Large-Scale RLSC Learning Without Agony

Wenye Li
wyli@cse.cuhk.edu.hk

Dept. Computer Science & Engineering
The Chinese University of Hong Kong

Joint work with Kin-Hong Lee, Kwong-Sak Leung

Introduction

- Supervised Learning
  - Given $(x_i; y_i)_{i=1}^{m}$, $where\ x_i \in R^d, y_i \in R$
  - Seek $y = f(x) + N(0, \Sigma)$
- An Effective Approach
  - Define $H_K = completion \quad of \ \left\{\sum_{x \in R^d} c_x K_x(\cdot), c_x \in R\right\}$, $\left(e.g. K_x(\cdot) = e^{\frac{-\|x-\cdot\|^2}{\sigma^2}} \right)$
    (inner product given by the kernel K)
  - Seek $f(\cdot) \in H_K$ to minimize $L = \frac{1}{m}\sum_{i=1}^{m}(y_i - f(x_i))^2 + \gamma \|f\|_K^2, \gamma \geq 0$
    - We are seeking a simple function that has small empirical error.
    - For classification → Regularized Least-Squares Classification.

Vapnik 1998, Poggio et. al. 1990

Introduction

$$\min L = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - f(x_i)\right)^2 + \gamma \|f\|_K^2$$

- Representer Theorem
  - It's tractable to seek a solution in the function space with (infinite) dimensions H_K
  - Only finite parameters are not zero $f(\cdot) = \sum_{i=1}^{m} c_i K_{x_i}(\cdot)$
- Computation
  - The parameters come from a positive definite linear system

$$\mathbf{Ac} = \mathbf{y}, \textit{ where } \mathbf{A} = \mathbf{K} + \gamma m \mathbf{I}$$

$$K_{ij} = K_{x_i}\left(x_j\right), \mathbf{I} : \textit{identity matrix}, \mathbf{c} = \left(c_1, \cdots, c_m\right)^T, \mathbf{y} = \left(y_1, \cdots, y_m\right)^T$$

Kimerldorf & Wahba 1971, Girosi et.al. 1995, Schölkopf et.al. 2001

Introduction

$$\mathbf{Ac} = \mathbf{y}$$

- Computation (cont.)
  - Simple, efficient
  - Cholesky factorization for small-medium sized problems
    - Time: $O(m^3)$, space: $O(m^2)$
  - Yet, how about large-scale problems?
    - Suppose m=1,000 takes 8MB memory & 1 second
    - m=20,000 takes ~3.2GB memory & 8,000 seconds
    - m=1,000,000 takes ~8,000GB memory & ~31.7 years

Direct methods impractical!!!

Previous Work

$$\mathbf{Ac} = \mathbf{y}$$

- Approximation methods
  - Low rank approximations
    - choose a subset of the training dataset and represent these points exactly in RKHS, while representing other points approximately as linear combinations of the selected points
    - Assumption: most eigenvalues of the kernel matrix are zero

Smola & Sch?lkopf, 2000, Williams & Seeger, 2001, Fine & Scheinberg, 2001, Lee & Mangasarian, 2001 …

Previous Work

- Deterministic Approach
  - Sherman-Morrison-Woodbury formula to calculate the inverse of the kernel matrix
    - Works well on a linear kernel with a small dimension.

Previous Work

- Deterministic Approaches
  - Iterative methods
    - Begin with a given approximated solution, modify the solution successively until convergence.
    - Two vital elements
      - Fast Matrix-Vector Multiplication
        - Fast Gauss Transform et. al.
      - Iterative Scheme
        - Conjugate gradient is generally suggested

Sun & Pitsianis, 2001, Shen et. al, 2006, Rifkin, 2006, Yang et. al., 2005, Freitas et. al. 2006

Previous Work

$$\mathbf{Ac} = \mathbf{y}$$

- Conjugate gradient for p.d. system
  - Instead of solving the system directly, CG tries to minimize
    $$\frac{1}{2}\mathbf{c}^T\mathbf{A}\mathbf{c} + \mathbf{y}^T\mathbf{c}$$
  - Starting arbitrarily, CG slides successively to another point along the conjugate direction to minimize the quadratic function.
  - The minimizer is equivalent to the solution to the linear system when CG converges.
  - Coupled with incomplete Cholesky factorization as a pre-conditioner, CG is de facto iterative solver for sparse p.d. systems.

Barrett et. al. 1994, Golub & van Loan 1996, Saad & van der Vorst 2000

Challenge

- In machine learning, the kernel matrix is generally not sparse.
  - Think of a Gaussian RBF kernel: none of the elements are zero.
  - It is generally difficult to find a pre-conditioner for such dense systems. Will a naïve application of CG still be the best solution?

