A Data Mining Approach to Matrix Preconditioning Problem *

Shuting Xu, † Jun Zhang ‡
Laboratory for High Performance Scientific Computing and Computer Simulation, Department of Computer Science, University of Kentucky, Lexington, KY 40506-0046, USA
February 21, 2005

Abstract
Preconditioned Krylov subspace methods are generally used to solve large sparse linear systems with sparse coefficient matrices. However, choosing an efficient preconditioner for a specific sparse matrix arising from a particular application presents a formidable challenge for many design engineers and application scientists who have little knowledge of preconditioned iterative methods. We propose to build an Online Preconditioner Prediction System, which can predict the solvability of general sparse matrices with structure-based sparse matrix preconditioners using data mining techniques. We first group matrices into clusters, then use Support Vector Machine Classification technique to predict the solving status of the sparse matrices in clusters with low purity values. The prediction method can predict whether a matrix can be solved by a preconditioner (in a preconditioned iterative solver). In the case of a negative prediction, the method can also predict the possible reasons why the matrix cannot be solved by this preconditioner. Our experimental results show that the overall accuracy of the prediction is above 90% for the ILU0 preconditioner and above 87% for the ILUK preconditioners.

Key words: sparse matrix, support vector machine, preconditioning

1 Introduction
There are many scientific applications in which there is a need to solve large sparse linear systems with sparse coefficient matrices, e.g., the applications to develop the next generation electromagnetics simulators [12], to track nerve fibers in human brains [10], and to simulate laminar diffusion flames [11], to name just a few. The class of preconditioned Krylov subspace methods [19] (with a Krylov iterative solver and a preconditioner) is considered the preferred methods in this field. The preconditioners employed in the preconditioned iterative solvers usually determine the overall convergence rate of the iteration procedure [25]. However, choosing a good preconditioner for a specific sparse matrix arising from a particular application is the combination of art and science, and presents a formidable challenge for many design engineers and application scientists who have little knowledge of preconditioned iterative methods.

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 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.

Current approaches to recommending iterative solvers and preconditioners are quantitative and categorical. Although mathematical theorems can be proved to guarantee the convergence of certain preconditioned iterative methods for a given model problem, they are not very useful for solving problems encountered in practical applications. Thus a recommendation approach that permits tradeoffs and that can be built and modified incrementally, based on increased knowledge, is certainly useful. The success of the software environment approach for other problems in scientific computing [6] suggests that it can be very useful when a recommendation system or a software environment approach is applied to the problem of finding the right path to solving a sparse matrix.

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 [15]. 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 [4]. 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 which 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. Furthermore, solving some of the matrices in the MatrixMarket does not provide much information to help application scientists and engineers whose matrices in question may not match any of the matrices in the MatrixMarket exactly.

The first idea of using matrix features and data mining techniques to predict the possibility of solving a sparse matrix was proposed in [26]. We are now working on a project to build an online Intelligent Preconditioner Recommendation System (IPRS) [24, 23], which can provide expert advice on choosing a high performance preconditioner as well as the suitable parameters for a given sparse matrix. The first step of the project is to design and implement an Online Preconditioner Prediction System (OPPS), which is the main topic of this paper. That is, given a sparse matrix, predicting whether the matrix can be solved by a certain preconditioned solver (preconditioner). If not, the system will try to predict the reason why it fails.

The OPPS system works as follows. First, we extract features from matrices, then these features are used to classify the matrices into groups, with matrices in the same group having the same solving status. There are many classification algorithms in literature. We choose Support Vector Machine Classification (SVC) [20, 21] because of its success in many classification problems. We also propose a new prediction method. The matrices are first divided into clusters according to their similarity in the features. We define the purity of a cluster to be the maximum percentage of the sparse matrices in the cluster that have the same solving status. Then SVC is applied only to the clusters with low purity values. Our experimental results show that we not only improve the accuracy of the prediction by the clustered SVC method but also gain some insights into the relationship between matrix features and solvability of the matrix by the preconditioned solvers.

