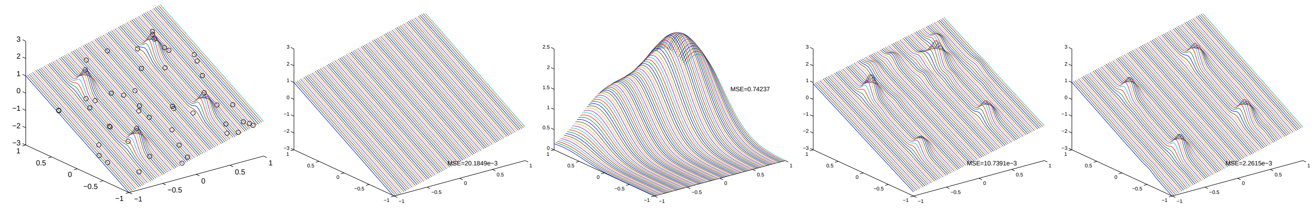

Figure 1: A two dimensional toy regression example. ***Left*, the target function with 50 training points; *mid-left*, linear kernel; *mid*, RBF kernel; *mid-right*, composite (linear+RBF); *right*, mixture (linear and RBF).**

nel on the training data). We construct models that are a mixture of models, where each model is based on one kernel choice from a library of kernels. Our algorithm automatically determines the kernels to be used in the mixture model. The decision boundary represented by this approach is

$$f(\mathbf{x}) = \sum_p \sum_j \alpha_j^p K_p(\mathbf{x}, \mathbf{x}_j). \qquad (3)$$

Previous kernel methods have employed similar strategies to improve generalization and reduce training and prediction computational costs. MARKING [2] optimized a heterogeneous kernel using a gradient descent algorithm in function space. GSVC and POKER [16, 15] grew mixtures of kernel functions from a family of RBF kernels. GSVC and POKER, however, were designed to be used only with weighted least squares SVMs. The proposed approach can be used in conjunction with any linear or quadratic generalized SVM formulations and algorithms.

In this article, we devise algorithms that boost on kernel columns via column generation (CG) techniques. Our goal is to develop approaches to construct inference models that make use of various geometry of the feature spaces introduced by a family of kernels other than a less expressive feature space induced by a single kernel. The 1-norm and 2-norm regularization methods can be employed to restrict the capacity of mixture models of form (3). We expect the resulting solution of mixture models to be more sparse than models with composite kernels. The CG-based techniques are adopted to obtain sparsity. We extend LPBoost, a CG boosting algorithm originally proposed for linear programs (LPs) [9], to quadratic programs (QPs). Our algorithm therefore becomes more general and is suitable for constructing a much larger variety of inference models.

We outline this paper now. In Section 2, we discuss the proposed mixture-of-kernels model and describe its characteristics. Section 3 extends LPBoost to QPs that use 2-norm regularization. Various CG-based algorithms are presented in Section 4 including classification and regression SVMs. Then in Section 5, we address some computational issues encountered when applying CG-based boosting methods. Experimental results on benchmark problems are included in Section 6 to examine the performance of the proposed method. The last section 7 concludes this paper.

2. MIXTURE OF KERNELS

In this section, we discuss the characteristics of the mixture-of-kernels method and compare it with other approaches.

2.1 Approximation capability

Models (3) that are based on a mixture of kernels are not necessarily equivalent to models (2) based on a composite kernel. Rewriting a composite kernel model (2) as $f(\mathbf{x}) = \sum_p \sum_i \alpha_i \mu_p K_p(\mathbf{x}_i, \mathbf{x})$, we can see it is equivalent to a mixture-of-kernels model (3) with $\alpha_i^p = \alpha_i \mu_p$. The opposite is not necessarily true, however, since given any composite kernel model (2), for any two kernels K_p and K_q, the ratio α_i^p / α_i^q is fixed to μ_p/μ_q, for all i (assuming $\mu_q \neq 0$ without loss of generality). Note that for a mixture-of-kernels model (3), we do not have this restriction.

It follows that a mixture model of form (3) can potentially give a larger hypothesis space than a model that uses either a single kernel or a composite kernel. The hypothesis space tends to be more expressive, i.e., a mixture-of-kernels model can better approximate target functions of practical problems. Although models using a single RBF kernel are known to be capable of approximating any continuous functions in the limit, these single-kernel models can give very poor approximations for many target functions. For example, suppose the target function can be expressed as a linear combination of several RBF kernels of different bandwidths. If we approximate it using a single RBF kernel (with a fixed bandwidth), then many basis functions have to be used. This leads to a dense and poor approximation, which is also difficult to learn. Therefore, using a single RBF kernel, we may have to use much more training data than necessary to learn a target function that can be better approximated by a mixture model with different kernels. Due to the better approximation capability, a mixture model (3) tends to give more sparse solutions which have two important consequences: the generalization performance tends to be improved since it is much easier to learn a sparse model due to the Occam's Razor principle (which says that the simplest solution is likely to be the best solution among all possible alternatives); the scalability is enhanced since sparse models require both less training and less testing time.

We use a simple regression problem to illustrate this point. The target function is defined as a "linear+RBF" function $f(\mathbf{x}) = x_1 + 2x_2 + \sum_i \exp(-\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{\sigma^2})$ where $\mathbf{c}_i$'s are $[\frac{1}{2}, \frac{1}{2}]$, $[-\frac{1}{2}, \frac{1}{2}]$, $[\frac{1}{2}, -\frac{1}{2}]$ and $[-\frac{1}{2}, -\frac{1}{2}]$. Figure 1(left) shows the target function together with 50 training points each generated as $y_i = f(\mathbf{x}_i) + \varepsilon$ where ε is a small Gaussian noise. The other four graphs in Figure 1 demonstrate the following: if we use the linear kernel, there is no way to adapt to the local RBF behavior; if a RBF kernel is used, the linear part of the function cannot be learned well. Using either a composite kernel model or a mixture kernel model achieves a better approximation of the target function. However, the mixture-of-kernel model is more sparse (which means better approximation to the target), and has better generalization (as seen in Figure 1). Moreover, the composite kernel method has significantly greater training and testing computational costs.

