COMMUNICATIONS IN INFORMATION AND SYSTEMS
Vol. 4, No. 4, pp. 325-340, 2004

004

A NEW DATA MINING APPROACH TO PREDICTING MATRIX CONDITION NUMBERS*

SHUTING XU† AND JUN ZHANG‡

Abstract. Condition number of a matrix is an important measure in numerical analysis and linear algebra. The general approach to obtaining it is through direct computation or estimation. The time and memory cost of such approaches are very high, especially for large size matrices. We propose a totally different approach to estimating the condition number of a sparse matrix. That is, after computing the features of a matrix, we use support vector regression (SVR) to predict its condition number. We also use feature selection strategies to further reduce the response time and improve accuracy. We use a feature selection criterion which combines the weights from SVR and the weights from comparison of matrices with their preconditioned counterparts. Our experiments show that the response time of the prediction method is on average 15 times faster than the direct computation approaches, which makes it suitable for online condition number query. The accuracy of our prediction method is not as precise as the general direct computation methods. However, many people only care about whether a matrix is *well-conditioned* or *ill-conditioned* or the order of the condition number, not the exact value of the condition number. For such users, a rough prediction with quick response time probably is a better choice than a precise value after waiting for hours or days.

Key words: condition number, support vector machine, feature selection, preconditioning

1. Introduction.

The condition number $k(A)$ of a nonsingular matrix A with respect to a matrix norm is formally defined as $||A|| \cdot ||A^{-1}||$ [6]. The condition number corresponding to the Frobenius norm will be denoted by $k_F(A)$ and the condition number corresponding to the p-norm will be denoted by $k_p(A)$. There are some relationships among the condition numbers based on different norms. For example, if $A \in R^{n \times n}$, then

$$\frac{1}{n} k_2(A) \leq k_1(A) \leq n k_2(A),$$

$$\frac{1}{n} k_\infty(A) \leq k_2(A) \leq n k_\infty(A),$$

$$\frac{1}{n^2} k_1(A) \leq k_\infty(A) \leq n^2 k_1(A).$$

*(Eds.) Wai Lam, Rui-song Ye, Haiying Wang, and Jun Zhang. The research work of S. Xu was supported by NSF under grant ACR-0234270. The research work of J. Zhang was supported in part by NSF under grants CCR-0092532 and ACR-0202934, by DOE under grant DE-FG02-02ER45961, and by the University of Kentucky Faculty Research Support Program.

†Laboratory for High Performance Scientific Computing and Computer Simulation, Department of Computer Science, University of Kentucky, Lexington, KY 40506–0046, USA. E-mail: sxu2@uky.edu, URL: http://www.csr.uky.edu/∼sxu2.

‡Correspondent. Laboratory for High Performance Scientific Computing and Computer Simulation, Department of Computer Science, University of Kentucky, Lexington, KY 40506–0046, USA. E-mail: jzhang@cs.uky.edu, URL: http://www.cs.uky.edu/∼jzhang.

In this paper, we only stress on the condition number corresponding to the 1-norm $k_1(A)$. If $k(A)$ is relatively small, then the matrix A is called a *well-conditioned matrix*, but if $k(A)$ is large, then A is an *ill-conditioned matrix* (e.g., around 10^5 for a 5×5 Hilbert matrix).

Condition number is a widely used matrix feature in many areas, such as in numerical analysis and linear algebra. In numerical analysis, the condition number is basically a measure of stability or sensitivity of a matrix (or the linear system it represents) to numerical operations. For example, the condition number associated with the linear equation $Ax = b$ gives a bound on how inaccurate the solution will be after the numerical solution. Suppose A is nonsingular, $\hat{x}$ is an approximate solution to x, r is the residual, and $b \neq 0$, then:

$$\frac{1}{k(A)} \frac{||r||}{||b||} \leq \frac{||x - \hat{x}||}{||x||} \leq k(A) \frac{||r||}{||b||}.$$

It means that the relative error in the computed solution is bounded by the condition number of the matrix A times the relative size of the residual. If A is ill-conditioned, the relative error may not be small even if the relative size of the residual is small. $k(A)$ can also measure how close A is to being singular:

$$\frac{1}{k(A)} = \min \frac{||A - B||}{||A||}, \ B \text{ is singular.}$$

A can be approximated by a singular matrix B if and only if $k(A)$ is large.