We introduce preconditioners in Section 2 of this paper. In Section 3, we describe the structure of the Online Preconditioner Prediction System. Some of the identified matrix features are explained in Section 4. We briefly review the clustering and classification algorithms used in the system in Section 5. The proposed prediction method is given in Section 6. The computational experiments are carried out and the results are discussed in Section 7. We sum up this paper in Section 8.

2 Preconditioners

The most promising general purpose iterative solution methods for solving large sparse matrices is the class of preconditioned Krylov methods [19]. Preconditioners in an explicit form consist of an explicitly constructed matrix $M$, which is close to $A$ in some sense. The key criteria for a good explicit preconditioner $M$ are that the construction cost of $M$ as well as the solution cost of an auxiliary linear system with $M$ should be inexpensive.

The Generalized Minimal Residual Method (GMRES) is a projection method based on the Krylov subspace. We apply the PGMRES [19] solver which uses the $L$ and $U$ matrices generated from the respective preconditioner $M = LU$ to precondition the original matrix before the application of the GMRES algorithm. The preconditioner is applied to the right hand side. The preconditioners that we choose to study in this paper are ILU0 and ILUK.

ILU0 is the incomplete LU (ILU) factorization technique with no fill-in, which takes the zero pattern $P$ to be precisely the zero pattern of the matrix $A$ [14, 19]. The definition of the ILU0 factorization is: “any pair of the matrices $L$ (unit lower triangular) and $U$ (upper triangular) so that the elements of $A - LU$ are zero in the locations of $NZ(A)$”, where $NZ(A)$ is the nonzero pattern of the matrix $A$. ILU0 is simple to implement, and its computational cost is inexpensive. It is effective for some problems, such as those from low-order discretizations of scalar elliptic PDEs and diagonally dominant matrices. However, for more difficult and realistic problems the no fill-in ILU0 factorization results in too crude an approximation to the original matrix, and more sophisticated preconditioners, which allow some fill-in in the incomplete LU factors, are needed [1].

ILUK is the ILU factorization with level of fill to be $k$. The initial level of fill of an element $a_{ij}$ of a sparse
matrix \( A \) is defined by:
\[
\text{lev}_{ij} = \begin{cases} 
0, & \text{if } a_{ij} \neq 0, \text{ or } i = j \\
\infty, & \text{otherwise}
\end{cases}
\]

Each time the value of \( a_{ij} \) is modified in the Gaussian Elimination procedure in the preconditioner construction, its level of fill is updated by:
\[
\text{lev}_{ij} = \min\{\text{lev}_{ij}, \text{lev}_{ik} + \text{lev}_{kj} + 1\}.
\]

In ILUK, all fill-in elements whose level of fill does not exceed \( k \) are kept. In this paper, we consider only \( k = 1, 2, 3 \) respectively. There are also some drawbacks of ILUK. For example, the amount of fill-in cannot be predicted in advance and the cost of updating the levels can be quite high. Furthermore, the algorithms may drop large size elements and result in an inaccurate incomplete factorization for certain indefinite matrices [18].

We choose ILU0 and ILUK because they are relatively simple, and they do not have many free parameters used in the construction of the preconditioners that can influence their performance. Thus the results of such preconditioned iterative solvers are easier to predict. We will deal with more sophisticated preconditioners in our later work.

3 Online Preconditioner Prediction System

![Feature Extraction Diagram](https://via.placeholder.com/150)

Figure 1: Online Preconditioner Prediction System.

The purpose of the Online Preconditioner Prediction System (OPPS) is to predict whether a sparse matrix can be solved by a preconditioned solver accurately and with a quick response time. The main structure of the OPPS is described in Figure 1. After a matrix is submitted, its features are computed and saved in a feature matrix. Some data mining algorithms are applied on the feature matrix to find out models, rules or other kinds of knowledge about the matrix features and their relationship with the preconditioned solvers. Such models, rules or knowledge are used to predict the solving status of the matrix, such as whether the sparse matrix can be successfully solved by a particular preconditioner, if not, which kind of problems will occur, etc. The procedures like mining the feature matrix are done in background and are repeated after some new matrices are submitted to the system. With more and more matrices added to the system, the prediction accuracy will improve. The response time of the system is just the time used to extract the matrix features and to predict the solving status based on the known knowledge.