2.2 Connection to RBF nets

Consider the case of sets of RBF functions (kernels), we discuss the relationship among different basis expansion models, such as models obtained by SVMs, our approach and RBF nets [5]. In basis expansion methods [11], the model takes a form of $f(\mathbf{x}) = \sum \alpha_j \phi_j(\mathbf{x})$, where each function ϕ_j is a basis function of a form $\exp(-||\mathbf{x} - \mathbf{c}^j||^2/\sigma_j)$. In RBF networks, the centers $\mathbf{c}$ and the bandwidths σ of the basis functions are heuristically determined using unsupervised techniques. Therefore to construct the decision rule, one has to estimate: 1. the number of RBF centers; 2. the estimates of the centers; 3. the linear model parameter $\boldsymbol{\alpha}$, and 4. the bandwidth parameter σ. Compared with RBF networks, the benefits of using SVMs have been studied [19, 18]. The first three sets of parameters can be automatically determined by SVMs when learning support vectors. The last parameter is usually obtained by cross-validation tuning. Classic SVMs, however, use only a single parameter σ, which means that all centers (support vectors) are associated with a single choice of σ. Contrary to SVMs, RBF networks estimate a different σ for every center $\mathbf{c}$ of the RBF basis. More generally, the bandwidth σ can be different on different directions. Our model has a flexibility in between SVMs and RBF networks. Our model still benefits from the SVM-like algorithms, so parameters except σ can be learned by solving an optimization problem. In addition, our model allows the RBF basis positioned at different centers (support vectors) to associate with different bandwidths.

2.3 Regularization

To achieve good generalization performance, it is important to introduce appropriate regularization conditions on the model class. For a mixture model of form (3), the natural extension to the reproducing kernel Hilbert space (RKHS) regularization, commonly used by single-kernel methods such as SVMs, is to use the following generalized RKHS regularization $R(f)$:

$$R(f) = \sum_p \sum_{i,j} \alpha_i^p \alpha_j^p K_p(\mathbf{x}_i, \mathbf{x}_j),$$

This regularization condition, however, requires positive semi-definiteness of the kernel matrix K_p. This requirement can be removed by introducing other regularization conditions that are equally suitable for capacity control [13]. In particular, we consider penalizing the 1-norm or 2-norm of $\boldsymbol{\alpha}$. These regularization methods are more generally applicable since they do not require the kernel matrix to be positive semi-definite. This can be an important advantage for certain applications. Moreover, it is well known that the 1-norm regularization $||\boldsymbol{\alpha}||_1 = \sum |\alpha_j|$ leads to sparse solutions, which as we have explained earlier, is very desirable.

The mixture-of-kernels method investigated in this work has interesting properties concerning its learning and approximation behaviors. Due to the space limitation, these theoretical issues will be addressed elsewhere. This paper focuses on algorithmic design issues such as achieving sparsity of the solutions and the scalability to large-scale problems.

3. COLUMN GENERATION FOR QP

The column generation techniques have been widely used for solving large-scale LPs or difficult integer programs since the 1950s [14]. In the primal space, the CG method solves LPs on a subset of variables $\boldsymbol{\alpha}$, which means not all columns of the kernel matrix are generated at once and used to construct the function f. More columns are generated and added to the problem to achieve optimality. In the dual space, the columns in the primal problem correspond to the constraints in the dual problem. When a column is not included in the primal, the corresponding constraint does not appear in the dual. If a constraint absent from the dual problem is violated by the solution to the restricted problem, this constraint (a cutting plane) needs to be included in the dual problem to further restrict its feasible region. Thus these techniques are also referred to as cutting plane methods [1]. We first briefly review the existing LPBoost with 1-norm regularization. Then we formulate the QP optimization problem with 2-norm regularization and discuss the extension of CG techniques to QPs.

For notational convenience, we can re-index the columns in different kernels to form a single multi-kernel. Given a library of kernels $\mathcal{S} = \{K_1,\ K_2, \cdots, K_P\}$, a Gram matrix $\mathbf{K}^p$ can be calculated for each kernel in $\mathcal{S}$ on sample data with the column $K_p(\cdot, \mathbf{x}_j)$ corresponding to the j^{th} training example. Let us line up all these kernel matrices together $\mathbf{K} = [\mathbf{K}^1\ \mathbf{K}^2\ \cdots\ \mathbf{K}^P]$, and let index j run through the columns and index i run along the rows. Hence $\mathbf{K}_{i\cdot}$ denotes the i^{th} row of $\mathbf{K}$, and $\mathbf{K}_{\cdot j}$ denotes the j^{th} column. There are $d = \ell \times P$ columns in total.

3.1 LP formulation

If the hypothesis $\mathbf{K}_{\cdot j}\alpha_j$ based on a single column of the matrix $\mathbf{K}$ is regarded as a weak model or base classifier, we can rewrite LPBoost using our notation and following the statement in [3, 9]:

$$\begin{array}{lll} \min_{\boldsymbol{\alpha},\boldsymbol{\xi}} & \sum_{j=1}^{d} \alpha_j + C \sum_{i=1}^{\ell} \xi_i & \\ \text{s.t.} & y_i \sum_j \mathbf{K}_{ij}\alpha_j + \xi_i \geq 1, \quad \xi_i \geq 0, \quad i = 1, \cdots, \ell & (4) \\ & \alpha_j \geq 0, \quad j = 1, \cdots, d, & \end{array}$$

for regularization parameter $C > 0$. The dual of LP (4) is

$$\begin{array}{lll} \max_{\mathbf{u}} & \sum_{i=1}^{\ell} u_i & \\ \text{s.t.} & \sum_{i=1}^{\ell} u_i y_i \mathbf{K}_{ij} \leq 1, \quad j = 1, \cdots, d, & (5) \\ & 0 \leq u_i \leq C, \quad i = 1, \cdots, \ell. & \end{array}$$

These problems are referred to as the master problems. The CG method solves LPs by incrementally selecting a subset of columns from the simplex tableau and optimizing the tableau restricted on the subset of variables (each corresponding to a selected column). After a primal-dual solution

$(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}}, \hat{\mathbf{u}})$ to the restricted problem is obtained, we solve

$$\tau = \max_j \sum_i \hat{u}_i y_i \mathbf{K}_{ij}, \tag{6}$$

where j runs over all columns of $\mathbf{K}$. If $\tau \leq 1$, the solution to the restricted problem is already optimal to the master problems. If $\tau > 1$, then the solution to (6) provides a column to be included in the restricted problem.

