computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

FAST SOLVER FOR INTERIOR POINT METHOD OF SVM TRAINING BY PARALLEL GMRES AND HSS

Di Zhao
Center for Cognitive and Brain Science, The Ohio State University
College of Medicine, The Ohio State University
e-mail: zhao.1029@osu.edu
Web address (URL): https://cog.osu.edu/people/zhao

Abstract: Support Vector Machine (SVM) is one of the latest statistical models for machine learning. The key problem of SVM training is an optimization problem (mainly Quadratic Programming). Interior Point Method (IPM) is one of mainstream methods to solve Quadratic Programming problem. However, when large-scale dataset is used in IPM based SVM training, computational difficulty happens because of computationally expensive matrix operations. Preconditioner, such as Cholesky factorization (CF), incomplete Cholesky factorization and Kronecker factorization, is an effective approach to decrease time complexity of IPM based SVM training. In this paper, we reformulate SVM training into the saddle point problem. As the research question that motivates this paper, based on parallel GMRES and recently developed preconditioner Hermitian/Skew-Hermitian Separation (HSS), we develop a fast solver HSS-pGMRES-IPM for the saddle point problem from SVM training. Computational results show that, the fast solver HSS-pGMRES-IPM significantly increases the solution speed for the saddle point problem from SVM training than the conventional solver CF. *Copyright © Research Institute for Intelligent Computer Systems, 2014. All rights reserved.*

Keywords: Interior Point Method, fast solver, parallel GMRES, Hermitian/Skew-Hermitian Separation, Support Vector Machine, Quadratic Programming.

1. INTRODUCTION

Support Vector Machine (SVM) is one of the latest statistical models for machine learning [1-6]. SVM is invented by Vladimir N. Vapnik of Columbia University, and soft margin SVM is published in 1995. The key problem of SVM training is an optimization problem [7, 8] which includes Linear Programming and Quadratic Programming.

While Linear Programming can be highly efficiently solved by methods such as Interior Point Method (IPM), active set and Simplex method, Quadratic Programming can be solved by multiple existing methods such as IPM, active set, augmented Lagrangian and Conjugate Gradient. In these methods, IPM is one of mainstream methods to solve Quadratic Programming from SVM training.

However, when large dataset is trained by IPM based SVM, computational difficulty happens because of computationally expensive matrix operations. Decreasing the time complexity of IPM based SVM training can be realized by methods such as chunking, decomposition, sequential minimal optimization and factorization.

IPM calculates the best solution by searching the interior of the optimization space [9-13] in Linear Programming [14] and Quadratic Programming [9]. IPM can be implemented by multiple algorithms, and Mehrotra predictor–corrector algorithm is the most popular one among them [15-20]. The main idea of Mehrotra predictor-corrector algorithm is to firstly calculate a search direction by the first-order predictor term, then to calculate the second-order corrector term, and finally to combine the predictor term and the corrector term into the complete search direction.

The most time-consuming part of IPM based SVM training is to solve the linear systems. In Mehrotra predictor-corrector algorithm, solving the linear systems happens twice in every iteration. Directly solving the linear systems by the non-factorization solver Gauss Jordan Elimination (GJE) is expensive, which needs the time complexity of $O(n^3)$ [21-25].

Factorization can be applied to decrease the time complexity of IPM based SVM training. Theoretically, factorizations such as LU factorization, LDU factorization, full rank factorization, QR factorization, LDL factorization, Cholesky factorization (CF) and Kronecker factorization can be apply to IPM [26-28]. For IPM based SVM training, since the kernel matrix Q is positive semi-definite matrix, CF is the conventional method to factorize the kernel matrix. The time complexity of CF is $O\left(\frac{1}{3}n^3 + 2n^2\right)$.

Hermitian/Skew-Hermitian Separation (HSS) is a newly developed method for matrix factorization. Can HSS accelerate IPM based SVM training? In this paper, we reform SVM training into the saddle point problem, we develop a fast solver HSS-pGMRES-IPM for the saddle point problem from SVM training, and theoretical analysis and computational results are also provided.

2. METHODS

In this section, we briefly introduce HSS-pGMRES for the saddle point problem, we reform SVM training into the saddle point problem, and we develop a quick solver, HSS-pGMRES-IPM, for solving the saddle point problem from SVM training.

2.1. HERMITIAN/SKEW–HERMITIAN SEPARATION–PARALLEL GMRES FOR SADDLE POINT PROBLEM

Saddle point problem is a linear system with the form:

$$\begin{bmatrix} F & D^T \\ D & -E \end{bmatrix} \begin{bmatrix} du \\ dp \end{bmatrix} = \begin{bmatrix} R_d \\ r_d \end{bmatrix}, \tag{1}$$

where F and E are usually symmetric matrices [29], du and dp are unknown variables, and R_d and r_d are right-hand-sides. Saddle point problems appear with high-frequency in scientific and engineering applications. Golub reviewed solution methods for saddle point problem in [29], and his solution methods for saddle point problem include Schur complement reduction, null space methods, coupled direct solvers, stationary iterations, Krylov subspace methods, preconditioner and multilevel methods [29].

