Column-Generation Boosting Methods for Mixture of Kernels

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 Inst.
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 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 sparser solutions, and thus significantly reduces the testing time. By extending the column generation (CG) optimization which existed for linear programs with 1-norm regularization to quadratic programs with 2-norm regularization, we are able to solve many learning formulations by leveraging various algorithms for constructing single kernel models. By giving different priorities to columns to be generated, we are able to scale CG boosting to large datasets. Experimental results on benchmark data are included to demonstrate its effectiveness.

Categories and Subject Descriptors:
I.5.2 [Pattern Recognition]: Design Methodology—classifier design and evaluation

General Terms: Algorithms, Design

Keywords: Kernel methods, Boosting, Column generation

1. INTRODUCTION

Kernel-based algorithms have proven to be very effective for solving 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 decision boundary f obtained by SVM classification is expressed as

$$f(\mathbf{x}) = \sum\nolimits_j \alpha_j K(\mathbf{x}, \mathbf{x}_j), \tag{1}$$

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. 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 [9, 6, 4, 12, 5, 2] has attempted to design kernels that adapt to the data of a particular task to be solved. For example, Lanckriet et al. [9] and Crammer et al. [5] 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 [9] or by boosting as in [5]. The decision boundary f thus becomes

$$f(\mathbf{x}) = \sum\nolimits_j \alpha_j \left(\sum\nolimits_p \mu_p K_p(\mathbf{x}, \mathbf{x}_j) \right). \tag{2}$$

In our approach, we do not make an effort to form a new kernel or a kernel matrix (the Gram matrix induced by a kernel on the training data). We construct models that are a mixture of models, each 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

$$f(\mathbf{x}) = \sum\nolimits_p \sum\nolimits_j \alpha_j^p K_p(\mathbf{x}, \mathbf{x}_j). \tag{3}$$

Previous kernel methods have employed similar strategies to improve generalization and reduce training and prediction computational costs. MARKING [1] optimized a heterogeneous kernel using a gradient descent algorithm in function space. GSVC and POKER [14, 13] grew mixtures of kernel

KDD'04, August 22–25, 2004, Seattle, Washington, USA.

functions from a family of RBF kernels, but they 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.

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 geometries of the feature spaces introduced by a family of kernels other than the 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) [7], 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, and then experimental results on benchmark data are presented to demonstrate the performance of the proposed method. Section 6 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 sparser 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.

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 [3]. In basis expansion methods [8], 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 [17, 16]. 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 $R(f)$:

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

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 [10]. 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 fo-

cuses 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 [11]. In the primal space, the CG method solves LPs restricted 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 column in the primal) needs to be included in the restricted problem in order to obtain optimality. We first briefly review the existing LPBoost with 1-norm regularization. Then we extend CG techniques to QPs with 2-norm regularization so that may successful formulations, such as classic SVMs, ridge regression, etc. can also benefit from CG techniques.

For notational convenience, we 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 the LPBoost using our notations and following the statement in [7]:

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

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

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

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

$$\tau = \max_j \sum\nolimits_i \hat{u_i} y_i \mathbf{K}_{ij}, \tag{6}$$

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:

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

The Lagrangian dual function is the following:

$$\begin{aligned} L = \frac{1}{2}\sum_{j=1}^{d} \alpha_j^2 + C\sum_{i=1}^{\ell} \xi_i - \sum\nolimits_i u_i (y_i \sum_{j=1}^{d} \mathbf{K}_{ij}\alpha_j + \xi_i - 1) \\ - \sum\nolimits_i s_i \xi_i - \sum\nolimits_j t_j \alpha_j \end{aligned}$$

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

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

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

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

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. The α_j in N can be interpreted as $\alpha_j = 0$. 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).*

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 $\sum_i \hat{u}_i y_i \mathbf{K}_{ij} \leq \alpha_j = 0,\ j \in N$. □

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 column 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 constraint to be added to the dual problem.

