SVM Classification for Predicting Sparse Matrix Solvability
with Parameterized Matrix Preconditioners

Shuting Xu[1] [*] Jun Zhang[2] [†]

[1]Department of Computer Information Systems, Virginia State University,
Petersburg, VA 23806, USA

[2]Laboratory for High Performance Scientific Computing and Computer Simulation,
Department of Computer Science, University of Kentucky,
Lexington, KY 40506–0046, USA

February 14, 2006

Abstract

Large sparse linear systems are routinely solved using pre-conditioned Krylov subspace methods, in which the suitability and quality of the preconditioners play the vital role in determining the convergence rate of the iteration scheme. Selecting a suitable preconditioner for a specific sparse matrix arising from a particular application presents a challenging task for many application scientists and engineers who have little knowledge of preconditioned iterative methods. In this paper, we propose to use data mining techniques to predict the solving status of general sparse matrices with incomplete LU factorization type preconditioners with parameters. This work follows our previous work to use data mining techniques to predict the solvability of general sparse matrices with parameter-free matrix structure-based incomplete LU type preconditioners. Our proposed method chooses some points in the parameter space as samples. Studying the performance of a typical preconditioner, e.g., ILUT, with these sample parameters, we can get the main idea of what kinds of combination of the parameters are favorable for a given sparse matrix. If we can correctly predict the solving status at these sample points, we may obtain an outline of the parameter area(s) in which the sparse matrix can be solved. We use support vector machine (SVM) classification to predict the solving status of the sparse linear systems by ILUT with a specific set of parameters. We also use singular value decomposition (SVD) and sparsified SVD to preprocess the matrix features to improve the accuracy of prediction. We focus our work on ILUT preconditioner, but the proposed strategies should be applicable to other preconditioners with parameters.

Key words: sparse matrix, support vector machine, preconditioning

1 Introduction

Mathematical modeling problems can usually be formulated in the form of a system of linear or nonlinear partial differential equations. The first task of computer simulation of a mathematical modeling problem is to convert the continuous modeling problem into a discrete problem to be solved using a computer. The system of partial differential equations can be discretized by many numerical discretization methods which, in most cases, yields a system of linear algebraic equations. The coefficient matrices of such linear systems are usually large and sparse, i.e., many entries have the numerical value of zero.

The solution of large sparse linear systems (sparse matrices) is one of the most important problems in large scale scientific computing. For the past 50 years, many direct and iterative methods have been developed for this purpose [5, 20]. Among them, the preconditioned Krylov subspace methods [20] (with a Krylov iterative solver and a preconditioner) is considered the preferred methods. The preconditioners employed in the preconditioned iterative solvers usually determine the overall convergence rate of the iteration procedure [32]. However, selecting a suitable preconditioner for a specific sparse matrix arising from a particular application to achieve fast convergence is the combination of art and science, and presents a challenging problem for many application scientists and engineers who have little knowledge of the preconditioned iterative methods [14, 15, 16].

There is an enlarging gap between the development of more and more sophisticated preconditioned iterative solvers by the computational linear algebra community

and the ability to understand and to properly use these solvers by the application scientists and engineers to solve their more and more complex modeling and simulation problems. High performance computers and numerical algorithms will be less useful if they are not matched with the intended application problems. In the context of preconditioning, the use of a wrong preconditioner may render an iteration process diverge.

There has been a considerable amount of effort made by several researchers and organizations to collect various sparse matrices in order to use them for test purposes. The National Institute of Standard and Technology (NIST) has been playing a leading role in this endeavor and currently hosts one of the largest such repositories: MatrixMarket [18]. Several other collections have been contributed by engineers, scientists and numerical analysts, e.g., the well-known Harwell-Boeing sparse matrix collection and the University of Florida sparse matrix collections [2]. NIST has done some categorization work and published some preliminary information on these matrices. For each matrix this information includes its type, dimensions, condition number, nonzero structure, etc. MatrixMarket is becoming a standard source of sparse matrices for testing various direct and iterative solution methods. However, there is no information regarding which matrix can be solved by what method using what parameters. Such information would be extremely helpful for application scientists and engineers as it would enable them to choose suitable sparse matrix solvers for certain class of applications.

It is attractive to use machine learning techniques to help application scientists and engineers to choose suitable preconditioners for their particular application problems. Sparse matrices arising from different applications do have certain different features. These features may be represented by the sizes and the locations of their nonzero entries. If we can determine and extract these matrix features, and study and learn how the performance of the preconditioned Krylov subspace methods is related to these matrix features, we may be able to predict the performance of these preconditioned iterative methods to solve other sparse matrices that may have the same or similar features.