Domain Decomposition: Ac=y

01: $s = \{1, \cdots, m\}$
02: $\textit{Divide } s \textit{ to subsets } s_i, i = 1, \cdots, \ell$
03: $t = 0, \mathbf{c} = 0$
04: $\textit{repeat}$
05: $\quad t = t + 1, \mathbf{c}^t = 0$
06: $\quad \textit{for } i = 1 \textit{ to } \ell \textit{ do}$
07: $\quad\quad \textit{Solve } \mathbf{c}_{s_i}{}^t \textit{ by } \mathbf{A}_{s_i, s_i} \mathbf{c}_{s_i}{}^t = \mathbf{y}_{s_i}$
08: $\quad\quad \mathbf{y} = \mathbf{y} - \mathbf{A}_{s, s_i} \mathbf{c}_{s_i}{}^t$
09: $\quad\quad \mathbf{c}_{s_i} = \mathbf{c}_{s_i} + \mathbf{c}_{s_i}{}^t$
10: $\quad \textit{endfor}$
11: $\textit{until } \mathbf{c} \textit{ converges}$
12: $\textit{return } \mathbf{c}$

- Notes:

- Assume the size each sub-system is $k = \left\lceil \frac{m}{\ell} \right\rceil$

- Time complexity per iteration (step 5—10): $O(\ell k^2) + O(m^2) = O(km) + O(m^2)$

- Memory requirement:
  $O(m^2)$ *if we store the whole* $\mathbf{A}$;
  $O(km)$ *if we only store* $\mathbf{A}_{s,s_i}$

- The major computation, which is from step 8, can be parallelized.

Convergence

- **Observation 1:** For any $m \times m$ positive definite matrix $\boldsymbol{A}$, there exist m points $X = \{x_1, \cdots, x_m\}$ in a space $\boldsymbol{R}^{\mathrm{d}}$ and a kernel function K defined on $\boldsymbol{R}^{\mathrm{d}}$ such that $A_{ij} = K(x_i, x_j), 1 \le i, j \le m.$
- Define $H_K \equiv \left\{ \sum_{j=1}^{m} c_{x_j} K_{x_j}(\cdot), c_{x_j} \in R \right\}$ with an inner product by K, then:
  - a solution to $\boldsymbol{Ac}=\boldsymbol{y}$ ↔ a function f in $\boldsymbol{H}_K$ such that $f(x_j)=y_j$ for all x_j.
- Given $X_i \subseteq X, 1 \le i \le \ell$ *such that* $\bigcup X_i = X$, $\boldsymbol{H}_i$: subspace of functions of $\boldsymbol{H}_K$ associated with $\boldsymbol{X}_i$. That is: $H_i = \left\{ \sum_{x \in X_i} c_x K_x(\cdot), c_x \in R \right\}$
- Given f in $\boldsymbol{H}_K$, define ***interpolation operators*** P_i**:**$\boldsymbol{H}_K \rightarrow \boldsymbol{H}_i$**,** $i=1,..,l$ *by*

$$P_i f = \sum_{x \in X_i} c_x K_x \;\; \mathit{and} \;\; (P_i f)(z) = f(z) \;\; \mathit{for\ all} \;\; z \in X_i$$

Convergence

- **Observation 2:**
  - Each interpolation operator P_i in fact defines the orthogonal projection from $\boldsymbol{H}_K$ to $\boldsymbol{H}_i$.
  - With the two observations, we associate the process of solving a p.d. linear system with alternating projections in an r.k.h.s.

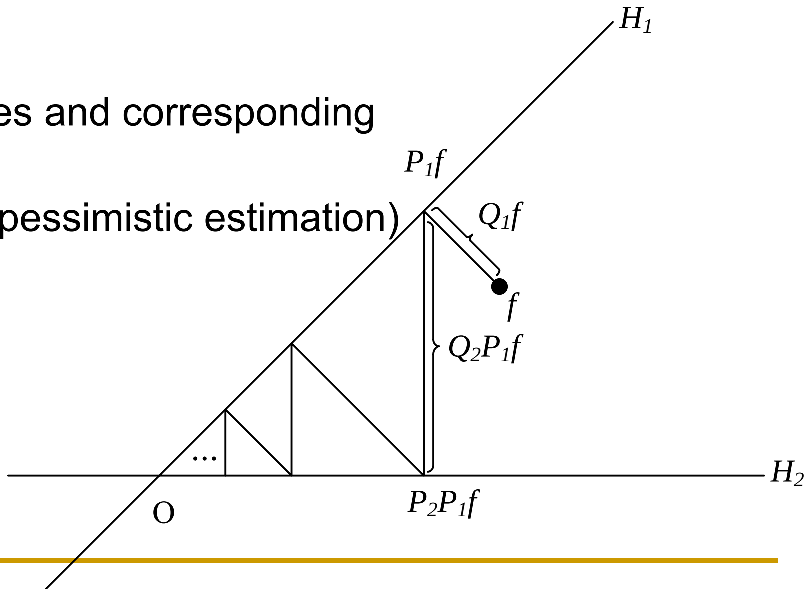
Faul & Powell 1999, Schaback & Wendland 2000, Beatson et. al. 2000

