Predictive low-rank decomposition for kernel methods

Francis R. Bach FRANCIS.BACH@MINES.ORG
Centre de Morphologie Mathématique, Ecole des Mines de Paris
35 rue Saint-Honoré, 77300 Fontainebleau, France

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science Division and Department of Statistics
University of California, Berkeley, CA 94720, USA

Abstract

Low-rank matrix decompositions are essential tools in the application of kernel methods to large-scale learning problems. These decompositions have generally been treated as black boxes—the decomposition of the kernel matrix that they deliver is independent of the specific learning task at hand—and this is a potentially significant source of inefficiency. In this paper, we present an algorithm that can exploit side information (e.g., classification labels, regression responses) in the computation of low-rank decompositions for kernel matrices. Our algorithm has the same favorable scaling as state-of-the-art methods such as incomplete Cholesky decomposition—it is linear in the number of data points and quadratic in the rank of the approximation. We present simulation results that show that our algorithm yields decompositions of significantly smaller rank than those found by incomplete Cholesky decomposition.

1. Introduction

Kernel methods provide a unifying framework for the design and analysis of machine learning algorithms (Schölkopf and Smola, 2001, Shawe-Taylor and Cristianini, 2004). A key step in any kernel method is the reduction of the data to a *kernel matrix*, also known as a *Gram matrix*. Given the kernel matrix, generic matrix-based algorithms are available for solving learning problems such as classification, prediction, anomaly detection, clustering and dimensionality reduction. There are two principal advantages to this division of labor: (1) any reduction that yields a positive semidefinite kernel matrix is allowed, a fact that opens the door to specialized transformations that exploit domain-specific knowledge; and (2) expressed in terms of the kernel matrix, learning problems often take the form of convex optimization problems, and powerful algorithmic techniques from the convex optimization literature can be brought to bear in their solution.

An apparent drawback of kernel methods is the naive computational complexity associated with manipulating kernel matrices. Given a set of n data points, the kernel matrix K is of size $n \times n$. This suggests a computational complexity of at least $O(n^2)$; in fact most kernel methods have at their core operations such as matrix inversion or eigenvalue decomposition which scale as $O(n^3)$. Moreover, some kernel algorithms make use of sophisticated tools such as semidefinite programming and have even higher-order polynomial complexities (Lanckriet et al., 2004).

These generic worst-case complexities can often be skirted, and this fact is one of the major reasons for the practical success of kernel methods. The underlying issue is that the kernel matrices often have a spectrum that decays rapidly and are thus of small numerical rank (Williams and Seeger, 2000). Standard algorithms from numerical linear algebra can thus be exploited to compute an approximation of the form $L = GG^\top$, where $G \in \mathbb{R}^{n \times m}$, and where the rank m is generally significantly smaller than n. Moreover, it is often possible to reformulate kernel-based learning algorithms to make use of G instead of K. The resulting computational complexity generally scales as $O(m^3 + m^2 n)$. This linear scaling in n makes kernel-based methods viable for large-scale problems.

To achieve this desirable result, it is of course necessary that the underlying numerical linear algebra rou-

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

tines scale linearly in n, a desideratum that *inter alia* rules out routines that inspect all of the entries of K. Algorithms that meet this desideratum include the Nyström approximation (Williams and Seeger, 2000), sparse greedy approximations (Smola and Schölkopf, 2000) and incomplete Cholesky decomposition (Fine and Scheinberg, 2001, Bach and Jordan, 2002).

One unappealing aspect of the current state-of-the-art is that the decomposition of the kernel matrix is performed independently of the learning task. Thus, in the classification setting, the decomposition of K is performed independently of the labels, and in the regression setting the decomposition is performed independently of the response variables. It seems unlikely that a single decomposition would be appropriate for all possible learning tasks, and unlikely that a decomposition that is independent of the predictions should be optimized for the particular task at hand. Similar issues arise in other areas of machine learning; for example, in classification problems, while principal component analysis can be used to reduce dimensionality in a label-independent manner, methods such as linear discriminant analysis that take the labels into account are generally viewed as preferable (Hastie et al., 2001). The point of view of the current paper is that that there are likely to be advantages to being "discriminative" not only with respect to the parameters of a model, but with respect to the underlying matrix algorithms as well.

Thus we pose the following two questions:

1. Can we exploit side information (labels, desired responses, etc.) in the computation of low-rank decompositions of kernel matrices?
2. Can we compute these decompositions with a computational complexity that is linear in n?

The current paper answers both of these questions in the affirmative. Although some new ideas are needed, the end result is an algorithm closely related to incomplete Cholesky decomposition whose complexity is a constant factor times the complexity of standard incomplete Cholesky decomposition. As we will show empirically, the new algorithm yields decompositions of significantly smaller rank than those of the standard approach.