3.2 QP formulation

Using the 2-norm regularization approach, constructing a model as in LP (4) corresponds to solving this QP:

$$\begin{array}{ll} \min_{\boldsymbol{\alpha},\boldsymbol{\xi}} & \frac{1}{2}\sum\limits_{j=1}^{d} \alpha_j^2 + C\sum\limits_{i=1}^{\ell} \xi_i \\ \text{s.t.} & y_i \sum\limits_j \mathbf{K}_{ij}\alpha_j + \xi_i \geq 1, \quad \xi_i \geq 0, \quad i = 1, \cdots, \ell \\ & \alpha_j \geq 0, \quad j = 1, \cdots, d. \end{array} \tag{7}$$

The Lagrangian dual function is the following:

$$\begin{array}{l} L = \frac{1}{2}\sum\limits_{j=1}^{d} \alpha_j^2 + C\sum\limits_{i=1}^{\ell} \xi_i - \sum_i u_i(y_i \sum\limits_j \mathbf{K}_{ij}\alpha_j + \xi_i - 1) \\ \quad - \sum_i s_i\xi_i - \sum_i t_j\alpha_j \end{array}$$

where u_i, s_i and t_j are nonnegative Lagrange multipliers.

Taking the derivative of the Lagrangian function with respect to the primal variables yields

$$\begin{array}{ll} \frac{\partial L}{\partial \alpha_j}: & \alpha_j - \sum_i u_i y_i \mathbf{K}_i j = t_j, \\ \frac{\partial L}{\partial \xi_i}: & C - u_i = s_i. \end{array} \tag{8}$$

Substituting (8) into the Lagrangian yields the dual problem as follows:

$$\begin{array}{ll} \max_{\mathbf{u}} \ \min_{\boldsymbol{\alpha}} & \sum\limits_{i=1}^{\ell} u_i - \frac{1}{2}\sum\limits_{j=1}^{d} \alpha_j^2 \\ \text{s.t.} & \sum\limits_{i=1}^{\ell} u_i y_i \mathbf{K}_{ij} \leq \alpha_j, \quad j = 1, \cdots, d, \\ & 0 \leq u_i \leq C, \quad i = 1, \cdots, \ell. \end{array} \tag{9}$$

The CG method partitions the variables α_j into two sets, the working set W that is used to build the model and the remaining set denoted as N that is eliminated from the model as the corresponding columns are not generated. Each CG step optimizes a subproblem over the working set W of variables and then selects a column from N based on the current solution to add to W. At each iteration, the α_j in N can be interpreted as $\alpha_j = 0$. Hence once a solution $\boldsymbol{\alpha}^W$ to the restricted problem is obtained, $\hat{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}^W \ \boldsymbol{\alpha}^N = 0)$ is feasible to the master QP (7). The following theorem examines when an optimal solution to the master problem is obtained in the CG procedure.

Theorem 3.1 (Optimality of QP CG). *Let $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}}, \hat{\mathbf{u}})$ be the primal-dual solution to the current restricted problems. If for all $j \in N$, $\sum_i \hat{u}_i y_i \mathbf{K}_{ij} \leq 0$, then $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}}, \hat{\mathbf{u}})$ is optimal to the corresponding master primal (7) and dual (9) problems.*

Proof. By the KKT conditions, to show the optimality, we need to confirm primal feasibility, dual feasibility and the equality of primal and dual objectives. Recall how we define $\hat{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}^W \ \boldsymbol{\alpha}^N = 0)$, so $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}})$ is feasible for QP (7). The primal objective must be equal to the dual objective since $\alpha_j = 0$, $\forall\ j \in N$. Now the key issue is dual feasibility. Since $(\boldsymbol{\alpha}^W, \hat{\boldsymbol{\xi}}, \hat{\mathbf{u}})$ is optimal for the restricted problem, it satisfies all constraints of the restricted dual problem, including $\sum_i \hat{u}_i y_i \mathbf{K}_{ij} \leq \alpha_j,\ j \in W$. Hence the dual feasibility is validated if $\sum_i \hat{u}_i y_i \mathbf{K}_{ij} \leq \alpha_j = 0,\ j \in N$. $\square$

Any column that violates dual feasibility can be added. For LPs, a common heuristic is to choose the column $\mathbf{K}_{\cdot j}$ that maximizes $\sum_i u_i y_i \mathbf{K}_{ij}$ over all $j \in N$. Similar to LPBoost, the CG boosting for QPs can also use the magnitude of the violation to pick the kernel columns or kernel basis function. In other words, the column $\mathbf{K}_{\cdot j}$ with the maximal score $\sum_i u_i y_i \mathbf{K}_{ij}$ will be included in the restricted problem.

Remark 3.1 (Column generation when $\boldsymbol{\alpha} \geq 0$). *Let $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}}, \hat{\mathbf{u}})$ be the solution to the restricted QP (7) and (9), and W, N be the current working and non-working sets. Solve*

$$\tau = \max_{j \in N} \sum_{i=1}^{\ell} \hat{u}_i y_i \mathbf{K}_{ij}, \tag{10}$$

and let the solution be $\mathbf{K}_{\cdot\hat{j}}$. If $\tau \leq 0$, the optimality is achieved; otherwise, the solution $\mathbf{K}_{\cdot\hat{j}}$ is a column for inclusion in the primal problem and also provides a cutting plane to the dual problem.

After the column has been generated, we can either solve the updated primal problem or the updated dual problem. The original LPBoost [9] solves the dual problem at each iteration. Any suitable algorithm can be used to solve the primal or dual master problem. From the optimization point of view, it is not clear which problem will be computationally cheaper than the other. In our implementation, we solve the primal problem which has the advantage of reducing the cost of finding the first feasible solution for the updated primal. For LPs, the tableau is optimized starting from the current solution after the new column is generated. Similar to LPs, the QP solver can have a warm-start by setting the initial guess of solution for the updated primal to the current solution. Therefore solving each restricted primal can be cheap in CG algorithms. Since we extend the column-generation boosting for LPs to QPs (7), we name the family of column-generation approaches CG-Boost.

4. VARIANTS OF CG-BOOST

We have extended the CG optimization for LPs to QPs where a 1-norm error metric with 2-norm regularization is minimized to construct models in the form of $\sum_j \alpha_j^p K_p(\mathbf{x}, \mathbf{x}_j)$, $\boldsymbol{\alpha} \geq 0$. However, typical kernel methods do not require the model parameters α_j to be nonnegative. Moreover, the model may contain a bias term b, i.e., $\sum \alpha_i^p K_p(\mathbf{x}_i, \mathbf{x}) + b$. In this section, we investigate these various models and devise variants of CG-Boost, including those suitable for classification and regression SVMs. Note that this general approach can be applied to other support vector methods including novelty detection and the nu-SVM formulations.