von Neumann's Alternating Projections

- von Neumann's alternating projection theorem:
  - $\boldsymbol{H_1}$ and $\boldsymbol{H_2}$: two closed subspaces of a general Hilbert space $\boldsymbol{H}$
  - $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$: two orthogonal projections onto $\boldsymbol{H_1}$ and $\boldsymbol{H_2}$.
  - How to get the orthogonal projection $\boldsymbol{P_1}$^$\boldsymbol{P_2}$ onto $\boldsymbol{H_1} \cap \boldsymbol{H_2}$?

$$\lim_{t\to\infty}\left(P_1P_2\right)^t f = \left(P_1 \wedge P_2\right) f$$

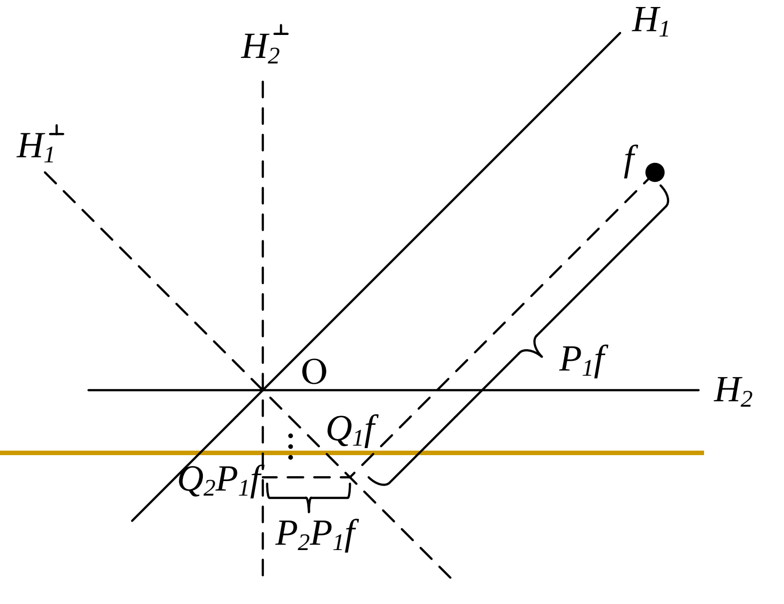
  - Generalizes to any finite number of subspaces and corresponding projections.
  - The convergence speed is at least linear. (a pessimistic estimation)



von Neumann, 1955, Smith et. al 1977

A Domain Decomposition Approach

- $f_0=f$ and $f_{lt+i}=f_{lt+i-1}-P_i f_{lt+i-1}$, where $t=1, 2, \ldots,$ and $i=1,\ldots,l$
-
- Correspondingly,
-
- $c_0=0$ and $c_{lt+i}=c_{lt+i-1}+P_i f_{lt+i-1}$, where $t=1, 2, \ldots,$ and $i=1,\ldots,l$
- Angle between subspaces is vital!!!