The paper is organized as follows. In Section 2, we review classical incomplete Cholesky decomposition with pivoting. In Section 3, we present our new predictive low-rank decomposition framework, and in Section 4 we present the details of the computations performed at each iteration, as well as the exact cost reduction of such steps. In Section 5, we show how the cost reduction can be efficiently approximated via a look-ahead method. Empirical results are presented in Section 6 and we present our conclusions in Section 7.

We use the following notations: for a rectangular matrix M, $\|M\|_F$ denotes the Frobenius norm, defined as $\|M\|_F = (\operatorname{tr} MM^\top)^{1/2}$; $\|M\|_1$ denotes the sum of the singular values of M, which is equal to the sum of the eigenvalues of M when M is square and symmetric, and in turn equal to $\operatorname{tr} M$ when the matrix is in addition positive semidefinite. We also let $\|x\|_2$ denote the 2-norm of a vector x, equal to $\|x\|_2 = (x^\top x)^{1/2} = \|x\|_F$. Given two sequences of distinct indices I and J, $M(I,J)$ denotes the submatrix of M composed of rows indexed by I, and columns indexed by J. Note that the sequences I and J are not necessarily increasing sequences. The notation $M(:,J)$ denotes the submatrix of the columns of M indexed by the elements of J, and similarly for $M(I,:)$. Also, we refer to the sequence of integers from p to q as $p{:}q$. Finally, we denote the concatenation of two sequences I and J as $[I\ J]$. We let $Id_q \in \mathbb{R}^{q\times q}$ denote the identity matrix and $1_q \in \mathbb{R}^q$ denote the vector of all ones.

2. Incomplete Cholesky decomposition

In this section, we review incomplete Cholesky decomposition with pivoting, as used by Fine and Scheinberg (2001) and Bach and Jordan (2002).

2.1. Decomposition algorithm

Incomplete Cholesky decomposition is an iterative algorithm that yields an approximation $L = GG^\top$, where $G \in \mathbb{R}^{n\times m}$, and where the rank m is generally significantly smaller than n.[1] The algorithm depends on a sequence of *pivots*. Assuming temporarily that the pivots $i_1, i_2, \ldots$ are known, and initializing a diagonal matrix D to the diagonal of K, the k-th iteration of the algorithm is as follows:

$$
\begin{aligned}
&G(i_k,k) = D(i_k)^{1/2}\\
&G(J_k,k) = \tfrac{1}{G(i_k,k)}\Big(K(J_k,i_k) - \textstyle\sum_{j=1}^{k-1} G(J_k,j)\,G(i_k,j)\Big)\\
&D(j) = D(j) - G(j,k)^2, \quad \forall j \notin \{i_1,\ldots,i_k\},
\end{aligned}
$$

where $I_k = (i_1,\ldots,i_k)$ and J_k denotes the sorted complement of I_k. The complexity of the k-th iteration is $O(kn)$, and thus the total complexity after m steps is $O(m^2 n)$. After the k-th iteration, $G(I_k, 1{:}k)$ is a lower triangular matrix and the approximation of K is $L_k = G_k G_k^\top$, where G_k is the matrix composed of the first k columns of G, i.e., $G_k = G(:, 1{:}k)$. We let denote D_k the diagonal matrix D after the k-th

[1] In this paper, the matrices G will always have full rank, i.e, the rank will always be the number of columns.

iteration.

2.2. Pivot selection and early stopping

The algorithm operates by greedily choosing the column such that the approximation of K obtained by adding that column is best. In order to select the next pivot i_k, we thus have to rank the gains in approximation error for all remaining columns. Since all approximating matrices $G_k G_k^\top$ are such that $L_k = G_k G_k^\top \preccurlyeq K$, the 1-norm $\|K - L_k\|_1$, which is defined as the sum of the singular values, is equal to $\|K - L_k\|_1 = \operatorname{tr}(K - L_k)$.

To compute the exact gain of approximation after adding column i_k is an $O(kn)$ operation. If this were to be done for all remaining columns at each iteration, we would obtain a prohibitive total complexity of $O(m^2n^2)$. The algorithm avoids this cost by using a lower bound on the gain in approximation. Note in particular that at every step we have $\operatorname{tr} L_k = \sum_{q=1}^{k} \|G(:,q)\|_2^2$; thus the gain of adding the k-th column is $\|G(:,k)\|_2^2$, which is lower bounded by $G(i_k,k)^2$. Even before the k-th iteration has begun, we know the final value of $G(i_k,k)^2$ if i_k were chosen, since this is exactly $D_{k-1}(i_k)$.

We thus chose the pivot i_k that maximizes the lower bound $D_{k-1}(i_k)$ among the remaining indices. This strategy also provides a principled early stopping criterion: if no pivot is larger than a given precision ε, the algorithm stops.