The idea of using matrix features and data mining techniques to predict the possibility of solving a sparse matrix by come preconditioners was first proposed in [33]. Over the years, we have made significant progress in extracting matrix features [25, 24], in using data mining techniques and matrix features to predict the condition numbers of sparse matrices [27, 26], and in using data mining techniques and matrix features to predict the solving status of sparse matrices by matrix structure-based incomplete LU type precon-

ditioners such as ILU(0) and ILU(k) [28, 30, 29].

Different approaches with similar aims in scientific computing have been proposed by other researchers. Houstis et al. successfully built PYTHIA II and other recommender systems for computational science applications [10, 11, 12]. Eijkhout et al. are exploiting the concept of automatic and self-adapting numerical linear algebra algorithms and software for, among other things, solving sparse linear systems [6, 7, 4, 31].

The process of extracting sparse matrix features and the identified 66 matrix features are described elsewhere [24, 25, 23]. In this paper we continue our previous work to predict the solving status of a sparse linear system with a certain preconditioned solver. The preconditioner we work on in this paper is ILUT [19], one of the popular preconditioners with many successful applications. Unlike ILU(0) or ILU(k), ILUT needs two preset parameters and it only works well under some sets of values of these parameters. Different sparse linear systems usually can be solved with ILUT with different parameters. Our aim in this paper is to predict with which parameter sets the sparse linear systems can be solved by ILUT.

This paper is organized as follows: Section 2 explains the problem encountered in predicting the solving status of a sparse linear system using ILUT and our method to resolve the problem. Singular value decomposition (SVD) and sparsified SVD are introduced in Section 3. The experiments are carried out and the results are reported in Section 4. The conclusion of this paper is in Section 5.

2 ILUT Parameter Space

ILUT is a kind of incomplete LU preconditioner with double dropping strategies. It is one of the most popular preconditioners with many successful applications [16, 14]. In this paper, we will still use the preconditioned GMRES (PGMRES) as our choice of preconditioned Krylov subspace method, i.e., we use the iteration solver GMRES with the ILUT preconditioner. PGMRES with ILUT can solve some sparse linear systems that would fail other preconditioners like ILU(0) (e.g., the matrix F2DB in the Harwell-Boeing collection). With ILUT, PGMRES may reduce the number of iterations to lower the computational time [20].

Corresponding to the two dropping strategies, there are two parameters used in ILUT. One is tolerance value tol, the other is the number of fill-in $lfil$. Since the $lfil$ parameter may be related to the size of the matrix in question, we normalize it by using the number of nonzero entries of the row in question. This parameter is called the fill-in rate $filr$, i.e., the ratio of the number of fill-in elements to the number of original nonzero entries

of each row. In this paper, we will use the fill-in rate $filr$ instead of $lfil$ in ILUT. Whether a sparse linear system can be solved by ILUT and the number of iterations the PGMRES will take are closely related to the values of these two parameters. Given a sparse linear system, we want to predict all the possible combination of the parameters with which the linear system can be solved. This is a quite hard problem, as both of the parameters may take real positive values, and the possible combination of the parameters may construct arbitrary areas in a two-dimensional parameter space. The left subfigure in Figure 1 illustrates the parameter space of ILUT for a particular sparse linear system. The closed grey area represents the parameter area within which the sparse linear system can be solved. As the area may be irregular and open and there may exist more than one such areas, it is very difficult to find some analytic functions to describe such area(s).

We propose to solve this problem by first choosing some points in the parameter space as samples. Studying the performance of ILUT with these sample parameters, we can get the main idea of what kinds of combination of the parameters are favorable for a given sparse linear system. In the right subfigure in Figure 1, the dots represent sample points. If we can correctly predict the solving status of a spars linear system at these sample points (red dot means that with the sample parameters the sparse linear system cannot be solved; black dot means it can be solved), we may obtain an outline of the parameter area(s) in which the sparse linear system can be solved. We use support vector machine (SVM) classification to predict the solving status of the sparse linear systems by ILUT with a specific set of parameters. We also use singular value decomposition (SVD) and sparsified SVD to preprocess the matrix features to improve the accuracy of prediction.

3 SVD and Sparsified SVD

3.1 Singular Value Decomposition
Singular Value Decomposition (SVD) [9] is a popular method in data mining and information retrieval [3]. It is usually used to reduce the dimensionality of the original dataset.

