Large-scale RLSC Learning Without Agony

Wenye Li   WYLI@CSE.CUHK.EDU.HK
Kin-Hong Lee   KHLEE@CSE.CUHK.EDU.HK
Kwong-Sak Leung   KSLEUNG@CSE.CUHK.EDU.HK
Dept. Computer Science & Engineering, The Chinese University of Hong Kong, Hong Kong, China

Abstract

The advances in kernel-based learning necessitate the study on solving a large-scale non-sparse positive definite linear system.

To provide a deterministic approach, recent researches focus on designing fast matrix-vector multiplication techniques coupled with a conjugate gradient method. Instead of using the conjugate gradient method, our paper proposes to use a domain decomposition approach in solving such a linear system. Its convergence property and speed can be understood within von Neumann's alternating projection framework. We will report significant and consistent improvements in convergence speed over the conjugate gradient method when the approach is applied to recent machine learning problems.

1. Introduction

Given training data $(\mathbf{x}_i; y_i)_{i=1}^m$ where $\mathbf{x}_i \in \mathcal{R}^d$ and $y_i \in \mathcal{R}$, the supervised learning tries to seek a predictive function $f : \mathcal{R}^d \to \mathcal{R}$ that explains the relationship between $\mathbf{x}$ and y. Following Tikhonov(Tikhonov & Arsenin, 1977), a simple yet effective approach stems from a regularized learning framework(Poggio & Girosi, 1990)(Poggio & Smale, 2003) by

$$\min_{f \in \mathcal{H}_K} \frac{1}{m} \sum_{i=1}^{m} (y_i - f(\mathbf{x}_i))^2 + \gamma \|f\|_K^2, \quad (1)$$

where $\|f\|_K$ is the norm in $\mathcal{H}_K$ - the Reproducing Kernel Hilbert Space (RKHS), induced by a symmetric positive definite function $K_\mathbf{x}(\mathbf{x}') = K(\mathbf{x}, \mathbf{x}')$, where (a kernel). The last term in Equation (1), where

$\gamma > 0$, forces smoothness and uniqueness of the solution. According to the representer theorem(Kimeldorf & Wahba, 1971), the solution has the form $f = \sum_{j=1}^{m} c_j K_{x_j}$ where $\mathbf{c} = (c_1, \cdots, c_m)^T$ and

$$(\mathbf{K} + \gamma m \mathbf{I}) \mathbf{c} = \mathbf{y} \quad (2)$$

where $\mathbf{K}$ is an $m \times m$ positive definite kernel matrix with elements $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{I}$ is the identity matrix and $\mathbf{y} = (y_1, \cdots, y_m)^T$.

The matrix $\mathbf{A} = \mathbf{K} + \gamma m \mathbf{I}$ is strictly positive definite. The linear system $\mathbf{Ac} = \mathbf{y}$ can be solved in $O(m^3)$ basic floating point operations and $O(m^2)$ storage by direct methods such as Gaussian elimination and LU factorization, where m is the number of training data. And it soon becomes evident that the direct solvers are impractical for large-scale problems, e.g., $m > 10000$.

To deal with this problem, one line of work suggested to use low-rank approximations to the kernel matrix(Smola & Schölkopf, 2000)(Williams & Seeger, 2001)(Fine & Scheinberg, 2001)(Lee & Mangasarian, 2001). These methods try to choose a reduced subset of the training dataset and represent the selected points exactly in the Hilbert space while representing the other points approximately as linear combinations of the selected points. Some success has been achieved, yet these methods do not guarantee the approximation of the kernel matrix in a deterministic sense. These methods assume that most eigenvalues of the matrix are zero. This is not always true and its violation might incur limited reduction in computational time and memory.

To provide a deterministic method to speed up kernel machines learning, (Fung & Mangasarian, 2001) used Sherman-Morrison-Woodbury formula to calculate the inverse of $(\mathbf{K} + \gamma m \mathbf{I})$. This approach works well on a linear kernel with a small d (recall d is the number of dimensions). For other problems, people still need to find general approaches.

Another way for a deterministic solution is to explore

the iterative techniques. Beginning with a given approximate solution, the methods modify the components of the solution successively, until convergence is achieved(Barrett et al., 1994)(Golub & van Loan, 1996)(Saad & van der Vorst, 2000). They do not guarantee a solution for all systems of equations. However, when they do yield a solution, they are usually less expensive than direct methods.