4 Matrix Feature Extraction

The features of a matrix used are directly related to the precision of the prediction system. We will compute the features of a matrix first and then use such information to predict its solving status. The features of a matrix are a reflection of its sparsity and the locations and the size of the nonzero elements. We have extracted about 66 features such as the matrix structure, value, bandwidth and diagonal related statistics. There may be other useful features that we can extract in the future and add them to the feature space.

Figure 2 is an illustration of the sparsity pattern structures of two different sparse matrices from the MatrixMarket [15]. Figure 2(a) shows the sparsity pattern of the matrix NOS7, which has 729 columns and 2673 nonzero elements. The matrix NOS7 is from finite difference approximation to diffusion equation with varying diffusivity in a 3D unit cube with Dirichlet boundary conditions. Figure 2(b) depicts the sparsity pattern of the matrix NNC261, which has 261 columns and 1500 nonzero elements. The matrix NNC261 is from a nuclear reactor model. It can be seen that the two matrices have quite different sparsity patterns. Our aim is to extract features that can represent these and other properties of the sparse matrices and to use data mining techniques to predict the solvability of the sparse matrices by the preconditioned iterative solvers, based on their sparsity pattern and properties (features).

4.1 Structure This group of features describe the distribution of nonzero entries of a matrix. For example, we compute the sparsity rate (the number of nonzero elements divided by the number of all elements) of the whole matrix (nnzrt), of the lower diagonal part (lowfillrt), of the upper diagonal part (upfillrt) and of the main diagonal (diagfillrt). Other features include the average nonzero entries per row (avnnzrow) and the standard deviation (stdavnnzrow), the average nonzero...
entries per column (avnnzpcol) and the standard deviation (sdavnnzpcol), the maximum and minimum number of nonzero elements per column and per row (maxnnzpcol, minnnzpcol, maxnnzrow, minnnzrow). Sometimes we also want to know the total number of non-void diagonals (nzdiags), i.e., the number of diagonals which have at least one nonzero element among the 2n − 1 diagonals of the matrix.

The attribute symmc measures whether a matrix is symmetric, i.e., $A = A^T$. relsymm describes the relative symmetry rate of a matrix. It is the ratio of the number of elements that match divided by nnz. An element $a(i,j)$ in the matrix $A$ matches if it satisfies the following condition: if $a(i,j)$ is nonzero then $a(j,i)$ is nonzero. If a matrix is a normal matrix, normal is equal to 1, otherwise it is 0.

The attribute blocksizes reflects whether a matrix has a block structure or not. The matrix has a block structure if it consists of square blocks that are dense. The value of blocksizes greater than one represents the size of the largest block.

### 4.2 Value

The attributes in this group sum up the value distribution of a matrix, e.g., the one norm (onenorm), infinity norm (infnorm), and the Frobenius norm (frofnorm) of a matrix. This group also includes the minimum of the sum of the columns (minonenorm), the minimum of the sum of the rows (mininfnorm), the Frobenius norm of the symmetric part of a matrix (symfnorm), and of the unsymmetric part (nsymfnorm).

We also consider average value of all nonzero entries (avnnzval) and the standard deviation (sdavnnzval), the average of the main diagonal entries (avdiag) and the standard deviation (sdavdiag), the average of the upper triangular entries (avuptrig) and the standard deviation (sdavuptrig), as well as the average of the lower triangular entries (avlowtrig) and the standard deviation (sdavlowtrig).

### 4.3 Bandwidth

This group of features describe the bandwidth of a matrix. Bandwidth provides a measure of the clustering of nonzero entries about the main diagonal. Lower bandwidth of a matrix (lowband) is defined as the largest value of $i - j$, where $a(i,j)$ is nonzero. On the contrary, upper bandwidth of a matrix (upband) is defined as the largest value of $j - i$. Maximum bandwidth (maxband) is defined as $\max(\max(j) - \min(j))$. Average bandwidth (avband) is defined as the average width of all columns.