After a column has been generated, we can either solve the updated primal problem or the updated dual problem. The original LPBoost [7] solves the dual problem at each iteration. From the optimization point of view, it is not clear which problem, the primal or the dual, will be solved with cheaper computational cost. Since the current solution at each CG iteration is primal feasible to the updated problem, no cost is needed to find the first feasible solution for the updated primal. Due to this advantage, we solve the primal problem in our implementation. For LPs, the tableau is optimized starting from the current solution after the new column is generated. For QPs, we set 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 CG boosting for LPs to QPs (7), we name the family of CG approaches CG-Boost.

4. VARIANTS OF CG-BOOST

We have extended the CG optimization for LPs to QPs for constructing 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 $\boldsymbol{\alpha} \geq 0$. Moreover, the model may contain an offset term b, i.e., $\sum \alpha_i^p K_p(\mathbf{x}_i, \mathbf{x}) + b$. We investigate these various models here 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 ν-SVMs.

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:

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

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*

$$\tau = \max_{j \in N} \ \left| \sum_{i=1}^{\ell} \hat{u}_i y_i \mathbf{K}_{ij} \right| \tag{12}$$

and let $\mathbf{K}_{\cdot \hat{j}}$ be the solution. If $\tau = 0$, the current solution $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}}, \hat{\mathbf{u}})$ is optimal to the master problem; otherwise, add the column $\mathbf{K}_{\cdot \hat{j}}$ to the restricted problem.

Unlike for problem (7) with $\boldsymbol{\alpha} \geq 0$, choosing the column with the largest violation score $|\sum_i u_i y_i \mathbf{K}_{ij}|$, i.e., generating columns by solving problem (12) 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 by solving problem (12) is the column in N that decreases the duality gap the most at the current solution.*

PROOF. Let $\mathbf{z}$ be the column produced by problem (12). The dual objective value at the current solution $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\xi}}, \hat{\mathbf{u}})$ is

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

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 $\mathbf{z}$. □

4.3 Apply to SVM regression

The CG-Boost approach can be easily generalized to solving regression problems with the ϵ-insensitive loss function $\max\{|y - f(\mathbf{x})| - \epsilon, 0\}$ [17]. To construct regression models f, we penalize points as errors that are predicted by f at least ϵ off from the true response. Using the 2-norm regularization (see [15] for 1-norm regularization), the QP becomes:

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

Note that each column in the primal has its size doubled in comparison with 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 $\sum_i (u_i - v_i)\mathbf{K}_{ij} = \alpha_j$. Again, 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*

$$\tau = \max_{j \in N} \left| \sum_{i=1}^{\ell} (\hat{u}_i - \hat{v}_i)\mathbf{K}_{ij} \right| \tag{14}$$

and let $\mathbf{K}_{\cdot \hat{j}}$ be the solution. If $\tau = 0$, 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, add the column $\mathbf{K}_{\cdot \hat{j}}$ to the restricted problem.

5. ALGORITHMS AND EXPERIMENTS

In the mixture-of-kernels method, the kernel library $\mathcal{S}$ contains finite choices, so an effective column, i.e., the solution of (12) can be found by scanning columns of $\mathbf{K}$. However, it requires access to the entire kernel matrix at each iteration. When the amount of training data is large, it is not desirable. To make CG-Boost more efficient, we propose a stratified CG process: columns from different training examples and different types of kernels are given different priorities to be generated.

First, columns corresponding to error points ($\xi_i > 0$) can be selected in order to fit the error points better in the next iteration. Hence (step i) we first choose the column with the largest violation score of (12) from the kernel columns corresponding to error points; (step ii) once all such columns are satisfied by the current solution, a full scan on all columns is performed. Moreover, different kernels correspond to feature spaces of different geometric character. 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 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. In either of the above steps (i) and (ii), the columns from the linear kernel will be considered first. If no appropriate linear kernel column can be added, then we consider columns from more complicated kernels.

We evaluated the CG-Boost mixture-of-kernels approach in terms of the prediction accuracy, sparsity of solutions and compared it to other approaches such as composite-kernel methods. The MNIST hand-written digit database of 60,000 28×28 images was used in our experiments. We discriminated between odd and even numbers. We randomly chose 1000 examples for training, 2000 examples as the validation set and 10000 examples in test. The training, validation and test sets were mutually exclusive. We used the commercial optimization package ILOG CPLEX 8.0 to solve the restricted problems. Three kernel types were adopted: linear, quadratic and RBF kernels, denoted respectively as L, Q and R in Table 1 where L+Q means a combination of linear and quadratic kernels with $\mu = 1$ and L,Q,R indicates a mixture of linear, quadratic and RBF kernels. We employed a fixed strategy to find a value for σ in the RBF kernel. First we calculated 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.