Let A be a feature matrix of dimension $n \times m$ representing the original dataset. The rows of the matrix correspond to data objects and the columns to attributes. The singular value decomposition of the matrix A is [9]

$$A = UHV^T,$$

where U is an $n \times n$ orthonormal matrix, $H = \text{diag}[\sigma_1, \sigma_2, \ldots, \sigma_s]$ ($s = \min\{m, n\}$) is an $n \times m$ diagonal matrix whose nonnegative diagonal entries are in a descending order, and V^T is an $m \times m$ orthonormal

matrix. The number of nonzero diagonals of H is equal to the rank of the matrix A.

Due to the arrangement of the singular values in the matrix H (in a descending order), the SVD transformation has the property that the maximal variation among the objects is captured in the first dimension, as $\sigma_1 \geq \sigma_i$ for $i \geq 2$. Similarly much of the remaining variations is captured in the second dimension, and so on. Thus, a transformed matrix with a much lower dimension can be constructed to represent the original matrix faithfully. Define

$$A_k = U_k H_k V_k^T,$$

where U_k contains the first k columns of U, H_k contains the first k nonzero diagonals of H, and V_k^T contains the first k rows of V^T. The rank of the matrix A_k is k. With k being usually small, the dimensionality of the dataset has been reduced dramatically from $\min\{m, n\}$ to k (assuming all attributes are linearly independent). It has been established that A_k is the best k dimensional approximation of A in the sense of Frobenius norm.

In data mining applications, the use of A_k to represent A has another important function. The removed part $E_k = A - A_k$ can be considered as the noise in the original dataset A [1]. Thus, in many cases, mining on the reduced dataset A_k may yield better results than mining on the original dataset A.

To avoid a potential confusion, we point out here that the matrix A is the feature space of all test sparse matrices with all the extracted features. It is not any of the sparse matrix in the linear systems to be solved.

3.2 Sparsified SVD
The SVD sparsification concept was proposed by Gao and Zhang in [8] for reducing the storage cost and enhancing the performance of SVD in text retrieval applications. Several sparsification strategies were proposed and experimented in [8]. The one that we used in this paper is the simplest one.

After reducing the rank of the SVD matrices, we set some small size entries, which are smaller than a certain threshold ϵ, in U_k and V_k^T, to zero. We refer to this operation as the dropping operation [8]. For example, given a threshold value ϵ, we drop u_{ij} in U_k if $|u_{ij}| < \epsilon$. Similarly, an element v_{ij} in V_k^T is also dropped if $|v_{ij}| < \epsilon$. Let $\overline{U}_k$ denote U_k with dropped elements and $\overline{V}_k^T$ denote V_k^T with dropped elements, we can represent the sparsified data matrix $\overline{A}_k$, with

$$\overline{A}_k = \overline{U}_k H_k \overline{V}_k^T.$$

The sparsified SVD method is equivalent to further removing noise from the dataset A_k. Denote $E_\epsilon =$
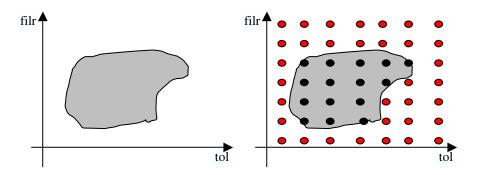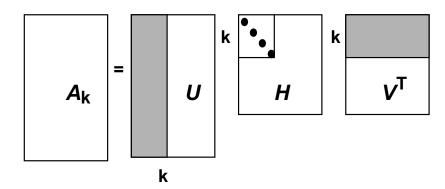
Figure 1: Parameter space of ILUT.

Figure 2: Singular value decomposition and reduced dimension.

$A_k - \overline{A}_k$, we have

$$A = \overline{A}_k + E_k + E_\epsilon.$$

4 Experiments and Results

We conduct some experiments to test the prediction accuracy of the solving status of the 319 sparse linear systems by PGMRES with preconditioner ILUT with different parameter sets. The sparse linear systems are constructed by using sparse matrices from MatrixMarket [18]. The parameter setting in PGMRES is the same as that used in our previous paper [29]. To be specific, the right hand sides of the linear systems are constructed by assuming that the solutions are a vector of all ones. The initial guessed solutions are a vector of all zeros. The maximum number of iterations is 500. The convergence stopping criterion is that the 2-norm of the residual vectors is reduced by 7 orders of magnitude. The iterative method used is GMRES(20) and the preconditioner is ILUT. We use SVM^{Light} [13] for SVM classification [21, 22] and Matlab [17] for SVD. The results are obtained by using a 5-fold cross validation. Detailed description about the SVM classification and its application to the matrix preconditioner prediction can be found in [23, 29].