### 4.4 Diagonal

The features in this group are diagonal related. For instance, they include the average distance from each entry to the diagonal (adisfd) and the standard deviation (sdadisfd), the average of the difference from each of the entry to its diagonal value (avwalfd) and the standard deviation (sdawalfd), the average of the difference from the largest value in a row to the diagonal value (avmaxwalfd) and the standard deviation (sdavmaxwalfd).

Other features in this category include the percentage of weakly diagonally dominant columns (diagondomcol) and the percentage of weakly diagonally dominant rows (diagondomrow). diagvalrate is the ratio
of the minimum absolute diagonal element value (except zero) to the maximum absolute diagonal element value.

4.5 Others We also include some other features, such as `strspiv` - the number of structural zero pivots, (a structural zero pivot is a null column above or null row to the left of a zero diagonal element); `zpirow` - whether a matrix has a null row to the left of a zero diagonal element; `zpivec` - whether a matrix has a null column above a zero diagonal element; `szdiag` - the smallest nonzero diagonal element with the dot product of its left vector and up vector being zero; `minvalcol` - the minimum of the smallest nonzero value in each column with a zero diagonal element.

For more detailed description on the identified matrix features, please see [23].

5 Clustering and classification

In this section, we briefly review the clustering and classification algorithms we use in the system.

5.1 K-means clustering The K-means algorithm [13] (with its many variants) is a popular clustering method in data mining. It gains its popularity due to its simplicity and intuition. The algorithm is an iteration procedure and requires that the number of clusters, k, be given a priori. Suppose that the k initial cluster centers are given, the algorithm iterates as follows:

1. It computes the Euclidean distance from each of the objects to each cluster center. An object is assigned to the cluster with the smallest distance.

2. Each cluster center is recomputed to be the mean of its constituent objects.

3. Repeat steps (1) and (2) until the convergence is reached.

The criterion function for the convergence can be computed, e.g., as

\[ f_r = \frac{1}{n} \sum_{i=1}^{n} \text{Edist}^2(d_i, c_j^{(r)}) \],

where \( r \) is the step of the iterations. The function `Edist(d_i, c_j)` computes the Euclidean distance between the object \( d_i \) and a cluster center \( c_j \). Given a convergence criterion \( \epsilon \), the K-means algorithm stops when \( |f_{r+1} - f_r| < \epsilon \). Note that \( f_r \) is a monotonically decreasing function with a lower bound. So its limit exists [5].

5.2 SVM classification The SVM (Support Vector Machine) is based on a structural risk minimization theory [20]. It has been successfully applied to many applications like face identification, text categorization, bioinformatics, etc [2, 3, 9].

In SVM classification, the goal is to find a hyper-plane that separates the examples with maximum margin. Given \( l \) examples \((x_1, y_1), ..., (x_l, y_l)\), with \( x_i \in \mathbb{R}^n \) and \( y_i \in \{-1, 1\} \) for all \( i \), SVM classification can be stated as a quadratic programming problem:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{l} \xi_i \\
\text{subject to} & \quad y_i (\langle w, x_i \rangle + b) \leq 1 - \xi_i \\
& \quad \xi_i \geq 0 \\
& \quad C > 0
\end{align*}
\]

where \( C \) is a user-selected regularization parameter, and \( \xi_i \) is a slack variable accounting for errors. After solving it, we can get the following decision function:

\[
(5.1) \quad f(x) = \sum_{i=1}^{l} \alpha_i y_i < x_i, x > + b.
\]

where \( 0 \leq \alpha_i \leq C \).