Newly developed matrix splitting based methods such as HSS provide an efficient way to solve saddle point problems [30-32]. Golub *et al.* developed HSS in [33], parameter optimization for HSS is proposed in [34], and preconditioned HSS is studied in [35-40].

To efficiently solve a linear system with the structure of saddle point problem ~~of~~ in equation (1) with the symmetric part H and the skew-symmetric part S, we firstly solve an uncoupled linear system:

$$(H + \alpha I_n) \cdot du^{k+\frac{1}{2}} = f_{uc}^k, \tag{2.1}$$
$$(E + \alpha I_m) \cdot dp^{k+\frac{1}{2}} = g_{uc}^k. \tag{2.2}$$

where α is a parameter, and $f_{uc}{}^k$ and $g_{uc}{}^k$ are right-hand-side. Then we solve a coupled linear system:

$$(\alpha I_n + S) \cdot du^{k+1} + D^T \cdot dp^{k+1} = f_c^k, \tag{3.1}$$
$$-D \cdot du^{k+1} + \alpha p^{k+1} = g_c^k. \tag{3.2}$$

where α is a parameter, and f^k and g^k are right-hand-side. By Schur complement reduction, we obtain:

$$[D(I_n + \alpha^{-1}S)^{-1}D^T + \alpha^2 I_m] \cdot dp^{k+1} = D(I_n + \alpha^{-1}S)^{-1}f^k + \alpha g^k. \tag{4}$$

Since the coefficient matrix $[D(I_n + \alpha^{-1}S)^{-1}D^T + \alpha^2 I_m]$ of equations (4) is a large and sparse matrix, GMRES is suitable to solve dp^{k+1}. After dp^{k+1} is solved, then we obtain du^{k+1}. The details of HSS is described in Algorithm 1:

Algorithm 1: The Hermitian/Skew-Hermitian Separation

```
• Initialization of HSS
• while $R < tol_{HSS}$
      Solve the coupled system equations (2)
      Solve the uncoupled system equations (3)
      end while
```

From Algorithm 1, we can see that the Hermitian/Skew-Hermitian Separation is built by a single loop, while the number of iteration is controlled by the tolerance $R < tol_{HSS}$. In every iteration, two linear systems are solved: the coupled system of equation (2) and the uncouple system of equation (3).

Convergence analysis of HSS (Algorithm 1) is analyzed in [35], the number of iterations can be found in [29, 32-37, 39, 41, 42], and the convergence speed of HSS (Algorithm 1) is decided by tol_{HSS}. However, the linear systems are unnecessary to be solved exactly, and the tolerance of the iterative solver for the linear systems can be loosened to increase the solution speed, which results in inexact HSS.

As we discussed, the uncoupled system of equations (4) in HSS (Algorithm 1) can be solved by sparse solvers such as GMRES, and the speed of the sparse solver decides the efficiency of HSS. We have developed a parallel Gram-Schmidt process based GMRES to simultaneously calculate vector projection in Gram-Schmidt process of GMRES.

Parallel Gram-Schmidt process based GMRES (pGMRES) is applied to HSS (Algorithm 1) to construct a fast solver HSS-pGMRES for saddle point problem. HSS-pGMRES consists of two loops: the outer loop for HSS and the inner loop for pGMRES, and the details of HSS-pGMRES are described in Algorithm 2:

Algorithm 2: Hermitian/Skew-Hermitian Separation−pGMRES

```
• Initialization of HSS
• while $R < tol_{HSS}$
    Solve the couple system equation (2)
    Solve the uncouple system equation (3)
        Initialization of pGMRES
        while $R_k < tol_{GMRES}$
            $w_0^{k+1} = A \cdot v^k$
            Calculate $v_{i+1}^{k+1}$ by parallel Gram-Schmidt Process
            Calculate $y_k$
        end while
        $u^k = u^0 + V_k y_k$
    end while
```

From Algorithm 2, we can see that HSS-pGMRES is built by a double loop: the outer loop of HSS and the inner loop of pGMRES. The number of outer iterations is controlled by the tolerance $R < tol_{HSS}$., and the number of outer iterations is controlled by the tolerance $R_k < tol_{GMRES}$. The inner loop pGMRES is responsible for solving the uncoupled linear system of equation (3), and every iteration of the outer loop HSS is responsible for solving the coupled linear system of equation (2).

In the conventional GMRES, we need k times computation of vector projection in k iteration of the inner loop GMRES and n iteration of the outer loop HSS (Algorithm 1). Therefore, we need total $\frac{m(m+1)n}{2}$ computation of vector projection with time complexity $O(m^2n)$ to build all orthogonal sets.

In HSS-pGMRES (Algorithm 2), we calculate the vector projection simultaneously in k iteration of the inner loop GMRES and n iteration of the outer loop HSS (Algorithm 1) in Fig. 1.