Among the iterative solvers, the conjugate gradient method is generally suggested to solve Equation (2) (Rifkin, 2002)(Yang et al., 2005)(Shen et al., 2006)(Freitas et al., 2006). It consecutively performs a matrix-vector multiplication $\mathbf{Kc}$. The computational cost is $O(m^2)$ per iteration. If $rank(\mathbf{K}) = r$, the method converges theoretically in $r+1$ steps. Then the computational complexity in solving the equation is $O(rm^2)$ and the storage requirement is $O(m^2)$ which is used to store $\mathbf{K}$. While this represents an improvement for some problems, the rank of the matrix may not be small. In machine learning tasks, $\mathbf{K}$ is often full rank and then the computational complexity becomes $O(m^3)$.

In this paper, we consider a different iterative method, a domain decomposition approach, to solve a positive definite linear system from machine learning problems. The approach can be understood and justified as the successive orthogonal projections among different subspaces of an RKHS. When applying this approach to machine learning tasks, we will report significant and consistent improvements in convergence speed over the conjugate gradient method.

A word on notation: Besides the K and $K_\mathbf{x}$ we have used in a kernel function, a bold capital letter denotes a matrix (e.g. $\mathbf{K}$), and a corresponding normal letter with subscript i, j refers to the (i, j)-th entry (e.g. K_{ij}). Similarly, a bold lower case letter denotes a vector (e.g. $\mathbf{c}$), and a corresponding normal letter with subscript i refers to the i-th entry (e.g. c_i). $\mathbf{0}$ denotes a vector of all 0's. Furthermore, let $\mathbf{A}$ be an $m \times m$ matrix and $\mathbf{c}$ be an $m \times 1$ vector. If $s_1, s_2 \subseteq \{1, \cdots, m\}$, then $\mathbf{A}_{s_1, s_2}$ is a matrix obtained from $\mathbf{A}$ by keeping only the rows with indices in s_1 and the columns with indices in s_2, and $\mathbf{c}_{s_1}$ is a vector obtained from $\mathbf{c}$ by keeping the elements with indices in s_1.

2. A Simple Algorithm

To solve a linear equation $\mathbf{Ac} = \mathbf{y}$, where $\mathbf{A}$ is an $m \times m$ strictly positive definite coefficient matrix and $\mathbf{y}$ is an m-dim vector, a simple yet efficient algorithm is shown below:

Algorithm 1 Solve a p.d. linear system $\mathbf{Ac} = \mathbf{y}$.

1: $s = \{1, \cdots, m\}$
2: Divide s into subsets $s_j, j = 1, \cdots, \ell$.
3: $t = 0, \mathbf{c} = \mathbf{0}$
4: **repeat**
5: $\quad t = t + 1, \mathbf{c}^t = \mathbf{0}$
6: $\quad$ **for** $j = 1$ to ℓ **do**
7: $\quad\quad$ Solve $\mathbf{c}_{s_j}^t$ by $\mathbf{A}_{s_j, s_j} \mathbf{c}_{s_j}^t = \mathbf{y}_{s_j}$.
8: $\quad\quad$ $\mathbf{y} = \mathbf{y} - \mathbf{A}_{s, s_j} \mathbf{c}_{s_j}^t$
9: $\quad\quad$ $\mathbf{c}_{s_j} = \mathbf{c}_{s_j} + \mathbf{c}_{s_j}^t$
10: $\quad$ **end for**
11: **until** $\mathbf{c}$ converges
12: **return** $\mathbf{c}$

2.1. Analysis

Some analysis of the algorithm: The algorithm is a standard domain decomposition approach. It solves the linear system $\mathbf{Ac} = \mathbf{y}$ iteratively. In an iteration from step 5 to step 10, the algorithm solves ℓ subsystems $\mathbf{A}_{s_j, s_j} \mathbf{c}_{s_j}^t = \mathbf{y}_{s_j}, 1 \leq j \leq \ell$. For simplicity, we assume the size of each subsystem is $k = \left[\frac{m}{\ell}\right]$, which may be regarded as a constant. The computational cost of each subsystem in step 7 is $O(k^3)$ by direct methods. There is a trick which greatly simplifies the computation. We first compute the Cholesky factorization of the coefficient matrix in each subsystem before the iterations start, which requires a cost of $O(\ell k^3) = O(k^2 m)$. Then we use the factorized matrices to solve the subsystems, which requires only $O(k^2)$ in solving one and $O(\ell k^2) = O(km)$ in solving all.