2.3. Low rank approximation and partitioned matrices

Incomplete Cholesky decomposition yields a decomposition in which the column space of L is spanned by a subset of the columns of K. As the following proposition shows, under additional constraints the subset of columns actually determines the approximation:

Proposition 1 *Let K be an $n \times n$ symmetric positive semidefinite matrix. Let I be a sequence of distinct elements of $\{1,\dots,n\}$ and J its ordered complement in $\{1,\dots,n\}$. There is an unique matrix L of size n such that:*
(i) L is symmetric,
(ii) the column space of L is spanned by by $K(:,I)$,
(iii) $L(:,I) = K(:,I)$.
This matrix L is such that

$$L([I\ J],[I\ J]) = \begin{pmatrix} K(I,I) & K(J,I)^\top \\ K(J,I) & K(J,I)K(I,I)^\dagger K(J,I)^\top \end{pmatrix}.$$

In addition, the matrices L and $K - L$ are positive semidefinite.

Proof If L satisfies the three conditions, then (i) and (iii) implies that $L(I,J) = L(J,I)^\top = K(J,I)^\top = K(I,J)$. Since the column space of L is spanned by $K(:,I)$, we must have $L(:,J) = K(:,I) \times E$, where E is a $|I| \times |J|$ matrix. By projecting onto the columns in I, we get $K(I,J) = K(I,I)E$, which implies that $L(J,J) = K(J,I)K(I,I)^\dagger K(I,J)$, where $M^\dagger$ denotes the pseudo-inverse of M (Golub and Loan, 1996). ■

Note that the approximation error for the block (J,J) is equal to the Schur complement $K(J,J) - K(J,I)K(I,I)^\dagger K(I,J)$ of $K(I,I)$. The incomplete Cholesky decomposition with pivoting builds a set $I = \{i_1,\dots,i_m\}$ iteratively and approximates the matrix K by L given in the previous proposition for the given I. To obtain L, a square root of $K(I,I)$ has to be computed that is easy to invert. The Cholesky decomposition provides such a square root which is built efficiently as I grows.

3. Predictive low-rank decomposition

We now assume that the $n \times n$ kernel matrix K is associated with side information of the form $Y \in \mathbb{R}^{n \times d}$. Supervised learning problems provide standard examples of problems in which such side information is present. For example, in the (multi-way) classification setting, d is the number of classes and each row y of Y has d elements such that y_i is equal to one if the corresponding data point belongs to class i, and zero otherwise. In the (multiple) regression setting, d is the number of response variables. In all of these cases, our objective is to find an approximation of K which (1) leads to good predictive performance and (2) has small rank.

3.1. Prediction with kernels

In this section, we review the classical theory of reproducing kernel Hilbert spaces (RKHS) which is necessary to justify the error term that we use to characterize how well an approximation of K is able to predict Y.

Let $x_i \in \mathcal{X}$ be an input data point and let $y_i \in \mathbb{R}^d$ denote the associated label or response variable, for $i = 1,\dots,n$. Let $\mathcal{H}$ be an RKHS on $\mathcal{X}$, with kernel $T(.,.)$. Given a loss function $c : \mathcal{X} \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$, the *empirical risk* is defined as $R(f) = \sum_{i=1}^{n} c(x_i, y_i, f(x_i))$ for functions f in $\mathcal{H}^d$. By a simple multivariate extension of the representer theorem (Schölkopf and Smola, 2001, Shawe-Taylor and Cristianini, 2004), minimizing the empirical risk subject to a constraint on the RKHS norm of f leads to a solution of the form $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$, where $\alpha_i \in \mathbb{R}^d$.

In this paper, we build our kernel approximations by considering the quadratic loss $c(x, y, f) = \frac{1}{2}\|x - f\|_F^2$. The empirical risk is then equal to $\frac{1}{2}\sum_{i=1}^{n} \|y_i - (K\alpha)_i\|_F^2 = \frac{1}{2}\|Y - K\alpha\|_F^2$, where $\alpha \in \mathbb{R}^{n\times d}$. When K is approximated by $GG^\top$, for G an $n \times m$ matrix, the optimal risk is equal to:

$$\min_{\alpha\in\mathbb{R}^{n\times d}} \tfrac{1}{2}\|Y - K\alpha\|_F^2 = \min_{\beta\in\mathbb{R}^{m\times d}} \tfrac{1}{2}\|Y - G\beta\|_F^2. \quad (1)$$

3.2. Global objective function

The global criterion that we consider is a linear combination of the approximation error of K and the loss as defined in Eq. (1), i.e:

$$J(G) = \lambda\|K - GG^\top\|_1 + \mu \min_{\beta\in\mathbb{R}^{m\times d}} \|Y - G\beta\|_F^2.$$

For convenience we use the following normalized values of μ and λ (which correspond to values of the corresponding terms in the objective for $G = 0$):

$$\lambda = \frac{1-\kappa}{\operatorname{tr} K} \text{ and } \mu = \frac{\kappa}{\operatorname{tr} Y^\top Y}.$$

The parameter κ thus calibrates the tradeoff between approximation of K and prediction of Y.