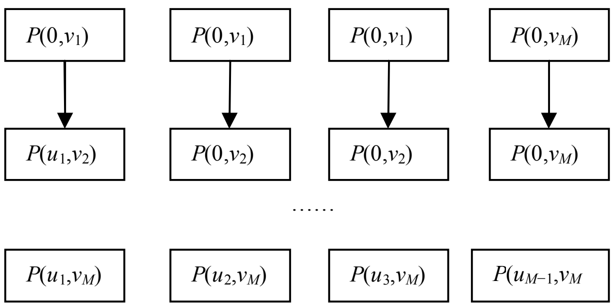


Fig. 1 – Parallel Grad-Schmidt process based pGMRES.

As Fig. 1 is showing, we only need mn computation of vector projection with time complexity $O(mn)$ to build the orthogonal set u.

2.2. SADDLE POINT EQUATION FROM IPM-SVM TRAINING

The primal form of SVM training can be represented by Quadratic Programming problem [43]:

$$\min_x \frac{1}{2} x^T Q x - e^T x,$$
$$ax = 0,$$
$$0 \le x \le C,$$

where x is the array of the Lagrange multipliers, a is the diagonal matrix of labels, and C is a parameter.

After Lagrange multiplier transformation, we obtain KKT conditions [44]. For details of algebra process from primal-dual problem to KKT conditions, the reader is referred to [9].

$$Xs = \sigma\mu e$$
$$(C - X)z = \sigma\mu e$$
$$a^T x = 0$$
$$-Qx + ay + s - z = -e$$
$$0 \le x \le c, s \ge 0, z \ge 0$$

By Mehrotra predictor-corrector algorithm [9, 45], we obtain the linear system in both the predictor step and the corrector step:

$$\begin{bmatrix} -Q & a & I & -I \\ a^T & 0 & 0 & 0 \\ S & 0 & X & 0 \\ -Z & 0 & 0 & (C-X) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \\ \Delta z \end{bmatrix} = \begin{bmatrix} r_c \\ r_b \\ r_s \\ r_z \end{bmatrix}, \quad (5)$$

where r_c, r_b, r_s and r_z are the right-hand-side. Eliminating Δs and Δz from the linear systems [43], we obtain the augmented linear system:

$$\begin{bmatrix} -(D+Q) & a^T \\ a & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} R_c \\ R_b \end{bmatrix}.$$

Also, the stop condition must be carefully selected for efficient convergence. For details of IPM implementation is comprehensively discussed in [9].

2.3 HSS-PGMRES-IPM FOR SVM TRAINING

In this subsection we apply HSS-pGMRES to solve the saddle point problem of equation (1) from IPM based SVM training. To efficiently solve a linear system with the structure of saddle point problem of equation (1) with the symmetric part $-D - \frac{1}{2}Q - \frac{1}{2}Q^T$ and the skew-symmetric part

$-\frac{1}{2}Q+\frac{1}{2}Q^T$, we firstly solve an uncoupled linear system:

$$\left(-D-\frac{1}{2}Q-\frac{1}{2}Q^T+\alpha I_n\right)\cdot dx^{k+\frac{1}{2}}=f_{uc}^k, \quad (6.1)$$

$$\alpha\cdot dy^{k+\frac{1}{2}}=g_{uc}^k. \quad (6.2)$$

where α is a parameter, and $f_{uc}{}^k$ and $g_{uc}{}^k$ are right-hand-side. Then we solve a coupled linear system:

$$\left(\alpha I_n-\frac{1}{2}Q+\frac{1}{2}Q^T\right)\cdot dx^{k+1}+a^T\cdot dy^{k+1}=f_c^k (7.1)$$

$$-a\cdot dx^{k+1}+\alpha y^{k+1}=g_c^k. \quad (7.2)$$

where α is a parameter, and f^k and g^k are right-hand-side. By Schur complement reduction, we obtain:

$$\left[a\left(I_n-\frac{1}{2\alpha}Q+\frac{1}{2\alpha}Q^T\right)^{-1}a^T+\alpha^2 I_m\right]\cdot dy^{k+1}$$
$$=a\left(I_n-\frac{1}{2\alpha}Q+\frac{1}{2\alpha}Q^T\right)^{-1}f^k+\alpha g^k. \quad (8)$$

Since the coefficient matrix $[a(I_n+\alpha^{-1}S)^{-1}a^T+\alpha^2 I_m]$ of equation (4) is large and sparse matrix, GMRES is suitable to solve dy^{k+1}. After dy^{k+1} is solved, then we obtain dx^{k+1}. The details of HSS-pGMRES-IPM is described in Algorithm 3:

Algorithm 3: HSS-pGMRES-IPM for SVM Training

```
• Initial IPM: calculate $(x^0,y^0,s^0,t^0)$ which satisfy the constraints
• while $R_t < tol_{IPM}$
      Solve $(\Delta x^{pre},\Delta y^{pre},\Delta s^{pre},\Delta t^{pre})$ from equation (1) by HSS-pGMRES (Algorithm 2)
      Calculate the step size $\alpha$ which satisfy the constraints
      Solve $(\Delta x^{cor},\Delta y^{cor},\Delta s^{cor},\Delta t^{cor})$ from equation (1) by HSS-pGMRES (Algorithm 2)
      Update the search direction by the formula:
          $(\Delta x,\Delta y,\Delta s,\Delta t) = (\Delta x^{pre},\Delta y^{pre},\Delta s^{pre},\Delta t^{pre}) + (\Delta x^{cor},\Delta y^{cor},\Delta s^{cor},\Delta t^{cor})$
      Update the optimization variable:
          $x^{k+1}=x^k+\alpha\Delta x^k$,
          $(y^{k+1},s^{k+1},t^{k+1}) = (y^k,s^k,t^k)+\alpha(\Delta y^k,\Delta s^k,\Delta t^k)$,
  end
```

From Algorithm 3 we can see, HSS-pGMRES-IPM is built by triple loops: the outer loop of IPM, the middle loop of HSS and the inner loop of pGMRES. The number of outer iterations is controlled by the IPM tolerance $R_t < tol_{IPM}$, the number of middle iterations is controlled by the tolerance $R < tol_{HSS}$, and the number of outer iterations is controlled by the tolerance $R_k < tol_{GMRES}$. The inner loop and the middle loop HSS-pGMRES are responsible for solving the predictor linear system of equation (1) and the corrector linear system of equation (1), and every iteration of the outer loop IPM is responsible for calculating the forward step $(\Delta x,\Delta y,\Delta s,\Delta t)$.

2.4 CONVERGENCE OF HSS-PGMRES-IPM

As we discussed previously, the fast solver HSS-pGMRES-IPM (Algorithm 3) consists of triple loops: the outer loop IPM and the middle and inner loop HSS-pGMRES. The general convergence theory of the outer loop IPM is described in [9], and the convergence analysis of the middle and inner loop HSS-pGMRES can be found in [35, 41, 42].

In the implementation of the fast solver HSS-pGMRES-IPM (Algorithm 3), three separated tolerances for every loop exist: the tolerance for the outer loop tol_{IPM}, the tolerance for the middle loop tol_{HSS} and the tolerance for the inner loop tol_{GMRES}. The three tolerances are unnecessary to be treated equally. The tolerance for the outer loop tol_{IPM} is often replaced by the number of total iterations $k < K$. The tolerance for the middle loop tol_{HSS} can be loosed, which is inexact HSS.

3. COMPUTATIONAL RESULTS

In this section, the performance of HSS-pGMRES-IPM (Algorithm 3) is illustrated by an example of SVM training problem.

3.1. The Problem

The dataset, "Classification of Human Lung Carcinomas by mRNA Expression Profiling Reveals Distinct Adenocarcinoma Sub-classes", comes from the cancer datasets of the Broad Institute of MIT [46]. The dataset includes 203 samples with 12600 genes in each sample. Kernel function is set as: $K(A,B)=(\langle x_i,x_j\rangle-min\langle x_i,x_j\rangle)$. We develop the SVM code on MATLAB, and we develop the code of the fast solver HSS-pGMRES-IPM (Algorithm 3) for SVM training, and existing codes are referenced [14, 47]. The workstation is Intel i5-2310 at 2.90GHz with 4GB memory.

3.2. THE PERFORMANCE OF HSS-PGMRES-IPM

To comparing time cost of the non-factorization solver GJE, the conventional factorization solver CF and the fast solver HSS-pGMRES-IPM (Algorithm 3) for SVM training, we set all conditions exactly the same except the solution method for the predictor linear system and the corrector linear system of equation (1). The time cost is plotted in Fig. 2.

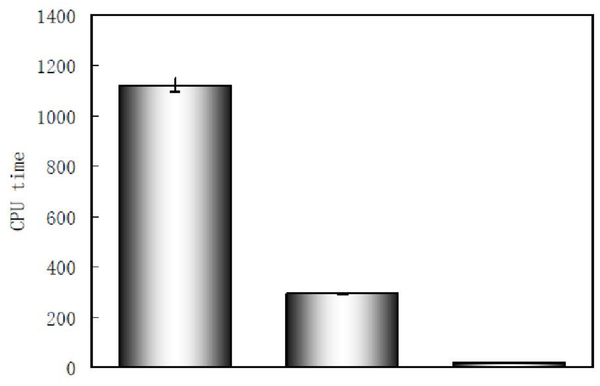


Fig. 2 – Time cost of SVM training with the non-factorization solver GJE (the left column), the conventional factorization solver CF (the middle column) and the fast solver HSS-pGMRES-IPM (the right column).

From Fig. 2 we can see, with maintaining the training accuracy of 90 ± 5%, the non-factorization solver GJE spends 1121.9 ± 25.4 seconds, the conventional factorization solver CF costs 292.9 ± 3.0 seconds, and the fast solver HSS-pGMRES-IPM (Algorithm 3) needs only 19.7 ± 0.0 seconds.