For the nonlinear case, we apply a mapping \( \Phi : X \to F \) to map the input space into some feature space \( F \). Here we use a kernel function, \( K(x, x_i) = \langle \Phi(x), \Phi(x_i) \rangle \), which is a symmetric function and satisfies the Mercer’s condition. We substitute \( K(x, x_i) \) for the dot product, which maps the input space into some reproduced kernel feature space. Then Equation (5.1) can be rewritten as:

\[
(5.2) \quad f(x) = \sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + b.
\]

The above methods only work for classifying two classes. If the matrix features are to be classified into more than two classes, we have to use one of the multi-class classification methods [7, 16, 22]. Two of the commonly used multi-class methods are “one-against-one” and “one-against-all”. Suppose there are \( n \) classes, “one-against-all” method constructs \( n \) classifiers. Classifier \( i \) divides the data into the class belonging to class \( i \) and those not belonging to class \( i \). The “one-against-one” method constructs classifiers for each of the class pairs, thus totally constructing \( n(n-1)/2 \) classifiers. Hsu et al showed in [7] that “one-against-one” is more suitable for practical use. However, a recent publication [17] argued that “one-against-all” is as accurate as any other approaches. Since using “one-against-all” can save more training time, we adopt the “one-against-all” method in this paper.
6 Prediction Method
We now explain how our prediction method works. The prediction method actually consists of two parts, the training algorithm and the classification algorithm.

Algorithm 6.1. Training Algorithm.  
1. Compute the feature matrix \( M_f \);  
2. Apply K-means(\( k,M_f \));  
3. For clusters \( c_1, \ldots, c_k \)  
4. If (\( Purity_i < \omega \))  
5. \( \text{SVC}_{\text{train}}(c_i) \).

In the training algorithm (Algorithm 6.1), we first calculate the features of each sparse matrix and save them in a feature matrix \( M_f \). Then we use the K-means algorithm to cluster all the objects (matrices) into \( k \) clusters according to their features. For each of the clusters, if its purity value is greater than a threshold value \( \omega \), most matrices in this cluster are considered to have the same solving status. If the purity value of a cluster is smaller than \( \omega \), we use the SVM classification algorithm to build training models for the cluster.

Algorithm 6.2. Classification Algorithm.  
1. Given a matrix \( A \), compute its features;  
2. Find its nearest cluster \( c_i \);  
3. If (\( Purity_i > \omega \))  
4. \( SS_A = SS_{c_i} \)  
5. Else  
6. \( SS_A = \text{SVC}_{\text{classify}}(c_i,A) \).

Given a new matrix \( A \), if we want to predict its solving status, we will apply the classification algorithm (Algorithm 6.2). First, the matrix features are computed. Then, the distance of the feature vector to the center of each cluster is calculated. The matrix is assigned to the cluster with the shortest distance. If the purity value of the cluster is greater than \( \omega \), the solving status of the matrix is defined as the solving status of the majority of the matrices in the cluster. Otherwise, the SVC training models of that cluster are used to classify the matrix to its corresponding solving status group.

After a certain number of new matrices have been inserted into each cluster, the training algorithm will be run again to reflect the changes brought by the newly added matrices.

7 Experiments and Results
We conduct some experiments to test the prediction method. The linear systems are constructed by using sparse matrices from MatrixMarket [15]. 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 preconditioners are matrix structure-based incomplete LU factorizations ILU0 and ILUK. We choose \( K \) to be 1, 2, 3 respectively and denote them as ILUK1, ILUK2 and ILUK3. We first predict whether a matrix can be solved by these preconditioned solvers. If we predict that a matrix cannot be solved by a particular preconditioner, we will also predict the reason(s) of the failure. Table 1 shows the possible solving status (SS) of running these preconditioned solvers and their meanings. If the prediction method encounters any problem during its processing, it will stop and return a corresponding status to indicate the problem. We use \( SVM^{Light} \) [8] for SVM classification.