Condition number can also be used to predict the convergence of iterative methods. For example, for the conjugate gradient (CG) method, the error can be bounded in terms of $k_2(M^{-1}A)$ [2], where M is a preconditioner. If A is symmetric positive definite, then for CG with a symmetric positive definite preconditioner M, it can be shown that:

$$||\hat{x}^{(i)} - x||_A \leq 2\alpha ||\hat{x}^{(0)} - x||_A,$$

where $\alpha = (\sqrt{k_2(M^{-1}A)} - 1)/(\sqrt{k_2(M^{-1}A)} + 1)$ [7, 13]. We will use condition number as one of the key features to predict the convergence of preconditioned iterative solvers in our Intelligent Preconditioner Recommendation System (IPRS) [25, 26]. This system provides recommendation on which preconditioned solver to choose for a given coefficient matrix.

There are several ways to obtain condition number. The direct method is to compute A^{-1} first and then multiply its norm with the norm of A. However, computing A^{-1} is equivalent to solving a linear system which is very time and memory consuming. Another method is using a less expensive algorithm to estimate the condition number. There are some estimation algorithms in literature [10]. For example, LAPACK uses subroutine SGECON to compute the condition number. Its time cost is $O(n^2)$ extra beyond the $O(n^3)$ cost of solving $Ax = b$, where n is the dimension of

A. If the size of the matrix is relatively large (e.g., $n > 20000$), the memory will be depleted before the computation is completed on our SunBlade 150 workstations. In most cases, the estimated condition number is within a factor of 10 of the true condition number, but there exist some counter-examples with large estimation errors. MATLAB uses LINPACK [5] for computing reciprocal of $k_1(A)$ and uses Higham's modification [10] of Hager's method to estimate $k_1(A)$. Such methods have similar problems with respect to memory and computing costs for large size matrices.

We propose a new approach to estimating the condition number of a *sparse* matrix. Instead of direct computation or estimation, we predict condition number from matrix features using data mining techniques. The predictor used is SVM regression (SVR) [9, 21, 23]. We also apply some feature selection methods [8, 17] to further reduce the time cost and improve precision. We propose a feature selection criterion which combines the weights from SVR with the weights from comparison of matrices with their preconditioned counterparts. Although the condition number predicted is not as precise as the above-mentioned direct computation methods, (in our experiments, if a relative deviation of 10^2 between the computed values and the predicated values of the condition number is acceptable, our prediction error is smaller than 25%), it has much smaller time and memory cost, especially for large size matrices, which is suitable for online condition number query. Furthermore, many users are only interested in knowing whether a matrix is *well-conditioned* or *ill-conditioned*, or the approximate order of the condition number. In these situations, response time and reliability are at least as important as accuracy.

The structure of the paper is as follows: We briefly review SVM regression in Section 2. In Section 3, we introduce the matrix features used to predict condition number. Three feature selection methods are described in Section 4. The computational experiments are carried out and the results are discussed in Section 5. We sum up this paper in Section 6.

2. SVM Regression. SVM regression is an approach to predicting real-valued outputs. It has been successfully applied in many areas such as financial forecasting [22, 27], image recognition [14, 23] and signal processing [23]. SVM regression using the ε-insensitive loss function is called ε-SV regression [24]. In ε-SV regression, the goal is to find a function $f(x)$ that has at most ε deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible [21]. We can visualize this as a tube of size 2ε around $f(x)$ and data fall out of the tube are errors [4].