To quantitatively comparing the solution speed among these solvers, we define the acceleration rate among two solvers as the following:

$$rate = \frac{t_i}{t_j},$$

where t_i is the time cost of the first solver, t_i is the time cost of the second solver, and *rate* is the calculated acceleration rate. The calculated acceleration rates for the three solvers: the non-factorization solver GJE, the conventional factorization solver CF and the fast solver HSS-pGMRES-IPM (Algorithm 3) are listed in Table 1.

From Table 1 we can see, the fast solver HSS-pGMRES-IPM (Algorithm 3) is approximately 56.95 times faster than the non-factorization solver GJE, and the fast solver HSS-pGMRES-IPM (Algorithm 3) is about 14.87 times faster than the conventional solver CF. From Fig. 2 and Table 1 we can see, the fast solver HSS-pGMRES-IPM (Algorithm 3) significantly accelerates the solution speed of saddle point problem from IPM based SVM training.

Table 1. Calculated acceleration rate among the three solvers: the non-factorization solver GJE, the conventional factorization solver CF and the fast solver HSS-pGMRES-IPM.

	GJE	CF	HSS-pGMRES-IPM
GJE	1.00	3.83	56.95
CF	–	1.00	14.87
HSS-pGMRES-IPM	–	–	1.00

3.3. THE SCALABILITY OF HSS-PGMRES-IPM BASED SVM TRAINING

To evaluate the scalability of three solvers the non-factorization solver GJE, the conventional factorization solver CF and the fast solver HSS-pGMRES-IPM (Algorithm 3) on small datasets, 250 genes are selected from the original dataset but keeping the number of sample of 203. The computational results of SVM training accuracy are plotted in Fig. 3.

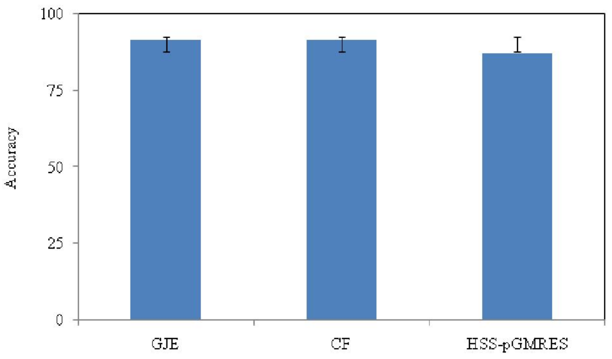


Fig. 3 – The accuracy of the three solvers the non-factorization solver GJE, the conventional factorization CF and the fast solver HSS-pGMRES-IPM on SVM training accuracy.

From Fig. 3 we can see, the difference among SVM training accuracy from the three solvers the non-factorization solver GJE, the conventional factorization CF and the fast solver HSS-pGMRES-IPM (Algorithm 3) are insignificant. The accuracy of the fast solver HSS-pGMRES-IPM (Algorithm 3) slightly decreases while the non-factorization solver GJE and the conventional factorization solver CF keep stable.

When training the small dataset by SVM, the time cost of the three solvers the non-factorization solver GJE, the conventional factorization solver CF and the fast solver HSS-pGMRES-IPM

(Algorithm 3) are measured. The computational experiments are repeated for three times, the average and the standard deviation are calculated. The calculated average time cost and the standard deviation are plotted in Fig. 4.

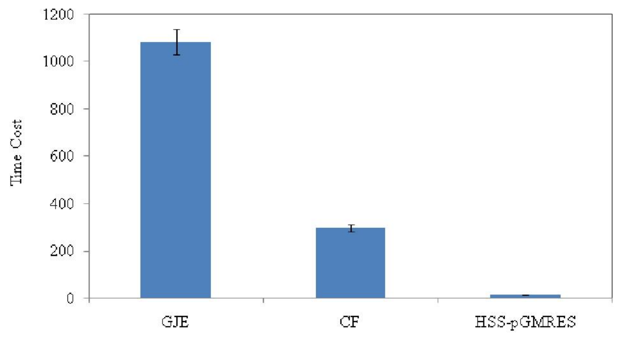


Fig. 4 – Effect of the parameter C of the fast solver HSS-pGMRES-IPM based SVM on time cost.

Comparing Fig. 2 with Fig. 4, no significant difference is presented between the time cost of the non-factorization solver GJE in Fig. 2 and that of in Fig. 4, of the conventional factorization solver CF and of the fast solver HSS-pGMRES-IPM (Algorithm 3).

The time cost of the solvers is decided by the size of the kernel matrix Q, and the size of the kernel matrix Q is decided by the number of samples. From Fig. 2 to Fig. 4, although the number of genes decreases, the number of samples keeps the same. Therefore, the size of the kernel matrix Q in Fig. 2 and Fig. 4 is the same, which leads to identical time cost between Fig. 2 and Fig. 4.