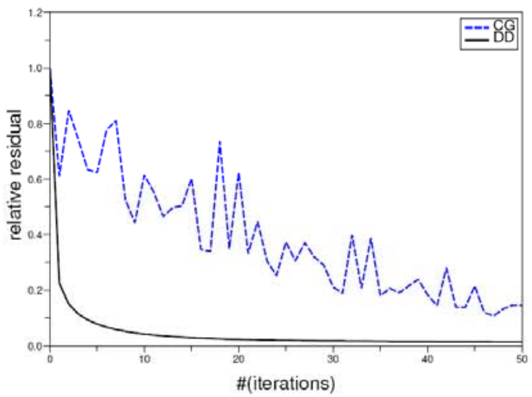


Experiments

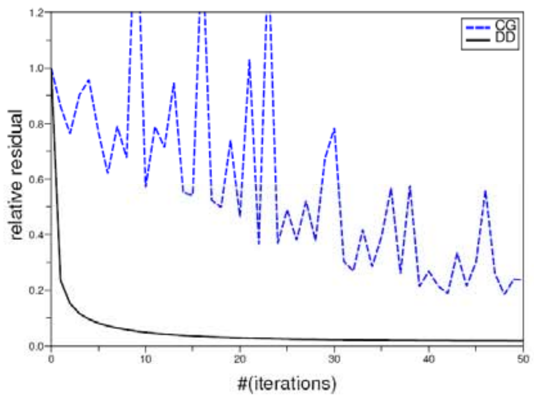
- Text Categorization
  - Three datasets from CMU text mining groups
    - 20-newsgroups: ~19,000 webpages in 20 classes
    - Webkb: ~8,300 pages in 7 classes
    - 7-sectors: ~4,600 pages in 7 classes
  - $Ac = y$
    - Linear&Gaussian RBF kernels are used.
    - Necessary parameters are selected via CV.
    - y: corresponding to the labels of each document
  - The domain decomposition approach reports better convergence property over conjugate gradient

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

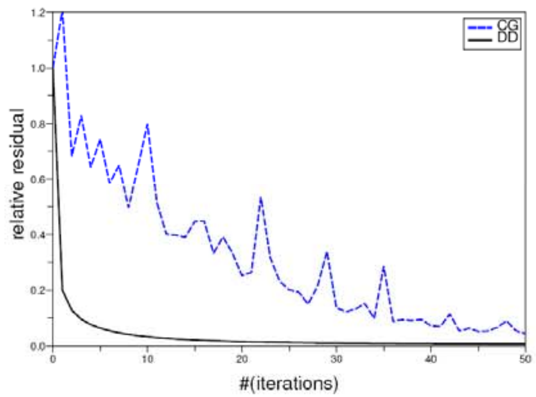
Experiments



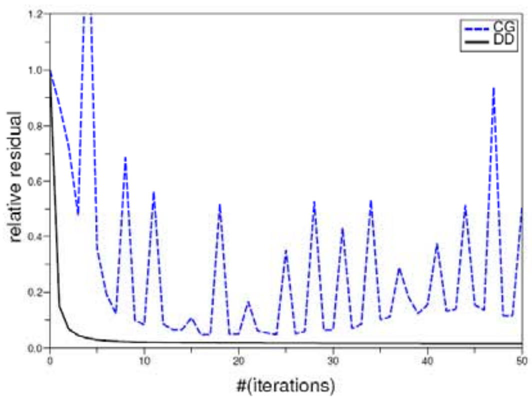
(a) 20-newsgroups (linear)



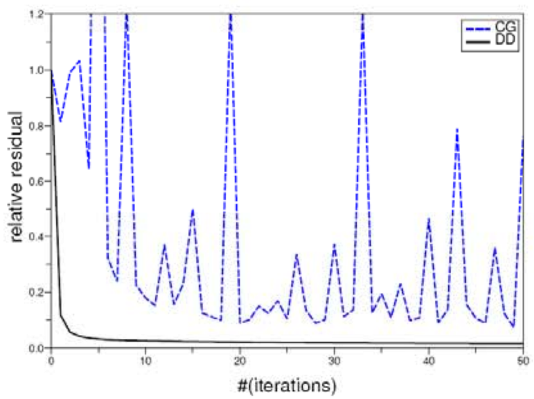
(b) webkb (linear)



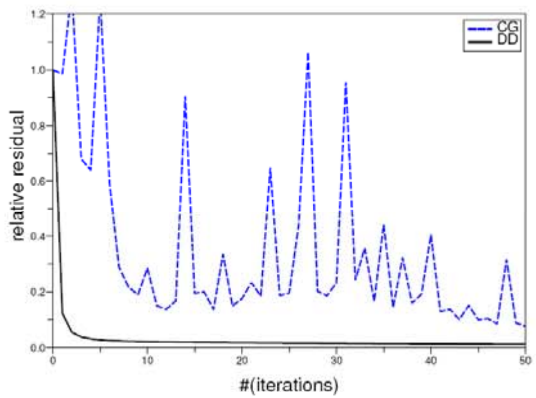
(c) 7-sectors (linear)



(d) 20-newsgroups (Gaussian)



(e) webkb (Gaussian)



(f) 7-sectors (Gaussian)

Conclusion

- An empirical study on iterative schemes for RLSC, GP …
  - A variant of block Gauss-Seidel method is suggested in solving p.d. linear systems in machine learning. Also a domain decomposition approach.
  - Divide and conquer works
- An analysis
  - Alternating projections in r.k.h.s.
  - Which factor is important to the convergence.
  - How to adjust this factor. (not included)
- Further work
  - Interleaving CG with Gauss-Seidel or other methods…
  - Comparisons with kernel conjugate gradient (Ratliff & Bagnell, 2007)

Thanks