4.1 Add offset b

If the decision boundary model or the regression model has an offset b, we need to discuss two cases. First, if the variable b is regularized, an ℓ vector of ones can be added to the kernel matrix to correspond to b. The CG-Boost will be exactly the same as for the model without an explicit b except for the constant column added to the mixture of kernels. Second, if b is not regularized, then there exists an extra equality constraint $\sum_i u_i y_i = 0$ in the dual problem (5) or (9). In the later case, we use b in the initial restricted problem and always keep b in the working set W. Thus the equality constraint always presents in the restricted dual problem. To evaluate the dual feasibility and assure optimality, we still just proceed as in Remark 3.1.

4.2 Remove bound constraints

The lower bound constraint on model parameters $\boldsymbol{\alpha} \geq 0$ is not a necessary condition to use the CG approach. Removing the lower bound from QP (7), we obtain the problem:

$$\begin{array}{ll} \min_{\boldsymbol{\alpha},\boldsymbol{\xi}} & \frac{1}{2}\sum_{j=1}^{d} \alpha_j^2 + C\sum_{i=1}^{\ell} \xi_i \\ \text{s.t.} & y_i \sum_j \mathbf{K}_{ij}\alpha_j + \xi_i \geq 1, \quad \xi_i \geq 0, \quad i = 1,\cdots,\ell. \end{array} \tag{11}$$

The corresponding dual can be obtained through Lagrangian duality analysis as usually done for SVMs. In the resulting dual, the inequality constraints in (9) become equality constraints, i.e., $\alpha_j = \sum_i u_i y_i \mathbf{K}_{ij}$. Usually we substitute them into the Lagrangian and eliminate the primal variables from the dual. The dual objective becomes

$$\sum_{i=1}^{\ell} u_i - \frac{1}{2}\sum_{j=1}^{d}\left(\sum_i u_i y_i \mathbf{K}_{ij}\right)^2.$$

In CG optimization iterations, the dual feasibility is the key to verifying optimality. To evaluate the dual feasibility of (11), the equality constraints should hold for all j. For the columns in the working set W, the corresponding constraints are satisfied by the current solution. For the columns $\mathbf{K}_{\cdot j}$ that do not appear in the current restricted problem, the corresponding $\alpha_j = 0$. Therefore if $\sum_i u_i y_i \mathbf{K}_{ij} = 0, \ \forall\, j \in N$, the current solution is optimal for the master problem. Optimality can be verified by the following method:

REMARK 4.1 (COLUMN GENERATION FOR $\boldsymbol{\alpha}$ FREE). *Let $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}}, \hat{\mathbf{u}})$ be the solution to the current restricted QP (7) and (9) without bound constraints on $\boldsymbol{\alpha}$, and W, N be the current working and non-working sets. Solve*

$$\tau^- = \min_{j\in N} \ \sum_{i=1}^{\ell} \hat{u}_i y_i \mathbf{K}_{ij}, \quad \tau^+ = \max_{j\in N} \ \sum_{i=1}^{\ell} \hat{u}_i y_i \mathbf{K}_{ij} \tag{12}$$

and let $\mathbf{K}_{\cdot j^*}$, $\mathbf{K}_{\cdot \hat{j}}$ be the solutions, respectively.
If $\tau^- = \tau^+ = 0$, the current solution $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}}, \hat{\mathbf{u}})$ is optimal to the master problem; otherwise,
(a) if $|\tau^-| > |\tau^+|$, add $\mathbf{K}_{\cdot j^*}$ to the restricted problem; or otherwise (b) add $\mathbf{K}_{\cdot \hat{j}}$ to the restricted problem.

Unlike for problem (7) with $\boldsymbol{\alpha} \geq 0$, choosing the column with the score $\sum_i u_i y_i \mathbf{K}_{ij}$ farthest from 0, i.e., generating columns using (a) and (b) is not simply a heuristic for problems with free $\boldsymbol{\alpha}$. Instead it is the "optimal" greedy step for CG optimization as shown in the following proposition.

PROPOSITION 4.1. *The column generated using (a) and (b) is the column in N that decreases the duality gap the most at the current solution.*

PROOF. Let $\mathbf{z}$ be the column generated using (a) and (b). The dual objective value at the current solution $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}}, \hat{\mathbf{u}})$ is

$$\begin{aligned} &\textstyle\sum_i \hat{u}_i - \frac{1}{2}\sum_j(\sum_i \hat{u}_i y_i \mathbf{K}_{ij})^2 \\ &= \textstyle\sum_i \hat{u}_i - \frac{1}{2}\sum_{j\in W}(\sum_i \hat{u}_i y_i \mathbf{K}_{ij})^2 - \frac{1}{2}\sum_{j\in N}(\sum_i \hat{u}_i y_i \mathbf{K}_{ij})^2 \\ &= \textstyle\frac{1}{2}\sum_{j\in W} \hat{\alpha}_j^2 + C\sum_i \hat{\xi}_i - \frac{1}{2}\sum_{j\in N}(\sum_i \hat{u}_i y_i \mathbf{K}_{ij})^2 \\ &= \textstyle\frac{1}{2}\sum_{j=1}^{d} \hat{\alpha}_j^2 + C\sum_i \hat{\xi}_i - \frac{1}{2}\sum_{j\in N}(\sum_i \hat{u}_i y_i \mathbf{K}_{ij})^2 \end{aligned}$$

The last equation holds due to $\hat{\boldsymbol{\alpha}} = (\hat{\boldsymbol{\alpha}}^W \ \hat{\alpha}^N = 0)$. Hence the duality gap at $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}}, \hat{\mathbf{u}})$ is $\frac{1}{2}\sum_{j\in N}(\sum_i \hat{u}_i y_i \mathbf{K}_{ij})^2$. Finding a column in N that minimizes the duality gap yields the solution $\mathbf{z}$. $\square$

4.3 Apply to SVM regression

The CG-Boost approach can be easily generalized to solving regression problems with the ϵ-insensitive loss function [19]. To construct regression models f, we penalize as errors points that are predicted by f at least ϵ off from the true response. The ϵ-insensitive loss is defined as $\max\{|y - f(\mathbf{x})| - \epsilon, 0\}$. Using the 2-norm regularization (see [17] for 1-norm regularization). The optimization problem becomes:

$$\begin{array}{ll} \min_{\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\eta}} & \frac{1}{2}\sum_{j=1}^{d} \alpha_j^2 + C\sum_{i=1}^{\ell} (\xi_i + \eta_i) \\ \text{s.t.} & \sum_j \mathbf{K}_{ij}\alpha_j + \xi_i \geq y_i - \epsilon, \quad i = 1,\cdots,\ell, \\ & -\sum_j \mathbf{K}_{ij}\alpha_j + \eta_i \geq -y_i - \epsilon, \quad i = 1,\cdots,\ell, \\ & \xi_i \geq 0, \quad \eta_i \geq 0, \quad i = 1,\cdots,\ell. \end{array} \tag{13}$$