The matrix β can be minimized out to obtain the following criterion:

$$\begin{aligned} J(G) \;=\; & \lambda\|K - GG^\top\|_1 \\ & +\mu \operatorname{tr}\left(Y^\top Y - Y^\top G(G^\top G)^{-1}G^\top Y\right). \end{aligned}$$

Finally, if we incorporate the constraint $K \preceq GG^\top$, we obtain the final form of the criterion:

$$\begin{aligned} J(G) \;=\; & \lambda \operatorname{tr}(K - GG^\top) \\ & +\mu \operatorname{tr}(Y^\top Y - Y^\top G(G^\top G)^{-1}G^\top Y). \end{aligned} \quad (2)$$

4. Cholesky with side information (CSI)

Our algorithm builds on incomplete Cholesky decomposition, restricting the matrices G that it considers to those which are obtained as incomplete Cholesky factors of K. In order to select the pivot, we need to compute the gain in the cost function J in Eq. (2) for each pivot at each iteration.

Let us denote the two terms in the cost function $J(G)$ as $\lambda J^K(G)$ and $\mu J^Y(G)$. The first term has been studied in Section 2, where we found that

$$J^K(G) = \operatorname{tr} K - \textstyle\sum_{k=1}^{m} \|G(:,k)\|_2^2. \quad (3)$$

In order to compute the second term,

$$J^Y(G) = \operatorname{tr}\left(Y^\top Y - Y^\top G(G^\top G)^{-1}G^\top Y\right), \quad (4)$$

efficiently, we need an efficient way of computing the matrix $G(G^\top G)^{-1}G^\top$ which is amenable to cheap updating as G increases. This can be achieved by QR decomposition.

4.1. QR decomposition

Given a rectangular $n \times m$ matrix M, such that $n \geqslant m$, the QR decomposition of M is of the form $M = QR$, where Q is a $n \times m'$ matrix with orthonormal columns, i.e., $Q^\top Q = Id_{m'}$, $m' \leqslant m$, and R is an $m' \times m$ upper triangular matrix. The matrix Q provides an orthonormal basis of the column space of M; if M has full rank m, then $m' = m$, while if not, m' is equal to the rank of m.

The QR decomposition can be seen as the Gram-Schmidt orthonormalization of the column vectors of M (Golub and Loan, 1996); moreover, the matrix $R^\top$ is the Cholesky factor of the $m \times m$ matrix $M^\top M$. A simple iterative algorithm to compute the QR decomposition of M follows the Gram-Schmidt orthonormalization procedure. The first column of Q and R are defined as $Q(:,1) = M(:,1)/\|M(:,1)\|_2$ and $R(1,1) = \|M(:,1)\|_2$. The k-th iteration, $k \leqslant m$, is the following:

$$\begin{aligned} & R(j,k) = Q(:,j)^\top M(:,k), \quad \forall j = 1,\dots,k-1 \\ & R(k,k) = \|M(:,k) - \textstyle\sum_{i=1}^{k-1} R(i,k)Q(:,i)\|_2 \\ & Q(:,k) = \frac{1}{R(k,k)}\left(M(:,i_k) - \textstyle\sum_{i=1}^{k-1} R(i,k)Q(:,i)\right). \end{aligned}$$

The algorithm stops whenever k reaches m or $R(k,k)$ vanishes. The complexity of each iteration is equal to $O(km)$ and thus the total complexity up to the m-th step is $O(m^2 n)$.

4.2. Parallel Cholesky and QR decompositions

While building the Cholesky decomposition G iteratively as described in Section 2.1, we update its QR decomposition at each step. The complexity of each iteration is $O(kn)$ and thus, if the algorithm stops after m steps, the total complexity is $O(m^2 n)$. We still need to describe the pivot selection strategy; as for the Cholesky decomposition we use a greedy strategy, i.e., we chose the pivot that most reduces the cost. In the following sections, we show how this choice can be performed efficiently.

4.3. Cost reduction

We use the following notation: $R_k = R(1:k, 1:k)$, $G_k = G(:, 1:k)$, $Q_k = Q(:, 1:k)$, $g_k = G(:,k)$ and $q_k = Q(:,k)$. After the k-th iteration the cost function is equal to

$$J_k = \lambda\left(\operatorname{tr} K - \sum_{q=1}^{m} \|g_q\|^2\right) + \mu\left(\operatorname{tr} Y^\top Y - \operatorname{tr} Y^\top Q_k Q_k^\top Y\right),$$

and the cost reduction is thus equal to

$$J_{k-1} - J_k = \lambda\Delta J^K + \mu\Delta J^Y, \quad (5)$$

where

$$\Delta J^K = \|g_k\|_2^2 \quad (6)$$

$$\Delta J^Y = \|Y^\top q_k\|_2^2 = \frac{\|Y^\top (Id_n - Q_{k-1}Q_{k-1}^\top)g_k\|_F^2}{\|(Id_n - Q_{k-1}Q_{k-1}^\top)g_k\|_F^2}. \quad (7)$$