The major computation comes from the modification of the residual in step 8. The total computation of $\mathbf{A}_{s, s_j} \mathbf{c}_{s_j}^t, 1 \leq j \leq \ell$ roughly requires $O(m^2)$ by direct matrix-vector multiplications. We can also resort to the fast matrix-vector multiplication methods[1] developed recently(Sun & Pitsianis, 2001)(Yang et al., 2005), which may simplify the computation to a linear complexity in some cases.

In summary, the computational complexity per iteration is $O(m^2)$, which is the same as conjugate gradient. However, in terms of the convergence speed, as we will see later, this approach needs far fewer iterations in solving a positive definite linear system from machine learning tasks.

The major memory consumption comes from the storage of $\mathbf{A}_{s, s_j}$ in step 8. The memory requirement is

---
[1] Instead of multiplying $\mathbf{Ac}$ directly, these methods first decompose $\mathbf{A} = \mathbf{D}^T \mathbf{D}$, where $\mathbf{D}$ has a much lower row dimension than $\mathbf{A}$. Then the methods compute $\mathbf{D}^T(\mathbf{Dc})$.

$O(m^2)$ if we store the whole coefficient matrix $\mathbf{A}$, which is the same as conjugate gradient. It is also possible to store only one $\mathbf{A}_{s,s_j}$ at a time, which requires only an $O(km)$ storage. When the fast matrix-vector multiplication methods are applicable, the memory requirement may be further reduced.

2.2. Parallelism

To tackle large-scale problems in practice, we often consider the concurrent execution when designing an algorithm. For the domain decomposition approach presented above, we can parallelize it easily.

As we have analyzed, the major computation comes from the iterative modification of the residual in step 8. Since this step only involves matrix-vector multiplication, it can be parallelized. The computations of $\mathbf{A}_{s,s_j}\mathbf{c}_{s_j}^t$ can be evenly partitioned to different computing nodes and then the results are summed up. The memory burden can also be distributed to all the nodes. Each node only needs to store one part of $\mathbf{A}$ instead of the whole kernel matrix, and the storage requirement becomes much smaller. As for the communication cost in parallelism, only the multiplication results need to be synchronized in each iteration, which is trivial.

3. Justification

The algorithm in section 2 may be justified by two steps. The first step maps the linear system in Equation (2) to an interpolation problem, and then establishes the relationship between the interpolation problem and the orthogonal projection in a Hilbert space. The second step analyzes the orthogonal projection by von Neumann's alternating projection theorem. The convergence property and speed of the algorithm become evident after the analysis.

The following lemma is a necessary step to understand a positive definite linear system $\mathbf{Ac} = \mathbf{y}$ from an interpolation point of view.

Lemma 1 *For any $m \times m$ positive definite matrix $\mathbf{A}$, there exist m points $\mathbf{x}_1, \cdots, \mathbf{x}_m$ in a space R^d and a kernel function K defined on R^d, such that $A_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.*

Proof. We only need to find such m points $\mathbf{x}_1, \cdots, \mathbf{x}_m$ and such a kernel K defined on a space R^d.

Since $\mathbf{A}$ is positive definite, $\mathbf{A}$ can be decomposed into

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T,$$

where $\mathbf{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_m)$ is an orthogonal matrix and $\mathbf{\Lambda}$ is a diagonal matrix with non-negative diagonal entries $\lambda_1, \cdots, \lambda_m$. Letting $d = m$, $\mathbf{x}_i = (\sqrt{\lambda_1}v_{1i}, \cdots, \sqrt{\lambda_m}v_{mi})^T$, $(1 \le i \le m)$ and K be a linear kernel defined on R^d completes the proof. ∎

With this lemma, we can see that the problem of finding a solution to $\mathbf{Ac} = \mathbf{y}$ is equivalent to the problem of finding a function $f = \sum_{j=1}^m c_j K_{\mathbf{x}_j}$ such that $f(\mathbf{x}_i) = y_i$ for all $\mathbf{x}_i$. Here we abuse the notation slightly. The K used here is a different kernel from the K in Equation (1).

We need the following definitions. Given a kernel function $K_\mathbf{x}(\mathbf{x}') = K(\mathbf{x}, \mathbf{x}')$ and a finite set of distinct points $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$ in R^d, let $\mathcal{H}_K$ denote the RKHS induced by K. That is, $\mathcal{H}_K$ is a finite dimensional space

$$\left\{ \sum_{j=1}^m a_j K_{\mathbf{x}_j}(\cdot) : a_1, \cdots, a_m \in R \right\}$$

endowed with the inner product

$$\langle f, g \rangle = \sum_{i,j=1}^m a_i b_j K(\mathbf{x}_i, \mathbf{x}_j)$$

where

$$f = \sum_{i=1}^m a_i K_{\mathbf{x}_i} \text{ and } g = \sum_{j=1}^m b_j K_{\mathbf{x}_j}.$$

3.1. Interpolation Operator and Orthogonal Projection

Let us recall the definition of orthogonal projection in a Hilbert space.