Note that in the primal the column generated at each CG iteration doubles its size in comparison to the classification case. The j^{th} column becomes $\mathbf{K}_{\cdot j}$ concatenated by $-\mathbf{K}_{\cdot j}$. Let u_i and v_i be the Lagrange multipliers corresponding to the first set of constraints and the second set of constraints, respectively. Then the dual constraints are $\sum_i (u_i - v_i)\mathbf{K}_{ij} = \alpha_j$. Thus solving the dual problem is more computationally attractive. Also as analyzed in Section 4.2, optimality can be verified by assessing if $\sum_i (u_i - v_i)\mathbf{K}_{ij} = 0, \ \forall\, j \in N$ as in the following remark.

REMARK 4.2 (COLUMN GENERATION FOR REGRESSION). *Let $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\eta}}, \hat{\mathbf{u}}, \hat{\mathbf{v}})$ be the solution to the current restricted QP (13) and its corresponding dual, and let W, N be the current working and non-working sets, respectively. Solve*

$$\begin{aligned} \tau^- &= \min_{j\in N} \ \textstyle\sum_{i=1}^{\ell} (\hat{u}_i - \hat{v}_i)\mathbf{K}_{ij}, \\ \tau^+ &= \max_{j\in N} \ \textstyle\sum_{i=1}^{\ell} (\hat{u}_i - \hat{v}_i)\mathbf{K}_{ij} \end{aligned} \tag{14}$$

and let $\mathbf{K}_{\cdot j^*}$, $\mathbf{K}_{\cdot \hat{j}}$ be the solutions, respectively.
If $\tau^- = \tau^+ = 0$, the current solution $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\eta}}, \hat{\mathbf{u}}, \hat{\mathbf{v}})$ is optimal to the master problem; otherwise,
(a) if $|\tau^-| > |\tau^+|$, add $\mathbf{K}_{\cdot j^*}$ to the restricted problem; or otherwise (b) add $\mathbf{K}_{\cdot \hat{j}}$ to the restricted problem.

5. STRATIFIED COLUMN GENERATION

When the CG method is applied as a general boosting approach, following the statement in [3], we need to solve $\max_{h\in\mathcal{H}} \sum_i \hat{u}_i y_i h(\mathbf{x})$ to generate a good column or base classifier h at each iteration. It is commonly difficult to estimate the entire hypothesis space $\mathcal{H}$ (often infinite) that can be implemented by the base learning algorithm. Typically we assume that an oracle exists to tell us the optimal column h. Even when we relax the problem and find a weak column that does not necessarily have the largest score, some heuristic strategy is still required. In mixture of kernels, the kernel library contains finite choices, so the matrix $\mathbf{K}$ comprises a finite number of columns. The generation of a column can be done by scanning all columns of $\mathbf{K}$ once requiring access to the entire kernel matrix at each iteration.. When the amount of training data is large, this is not desirable. To alleviate this difficulty and make CG-Boost more efficient, we design schemes to reduce the number of possible columns scanned in order to identify a violated dual constraint.

Examining the following complementarity conditions of QP (7) and (9) is insightful.

$$u_i \left(y_i \sum_j \mathbf{K}_{ij}\alpha_j + \xi_i - 1 \right) = 0, \qquad \forall i, \qquad (15)$$

$$\alpha_j \left(\sum_i u_i y_i \mathbf{K}_{ij} - \alpha_j \right) = 0, \qquad \forall j, \qquad (16)$$

$$\xi_i \left(u_i - C \right) = 0 \qquad \forall i. \qquad (17)$$

The first set of conditions tells us that $u_i = 0$ whenever $y_i \sum_j \mathbf{K}_{ij}\alpha_j > 1$. Since u_i can be interpreted as the "misclassification" cost, this implies that only examples with tight margin can have non-zero costs. The second group of conditions shows that whenever $\alpha_j > 0$, it is exactly equal to $\sum_i u_i y_i \mathbf{K}_{ij}$. From the last group of conditions, a point that produces margin error, i.e., $\xi_i > 0$, gets the largest "misclassification" cost C.

Based on the complementarity analysis, more effort is required to fit the error points ($\xi_i > 0$) in order to decrease the "misclassification" cost. Thus, it is reasonable to give high priority to the columns from various kernels in the kernel library that are centered at or correspond to the error points. Our toy regression example depicted in Figure 1 serves as a lucid example to illustrate this idea. Suppose we first fit a linear regression model. Then the 4 points close to the centers of the 4 RBF functions will introduce large error. This exhibits a need for nonlinear structures in order to improve the prediction on these points since the linear function cannot fit them well. When the 4 columns (corresponding to these 4 points) of the Gram matrix induced by the RBF kernel are added to the primal problem, the resulting model is dramatically improved.

Different kernels correspond to feature spaces of different geometric characters. Certain tasks may favor one kernel over others within the library. Domain insights may help identify good candidate kernels for a specific problem. Users can give high priority to more interpretable kernels by generating columns first from these kernels. If no such priori information exists, one philosophy is to have a preference for less complex and computationally cheap kernels. For example, assume the kernel library contains two types of kernels, linear and RBF. At an iteration, if the dual constraints based on columns from the linear kernel are violated, we choose a linear kernel column for inclusion in the restricted problem, and shall not continue considering columns from RBF kernels, thus giving higher priority to the simple kernels. This stratification biases the solution $\boldsymbol{\alpha}$ to be sparser on the more complex kernel basis functions (columns) than on the simple linear ones and serves as an extra means to control capacity of the mixture models.

5.1 The algorithm

We use the classification formulation (11) with an explicit bias term b as an example to describe the algorithm. Let $\mathbf{E}$ to denote the set of the columns in $\mathbf{K}$ corresponding to the error points ($\xi_i > 0$) at an iteration. Then $\mathbf{E} \cap \mathbf{K}^p$ indicates the columns, from the kernel K_p, that correspond to error points. Without loss of generality, we assume the kernels in $\mathcal{S} = \{K_1, \cdots, K_P\}$ are ordered with their priority decreasing. With a little abuse of notation, we use $\mathbf{N}$ to denote all columns in $\mathbf{K}$ that are not included in the current restricted problem. In the CG-Boost algorithm, TR1, TR2, and TR4 represent termination rules which will be discussed in the following section.