Table 1: Classification results for MNIST hand-written digit database. *Top*, 2-norm regularization; *bottom*, 1-norm regularization. R_{trn}, R_{tst}, T_{trn} and T_{tst} denote the training and test error percentages as well as training and test times in seconds, respectively. C_l, C_q and C_r give the numbers of columns from linear, quadratic and RBF kernels.

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

From the T_{tst} column of Table 1, the mixture models require much less execution time in the testing phase than composite models due to the sparsity on the RBF kernel bases achieved by the mixture models by referencing the last 3 columns of Table 1. Due to the characteristics of composite kernel models, different kernels have the same number of columns selected even though some columns might not be necessary. Solving the mixture kernel models can consume more time than composite-kernel models based on T_{trn} (provided we do not count the time needed to determine the parameter μ in the composite-kernel models. We simply set $\mu = 1$ in our comparison). However since CG-Boost is an iterative boosting process, it can be terminated earlier when a solution with desirable performance is obtained. Then the training time is also dramatically reduced.

We validated the effectiveness of our stratified process in generating the columns. Form Figure 1 (top), 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 at that iteration. Hence the number of columns that needs 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). CG-Boost is an extension to original LPBoost, so the scalability analysis for LPBoost in [7] is also suitable to CG-Boost.

We also examined the parameter tuning problem. An appropriate choice of the regularization parameter C determines the prediction accuracy. 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,

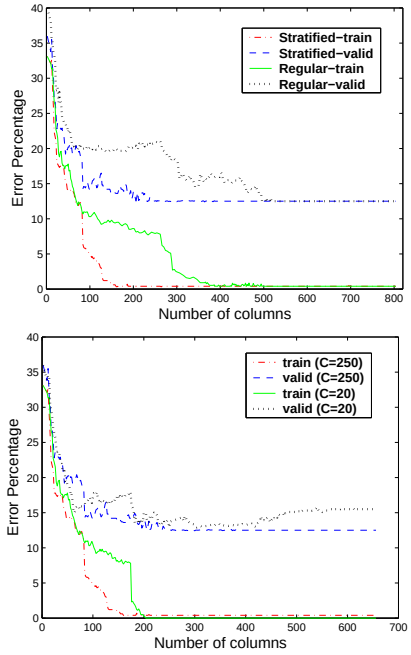

Figure 1: Top: comparison of regular CG boosting and stratified CG boosting methods. Bottom: improper regularization parameter values can be monitored during the CG iteration.

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 1 (bottom) where the validation curve corresponding to an improper C value goes up after 400 columns are generated.

6. CONCLUSIONS

We proposed a column generation boosting approach CG-Boost for classification and regression using mixture-of-kernels models. We argued that the proposed approach leads to models that are more expressive than models obtained by single-kernel or composite-kernel approaches. This means we are able to better approximate the target function using a smaller number of basis functions. CG-Boost produces sparse solutions, so the testing time can be significantly reduced, and also sparser models tend to have better generalization performance due to the Occam's Razor principle. Experimental results were presented to support our claims. In addition, by extending the current LPBoost method to handle quadratic programs, we are able to solve many learning formulations by leveraging existing and future algorithms for constructing single kernel models. Our method is also computationally more efficient than the composite kernel method, which often requires semi-definite programming techniques to find appropriate composite kernels.

7. REFERENCES

[1] 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.

[2] 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.

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

[4] O. Bousquet and D. J. L. Herrmann. On the complexity of learning the kernel matrix. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 399–406. MIT Press, Cambridge, MA, 2003.

[5] 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.

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

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

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

[9] 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.

[10] 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.

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

[12] C. S. Ong, A. J. Smola, and R. C. Williamson. Hyperkernels. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 478–485. MIT Press, Cambridge, MA, 2003.

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

[14] 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*, 36:1479–1488, 2003.

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

[16] 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.

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