Definition 1 (*Orthogonal Projection*) *Let $\mathcal{V}$ be a closed subspace of a Hilbert space $\mathcal{H}$. The linear operator $P : \mathcal{H} \to \mathcal{V}$ is called the orthogonal projection onto $\mathcal{V}$ if for any $f \in \mathcal{H}$ and any $v \in \mathcal{V}$*

$$\langle v, f - Pf \rangle = 0,$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product in $\mathcal{H}$.

The orthogonal projection is associated with the best approximation. For any $f \in \mathcal{H}$ and a closed subspace $\mathcal{V}$ of $\mathcal{H}$, the orthogonal projection of f onto $\mathcal{V}$ is the unique $g \in \mathcal{V}$ such that $\langle f - g, f - g \rangle$ is minimized.

We also define the interpolation operator which is relevant to our linear system.

Definition 2 (*Interpolation Operator*) *Let $X_1, \cdots, X_\ell$ be subsets of X, such that $\cup_{i=1}^\ell X_i = X$. Given $f \in \mathcal{H}_K$, define interpolation operators $P_i : \mathcal{H}_K \to \mathcal{H}_i, i = 1, \cdots, \ell$ by*

$$P_i f = \sum_{\mathbf{x} \in X_i} c_\mathbf{x} K_\mathbf{x} \text{ and } (P_i f)(\mathbf{z}) = f(\mathbf{z}) \text{ for all } \mathbf{z} \in X_i.$$

The next result identify each operator P_i as the orthogonal projection onto the function subspace $\mathcal{H}_i$. This is a key observation utilized in (Faul & Powell, 1999)(Schaback & Wendland, 2000)(Beatson et al. 2000). And we explained it in terms of the relationship between the interpolation operators and the orthogonal projections.

Lemma 2 *Let $X_i \subseteq X, (1 \le i \le \ell)$ be a finite set of distinct points in R^d and P_i denote the interpolation operator defined above. Then P_i is the orthogonal projection from $\mathcal{H}_K$ onto $\mathcal{H}_i$.*

Proof. Given $f = \sum_{\mathbf{x} \in X} a_\mathbf{x} K_\mathbf{x}$ and $\mathcal{H}_i$ with a basis $\{K_{\mathbf{v}_1}, \cdots, K_{\mathbf{v}_n}\}$, where $\{\mathbf{v}_1, \cdots, \mathbf{v}_n\} \subseteq X_i$. Suppose

$$P_i f = \sum_{j=1}^n c_{\mathbf{v}_j} K_{\mathbf{v}_j}.$$

exists. The coefficients $\mathbf{c} = (c_{\mathbf{v}_1}, \cdots, c_{\mathbf{v}_n})^T$ must solve the linear system

$$\mathbf{Gc} = \mathbf{b}$$

with

$$G_{jk} = \langle K_{\mathbf{v}_j}, K_{\mathbf{v}_k} \rangle, \; j, k = 1, \cdots, n$$

and

$$b_j = \langle K_{\mathbf{v}_j}, f \rangle = f(\mathbf{v}_j), \; j = 1, \cdots, n.$$

The Gram matrix $\mathbf{G}$ is strictly positive definite and the solution is unique. In fact, it can be verified that $P_i f(\mathbf{x}) = f(\mathbf{x})$ holds for all $\mathbf{x} \in X_i$. Hence $P_i f$ exists and is unique. It can also be easily verified that $P_i f$ is the best approximation of f in $\mathcal{H}_i$. That is, $P_i f$ is the unique g such that $\langle f - g, f - g \rangle$ is minimized. We omitted the verifications here.

The lemma now follows from the characterization of $f \in \mathcal{H}_K$ onto a closed subspace $\mathcal{H}_i$ of $\mathcal{H}_K$, which is the unique $g \in \mathcal{H}_i$ such that $\langle f - g, f - g \rangle$ is minimized. ∎

It is now possible to understand the algorithm in section 2 as a version of von Neumann's alternating projections. Each execution of step 7 in the algorithm corresponds to an orthogonal projection from $\mathcal{H}_K$ onto its subspace $\mathcal{H}_i$.