ALGORITHM 5.1. **CG-Boost**

```
1. Initialize the first column $\mathbf{K}_0 = \mathbf{e}$ (the vector of ones
   corresponding to the bias term $b$)
2. Let $\boldsymbol{\alpha}^0 = 0$
3. For $t = 1$ to $T$, do
4.   Solve problem (11) with the current kernel matrix $\mathbf{K}_{t-1}$,
     and obtain solution $(\boldsymbol{\alpha}^t, \boldsymbol{\xi}^t, \mathbf{u}^t)$.
5.   For $p = 1$ to $P$, (do a scan on $\mathbf{E} \cap \mathbf{K}$)
         Solve problem (12) on $\mathbf{E} \cap \mathbf{K}^p$,
         obtain $\tau = max\{|\tau^-|, |\tau^+|\}$,
         and the corresponding column $\mathbf{z}$
         If $\tau > 0$, $\mathbf{K}_t = \{\mathbf{K}_{t-1},\ \mathbf{z}\}$, break from loop on $p$
     End of loop $p$
6.   If no column is found, i.e., $\tau < 0$, terminate (TR1),
     or
7.   For $p = 1$ to $P$, (do a full scan on $\mathbf{N}$)
         Solve problem (12) on $\mathbf{N} \cap \mathbf{K}^p$,
         obtain $\tau = max\{|\tau^-|, |\tau^+|\}$,
         and the corresponding column $\mathbf{z}$
         If $\tau > 0$, $\mathbf{K}_t = \{\mathbf{K}_{t-1},\ \mathbf{z}\}$, break from loop on $p$
     End of loop $p$
8.   If no column is found, optimality is
     achieved, terminate (TR2)
9. End of loop $t$ (TR4)
```

5.2 Termination rules

Several termination strategies can be used to stop the optimizing process, depending on the desired level of quality of the solutions. Four schemes can be designed:

TR1. Stop when no columns corresponding to error examples can be added, in other words, all dual constraints formed by such columns are satisfied by the current solution. This termination rule is weak due to no guarantee that a good model has been obtained, but it requires a very small amount of computation, and a full scan on the kernel matrix is not needed.

Table 1: Datasets Summary

Dataset	dim	n_pts	n_positive	n_negative
Breast	30	569	212	357
Ionosphere	33	351	225	126
Pima	8	768	268	500
Digits	784	60,000	30,378	29,622
Forest	54	495,141	211,840	283,301

TR2. Stop when the problem is completely optimized, i.e., the algorithm converges to an optimal solution. This often requires several full scans on the full kernel matrix until all dual constraints are satisfied by the current solution to the restricted problem.
TR3. The first and second termination rules may be either too weak or too strong. CG-Boost allows us to stop in the middle when a suboptimal model achieves the desirable performance. Especially, for large databases, a validation set can be created independent of the training and test sets. Monitoring the prediction performance of each model generated in the CG iterations on the validation set can produce insights into a reasonable termination location.
TR4. Another simple way to stop the CG boosting process is to pre-specify a maximal number of iterations. When the limit is reached, the program is terminated.
We shall evaluate different termination strategies in our computational studies. Based on specific problems at hand, we can choose the termination rule most reasonable to the task.

6. COMPUTATIONAL RESULTS

The experiments were designed to evaluate the performance of the proposed approach in terms of the prediction accuracy, sparsity of solutions as well as scalability to large data sets, and compare it to other approaches such as composite-kernel methods. Small UCI datasets [6] were used primarily to evaluate the generalization performance. Two additional large databases were used: the Forest data from UCI KDD Archive, and the NIST hand-written digit database downloaded from *http://yann.lecun.com/exdb/mnist/*. Table 1 provides the basic statistics of the datasets[1]. The data was preprocessed by normalizing each original feature to have mean 0 and standard deviation 1. Three kernel types in the kernel library were used: K_1 is the linear kernel, K_2 is the quadratic kernel, and K_3 is a RBF kernel, denoted respectively as L, Q and R in the Tables 3 and 4. We employed a fixed strategy to find a σ for the RBF kernel in the experiments on all different datasets. First we calculate the mean of $||\mathbf{x}_i - \mathbf{x}_j||^2$ where i, j run through the training examples, and then set σ equal to the mean value. We use the commerical optimization package ILOG CPLEX 8.0 to solve the restricted problems. But note that any suitable algorithm for the appropriate single kernel problem can be extended to mixture of kernels by using it to solve the restricted problems in CG-Boost.

6.1 Performance evaluation

For small datasets, we preformed 5-fold cross validation for various algorithms using the linear kernel, the RBF kernel,

[1]The table 1 columns are dimension (dim), the number of points (n_pts), the number of positive examples (n_positive) and the number of negative examples (n_negative).

Table 2: Five-fold CV results of various methods (with 2-norm regularization) on small datasets.

Method	FP_{tst}	FN_{tst}	T_{trn}	T_{tst}	C_l	C_r
Breast						
linear	1.96	5.66	7.36	0.002	450	–
RBF	3.08	3.77	10.85	0.471	–	455
composite	1.96	**3.77**	10.37	0.471	453	453
mixture	**1.68**	4.25	17.45	0.243	453	174
Ionosphere						
linear	23.8	5.8	1.19	0.001	276	–
RBF	**11.2**	**3.1**	1.98	0.015	–	273
composite	15.8	3.6	2.10	0.018	276	276
mixture	14.3	3.6	4.08	0.006	276	89
Pima						
linear	19	35.4	12.9	0.002	610	–
RBF	21.8	32.1	21.7	0.965	–	609
composite	20.4	34.3	21.7	0.972	614	614
mixture	**18**	**32.8**	36.8	0.014	609	64

Table 3: Training and test classification accuracies on datasets created from Forest database. ***Top*, 2-norm regularization; *bottom*, 1-norm regularization.**