Suppose a linear function $f(x)$ is of the form:

$$f(x) =< w, x > +b, \quad w \in X, \quad b \in \Re,$$

where $< w, x >$ denotes the dot product of the vectors w and x. To account for the errors, we introduce slack variables ξ_i and ξ_i^*. ξ_i computes the error for underesti-

mating the function while ξ_i^* computes the error for overestimating the function. The ε-insensitive loss function $|\xi|_\varepsilon$ is expressed as:

$$|\xi|_\varepsilon := \begin{cases} 0, & \text{if } |\xi| \le \varepsilon, \\ |\xi| - \varepsilon, & \text{otherwise.} \end{cases}$$

Then we have the following convex optimization problem:

$$\text{minimize } \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{l} (\xi_i + \xi_i^*),$$

$$\text{subject to } \begin{cases} y_i - < w, x_i > -b \le \varepsilon + \xi_i, \\ < w, x_i > +b - y_i \le \varepsilon + \xi_i^*, \\ \xi_i,\ \xi_i^* \ge 0, \\ C > 0, \end{cases}$$

where C determines the trade-off between the flatness of $f(x)$ and the amount up to which deviation larger than ε is tolerated.

The above optimization problem can be solved more easily in its dual formulation [21]. We can use a standard dualization method utilizing Lagrange multipliers. After solving it, we can get:

$$w = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) x_i, \tag{1}$$

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) < x_i, x > +b. \tag{2}$$

The variable b can be computed as:

$$b = \begin{cases} y_i - < w, x_i > -\varepsilon, & \text{for } \alpha_i \in (0, C), \\ y_i - < w, x_i > +\varepsilon, & \text{for } \alpha_i^* \in (0, C). \end{cases}$$

For the nonlinear case, we apply a mapping $\Phi : X \to F$ to map input space into some feature space F. Here we use a kernel function, $K(x, x_i) =< \Phi(x), \Phi(x_i) >$, 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. The commonly used kernel functions are:

$$\text{Polynomial} : K(x, x_i) = (< x, x_i > +c)^d,$$

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

$$\text{Neural Network} : K(x, x_i) = \tanh(\eta < x, x_i > +\vartheta).$$

3. Matrix Feature Extraction. The features of a matrix used are directly related with the precision of the prediction system. We will compute the features of a matrix first and then use such information to predict its condition number. We list below some matrix features such as structure, value, bandwidth and diagonal related statistics. Yet there may be more useful features that we can extract in the future and add them in the feature space.

3.1. Structure. This group of features describe the distribution of nonzero entries of a matrix. For example, we consider 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 ($avnnzprow$) and the standard deviation ($sdavnnzprow$), 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 ($maxnnzpcol, minnnzpcol, maxnnzprow, minnnzprow$). 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 symmetric rate of a matrix. It is the ratio of the number of elements that matches 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 $blocksize$ reflects whether a matrix has a block structure. The matrix has a block structure if it consists of square blocks that are dense. The value of $blocksize$ greater than one represents the size of the largest block.

3.2. Value. The attributes in this group sum up the value distribution of a matrix. For example, the one norm ($onenorm$), the infinity norm ($infnorm$), and the Frobenius norm ($frnorm$) of a matrix are computed. 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 ($sdavlowtrig$).

3.3. 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.

3.4. Diagonal. The features in this group are diagonal related. For instance, we include the average distance from each entry to the diagonal ($avdisfd$) and the standard deviation ($sdavdisfd$), the average of the difference from each of the entry to its diagonal value ($avvalfd$) and the standard deviation ($sdavvalfd$), the average of the difference from the largest value in a row to the diagonal value ($avmaxvalfd$) and the standard deviation ($sdavmaxvalfd$).

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

3.5. Others. We also include some features used to predict the solvability of preconditioned iterative linear system solvers, such as *strzpiv* - the number of structural zero pivots, that is, a null column above or null row to the left of a zero diagonal element; *zpivrow* - whether a matrix has a null row to the left of a zero diagonal element; *zpivcol* - whether a matrix has a null column above a zero diagonal element; *zpivdiag* - whether a matrix has a zero diagonal element with the dot product of its left vector and up vector is zero; *szvdiag* - the smallest nonzero diagonal element with the dot product of its left vector and up vector being zero; *minvalcol* - if a diagonal element has value 0, find the smallest nonzero value in that column, and *minvalcol* is the minimum of such values among all columns. For more detailed description on the matrix features, please see [26].