3.2. von Neumann's Alternating Projection Theorem

The von Neumann's alternating projection theorem(von Neumann, 1955) concerns an orthogonal projection in a general Hilbert space $\mathcal{H}$. The motivation is as follows: It is well known that the orthogonal projection onto the intersection $\mathcal{U} \cap \mathcal{V}$ of the two closed subspaces $\mathcal{U}$ and $\mathcal{V}$ of $\mathcal{H}$ can be obtained by the product $P_u P_v$ if $P_u P_v = P_v P_u$, where P_u and P_v are the orthogonal projections onto $\mathcal{U}$ and $\mathcal{V}$, respectively. The question arises out of the orthogonal projection onto $\mathcal{U} \cap \mathcal{V}$ when $P_u P_v \neq P_v P_u$. Denote this desired projection by $P_u \wedge P_v$, and the answer is given by the limit of the sequence

$$\lim_{t\to\infty} (P_u P_v)^t f = (P_u \wedge P_v) f, \quad (3)$$

where the left-hand side converges to the right-hand side in the norm of $\mathcal{H}$ and f denotes any element of $\mathcal{H}$. Equation (3) generalizes by induction to any finite number of subspaces and their corresponding projections. This is von Neumann's alternating projection theorem.

3.3. Convergence Speed

We are able to analyze the convergence speed of the domain decomposition approach in section 2 by von Neumann's alternating projection algorithm, the convergence property of which has been well studied in mathematics. If we define $Q_i = I - P_i$, $i = 1, \cdots, \ell$, then Q_i is the orthogonal projection onto $\mathcal{H}_i^\perp$ (the subspace orthogonal and complementary to $\mathcal{H}_i$), and $(Q_\ell \cdots Q_1)^t f$ denotes t complete iterations of the alternating projections. The convergence speed of the alternating projection algorithm can be understood in terms of the angles among the subspaces $\mathcal{H}_i^\perp$.

Definition 3 *Let $\mathcal{U}_1$ and $\mathcal{U}_2$ be two closed subspaces of a Hilbert space $\mathcal{H}$ with $\mathcal{U} = \mathcal{U}_1 \cap \mathcal{U}_2$. Then the angle α between $\mathcal{U}_1$ and $\mathcal{U}_2$ is given by*

$$\cos\alpha = \sup\left\{ \left\langle \frac{\mathbf{u}}{\|\mathbf{u}\|}, \frac{\mathbf{v}}{\|\mathbf{v}\|} \right\rangle : \mathbf{u} \in \mathcal{U}_1 - \mathcal{U}, \mathbf{v} \in \mathcal{U}_2 - \mathcal{U} \right\}. \quad (4)$$

The following theorem(Smith et al., 1977) guarantees that the alternating projection algorithm has at least a linear convergence speed.

Theorem 1 Let $Q_1, \cdots, Q_\ell$ be ℓ orthogonal projections onto closed subspaces $\mathcal{U}_1, \cdots, \mathcal{U}_\ell$ of a Hilbert space $\mathcal{H}$. Let $\mathcal{U} = \cap_{i=1}^{\ell} \mathcal{U}_i$. Let $Q : \mathcal{H} \to \mathcal{U}$ be the orthogonal projection onto $\mathcal{U}$, and let α_j be the angle between $\mathcal{U}_j$ and $\cap_{i=j+1}^{\ell} \mathcal{U}_i$. Then for any $f \in \mathcal{H}$,

$$\left\| \left(Q_\ell \cdots Q_1\right)^t f - Qf \right\|^2 \leq c^{2t} \|f - Qf\|^2,$$

where

$$c^2 \leq 1 - \prod_{j=1}^{\ell-1} \sin^2 \alpha_j.$$

It is noteworthy that this is a pessimistic estimation of the convergence rate. In practice, as we will see in the experiments, the convergence speed is often much faster. It is also noteworthy that the angle between some $\mathcal{U}_j$ and $\cap_{i=j+1}^{\ell} \mathcal{U}_i$ might be zero and $c = 1$, which makes us not be able to verify the approach's convergence property by this theorem. Obviously, some trivial assumptions will rule out this possibility, for example, each $\mathcal{U}_j$, ($1 \leq j \leq \ell$) has an element which only belongs to it, and to no other $\mathcal{U}_j$'s.