4.1 Prediction with SVM Classification In this section we test the accuracy of predicting the solving status of sparse linear systems with the combination of parameters using SVM classification. For the parameter tol, we choose the most often used values 0.1, 0.01, 0.001, 0.0001 and 0.00001. For the parameter $filr$, the sample values we choose are 1, 2, 3, 4, and 5. The kernel we used in SVM classification is RBF, which is expressed by:

$$\text{RBF} : K(x, x_i) = e^{-\frac{||x - x_i||^2}{2\sigma^2}}.$$

Figure 3 shows the average prediction accuracy with different combination of the parameters. Here σ is set to be 0.1 in RBF kernel. We can see that the highest prediction accuracy 92.79% is obtained with $tol = 0.001$ and $filr = 1$. The lowest prediction accuracy 83.6991% is obtained with $tol = 0.0001$ and $filr = 3$. Generally speaking, when tol is high, e.g., $tol = 0.1$ and $filr$ is low, e.g. $filr = 1$, the prediction accuracy is better. With the increase of $filr$ and the decrease of tol, prediction accuracy becomes lower.

Figure 4 also shows the average prediction accuracy with different combination of parameters. The difference from Figure 3 is that σ is set to be 0.01 in RBF

tol					
0.1	0.921630	0.909091	0.902821	0.915361	0.915361
0.01	0.921630	0.909091	0.899687	0.902821	0.921630
0.001	0.927900	0.902821	0.868339	0.862069	0.858934
0.0001	0.912226	0.899687	0.836991	0.843260	0.884013
0.00001	0.924765	0.899687	0.843260	0.862069	0.865204
	1	2	3	4	5 filr

Figure 3: Prediction accuracy of SVM classification ($\sigma = 0.1$).

kernel. However, it does not depict the same prediction accuracy distribution as shown in Figure 3. Here the highest prediction accuracy area is on the upper-right corner of the figure, with tol from 0.1 to 0.01 and $filr$ from 4 to 5.

Figure 5 is obtained by setting σ to be 0.001 in RBF kernel. This figure shows another prediction accuracy pattern. This time the highest prediction accuracy is achieved with $tol = 0.01$ or $filr = 1$.

Figure 6 is obtained by setting σ to be 0.0001 in RBF kernel. Its highest prediction accuracy area is also on the upper-right corner of the figure, like in Figure 4 but much longer. To be specific, it is the area with tol from 0.1 to 0.01 and $filr$ from 2 to 5.

Table 1 describes the total prediction accuracy of SVM classification with different σ values. In calculating the total prediction accuracy we take into account all the combinations of parameters. The table shows that the value of σ does not affect the total prediction accuracy much. All the total average prediction accuracy are between 89% and 90%, which means that SVM classification works well for solving status prediction. In the latter experiments, we set $\sigma = 0.001$, as it provides a good prediction accuracy and takes less time to train than using $\sigma = 0.01$.

4.2 Applying SVD In many data mining applications, using SVD to preprocess data can improve the performance of data mining algorithms. In this section, we first apply SVD to the original data set of feature space and then use SVM classification to predict the solving status of the sparse linear systems to see if it can improve the prediction accuracy. To compare the results obtained with or without SVD, we set $\sigma = 0.001$ in RBF in this section. The size of the original feature matrix to apply SVD is 319×66.

Table 2 shows the prediction accuracy obtained by applying SVD with rank $k = 60$ in preprocessing. The highest prediction accuracy 92.163% is obtained with $tol = 0.01$ and $filr = 4$, which is similar to the highest

prediction accuracy in Figure 3. The lowest prediction accuracy 84.6395% is obtained with $tol = 0.0001$ and $filr = 3$, which is a little bit higher than the lowest prediction accuracy in Figure 3. The high prediction accuracy areas lie on the row with $tol = 0.01$ and the column with $filr = 1$.

Table 3 is obtained with using $k = 50$ in SVD. The high prediction accuracy pattern is exactly the same as the one showed in Table 2. Actually, most of the combinations of the tol and $filr$ parameters have same accuracy as in Table 2.

Table 4 is obtained with using $k = 40$ in SVD. Its high prediction accuracy pattern is quite different from the ones in Table 2 and Table 3. The high prediction accuracy area lies in the first two rows with $tol = 0.1$ and $tol = 0.01$.

Table 5 is obtained with using $k = 30$ in SVD. In this table the high prediction accuracy area lies in the column with $filr = 2$.