4. Feature Selection. Our experiments show that the accuracy of the condition number predicted based on the above features seems to be good. But such a collection of features may contain some redundant information. We apply feature selection methods to remove the redundancy. Feature selection may also bring other benefits: reduce the computation time, save memory space, remove noise and possibly optimize the prediction accuracy. For an online condition number prediction system, it is crucial to lower the response time and improve precision. In this section we investigate 3 feature selection methods.

4.1. Correlation. Correlation is one of the simplest feature selection methods. It computes the correlation of the input vector x_i and the target vector y as follows:

$$Cor_i = \frac{\sum_{k=1}^{m}(x_{k,i}-\bar{x}_i)(y_k-\bar{y})}{\sqrt{\sum_{k=1}^{m}(x_{k,i}-\bar{x}_i)^2 \sum_{k=1}^{m}(y_k-\bar{y})^2}},$$

where the bar stands for an average over the index k. In linear regression, Cor_i^2 represents the fraction of the total variance around the mean value $\bar{y}$ that is explained by the linear relation between x_i and y. Correlation criteria can only detect linear dependencies between variables and target [8].

4.2. Weights from SVR. There have been some feature selection methods based on the weights from the SVM classification model [3, 11, 16, 18]. Using the weights from SV-regression works in the same way.

Using the kernel function $K(x, x_i)$, Equations (1) and (2) can be written as:

$$w = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) K(x_i),$$

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) K(x_i, x) + b.$$

Like in neural networks, the output prediction is of the form:

$$predict(x) = G(\sum_j w_j x_j + b),$$

where $G(x)$ is an activation function. A feature j with a larger weight w_j has more effect on the prediction than a feature with a smaller weight [16, 18]. Shih *et al.* justified in [20] that the features with higher weights are more influential in determining the width of the margin. Thus $||w||^2$ is a suitable criterion for feature selection.

4.3. Combinational method. It is known that a good preconditioner can improve the condition number of a matrix [19]. The preconditioner M is a matrix which approximates A in some sense, such that the auxiliary linear system $Mx = b$ is inexpensive to solve. We compare the condition number as well as the matrix features of the original matrix and the preconditioned matrix to find out which features contribute more to the improvement of the condition number. Certain features have larger influence on the condition number and should be kept in feature selection. Assume we have l matrix examples, and k^A represents the condition number vector for all the original matrices, while k^M represents the condition number vector for all the preconditioned matrices. v_i^A is the vector of features for the ith original matrix, likewise, v_i^M is the vector of features for the ith preconditioned matrix. Then we can obtain the weight vector w_{cmp} to rank the features:

$$w_{cmp} = \sum_{i=1}^{l} \left(|k_i^A - k_i^M| * |v_i^A - v_i^M| \right). \tag{3}$$

w_{cmp} seems to be a reasonable criterion for feature selection. However, as we cannot successfully construct a useful preconditioner for all the general matrices, w_{cmp} is

biased towards the features of the matrices that can be preconditioned. To remedy this problem, we propose to use the weight w_{comb}, which combines w_{cmp} and the weight from SVM regression w_{SVM}, as

$$w_{comb} = \text{nml}(w_{cmp}) + \text{nml}(w_{SVM}) \tag{4}$$

where the function $\text{nml}(x)$ normalizes vector x. Thus w_{comb} is the sum of the normalized w_{cmp} and w_{SVM}.

5. Experiments and Results. In this section, we report our experiments on the accuracy and response time of the condition number prediction methods. We use SVM^{Light} [12] for SVM regression. There are 277 matrices from Matrix Market [15] tested in the experiments. We use altogether 60 matrix features, most of which are explained in Section 3. The experiments are carried out on a SunBlade 150 workstation.

5.1. Accuracy. First, we test how accurate the predicted condition numbers are, compared with the directly computed condition numbers. The accuracy is obtained using a 5-fold cross validation. We choose C to be 10000 for SVR which works best according to the results of the 5-fold cross validation. Other parameters for the kernels are chosen in the same way. The feature selection criteria used are correlation, w_{SVM}, w_{cmp}, and w_{comb}. We compare them together with the SVR without feature selection on three kernels: linear kernel, polynomial kernel and RBF kernel. Here all the feature selection methods choose 50% of the features.