Kernel	R_{trn}	R_{tst}	T_{trn}	T_{tst}	C_l	C_q	C_r
L	20.6	23.6	57.2	0.2	999	–	–
Q	16.8	23.2	57.2	2.4	–	975	–
R	21.5	25.9	143.3	382	–	–	987
L+Q	16.6	23.1	61.5	4.4	975	975	–
L+R	18.5	23.3	140.8	380	980	–	980
L+Q+R	16.8	23.0	140.0	388	980	980	980
L,Q,R	16.3	**22.7**	129.8	**2.6**	676	173	3
L	20.1	23.6	8.9	0.2	26	–	–
Q	19.4	25.1	15.2	0.36	–	70	–
R	27.9	30.2	97.4	24	–	–	66
L+Q	19.2	23.4	12.7	0.47	65	65	–
L+R	19.9	**23.0**	94.4	16.3	41	–	41
L+Q+R	21.5	25.5	97.0	21	53	53	53
L,Q,R	16.8	**23.0**	111.6	**2.5**	30	10	3

Table 4: Training and test classification accuracies on datasets created from NIST hand-written digit database. *Top*, **2-norm regularization;** *bottom*, **1-norm regularization.**

Kernel	R_{trn}	R_{tst}	T_{trn}	T_{tst}	C_l	C_q	C_r
L	1.1	18.7	73.1	0.2	969	–	–
Q	6	18.0	90.3	4.4	–	181	–
R	16.2	22.1	115.2	870	–	–	971
L+Q	3.1	15.2	90.9	0.82	184	184	–
L+R	15	20.0	125.4	882	980	–	980
L+Q+R	2.5	16.1	158.5	889	980	980	980
L,Q,R	0.4	**13.5**	130.2	**13.4**	122	20	13
L	4.8	15.6	33.1	0.2	171	–	–
Q	6.3	19.6	54.9	2.3	–	94	–
R	40	38.8	74.6	6.4	–	–	7
L+Q	6.0	16.5	73.0	4.5	98	98	–
L+R	4.8	15.8	83.7	84	93		93
L+Q+R	6.6	15.3	71.1	85.5	93	93	93
L,Q,R	0.9	**13.1**	95.5	**11.1**	77	10	12

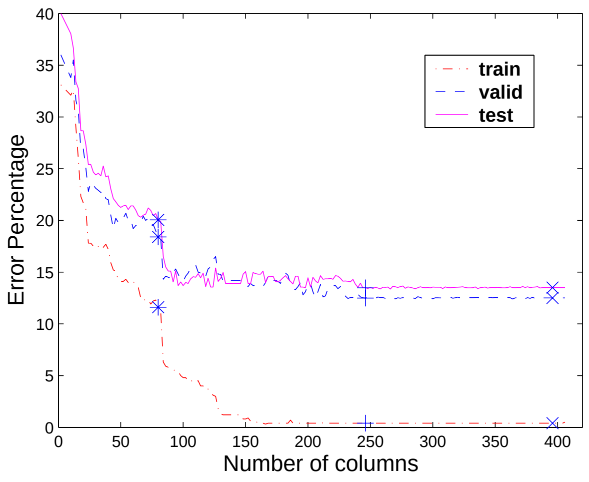

Figure 2: Comparison of termination strategies.

the composite kernel of linear and RBF, and our mixture kernel model. The parameter C was tuned within each training dataset from choices $\{1, 10, 100\}$ in the first fold. The optimized value of C is used in the successive folds. In our experiments, we did not tune the parameter μ in the composite kernel since efficient methods were not available. Therefore, μ is simply fixed to 1. CG-Boost was trained until optimality, i.e. termination rule TR1.

Table 2 lists the results obtained by algorithms using the 2-norm regularization method. Since these datasets are quite imbalanced, we provide the FP rates and FN rates together with the training time (T_{trn}) required to completely optimize the QPs (TR2), the execution time (T_{tst}) in the test phase and the number of columns from each kernel (C_l for linear kernel, C_r for RBF kernel). From Table 2, the composite model and mixture model have similar performance, but mixture models use significantly less time in the testing phase since solutions are sparse. Solutions are especially sparse on the computationally expensive kernels such as RBF kernels. Sparsity for nonlinear kernels is computationally important since it allows kernel methods to scale well to large problems. The sparsity of the linear kernel is less crucial since we can simplify $\sum_i \alpha_i K_1(x, x_i) = K_1(x, \sum_i \alpha_i x_i)$. Note the Ionosphere dataset inherently favors the RBF kernel. Since we emphasize the sparsity on RBF kernel columns, we obtained slightly worse performance.

Results of experiments using large databases are reported in Table 3 and 4. We generated these datasets by randomly choosing 1000 examples for training, 2000 examples as the validation set and 10000 examples in test. The training, validation and test sets are mutually exclusive. Since the amount of test data is increased, the difference on the execution time by mixture and composite models is magnified. This is due to the sparsity on the RBF kernel basis achieved by the mixture models by referencing the last 3 columns. Only 3 RBF kernel columns were selected for Forest data in the mixture models. Due to the characteristics of composite kernel models, all kernels have the same number of columns selected even though some columns might not be necessary. Notice that solving the mixture kernel models usually consumed more time based on T_{trn} (if we do not count the time needed to determine the parameter μ in the composite-kernel models). However when we investigate the behavior of CG-Boost, we can see we do not have to solve the problem completely. We can terminate the CG algorithm when an acceptable solution is obtained. Then the training time is also dramatically reduced.

6.2 Behavior of CG-Boost

First, we examine different termination rules using the experimental results obtained on digit datasets by QPs with 2-norm regularization. Figure 2 shows the error percentage versus the number of columns. The three curves represent the training, validation and test performance. All 3 curves are drawn together only for illustrating the results – no test data was used in training. If TR1 is used, we stop at the iteration marked by *, and if TR3 is used, we stop at the iteration marked by $+$. Completely solving the QPs requires more iterations as marked by $\times$. Since using TR3, we obtain models with performance similar to the optimal model, and the training time is reduced by more than half, TR3 appears to be a good choice for termination criteria. We adopted TR3 in later experiments.

Second, we validate the effectiveness of our stratified process in generating the columns. Recall the stratification strategy here is to give different priorities to different training examples and different types of kernels. By looking at Figure 3, more columns have to be generated in the CG approach if we do not stratify the column generation. The convergence was slower for the regular CG process. Moreover, using the stratified process, whenever a column from a simple kernel is identified to be included in the QPs, all other columns from complex kernels will not be calculated. Hence the number of kernel columns that need to be computed in order to generate a column is significantly reduced in comparison with the regular CG process. On average the stratified CG needed 255 kernel columns to be calculated at each iteration while the regular CG required a full scan ($\ell \times P$=3000 columns) in each iteration.