3.4. A Domain Decomposition Approach

Based on the justification, we come to the following domain decomposition approach. It generates the sequence of residuals $\{f_{t+i}\}$, where $t = 1, 2, \cdots$ and $i = 1, 2, \cdots, \ell$ via

$$f_0 = f \text{ and } f_{t+i} = f_{t+i-1} - P_i f_{t+i-1}.$$

And the sequence of approximations $\{c_{t+i}\}$ is generated by:

$$c_0 = 0 \text{ and } c_{t+i} = c_{t+i-1} + P_i f_{t+i-1}.$$

The algorithm in section 2 exactly implements this approach.

A graphical illustration is given in figure (1). Suppose a Hilbert space $\mathcal{H}_K$ can be decomposed into two subspaces: $\mathcal{U}$ and $\mathcal{V}$. For any unknown $f \in \mathcal{H}_K$, we project $f_0 = f$ onto $\mathcal{U}$ and the projection $P_u f_0$ can be computed. In the second step, we project $f_1 = Q_u f_0 = f_0 - P_u f_0$ onto $\mathcal{V}$ and $P_v f_1$ can be computed. Then we project $f_2 = Q_v f_1 = f_1 - P_v f_1$ onto $\mathcal{U}$, and so on and so forth. By von Neumann's alternating projection theorem, the projection comes to $\mathcal{U} \cap \mathcal{V}$ in the end, which is the zero function in our case. Correspondingly, we start from a zero function. By summing up the successive projections of f_i, ($i = 0, 1, 2, \cdots$) onto $\mathcal{U}$ or $\mathcal{V}$, we can approximate f to arbitrary precision in the norm of the space.

In some sense, this domain decomposition approach and the classical conjugate gradient method belong to two completely different categories of solvers of linear equations. As shown in figure (1), the domain decomposition method may be regarded as a *projection-based* method. Conjugate gradient, just as its name implies, is a *gradient*-based method. Instead of solving $\mathbf{Ac} = \mathbf{y}$ directly, the method studies the minimization of a quadratic form: $\frac{1}{2}\mathbf{c}^T \mathbf{Ac} + \mathbf{y}^T \mathbf{c}$. Starting with an arbitrary point $\mathbf{c}_0$, the method slides successively to another approximation along the conjugate gradient direction in minimizing the quadratic function. It reaches a minimizer when it converges, which is equivalent to the solution to $\mathbf{Ac} = \mathbf{y}$ if $\mathbf{A}$ is symmetric and positive definite.

4. Experiments

To compare the performance of the domain decomposition approach and the conjugate gradient method in machine learning tasks, we carried out a series of experiments in solving positive definite linear equations. The equations were generated in the regularized least-squares classification (RLSC) problems. Two commonly-used kernels, the linear kernel and Gaussian RBF kernel, were adopted in the experiments. Because the Gaussian kernel parameter and the regularization parameter γ affect the convergence speed of both algorithms, we used cross validation beforehand to select the parameters which gave the optimal classification accuracy[2].

We report the results of six experiments using the three datasets from CMU text mining group[3], which are representative of our many experiments on different datasets. The 20-newsgroups dataset has about 19,000 web pages in 20 classes. The webkb dataset has about 8,300 pages in 7 classes. The 7-sectors dataset has about 4,600 pages in 7 classes. In the experiments, a document is represented by *bag-of-words*. We first computed the kernel matrix $\mathbf{K}$ using the whole dataset with a linear kernel and a Gaussian kernel respectively. Then we got the matrix $\mathbf{A}$ by $\mathbf{K} + \gamma m \mathbf{I}$. Each element of $\mathbf{y}$ was set by a number specifying a document's class label, i.e., $1, 2, \cdots, 20$ for 20-newsgroups dataset[4]. Finally, we solved the linear system $\mathbf{Ac} = \mathbf{y}$ for the vector $\mathbf{c}$. For domain decomposition, we set the size of each subsystem X_i to be 1,000.

Figure (2)(a)-(f) depicts the results of relative residuals $\frac{\|\mathbf{Ac}-\mathbf{y}\|}{\|\mathbf{y}\|}$ versus the number of iterations. We can see that comparing with the conjugate gradient method which exhibits oscillatory behaviors in the experiments, the domain decomposition approach is more stable in reducing the relative residuals. It reaches an acceptable residual after about 10 iterations in the experiments, which is much fewer than the number of iterations required by the conjugate gradient method. As for the running time, the two methods do not differ much in each iteration, and so domain decomposition requires much less running time than conjugate gradient to converge.