Following Section 2.1, we can express g_k in terms of the pivot i_k and the approximation after the $(k-1)$-th iteration, i.e.,

$$g_k = \frac{(K - L_{k-1})(:, i_k)}{(K - L_{k-1})(i_k, i_k)^{1/2}} = \frac{(K - L_{k-1})(:, i_k)}{D_{k-1}(i_k)^{1/2}} \quad (8)$$

Computing this reduction before the k-th iteration for all $n + 1 - k$ available pivots is a prohibitive $O(kn^2)$ operation. As in the case of Cholesky decomposition, a lower bound on the reduction can be computed to avoid this costly operation. However, we have developed a different strategy, one based on a look-ahead algorithm that gives cheap additional information on the kernel matrix. This strategy is presented in the next section.

5. Look-ahead decompositions

At every step of the algorithm, we not only perform one step of Cholesky and QR, but we also perform several "look-ahead steps" to gather more information about the kernel matrix K. Throughout the procedure we maintain the following information: (1) decomposition matrices $G_{k-1} \in \mathbb{R}^{n\times(k-1)}$, $Q_{k-1} \in \mathbb{R}^{n\times(k-1)}$, $D_{k-1} \in \mathbb{R}^n$, and $R_{k-1} \in \mathbb{R}^{(k-1)\times(k-1)}$, obtained from the sequence of indices $I_{k-1} = (i_1, \ldots, i_{k-1})$; (2) additional decomposition matrices obtained by δ additional runs of Cholesky and QR decomposition: $G_{k-1}^{adv} \in \mathbb{R}^{n\times(k-1+\delta)}$, $Q_{k-1}^{adv} \in \mathbb{R}^{n\times(k-1+\delta)}$, $R_{k-1}^{adv} \in \mathbb{R}^{(k-1+\delta)\times(k-1+\delta)}$, $D_{k-1}^{adv} \in \mathbb{R}^n$. The first $k-1$ columns of G_{k-1}^{adv} and Q_{k-1}^{adv} are the matrices G_{k-1} and Q_{k-1}, and the δ additional columns that are added are indexed by $H_k = (h_1^k, \ldots, h_\delta^k)$. We now describe how this information is updated, and how it is used to approximate the cost reduction. A high-level description of the overall algorithm is given in Figure 1.

5.1. Approximation of the cost reduction

After the $(k-1)$-th iteration, we have the following approximations: $L_{k-1} = G_{k-1}G_{k-1}^\top$ and $L_{k-1}^{adv} = G_{k-1}^{adv}(G_{k-1}^{adv})^\top$. In order to approximate the cost reduction defined by Eqs. (5), (6), (7) and (8), we replace all currently unknown portions of the kernel matrix (i.e., the columns whose indices are not in $I_{k-1} \cup H_k$) by the corresponding elements of L_{k-1}^{adv}. This is equivalent to replacing g_k in Eq. (8) by

$$\hat{g}_k = \frac{(L_{k-1}^{adv} - L_{k-1})(:, i_k)}{(K(i_k, i_k) - L_{k-1}(i_k, i_k))^{1/2}}.$$

In order to approximate ΔJ^K, we also make sure that $g_k(i_k)$ is not approximated so that our error term reduces to the lower bound of the incomplete Cholesky decomposition when $\delta = 0$ (i.e., no look-ahead performed); this is obtained through the corrective term η in the following equations. We obtain:

$$\widehat{\Delta} J^K(i_k) = \frac{A_{k-1}(i_k) + \eta_{k-1}(i_k)}{D_{k-1}(i_k)} \quad (9)$$

$$\widehat{\Delta} J^Y(i_k) = \frac{C_{k-1}(i_k)}{B_{k-1}(i_k)} \quad (10)$$

with

$$\eta_{k-1}(i_k) = D_{k-1}(i)^2 - (D_{k-1}(i) - D_{k-1}^{adv}(i))^2$$
$$A_{k-1}(i_k) = \|(L_{k-1}^{adv} - L_{k-1})(:, i_k)\|_2^2$$
$$B_{k-1}(i_k) = \|(Id_n - Q_{k-1}Q_{k-1}^\top)(L_{k-1}^{adv} - L_{k-1})(:, i_k))\|_2^2$$
$$C_{k-1}(i_k) = \|Y^\top (Id_n - Q_{k-1}Q_{k-1}^\top)(L_{k-1}^{adv} - L_{k-1})(:, i_k))\|_F^2.$$

Note that when the index i_k belongs to the set of δ indices that were considered in advance, then the approximation is exact.