3.4. THE EFFECT OF PARAMETER C ON ACCURACY

How to select Parameter C of SVM is an long-term but important problem. We test the performance of HSS-pGMRES-IPM (Algorithm 3) with different selection of Parameter C. Results are listed in Fig. 5, which is similar to our former research results.

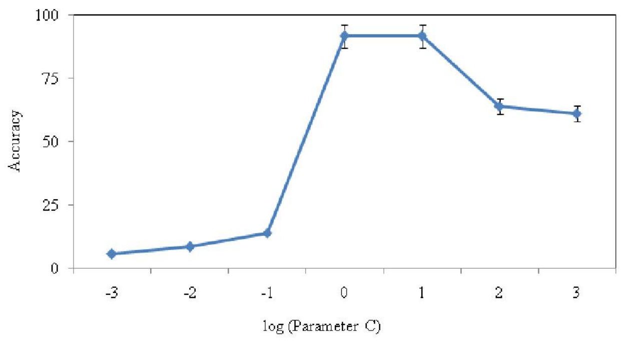


Fig. 5 – Effect of Parameter C of HSS-pGMRES-IPM based SVM on training accuracy.

From Fig. 5 we can see, different selection of Parameter C significantly affects the performance of HSS-pGMRES-IPM (Algorithm 3) based SVM, and parameter C should be selected at middle of the value range.

We also calculate the time cost of HSS-pGMRES-IPM (Algorithm 3) based SVM with different selection of Parameter C. The values of Parameter C are selected from 10^1 to 10^7, the tests are repeated for three times, and the average and the standard deviation are calculated, and the results are plotted in Fig. 6.

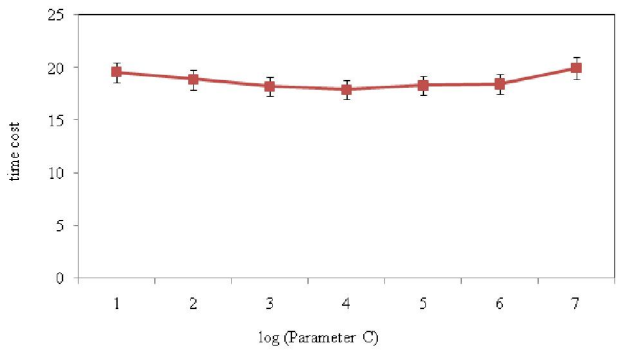


Fig. 6 – Effect of Parameter C of HSS-pGMRES-IPM based SVM on Time Cost.

From Fig. 6 we can see, different selection of parameter C does not affect the time cost of HSS-pGMRES-IPM (Algorithm 3) based SVM. The standard deviation of every selection from different Parameter C is small, which also proves that the time cost of HSS-pGMRES-IPM (Algorithm 3) based SVM is not significantly affected by Parameter C.

4. DISCUSSION

In this paper, by taking advantages of saddle point reformulation, we developed a fast solver, HSS-pGMRES-IPM, for SVM training problem. However, as discussed in [29], multiple other approaches exist for solving saddle point problem. However, HSS presents higher efficiency than the conventional approaches [32-36, 39, 41, 42], and similar acceleration is reported in this paper.

Since the linear systems from IPM are involved by HSS-pGRMES, and the fast solver applies to SVM, two problems should be considered: one is condition number of kernel matrix Q, and the other is the round off error in the solution of IPM with HSS-pGMRES.

Although no further mathematical explanation or proof, [48] describes that, if ill conditioning of Q in infeasible IPM for LP, there is in a serious loss of accuracy when solving the Newton equations. Unfortunately, kernel matrix Q is a matrix coming from the dataset, measures to change it are limited. [43] provides suggestion to ameliorate the problem: to change the kernel function.

Scaling dataset provides no help to decrease condition number. Let's try $q(q > 0)$ scaled dataset $A = (x_{ij})$ and dot product $\langle x_i, x_j \rangle$ as kernel function as example. Firstly, let scale the dataset: $qA = (qA_{ij})$. Secondly, let compute dot product $\langle qx_i, qx_j \rangle = q^2 \langle x_i, x_j \rangle$. That is, before scaling the kernel matrix Q, after scaling the kernel matrix is $q^2 Q$. Finally, let's compute condition number.

$$\|Q\|_2 = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |y_{ij}|^p\right)^{\frac{1}{p}}.$$

Since the kernel matrix Q is non-negative, $|y_{ij}| = y_{ij}$. Then,

$$\|Q\|_p = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |y_{ij}|^p\right)^{\frac{1}{p}}.$$

By definition of norm,

$$\|q^2 Q\| \cdot \|(q^2 Q)^{-1}\| = \|q^2 Q\| \left\|\frac{Q^{-1}}{q^2}\right\|.$$

According to the definition of matrix norm [49],

$$\begin{aligned} &\|q^2 Q\| \cdot \left\|\frac{Q^{-1}}{q^2}\right\| \\ &= q^2 \|Q\| \cdot \frac{1}{q^2} \|Q^{-1}\| \\ &= \|Q\| \cdot \|Q^{-1}\|. \end{aligned}$$

That is, scaling does not really help to change condition number of the kernel matrix.