Table 6 is obtained with using $k = 20$ in SVD. In this table the high prediction accuracy area lies in the area with $tol = 0.01$ and $filr$ from 3 to 5.

To sum up all the patterns of high prediction accuracy areas appear in Table 2 - Table 6, we can see that high prediction accuracy is usually obtained with $tol = 0.1$ and $tol = 0.01$, or $filr = 1$ and $filr = 2$.

Table 7 shows the total prediction accuracy after applying SVD with different rank. The total prediction accuracy without SVD with the same σ value is 0.894044, which is the same as applying SVD with $k = 60$. When we choose $k = 50$, the total prediction accuracy after applying SVD is 0.894169, a little higher than that without using SVD. But from then on, with the decrease of rank k, the total prediction accuracy also drops. This table shows that applying SVD can improve the accuracy a little in this experiment. However, even the the rank k is set to be very small, e.g., less than one third of the number of attributes, the total prediction accuracy will not drop much.
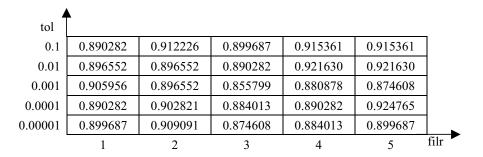
tol					
0.1	0.890282	0.912226	0.899687	0.915361	0.915361
0.01	0.896552	0.896552	0.890282	0.921630	0.921630
0.001	0.905956	0.896552	0.855799	0.880878	0.874608
0.0001	0.890282	0.902821	0.884013	0.890282	0.924765
0.00001	0.899687	0.909091	0.874608	0.884013	0.899687
	1	2	3	4	5 filr

Figure 4: Prediction accuracy of SVM classification ($\sigma = 0.01$).

tol					
0.1	0.905956	0.899687	0.890282	0.899687	0.899687
0.01	0.909091	0.909091	0.918495	0.921630	0.918495
0.001	0.905956	0.899687	0.846395	0.862069	0.868339
0.0001	0.905956	0.905956	0.874608	0.849530	0.902821
0.00001	0.905956	0.905956	0.880878	0.868339	0.896552
	1	2	3	4	5 filr

Figure 5: Prediction accuracy of SVM classification ($\sigma = 0.001$).

4.3 Applying Sparsified SVD In sparsified SVD (SSVD), we drop small entries in the U_k and V_k^T matrices. This method can save memory space, it can also improve the performance of data mining algorithms in some applications [8]. In this section, we conduct some experiments on preprocessing the original matrix features using sparsified SVD and then use SVM classification to predict the solving status of the sparse linear systems to see if it can improve the prediction accuracy. To compare the results obtained with or without SSVD and with SVD, we set $\sigma = 0.001$ in RBF and $k = 50$ in SVD.

Tables 8 - 10 list the prediction accuracy of using SSVD with a decreasing value for the dropping threshold ϵ. Table 8 shows the average prediction accuracy after applying SSVD with $\epsilon = 0.01$ using different combinations of tol and $filr$. The highest prediction accuracy 92.163% is obtained with $tol = 0.01$ and $filr = 4$, which is similar to the highest prediction accuracy in Figure 3 and the same as using SVD method. The lowest prediction accuracy 82.4451% is obtained with $tol = 0.001$ and $filr = 3$, which is a little lower than the lowest prediction accuracy obtained by without using SSVD and by using SVD. The high prediction accuracy area in this table lies in the area with $tol = 0.01$ and $filr$ from 3 to 5.

Table 9 shows the average prediction accuracy after applying SSVD with $\epsilon = 0.001$. Besides the high prediction accuracy area appears in Table 8, there is one more small area with $tol = 0.1$ and $filr$ from 4 to 5.

Table 10 shows the average prediction accuracy after applying SSVD with $\epsilon = 0.0001$. This time the high prediction accuracy area is exactly the same as the ones in Table 9.

We have also tested the prediction accuracy after applying SSVD with $\epsilon = 0.00001$. But the results are the same as the ones in Table 10 so we just omit it here. From the three tables in this section, we can see that the area with $tol = 0.01$ and $filr$ from 3 to 5 is the area that works best for SSVD. High prediction accuracy can be obtained by using parameters in this area with whichever ϵ values.

Table 11 shows the total prediction accuracy after applying SSVD with different dropping threshold ϵ. With the decrease of ϵ, the total accuracy drops as less elements in the matrix are set to be 0. Same accuracy is obtained with $\epsilon = 0.0001$ and $\epsilon = 0.00001$, as after a certain ϵ value, no more elements in the matrix will be dropped. The best accuracy obtained by SSVD in this example is 88.2132%, which is lower than using SVD or without using any preprocessing method.