A naive computation of the approximation would lead to a prohibitive quadratic complexity in n. We now present a way of updating the quantities defined above as well as a way of updating the δ look-ahead Cholesky and QR steps at a cost of $O(\delta n + dn)$ per iteration.

5.2. Efficient implementation

Updating the look ahead decompositions After the pivot i_k has been chosen, if it was not already included in the set of indices already treated in advance, we perform the additional step of Cholesky and QR decompositions with that pivot. If it was already chosen, we select a new pivot using the usual Cholesky lower bound defined in Section 2. Let G_k^{bef}, Q_k^{bef} and R_k^{bef} be those decompositions with $k + \delta$ columns. In both cases, we obtain a Cholesky decompositions whose k-th pivot is not i_k in general, since i_k may not be among the first look-ahead pivots from the previous iteration. In general, i_k is less than δ indices away from the k-th position. In order to compute G_k^{adv}, Q_k^{adv} and R_k^{adv}, we need to update the Cholesky and QR decompositions to advance pivot i_k to the k-th position. In Appendix A, we show how this can be done with worst-case time complexity $O(\delta n)$, which is faster than naively redoing δ steps of Cholesky decomposition in $O(k\delta n)$.

Updating approximation costs In order to derive update equations for $A_k(i)$, $B_k(i)$ and $C_k(i)$, the cru-

Input: kernel matrix $K \in \mathbb{R}^{n\times n}$,
target matrix $Y \in \mathbb{R}^{n\times d}$,
maximum rank m, tolerance ε,
tradeoff parameter $\kappa \in [0,1]$,
number of look-ahead steps $\delta \in \mathbb{N}$.

Algorithm:

1. Perform δ look-ahead steps of Cholesky (Section 2.1) and QR decomposition (Section 4.1), selecting pivots according to Section 2.2.
2. Initialization: $\eta = 2\varepsilon$, $k = 1$.
3. While $\eta > \varepsilon$ and $k \leqslant m$,
   a. Compute estimated gains for the remaining pivots (Section 5.1), and select best pivot,
   b. If new pivot not in the set of look-ahead pivots, perform a Cholesky and a QR step, otherwise perform the steps with a pivot selected according to Section 2.2,
   c. Permute indices in the Cholesky and QR decomposition to put new pivot in position k, using the method in Appendix A,
   d. Compute exact gain η; let $k = k + 1$.

Output: G and its QR decomposition.

Figure 1. High-level description of the CSI algorithm.

cial point is to notice that

$$\begin{aligned} L_k^{adv}(:,i) &= L_k^{bef}(:,i) \\ &= L_{k-1}^{adv}(:,i) + G_k^{bef}(i,k+\delta)G_k^{bef}(:,k+\delta). \end{aligned}$$

This makes most of the terms in the expansion of $A_k(i)$, $B_k(i)$ and $C_k(i)$ identical to terms in $A_{k-1}(i)$, $B_{k-1}(i)$ and $C_{k-1}(i)$. The total complexity of updating $A_k(i)$, $B_k(i)$ and $C_k(i)$, for all i, is then $O(dn+\delta n)$.

5.3. Computational complexity

The total complexity of the CSI algorithm after m steps is the sum of (a) $m+\delta$ steps of Cholesky and QR decomposition, i.e., $O((m+\delta)^2 n)$, (b) updating the look-ahead decompositions by permuting indices as presented in Appendix A, i.e., $O(\delta m n)$, and (c) updating the approximation costs, i.e., $O(mdn + m\delta n)$.

The total complexity is thus $O((m+\delta)^2 n + mdn)$. In the usual case in which $d \leqslant \max\{m, \delta\}$, this yields a total complexity equal to $O((m+\delta)^2 n)$, which is the same complexity as computing $m+\delta$ steps of Cholesky and QR decomposition. For large kernel matrices, the Cholesky and QR decompositions remain the most costly computations, and thus the CSI algorithm is only a few times slower than the standard incomplete Cholesky decomposition.

We see that the CSI algorithm has the same favorable linear complexity in the number of data points n as standard Cholesky decomposition. In particular, we do not need to examine every entry of the kernel matrix in order to compute the CSI approximation. This is particularly important when the kernel is itself costly to compute, as in the case of string kernels or graph kernels (Shawe-Taylor and Cristianini, 2004).

5.4. Including an intercept

It is straightforward to include an intercept in the CSI algorithm. This is done by replacing Y with $\Pi_n Y$, where $\Pi_n = (Id_n - \frac{1_n 1_n^\top}{n})$ is the centering projection matrix. The Cholesky decomposition is not changed, while the QR decomposition is now performed on $\Pi_n G$ instead of G. The rest of the algorithm is not changed.