5. Conclusions

Recent researches in machine learning necessitate the study on solving a large-scale positive definite linear system. To provide a deterministic solver for this type of equations, people generally resort to the conjugate gradient method. The conjugate gradient method with incomplete Cholesky factorizations is the de facto iterative solver for sparse positive definite linear systems which typically originate from the systems related to elliptic partial differential equations. However, the positive definite systems in machine learning problems are often not sparse, the Cholesky factorization becomes expensive for large-scale dense systems, and a direct application of conjugate gradient may not be the best choice.

In this paper, we considered a domain decomposition approach for the task. It is not a new approach, and can be regarded as a variant of block Gauss-Seidel method in solving linear equations. People may also found the similarity in the backfitting algorithm for generalized linear models(Hastie & Tibshirani, 1986).

The viewpoint in analyzing this approach was ever used in (Beatson et al., 2000), where the authors designed a fast algorithm for interpolation problems and studied its convergence property within von Neumann's alternating projection framework after having established the algorithm's relationship with the orthogonal projections in a Hilbert space. This paper showed that their analysis generalizes to all positive definite linear systems.

We applied the approach to the regularized least-squares classification problems and obtained significant and consistent improvements over the standard conjugate gradient method. Besides RLSC, a series of other models(Williams & Rasmussen, 1996)(Li et al., 2007) may also benefit from this approach. Another advantage of the approach lies in the parallelism, which provides the ability to solve large-scale problems in practice.

Two topics related to the approach need to be further investigated. In the analysis of the algorithm, the angles between different subspaces are shown to be relevant to the convergence speed. In fact, it is also possible to study the angles defined in Equation (4) explicitly. How to partition the data into different subsystems so as to maximize the angles and make the algorithm converge faster? One possible solution is to pre-process the data using, for example, some tree-based structures(Preparata & Shamos, 1985)(Shen et al., 2006).

The second investigation is relevant to the real applications. It is not unusual to see a machine learning task which has up to millions of training data. With such a domain decomposition approach and the fast matrix-vector multiplication techniques, the effectiveness of kernel-based method in large-scale learning may be practically verified.

After finishing the work of this paper, we came to know the recent work of (Ratliff & Bagnell, 2007) that gave a novel variant of the conjugate gradient method to speed up the training of kernel machines. Empirical comparisons between the work and the domain decomposition approach will be carried out in the future.

Figure 1. A graphical illustration of the domain decomposition approach. An unknown function f comes near the zero function after several projections. Accordingly, we can approximate f based on the projection information.

---

[2] For a *wide* range of parameter choices near the optimal ones, similar convergence trends were observed in the comparisons.

[3] http://www.cs.cmu.edu/~TextLearning/datasets.html

[4] Similar convergence trends were observed when each y is set to either -1 or $+1$ for practical binary classification.
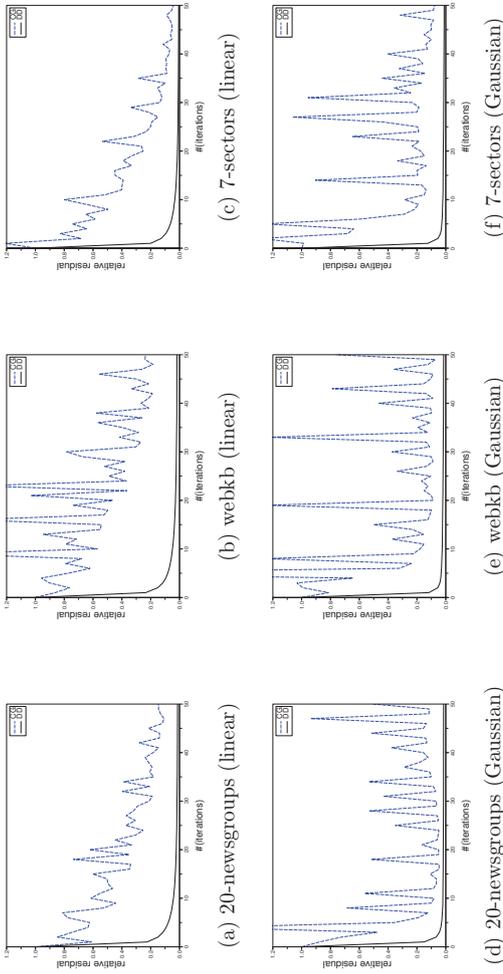
Figure 2. Comparisons of convergence speed between conjugate gradient (CG) and domain decomposition (DD) using different datasets and kernels. The domain decomposition approach requires much fewer iterations than the conjugate gradient method to converge.