5 Conclusion

We have proposed a method to predict the solving status of a sparse linear system using PGMRES with ILUT,
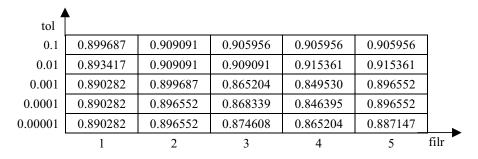
tol	1	2	3	4	5
0.1	0.899687	0.909091	0.905956	0.905956	0.905956
0.01	0.893417	0.909091	0.909091	0.915361	0.915361
0.001	0.890282	0.899687	0.865204	0.849530	0.896552
0.0001	0.890282	0.896552	0.868339	0.846395	0.896552
0.00001	0.890282	0.896552	0.874608	0.865204	0.887147

Figure 6: Prediction accuracy of SVM classification ($\sigma = 0.0001$).

σ	0.1	0.01	0.001	0.0001
Total Accuracy	0.892414	0.897304	0.894044	0.891285

Table 1: Total prediction accuracy of SVM classification with different σ.

which has two parameters. The problem is hard as it is difficult to predict all the possible areas in the parameter space that a given sparse linear system can be solved. We choose some sample points in the parameter space to predict the solving status of sparse linear systems at such sample points. Then we can have an idea of the outline of the area in the parameter space that the sparse linear system can be solved.

We used SVM classification in prediction and experimented SVD and sparsified SVD to preprocess the matrix features. The experimental results show that using SVM classification alone the total average accuracy of prediction on all the sample points are above 89%. The SVD method can improve the accuracy a little bit but the sparsified SVD method does not work well in this problem. We also analyzed in detail the area patterns in parameter space that can obtain high prediction accuracy in each prediction method.

Our study is one part of a broader effort to build a data mining techniques based intelligent preconditioner recommendation system for application scientists and engineers [23]. The intelligent recommendation system will work as follows: when a matrix is submitted through an interface, the preprocessing unit calculates the attributes of the matrix and passes on the attributes to the classification tool. The classification tool predicts which preconditioner would work best for the matrix as well as the suitable parameters. Then it sends the suggestions back to the interface. The matrix is saved in the database. It is solved using different sets of preconditioned solvers and the results are stored in the respective solver tables. The data mining tool periodically searches (mines) the database for rules and knowledge, and saves what it obtains in the knowledgebase. With more and more matrices stored in the database, the pre-

dictions (recommendations) given by the system will be more and more accurate.

References

[1] M. W. Berry, Z. Drmac, and E. R. Jessup. Matrix, vector space, and information retrieval. *SIAM Rev.*, 41:335–362, 1999.

[2] T. Davis. University of Florida sparse matrix collection, http://www.cise.ufl.edu/~davis/sparse. *NA Digest*, 97(23), June 7, 1997.

[3] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis, *J. Amer. Soc. Infor. Sci.*, 41:391–407, 1990.

[4] J. Demmel, J. Dongarra, V. Eijkhout, E. Fuentes, A. Petitet, R. Vuduc, R. C. Whaley, and K. Yelick. Self adapting linear algebra algorithms and software. *Proceedings of the IEEE*, 93:293–312, 2005.

[5] I. S. Duff, A. M. Erisman, J. K. Reid. Direct Methods for Sparse Matrices, Clarendon Press, New York, NY, 1986.

[6] V. Eijkhout and E. Fuentes. A proposed standard for matrix metadata. Technical Report ICL-UT-03-02, Innovative Computing Laboratory, University of Tennessee, Knoxville, TN, 2003.

[7] V. Eijkhout, E. Fuentes, T. Eidson, and J. Dongarra. The component structure of a self-adapting numerical software system. *Int. J. Parallel Programming*, 33(2), 2005.

[8] J. Gao and J. Zhang. Sparsification strategies in latent semantic indexing, in *Proceedings of the 2003 Text Mining Workshop*, M. W. Berry and W. M. Pottenger, (ed.), pp. 93–103, San Francisco, CA, May 3, 2003.