HSS-pGMRES-IPM (Algorithm 3) is a fast solver for SVM training. However, more theoretical study of HSS-pGMRES-IPM (Algorithm 3) for SVM training is needed to investigate the stability conditions for different datasets. Also, the linear systems in gene expression dataset are not huge, which is in size of *mn*, where *m* is a constant less than 10, and *n* the number of sample of dataset.

5. CONCLUSIONS

In this paper, we reformed SVM training into the saddle point equation, and we developed the fast solver, named HSS-pGMRES-IPM (Algorithm 3), for SVM training. Computational results show that the fast solver HSS-pGMRES-IPM (Algorithm 3) based SVM is significantly faster than the conventional factorization CF based SVM.

6. REFERENCES

[1] N. Cristianini, and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other Kernel-based Learning Methods*, Cambridge University Press, 2000.

[2] J. A. K. Suykens, T. Van Gestel, and J. De Brabanter, *Least Squares Support Vector Machines*, World Scientific, 2002.

[3] I. Steinwart, and A. Christmann, *Support Vector Machines*, Springer, 2008.

[4] S. Abe, *Support Vector Machines for Pattern Classification*, Springer, 2010.

[5] C. Campbell, and Y. Ying, *Learning with Support Vector Machines*, Morgan & Claypool Publishers, 2011.

[6] A. Statnikov, C. F. Aliferis, and D. P. Hardin, *A Gentle Introduction to Support Vector Machines in Biomedicine: Theory and Methods*, World Scientific, 2011.

[7] V. N. Vapnik, *Statistical Learning Theory*, Wiley, 1998.

[8] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 2000.

[9] S. J. Wright, *Primal-Dual Interior-Point Methods*, Society for Industrial and Applied Mathematics, 1997.

[10] Y. Ye, *Interior Point Algorithms: Theory and Analysis*, Wiley, 2011.

[11] J. Renegar, *A Mathematical View of Interior-Point Methods in Convex Optimization*, Society for Industrial and Applied Mathematics, 2001.

[12] C. Roos, T. Terlaky, and J. P. Vial, *Interior Point Methods for Linear Optimization*, Springer, 2006.

[13] T. Terlaky, *Interior Point Methods of Mathematical Programming*, Springer, 1996.

[14] Y. Zhang, User's guide to Lipsol linear-programming interior point solvers V0.4, *Optimization Methods and Software*, (11) 1-4 (1999), pp. 385-396.

[15] I. Lustig, R. Marsten, and D. Shanno, On implementing Mehrotra's predictor–corrector interior-point method for linear programming, *SIAM Journal on Optimization*, (2) 3 (1992), pp. 435-449.

[16] R. Tapia, Y. Zhang, M. Saltzman, and A. Weiser, The Mehrotra predictor-corrector interior-point method as a perturbed composite Newton method, *SIAM Journal on Optimization*, (6) 1 (1996), pp. 47-56.

[17] Y. Zhang, and D. Zhang, On polynomiality of the Mehrotra-type predictor-corrector interior-point algorithms, *Mathematical Programming*, (68) 1-3 (1995), pp. 303-318.

[18] T. Carpenter, I. Lusting, J. Mulvey, and D. Shanno, Higher-order predictor-corrector

interior point methods with application to quadratic objectives, *SIAM Journal on Optimization*, (3) 4 (1993), pp. 696-725.

[19] M. Salahi, J. Peng, and T. Terlaky, On Mehrotra-type predictor-corrector algorithms, *SIAM Journal on Optimization*, (18) 4 (2008), pp. 1377-1397.

[20] C. Cartis, Some disadvantages of a Mehrotra-type primal-dual corrector interior point algorithm for linear programming, *Applied Numerical Mathematics*, (59) 5 (2009), pp. 1110-1119.

[21] J. Hefferon, *Linear Algebra*, Department of Mathematics & Applied Mathematics, Virginia Commonwealth University, 2009.

[22] G. E. Shilov, *Linear Algebra*, Dover Publications, 2012.

[23] L. N. Trefethen, and D. Bau, *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, 1997.

[24] L. Smith, *Linear Algebra*, Springer, New York, 1998.

[25] S. Lang, *Linear Algebra*, Springer, 1987.

[26] G. Wu, Z. Zhang, and E. Chang, Kronecker factorization for speeding up kernel machines, *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2005.

[27] G. Wu, E. Chang, Y. K. Chen, and C. Hughes, Incremental approximate matrix factorization for speeding up support vector machines, *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, USA, 2006.

[28] D. Wanyu, Z. Kai, and Z. Qinghua, *Speed Up Kernel Projection Vector Machine Using Kronecker Decomposition*, *Proceedings of the 6th International Conference on New Trends in Information Science and Service Science and Data Mining (ISSDM)*, 23-25 October 2012, pp. 722-725.

[29] M. Benzi, G. H. Golub, J. Liesen, Numerical solution of saddle point problems, *Acta Numerica*, (14) (2005), pp. 1-137.