<table>
<thead>
<tr>
<th>SS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-100</td>
<td>zero pivot in constructing a preconditioner</td>
</tr>
<tr>
<td>-5</td>
<td>zero row in constructing a preconditioner</td>
</tr>
<tr>
<td>-8</td>
<td>large condest, unstable preconditioner</td>
</tr>
<tr>
<td>-1</td>
<td>solver cannot converge in 500 iterations</td>
</tr>
<tr>
<td>0</td>
<td>successfully solved</td>
</tr>
</tbody>
</table>

7.1 Solvability prediction Here we compare the prediction results obtained by using the proposed prediction method (CSCV) and by using SVM classification (SVC) alone. A total of 66 matrix features are extracted and used. In the K-means algorithm, we choose \( k \) to be 12. \( \omega \) is set to be 85%. Thus if more than 85% of the matrices in a cluster have the same SS, we will not do further classification on that cluster. In SVM classification, we apply the one-against-all method, constructing classifiers for each of the solving status and choosing the one with the highest value as a sparse matrix’s final predicted solving status. The results are obtained by using a 3-fold cross validation.

<table>
<thead>
<tr>
<th>SS</th>
<th>0</th>
<th>-100</th>
<th>-1</th>
<th>-8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>NM</td>
<td>143</td>
<td>132</td>
<td>32</td>
<td>12</td>
<td>319</td>
</tr>
<tr>
<td>( N_{M SVC} )</td>
<td>127</td>
<td>128</td>
<td>13</td>
<td>2</td>
<td>270</td>
</tr>
<tr>
<td>( C_{SVC} )</td>
<td>88.8%</td>
<td>97.0%</td>
<td>40.6%</td>
<td>16.7%</td>
<td>84.6%</td>
</tr>
<tr>
<td>( N_{M SVC} )</td>
<td>136</td>
<td>128</td>
<td>21</td>
<td>5</td>
<td>290</td>
</tr>
<tr>
<td>( C_{O SVC} )</td>
<td>95.1%</td>
<td>97.0%</td>
<td>65.6%</td>
<td>41.7%</td>
<td>90.9%</td>
</tr>
</tbody>
</table>

The predicted solving status related to ILU0 are
listed in Table 2. Here NM denotes the number of matrices, while CT denotes the correct rate (of successful prediction). Of all the 143 successfully solved matrices, we can correctly predict 136 of them by using CSVC, with the correct rate of 95.1%. On the other hand, if we use SVC, the correct rate is only 88.8%. If a matrix factorization encounters a zero pivot error, we can predict it with a correct rate of 97.0% with both SVC and CSVC. The accuracy is not high for predicting the cases of SS = -1 or SS = -8, the major reason for the poor predictions is that there are too few examples of such cases in the data sets. However, we can see the improvement of the prediction accuracy by using CSVC in stead of SVC. The correct rates are raised by 25% for both SS = -1 and SS = -8. Finally, the total correct rate for predicting ILU0 by SVC is 84.6%, while using CSVC it rises to 90.9%.

<table>
<thead>
<tr>
<th>SS</th>
<th>0</th>
<th>-5</th>
<th>-1</th>
<th>-8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>NM</td>
<td>197</td>
<td>74</td>
<td>37</td>
<td>11</td>
<td>319</td>
</tr>
<tr>
<td>CT</td>
<td>94.9%</td>
<td>82.4%</td>
<td>40.5%</td>
<td>54.6%</td>
<td>84.3%</td>
</tr>
<tr>
<td>NM</td>
<td>185</td>
<td>67</td>
<td>26</td>
<td>3</td>
<td>281</td>
</tr>
<tr>
<td>CT</td>
<td>93.9%</td>
<td>90.5%</td>
<td>70.3%</td>
<td>27.3%</td>
<td>88.1%</td>
</tr>
</tbody>
</table>

Table 3: Predicted solving status related to ILUK.

We compare the accuracy in predicting the solving status of ILUK1, ILUK2 and ILUK3 in Table 3. For ILUK1, C SVC excels SVC in predicting SS = -5 and SS = -1, but its correct rate of predicting SS = 0 is slightly lower than that of SVC. It performs worse than SVC in predicting SS = -8, too. The total correct rate of CSVC is 88.1%, which is higher than that of SVC (84.3%). Compared with the results of ILUK1, CSVC works better for ILUK2 and ILUK3. It has higher accuracy in predicting every solving status. For ILUK2, it increases the CT by around 17% for SS = -5 and SS = -1. The total correct rate is raised from 82.8% to 87.1%. For ILUK3, CSVC exceeds SVC for every solving status, too. It gets 87.8% in total correct rate while SVC only gets 82.1%.