6. Experiments

We have conducted a comparison of CSI and incomplete Cholesky decomposition for 37 UCI datasets, including both regression and (multi-way) classification problems. The kernel method that we used in these experiments is the least-squares SVM (Suykens and Vandewalle, 1999). The goal of the comparison was to investigate to what extent we can achieve a lower-rank decomposition with the CSI algorithm as compared to incomplete Cholesky, at equivalent levels of predictive performance.[2]

6.1. Least-squares SVMs

The least-squares SVM (LS-SVM) algorithm is based on the minimization of the following cost function:

$$\tfrac{1}{2n}\|Y - K\alpha - 1_n b\|_F^2 + \tfrac{\tau}{2}\operatorname{tr}\alpha^\top K\alpha,$$

where $\alpha \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^{1\times d}$. This is a classical penalized least-squares problem, whose estimating equations are obtained by setting the derivatives to zero:

$$b = \tfrac{1}{n}(-1_n^\top K\alpha + 1_n^\top Y),\ (K\Pi_n K + n\tau K)\alpha = K\Pi_n Y,$$

where $\Pi_n = (Id_n - \frac{1}{n}1_n 1_n^\top)$.

6.2. Least-squares SVM with incomplete Cholesky decomposition

We now approximate K by an incomplete Cholesky factorization obtained from columns in I, i.e., $L = GG^\top$. Expressed in terms of G, the estimating equa-

[2] A Matlab/C implementation can be downloaded from `http://cmm.ensmp.fr/~bach/`

tions for the LS-SVM become:

$$G(G^\top \Pi_n G + n\tau Id_m)G^\top \alpha = GG^\top \Pi_n Y \quad (11)$$

$$b = \tfrac{1}{n}(-1_n^\top GG^\top \alpha + 1_n^\top Y). \quad (12)$$

The solutions of Eq. (11) are the vectors of the form:

$$\alpha = G(G^\top G)^{-1}(G^\top \Pi_n G + n\tau Id_m)^{-1}G^\top \Pi_n Y + v,$$

where v is any vector orthogonal to the column space of G. Thus α is not uniquely defined; however, the quantity $K\alpha$ is uniquely defined, and equal to $K\alpha = G(G^\top \Pi_n G + \tau Id_m)^{-1}G^\top \Pi_n Y$. We also have $b = \frac{1}{n}(-1_n^\top G(G^\top \Pi_n G + \tau Id_m)^{-1}G^\top \Pi_n Y + 1_n^\top Y)$ and the predicted training responses in $\mathbb{R}^d$ are $K\alpha + b1_n = \Pi_n G(G^\top \Pi_n G + \tau Id_m)^{-1}G^\top \Pi_n Y$.

In order to compute the responses for previously unseen data z_j, for $j = 1, \ldots, n_{test}$, we consider the rectangular testing kernel matrix in $\mathbb{R}^{n_{test} \times n}$, defined as $(K_{test})_{ji} = T(x_i, z_j)$. We use the approximation of K_{test} based on the rows of K_{test} for which the corresponding rows of K were already selected in the Cholesky decomposition of K. If we let I denote those rows, the testing responses are then equal to $K_{test}(:, I)G(I, :)^{-\top}G^\top \alpha$, which is uniquely defined (while α is not). This also has the effect of not requiring the computation of the entire testing kernel matrix K_{test}—a substantial gain for large datasets.

In order to compute the training error and testing errors, we threshold the responses appropriately (by taking the sign for binary classification, or the closest basis vector for multi-class classification, where each class is mapped to a basis vector).

6.3. Experimental results - UCI datasets

We transformed all discrete variables to multivariate real random variables by mapping them to the basis vectors; we also scaled each variable to unit variance. We performed 10 random "75/25" splits of the data. We used a Gaussian-RBF kernel, $T(x, y) = \exp(-\gamma\|x - y\|_2^2)$, with the parameters γ and τ chosen so as to minimize error on the training split. The minimization was performed by grid search.

We trained and tested several LS-SVMs with decompositions of increasing rank, comparing incomplete Cholesky decomposition to the CSI method presented in this paper. The hyperparameters for the CSI algorithm were set to $\kappa = 0.99$ and $\delta = 40$. The value of δ was chosen to be large enough so that in most cases the final rank was the same as if the entire kernel matrix was used, and small enough so that the complexity of the lookahead was small compared to the rest of the Cholesky decomposition.

For both algorithms, the stopping criterion (the minimal gain at each iteration) was set to 10^{-4}. We imposed no upper bound on the ranks of the decompositions.

We report the minimal rank for which the cross-validation error is within a standard deviation of the average testing error obtained when no low-rank decomposition is used. As shown in Figure 2, the CSI algorithm generally yields a decomposition of significantly smaller rank than incomplete Cholesky decomposition; indeed, the difference in minimal ranks achieved by the algorithms can be dramatic.