[30] H. C. Elman, Preconditioners for saddle point problems arising in computational fluid dynamics, *Applied Numerical Mathematics*, (43) 1-2 (2002), pp. 75-89.

[31] H. C. Elman, D. J. Silvester, and A. J. Wathen, Performance and analysis of saddle point preconditioners for the discrete steady-state Navier-Stokes equations, *Numer. Math.*, (90) 4 (2002), pp. 665-688.

[32] M. Benzi, and A. Wathen, *Some Preconditioning Techniques for Saddle Point Problems*, Springer, Berlin Heidelberg, 2008.

[33] Z.-Z. Bai, G. H. Golub, and M. K. Ng, Hermitian and Skew-Hermitian splitting methods for non-Hermitian positive definite linear systems, *SIAM J. Matrix Anal. Appl.*, (24) 3 (2002), pp. 603-626.

[34] M. Benzi, M. Gander, and G. Golub, Optimization of the Hermitian and skew-Hermitian splitting iteration for saddle-point problems, *BIT Numerical Mathematics*, (43) 5 (2003), pp. 881-900.

[35] M. Benzi, and G. Golub, A preconditioner for generalized saddle point problems, *SIAM Journal on Matrix Analysis and Applications*, (26) 1 (2004), pp. 20-41.

[36] Z.-Z. Bai, G. H. Golub, and J.-Y. Pan, Preconditioned Hermitian and Skew-Hermitian splitting methods for non-Hermitian positive semidefinite linear systems, *Numer. Math.*, (98) 1 (2004), pp. 1-32.

[37] D. Bertaccini, G. H. Golub, S. S. Capizzano, and C. T. Possio, Preconditioned HSS methods for the solution of non-Hermitian positive definite linear systems and applications to the discrete convection-diffusion equation, *Numer. Math.*, (99) 3 (2005), pp. 441-484.

[38] G. Golub, C. Greif, and J. Varah, An algebraic analysis of a block diagonal preconditioner for saddle point systems, *SIAM Journal on Matrix Analysis and Applications*, (27) 3 (2005), pp. 779-792.

[39] Z.-Z. Bai, G. H. Golub, L.-Z. Lu, and J.-F. Yin, Block triangular and Skew-Hermitian splitting methods for positive-definite linear systems, *SIAM J. Sci. Comput.*, (26) 3 (2005), pp. 844-863.

[40] M. Botchev, and G. Golub, A class of nonsymmetric preconditioners for saddle point problems, *SIAM Journal on Matrix Analysis and Applications*, (27) 4 (2006), pp. 1125-1149.

[41] Z.-Z. Bai, On semi-convergence of Hermitian and Skew-Hermitian splitting methods for singular linear systems, *Computing*, (89) 3-4 (2010), pp. 171-197.

[42] Z.-Z. Bai, G. H. Golub, and C.-K. Li, Optimal parameter in Hermitian and Skew-Hermitian splitting method for certain two-by-two block matrices, *SIAM J. Sci. Comput.*, (28) 2 (2006), pp. 583-603.

[43] S. Fine, and K. Scheinberg, Efficient SVM training using low-rank kernel representations, *J. Mach. Learn. Res.*, (2) (2002), pp. 243-264.

[44] K. Scheinberg, An efficient implementation of an active set method for SVMs, *J. Mach. Learn. Res.*, (7) (2006), pp. 2237-2257.

[45] S. Mehrotra, On the implementation of a primal-dual interior point method, *SIAM*

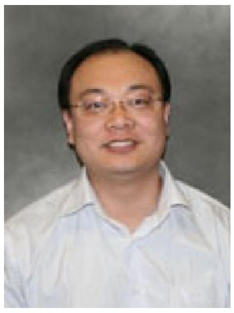
Journal on Optimization, (2) 4 (1992), pp. 575-601.

[46] A. Bhattacharjee, W. G. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, M. Gillette, M. Loda, G. Weber, E. J. Mark, E. S. Lander, W. Wong, B. E. Johnson, T. R. Golub, D. J. Sugarbaker, and M. Meyerson, Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses, *Proceedings of the National Academy of Sciences*, (98) 24 (2001), pp. 13790-13795.

[47] T. R. Kruth, *Interior-Point Algorithms for Quadratic Programming*, Technical University of Denmark, Lyngby, Denmark, 2008.

[48] E. Anderson, Jacek Gondzio, Csaba Meszaros, and Xiaojie Xu, Implementation of Interior Point Methods for Large Scale Linear Programming, *Technical report, Logilab, HEC Geneva, Section of Management Studies, University of Geneva*, Geneva, Switzerland, January 1996.

[49] G. H. Golub, and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 2012.



__Di Zhao__ is a postdoctoral researcher at The Ohio State University. He received his PhD of Computational Analysis and Modeling from Louisiana Tech University in 2010, and he worked in Columbia University as Postdoctoral Research Scientist from 2010 to 2012. His research interests are parallel algorithm and GPU computing