In conclusion, the prediction accuracy is improved in almost all the categories with CSVC for ILUK. Using CSVC, the correct rate of predicting whether a sparse matrix can be successfully solved is above 92% for all levels of the ILUKs. And the total correct rate is above 87%.

7.2 Choice of ω In the previous subsection, we choose ω to be 85% because it works best for our system, after a large number of experiments. Figure 3 shows the change of total correct rates of using CSVC for ILU0 and ILUKs with the increase of ω. We can see that the trend of the lines representing ILU0 and ILUKs are very similar. When ω is not large enough, with the increase of ω, CT increases. It means when the cluster is not very pure, choosing the solving status of the majority of the matrices as the solving status for every matrix will incur a large error. In this case, it is better to construct classification models for each of the solving status group in the cluster and predict the solving status of each matrix using these models. However, after ω reaches some point (85% in our experiments), the prediction error of the classification models is larger than not using these models. We do not need to construct classification models in this situation and can save training time.

Figure 3: Relation of ω and the total correct rate.

Another point of Figure 3 is that even the lowest CT of ILU0 and ILUK in the figure is higher than the CT obtained by using SVC alone. It suggests that even if the ω value used is not the best one, it can also improve the prediction accuracy by using the CSVC method.

7.3 Cluster analysis Table 4 lists some statistics of the 12 clusters. P_{g \neq 0} denotes the purity of the cluster with respect to the solving status related to ILU0, and SS_{maj \neq 0} denotes the solving status of the majority of the matrices in the cluster. Other columns for ILUKs have the similar meanings. There are 6 clusters with
the purity value equal to 1 for both ILU0 and ILUKs. Matrices in Cluster 1 and Cluster 9 all have SS = −100 for ILU0 and SS = −5 for ILUKs. All matrices in Clusters 7, 10, 11, and 12 can be successfully solved. The last line of the table shows the average purity values for ILU0 and ILUKs. We can get:

\[
\text{average}(P_{ilu0}) > \text{average}(P_{iluk3}) > \\
\text{average}(P_{iluk2}) > \text{average}(P_{iluk1})
\]

If we compare that with the improvement of total correct rate obtained by using CSVC instead of SVC, we can get a similar sequence:

\[
\text{improvement}(CT_{ilu0}) > \text{improvement}(CT_{iluk3}) > \\
\text{improvement}(CT_{iluk2}) > \text{improvement}(CT_{iluk1})
\]

It seems that CSVC works better with higher average purity values. Our experiences in the experiments also show this trend. It suggests that if we use some clustering methods which can achieve higher purity value, we can gain higher total correct rate. We will do some experiments in the future to verify whether this hypothesis is true or not.

Table 5 describes the composition of the 6 purist clusters. It tells us which matrices have similar features. Matrices with the same name come from the same source. IMPCOL(3) means 3 matrices of IMPCOL type. If given a new matrix with the feature vector very close to one of these cluster centers, then we can predict that it will show similar property as the other matrices in the cluster when solved by some preconditioned solvers. We will investigate the features of these matrices and find out which features have large influence on deciding whether it can be solved by a preconditioner.

### 8 Concluding Remarks

The experimental results show that the prediction method gets promising results in predicting the solving status of sparse matrices by the matrix structure-based ILU type preconditioners. Using CSVC can improve the prediction accuracy over using SVC alone, the purest clusters generated can also give us some suggestions on which kind of matrices can get what results when solved by these preconditioned solvers.

This implementation and the reported experimental results are the first indication and evaluation of our study in using data mining techniques in large scale scientific computing applications. We will study and experiment the prediction method on more sophisticated preconditioners like ILUT and use feature selection method to reduce the computational cost in our future work.