(a) 20-newsgroups (linear)

(b) webkb (linear)

(c) 7-sectors (linear)

(d) 20-newsgroups (Gaussian)

(e) webkb (Gaussian)

(f) 7-sectors (Gaussian)

References

Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., & van der Vorst, H. (1994). *Templates for the solution of linear systems: Building blocks for iterative methods.* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Beatson, R., Light, W., & Billings, S. (2000). Fast solution of the radial basis function interpolation equations: Domain decomposition methods. *SIAM J. Sci. Comput*, *22*, 1717–1740.

Faul, A., & Powell, M. (1999). Proof of convergence of an iterative technique for thin plate spline interpolation in two dimensions. *Advances in Computational Mathematics*, *11*, 183–192.

Fine, S., & Scheinberg, K. (2001). Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, *2*, 243–264.

Freitas, N. D., Wang, Y., Mahdaviani, M., & Lang, D. (2006). Fast Krylov methods for n-body learning. In Y. Weiss, B. Schölkopf and J. Platt (Eds.), *Advances in neural information processing systems 18*, 251–258. Cambridge, MA: MIT Press.

Fung, G., & Mangasarian, O. (2001). Proximal support vector machine classifiers. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)* (pp. 77–86). New York: ACM Press.

Golub, G. H., & van Loan, C. F. (1996). *Matrix computations.* John Hopkins Studies in the Mathematical Sciences. Baltimore, Maryland: Johns Hopkins University Press. Third edition.

Hastie, T., & Tibshirani, R. (1986). Generalized additive models. *Statistical Science*, *1*, 297–318.

Kimeldorf, G., & Wahba, G. (1971). Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.*, *33*, 82–95.

Lee, Y.-J., & Mangasarian, O. (2001). Rsvm: Reduced support vector machines. *First SIAM International Conference on Data Mining.* Chicago.

Li, W., Lee, K.-H., & Leung, K.-S. (2007). Generalized regularized least-squares learning with predefined features in a Hilbert space. In B. Schölkopf, J. Platt and T. Hoffman (Eds.), *Advances in neural information processing systems 19.* Cambridge, MA: MIT Press.

Poggio, T., & Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, *247*, 978–982.

Poggio, T., & Smale, S. (2003). The mathematics of learning: Dealing with data. *Not. Am. Math. Soc*, *50*, 537–544.

Preparata, F. P., & Shamos, M. I. (1985). *Computational geometry: An introduction.* Springer.

Ratliff, N. D., & Bagnell, J. A. (2007). Kernel conjugate gradient for fast kernel machines. *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007* (pp. 1017–1022).

Rifkin, R. (2002). *Everything old is new again: A fresh look at historical approaches in machine learning.* Doctoral dissertation, Massachusetts Institute of Technology.

Saad, Y., & van der Vorst, H. A. (2000). Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics*, *123*, 1–33. Numerical analysis 2000, Vol. III. Linear algebra.

Schaback, R., & Wendland, H. (2000). Numerical techniques based on radial basis functions. *Curve and Surface Fitting* (pp. 359–374). Nashville, TN: Vanderbilt University Press.

Shen, Y., Ng, A., & Seeger, M. (2006). Fast Gaussian process regression using kd-trees. In Y. Weiss, B. Schölkopf and J. Platt (Eds.), *Advances in neural information processing systems 18*, 1227–1234. Cambridge, MA: MIT Press.

Smith, K., Solomon, D., & Wagner, S. (1977). Practical and mathematical aspects of the problem of reconstructing objects from radiographs. *Bulletin of the American Mathematical Society*, 1227–1270.

Smola, A. J., & Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 911–918). Morgan Kaufmann.

Sun, X., & Pitsianis, N. P. (2001). A matrix version of the fast multipole method. *SIAM Review*, *43*, 289–300.

Tikhonov, A., & Arsenin, V. (1977). *Solutions of ill-posed problems.* Winston and Sons.

von Neumann, J. (1955). *Mathematical foundations of quantum mechanics.* Princeton University Press.

Williams, C. K. I., & Rasmussen, C. E. (1996). Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer and M. E. Hasselmo (Eds.), *Advances in neural information processing systems*, vol. 8. Cambridge, MA: MIT Press.

Williams, C. K. I., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Dietterich and V. Tresp (Eds.), *Advances in neural information processing systems 13*, 682–688. MIT Press.

Yang, C., Duraiswami, R., & Davis, L. (2005). Efficient kernel machines using the improved fast Gauss transform. In L. K. Saul, Y. Weiss and L. Bottou (Eds.), *Advances in neural information processing systems 17*, 1561–1568. Cambridge, MA: MIT Press.