Third, we look at the parameter tuning problem. For classi-

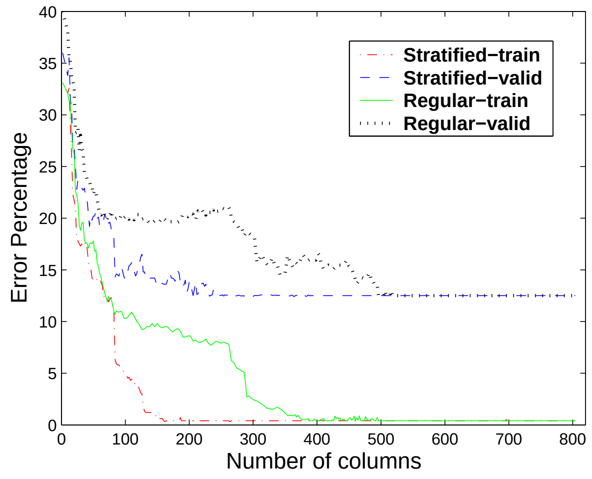

Figure 3: Comparison of regular CG boosting and stratified CG boosting methods.

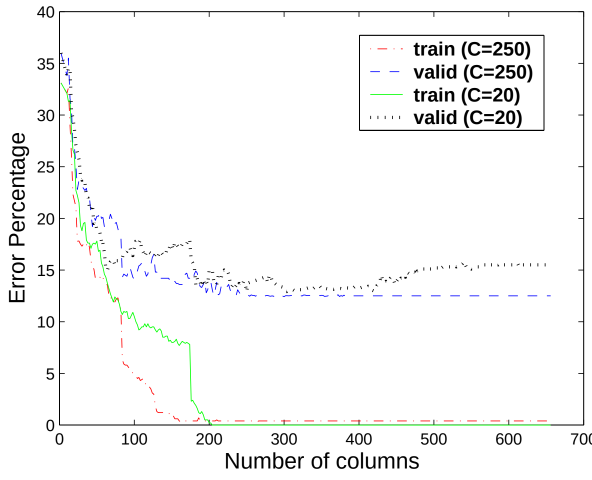

Figure 4: Improper regularization parameter value can be monitored during the CG iteration.

fication problems, prediction accuracy depends on an appropriate choice of the regularization parameter C. In the usual tuning process, we choose a value for C, then solve the QPs completely, and then examine the validation performance of the obtained model to decide if the choice is appropriate. In CG-Boost, by monitoring the validation performance at each iteration, we can assess if it is overfitting due to an improper choice of C, without having to fully solve the QPs as shown in Figure 4.

Forth, we assess the scalability of the proposed CG-Boost. Our CG-Boost is an extension to original LPBoost, so the scalability analysis for LPBoost in [9] is also suitable to CG-Boost. We conducted experiments using the Forest database. We randomly took 1000, 2000, 5000 and 10000 examples from the database as the training sets, we still use 10000 examples in test as in the previous experiments. The number of columns and running times are included in Table 5. The number of columns that were generated does not increase as

Table 5: The number of iterations as a function of data amount.

size	1000	2000	5000	10000
Iter	253	302	520	613
Time	141	315	740	2530

the number of examples increases. The rate of increase is not even linear. In addition, the testing time has a very flat rate of growth since the numbers of columns included in the models are similar for datasets of different sizes and most of the terms are linear.

7. CONCLUSIONS

We proposed a CG Boosting method for classification and regression using mixture-of-kernels models. We argued that the mixture of kernel approach leads to models that are more expressive than models based on the traditional single-kernel or the composite kernel approach. This means that for many problems, we are able to better approximate the target function with a fixed number of basis functions. A simple example is given to demonstrate this phenomenon.

Due to the better approximation behavior, the mixture of kernel method often leads to sparser solutions. This property is computationally important since with a sparser solution, it is possible to handle larger problems, and the testing time can be significantly reduced. Moreover, sparse models tend to have better generalization performance due to the Occam's Razor principle. Experiments on various datasets were presented to support our claims. In addition, by extending the current LPBoost column generation boosting method to handle quadratic programming, we are able to solve many learning formulations by leveraging existing and future algorithms for constructing single kernel models. Our method is also computational significantly more efficient than the composite kernel method, which requires semi-definite programming techniques to find appropriate composite kernels.

8. REFERENCES

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms.* John Wiley & Sons, Inc., New York, NY, 1993.

[2] K. Bennett, M. Momma, and M. Embrechts. MARK: A boosting algorithm for heterogeneous kernel models. In *Proceedings of SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 24–31, 2002.

[3] K. P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. In *Proceedings of the 17th International Conference on Machine Learning*, pages 65–72. Morgan Kaufmann, San Francisco, CA, 2000.

[4] J. Bi. Multi-objective programming in SVMs. In T. Fawcett and N. Mishra, editors, *Proceedings of the Twentieth International Conference on Machine*

Learning, pages 35–42, Menlo Park, CA, 2003. AAAI Press.

[5] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[6] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[7] K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 537–544. MIT Press, Cambridge, MA, 2003.

[8] N. Cristianini, A. Elisseef, and J. Shawe-Taylor. On optimizing kernel alignment. Technical Report NC2-TR-2001-087, NeuroCOLT, 2001.

[9] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1–3):225–254, 2002.

[10] B. Hamers, J. Suykens, V. Leemans, and B. Moor. Ensemble learning of coupled parameterised kernel models. In *Supplementary Proc. of the International Conference on Artificial Neural Networks and International Conference on Neural Information Processing (ICANN/ICONIP)*, pages 130–133, 2003.

[11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: data mining, inference and prediction*. Springer-Verlag, New York, 2001.

[12] G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2003.

[13] O. L. Mangasarian. Generalized support vector machines. In P. Bartlett, B. Schölkopf, D. Schuurmans, and A. Smola, editors, *Advances in Large Margin Classifiers*, pages 135–146. MIT Press, 2000.

[14] S. G. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, New York, NY, 1996.

[15] E. Parrado-Hernandez, J. Arenas-Garca, I. Mora-Jimenez, , and A. Navia-Vazquez. On problem oriented kernel refining. *Neurocomputing*, 55:135–150, 2003.

[16] E. Parrado-Hernandez, I. Mora-Jimenez, J. Arenas-Garcia, A. R. Figueiras-Vidal, and A. Navia-Vazquez. Growing support vector classifiers with controlled complexity. *Pattern Recognition Letters*, 36:1479–1484, 2003.

[17] G. Rätsch, A. Demiriz, and K. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48(1–3):193–221, 2002.

[18] B. Schölkopf, K.Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, 45(11):2758–2765, 1997.

[19] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., New York, 1998.