### References


### Table 4: Cluster statistics.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>NM</th>
<th>$P_{\delta \mu 0}$</th>
<th>SS$_{\delta \mu 0}$</th>
<th>$P_{\delta \mu k1}$</th>
<th>SS$_{\delta \mu k1}$</th>
<th>$P_{\delta \mu k2}$</th>
<th>SS$_{\delta \mu k2}$</th>
<th>$P_{\delta \mu k3}$</th>
<th>SS$_{\delta \mu k3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>11</td>
<td>1.0</td>
<td>-100</td>
<td>1.0</td>
<td>-5</td>
<td>1.0</td>
<td>-5</td>
<td>1.0</td>
<td>-5</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>22</td>
<td>0.69</td>
<td>-100</td>
<td>0.48</td>
<td>0</td>
<td>0.50</td>
<td>0</td>
<td>0.96</td>
<td>0</td>
</tr>
<tr>
<td>Cluster 3</td>
<td>26</td>
<td>0.88</td>
<td>-100</td>
<td>0.62</td>
<td>0</td>
<td>0.69</td>
<td>0</td>
<td>0.73</td>
<td>0</td>
</tr>
<tr>
<td>Cluster 4</td>
<td>27</td>
<td>0.57</td>
<td>-100</td>
<td>0.26</td>
<td>-1</td>
<td>0.30</td>
<td>-1</td>
<td>0.37</td>
<td>-1</td>
</tr>
<tr>
<td>Cluster 5</td>
<td>16</td>
<td>0.69</td>
<td>0</td>
<td>0.75</td>
<td>0</td>
<td>0.81</td>
<td>0</td>
<td>0.81</td>
<td>0</td>
</tr>
<tr>
<td>Cluster 6</td>
<td>18</td>
<td>0.78</td>
<td>0</td>
<td>0.83</td>
<td>0</td>
<td>0.83</td>
<td>0</td>
<td>0.83</td>
<td>0</td>
</tr>
<tr>
<td>Cluster 7</td>
<td>5</td>
<td>1.0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
</tr>
<tr>
<td>Cluster 8</td>
<td>50</td>
<td>0.92</td>
<td>0</td>
<td>0.90</td>
<td>0</td>
<td>0.86</td>
<td>0</td>
<td>0.96</td>
<td>0</td>
</tr>
<tr>
<td>Cluster 9</td>
<td>31</td>
<td>1.0</td>
<td>-100</td>
<td>1.0</td>
<td>-5</td>
<td>1.0</td>
<td>-5</td>
<td>1.0</td>
<td>-5</td>
</tr>
<tr>
<td>Cluster 10</td>
<td>41</td>
<td>1.0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
</tr>
<tr>
<td>Cluster 11</td>
<td>7</td>
<td>1.0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
</tr>
<tr>
<td>Cluster 12</td>
<td>15</td>
<td>1.0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
</tr>
<tr>
<td>Average</td>
<td>26.6</td>
<td>0.86</td>
<td>-</td>
<td>0.82</td>
<td>-</td>
<td>0.83</td>
<td>-</td>
<td>0.85</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table 5: Matrix composition of the pure clusters.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Matrices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>IMPCOL(3), QH(2), RW(2), TOLS(2), WEST0132, ZENIOS</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>BCSSTM(5)</td>
</tr>
<tr>
<td>Cluster 3</td>
<td>BP(9), GEMAT(2), IMPCOL(2), PSMIGR(2), SHL(3), STR(4), WEST(10)</td>
</tr>
<tr>
<td>Cluster 4</td>
<td>FS(4), STEAM(2), BUS(4), BCSSTK(26), BCSSTM(3), GR30,30, LUND_B</td>
</tr>
<tr>
<td>Cluster 5</td>
<td>BCSSTM(7)</td>
</tr>
<tr>
<td>Cluster 6</td>
<td>FS,183, MCCA, MCCE, PORES, SAYLR1, STEAM3, BCSSTK(3), LUND_A, NOS(5)</td>
</tr>
</tbody>
</table>