[9] G. H. Golub and C. F. van Loan. *Matrix Computation.* John Hopkins Univ. Press, Baltimore, 3rd Edition, 1996.

tol	$filr = 1$	$filr = 2$	$filr = 3$	$filr = 4$	$filr = 5$
0.1	0.905956	0.899687	0.890282	0.899687	0.899687
0.01	0.909091	0.909091	0.918495	0.92163	0.918495
0.001	0.905956	0.899687	0.846395	0.862069	0.868339
0.0001	0.905956	0.905956	0.874608	0.84953	0.902821
0.00001	0.905956	0.905956	0.880878	0.868339	0.896552

Table 2: Prediction accuracy after applying SVD with $k = 60$.

tol	$filr = 1$	$filr = 2$	$filr = 3$	$filr = 4$	$filr = 5$
0.1	0.905956	0.899687	0.887147	0.899687	0.899687
0.01	0.909091	0.912226	0.918495	0.92163	0.918495
0.001	0.905956	0.899687	0.846395	0.862069	0.868339
0.0001	0.905956	0.905956	0.877743	0.84953	0.902821
0.00001	0.905956	0.905956	0.880878	0.868339	0.896552

Table 3: Prediction accuracy after applying SVD with $k = 50$.

[10] E. N. Houstis, A. C. Catlin, J. R. Rice, V. S. Verykios, N. Ramakrishnan, and C. E. Houstis. PYTHIA-II: A knowledge/database system for managing performance data and recommending scientific software. *ACM Trans. Math. Soft.*, 26(2):227–253, 2000.

[11] E. N. Houstis, J. R. Rice, E. Gallopoulos, and R. Bramley. *Enabling Technologies for Computational Sciences: Frameworks, Middleware and Environments.* Kluwer Academic Publishers, Boston, MA, 2000.

[12] E. N. Houstis, J. R. Rice, and R. Vichnevetsky. *Intelligent Mathematical Software Systems*, Proceedings of the first IMACS/IFAC International Conference on Expert Systems for Numerical Computing, North-Holland, New York, 1990.

[13] T. Joachims. Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges and A. Smola (ed.), MIT-Press, 1999.

[14] N. Kang, J. Zhang, and E. S. Carlson. Parallel simulation of anisotropic diffusion with human brain DT-MRI data. *Computers and Structures*, 82(28):2389–2399, 2004.

[15] S. Karaa, J. Zhang, and C. C. Douglas. Preconditioned multigrid simulation of an axisymmetric laminar diffusion flame, *Math. Comput. Model.*, 38:269–279, 2003.

[16] J. Lee, J. Zhang, and C.-C. Lu. Performance of preconditioned Krylov iterative methods for solving hybrid integral equations in electromagnetics, *J. of Applied Comput. Electromagnetics Society*, 18:54–61, 2003.

[17] http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml.

[18] http://math.nist.gov/MatrixMarket.

[19] Y. Saad. ILUT: a dual threshold incomplete LU preconditioner. *Numer. Linear Algebra Appl.*, 1(4):387–402, 1994.

[20] Y. Saad. *Iterative Methods for Sparse Linear Systems*, PWS, New York, 1996.

[21] V. N. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, New York, 1998.

[22] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[23] S. Xu. *Study and Design of An Intelligent Preconditioner Recommendation System.* PhD thesis, University of Kentucky, Lexington, KY, 2005.

[24] S. Xu, E. Lee, and J. Zhang. An interim analysis report on preconditioners and matrices. Technical Report No. 388-03, Department of Computer Science, University of Kentucky, Lexington, KY, 2003.

[25] S. Xu, E. Lee, and J. Zhang. Designing and building an intelligent preconditioner recommendation system (a progress report). In *Abstracts of the 2003 International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Scientific and Industrial Applications*, Napa, CA, 2003.

[26] S. Xu and J. Zhang. Matrix condition number prediction with SVM regression and feature selection, in *Proceedings of the Fifth SIAM International Conference on Data Mining*, Newport Beach, CA, April, 2005.

[27] S. Xu and J. Zhang. A new data mining approach to predicting matrix condition numbers. *Commun. Inform. Systems*, 4(4):325–340, 2004.

[28] S. Xu and J. Zhang. A data mining approach to matrix preconditioning problem, in *Proceedings of the Eighth Workshop on Mining Scientific and Engineering Datasets (MSD05), in conjunction with the Fifth SIAM International Conference on Data Mining*, Newport Beach, CA, April, 2005.