7. Conclusions

A major theme of machine learning research is the advantages that accrue to "discriminative" methods—methods that adjust all of the parameters of a model to minimize a task-specific loss function. In this paper we have extended this point of view to the matrix algorithms that underlie kernel-based learning methods. With the incomplete Cholesky decomposition as a starting point, we have developed a new low-rank decomposition algorithm for positive semidefinite matrices that can exploit side information (e.g., classification labels). We have shown that this algorithm yields decompositions of significantly lower rank than those obtained with current methods (which ignore the side information). Given that the computational requirements of the new algorithm are comparable to those of standard incomplete Cholesky decomposition, we feel that the new algorithm can and should replace incomplete Cholesky in a variety of applications.

There are several natural extensions of the research reported here that are worth pursuing, most notably the extension of these results to situations in which two or more related kernel matrices have to be approximated conjointly, such as in kernel canonical correlation analysis (Bach and Jordan, 2002) or multiple kernel learning (Lanckriet et al., 2004).

Appendix A. Efficient pivot permutation

In this appendix we describe an efficient algorithm to advance the pivot with index q to position $p < q$ in an incomplete Cholesky and QR decomposition. This can be achieved by $q - p$ transpositions between successive pivots. Permuting two successive pivots $p, p{+}1$ can be done in $O(n)$ as follows (we let denote $P = p{:}p{+}1$):

1. Permute rows p and $p{+}1$ of Q and G
2. Perform QR decomposition $G(P, P)^\top = Q_1 R_1$
3. $R(:, P) \leftarrow R(:, P)Q_1$, $G(:, P) \leftarrow G(:, P)Q_1$

dataset	n_f	n_c	n_p	Chol	CSI
ringnorm	20	2	1000	14	3
kin-32fh-c	32	2	2000	25	6
pumadyn-32nm	32	–	4000	93	23
pumadyn-32fh	32	–	4000	30	8
kin-32fh	32	–	4000	34	10
cmc	12	3	1473	10	3
bank-32fh	32	–	4000	221	72
page-blocks	8	2	5473	451	155
spambase	49	2	4000	90	31
isolet	617	8	1798	254	89
twonorm	20	2	4000	8	3
dermatology	34	2	358	32	14
comp-activ	21	–	4000	159	73
abalone	10	–	4000	27	13
yeast	7	3	673	8	4
titanic	8	2	2201	4	2
kin-32nm-c	32	2	4000	122	68
pendigits	16	4	4485	111	63
adult	3	2	4000	7	4
ionosphere	33	2	351	76	45
liver	6	2	345	15	9
pi-diabetes	8	2	768	10	6
segmentation	15	3	660	5	3
waveform	21	3	2000	8	5
splice	240	3	3175	487	305
census-16h	16	–	1000	42	28
kin-32nm	32	–	2000	307	211
add10	10	–	2000	280	204
mushroom	116	2	4000	60	44
bank-32-nm	32	–	4000	413	328
kin-32nm	32	–	4000	586	479
vehicle	18	2	416	31	27
breast	9	2	683	2	2
thyroid	7	4	1000	1	1
satellite	36	3	2000	2	2
vowel	10	4	360	70	73
optdigits	58	6	2000	68	72
boston	12	–	506	48	61

Figure 2. Simulation results on UCI datasets, where n_f is the number of features, n_c the number of classes ('–' for regression problems), and n_p the number of data points. For both classical incomplete Cholesky decomposition (Chol) and Cholesky decomposition with side information (CSI), we report the minimal rank for which the prediction performance with a decomposition of that rank is within one standard deviation of the performance with a full-rank kernel matrix. Datasets are sorted by the values of the ratios between the last two columns.

4. Perform QR decomposition $R(P,P) = Q_2 R_2$
5. $R(P,:) \leftarrow Q_2^\top R(P,:)$, $Q(:,P) \leftarrow Q(:,P)Q_2$.

The total complexity of permuting indices p and q is thus $O(|p-q|n)$. Note all columns of G and Q between p and q are changed but that the updates involve $|p-q|$ shuffles between successive columns of Q and G.

Acknowledgements

We wish to acknowledge support from a grant from Intel Corporation, and a graduate fellowship to Francis Bach from Microsoft Research. We also wish to acknowledge Grant 0412995 from the National Science Foundation.

References

F. R. Bach and M. I. Jordan. Kernel independent component analysis. *J. Mach. Learn. Res.*, 3:1–48, 2002.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *J. Mach. Learn. Res.*, 2:243–264, 2001.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. J. Hopkins Univ. Press, 1996.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

G. R. G. Lanckriet, N. Cristianini, L. El Ghaoui, P. Bartlett, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, 5:27–72, 2004.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2001.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge Univ. Press, 2004.

A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proc. ICML*, 2000.

J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Proc. Let.*, 9 (3):293–300, 1999.

C. K. I. Williams and M. Seeger. Effect of the input density distribution on kernel-based classifiers. In *Proc. ICML*, 2000.