[29] S. Xu and J. Zhang. A data mining approach to matrix preconditioning problem, in *Proceedings of the Eighth Workshop on Mining Scientific and Engineering Datasets (MSD'05)*, pp. 49–57, Newport Beach, CA,

tol	$filr = 1$	$filr = 2$	$filr = 3$	$filr = 4$	$filr = 5$
0.1	0.902821	0.905956	0.887147	0.902821	0.902821
0.01	0.902821	0.912226	0.915361	0.92163	0.918495
0.001	0.893417	0.896552	0.84326	0.865204	0.868339
0.0001	0.887147	0.902821	0.874608	0.84953	0.915361
0.00001	0.896552	0.902821	0.877743	0.874608	0.899687

Table 4: Prediction accuracy after applying SVD with $k = 40$.

tol	$filr = 1$	$filr = 2$	$filr = 3$	$filr = 4$	$filr = 5$
0.1	0.902821	0.902821	0.896552	0.899687	0.899687
0.01	0.896552	0.905956	0.802508	0.9279	0.915361
0.001	0.902821	0.899687	0.852665	0.880878	0.884013
0.0001	0.896552	0.905956	0.884013	0.805643	0.909091
0.00001	0.899687	0.902821	0.887147	0.874608	0.896552

Table 5: Prediction accuracy after applying SVD with $k = 30$.

April, 2005.

[30] S. Xu and J. Zhang. Solvability prediction of sparse matrices with matrix structure-based preconditioners, in *Abstracts of the 2005 International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Scientific and Industrial Applications*, Atlanta, Georgia, 3 pages, May 2005.

[31] Q. Yi, K. Kennedy, H. You, K. Seymour, and J. Dongarra. Automatic blocking of QR and LU factorization for locality. In *2nd ACM SIGPLAN Workshop on Memory System Performance (MSP 2004)*, Washington, DC, 2004.

[32] J. Zhang. Preconditioned Krylov subspace methods for solving nonsymmetric matrices from CFD applications. *Comput. Methods Appl. Mech. Engrg.*, 189(3):825–840, 2000.

[33] J. Zhang. Performance of ILU preconditioners for stationary 3D Navier-Stokes simulation and the matrix mining project. In *Proceedings of the 2001 International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Scientific and Industrial Applications*, p. 89–90, Tahoe City, CA, 2001.

tol	$filr = 1$	$filr = 2$	$filr = 3$	$filr = 4$	$filr = 5$
0.1	0.877743	0.887147	0.880878	0.887147	0.887147
0.01	0.862069	0.890282	0.909091	0.918495	0.915361
0.001	0.874608	0.893417	0.858934	0.852665	0.880878
0.0001	0.880878	0.902821	0.874608	0.855799	0.890282
0.00001	0.884013	0.899687	0.884013	0.868339	0.890282

Table 6: Prediction accuracy after applying SVD with $k = 20$.

K	60	50	40	30	20
tol	0.894044	0.894169	0.892790	0.889279	0.884263

Table 7: Total prediction accuracy after applying SVD with different rank.

tol	$filr = 1$	$filr = 2$	$filr = 3$	$filr = 4$	$filr = 5$
0.1	0.909091	0.896552	0.877743	0.896552	0.896552
0.01	0.896552	0.899687	0.902821	0.92163	0.912226
0.001	0.902821	0.887147	0.824451	0.84953	0.858934
0.0001	0.887147	0.890282	0.852665	0.846395	0.890282
0.00001	0.893417	0.890282	0.852665	0.846395	0.871473

Table 8: Prediction accuracy after applying SSVD with $\epsilon = 0.01$.

tol	$filr = 1$	$filr = 2$	$filr = 3$	$filr = 4$	$filr = 5$
0.1	0.899687	0.899687	0.887147	0.902821	0.902821
0.01	0.899687	0.899687	0.902821	0.915361	0.912226
0.001	0.902821	0.890282	0.84326	0.858934	0.865204
0.0001	0.887147	0.890282	0.858934	0.84953	0.899687
0.00001	0.890282	0.893417	0.868339	0.846395	0.887147

Table 9: Prediction accuracy after applying SSVD with $\epsilon = 0.001$.

tol	$filr = 1$	$filr = 2$	$filr = 3$	$filr = 4$	$filr = 5$
0.1	0.899687	0.899687	0.887147	0.902821	0.902821
0.01	0.899687	0.899687	0.902821	0.915361	0.912226
0.001	0.902821	0.890282	0.84326	0.862069	0.865204
0.0001	0.887147	0.890282	0.858934	0.84953	0.899687
0.00001	0.890282	0.893417	0.868339	0.84953	0.887147

Table 10: Prediction accuracy after applying SSVD with $\epsilon = 0.0001$.

ϵ	0.01	0.001	0.0001	0.00001
Total Accuracy	0.882132	0.886144	0.886395	0.886395

Table 11: Total prediction accuracy after applying SSVD with different dropping threshold.