Figure 1 shows the accuracy comparison using a RBF kernel ($\gamma = 0.1$). The figure illustrates the percentage of all matrices for which the relative differences between the computed values and the predicted values of the condition number are within 10, 10^2, 10^3, 10^4, respectively. For the RBF kernel, w_{cmp} does not work well. Its accuracy is the lowest. For all the other methods, we can safely say that more than 70% of the matrices have relative differences smaller than 10^2. Among them, feature selection with w_{comb} works best for all the difference scales except the first one. Using feature selection with w_{comb}, 76.2% of the matrices have relative differences smaller than 10^2. It has better accuracy than using SVR alone, although it only uses half of the features. Other feature selection methods can also obtain similar accuracy to that obtained from SVR without feature selection.

Figure 2 displays the accuracy comparison using a linear kernel. Here both feature selection criteria w_{comb} and w_{SVM} work well. Their accuracy are higher than using SVR alone for all the difference scales. For the first two difference scales, w_{SVM} is better than w_{comb}, while for the last two, w_{comb} exceeds w_{SVM}. Correlation criterion performs worst for the linear kernel.

The accuracy obtained using a polynomial kernel ($d = 2$) is depicted in Figure 3. In this figure, all the feature selection methods obtain much better accuracy than that

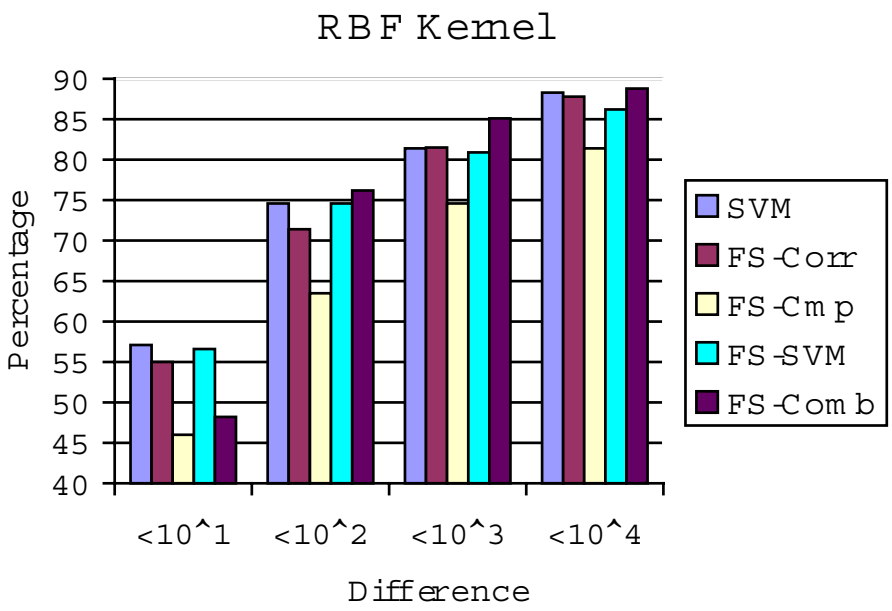

FIG. 1. *Comparison of accuracy with a RBF kernel.*

without feature selection. Here w_{SVM} and w_{comb} perform similarly, both are better than the other methods.

Put these 3 figures together, we can see that the best accuracy is obtained using the RBF kernel, then the linear kernel, and the polynomial kernel does not seem to fit for this job. For the RBF kernel SVR without feature selection works rather well, thus the advantage of the feature selection methods over it is moot. However, for the polynomial kernel, when SVR without feature selection performs poorly, using feature selection methods can remarkably improve accuracy. Among the feature selection methods, the performance of the criteria using w_{comb} or w_{SVM} are consistently good.

Next, we compare the performance of the feature selection methods with different number of features selected. We test the accuracy using 25%, 50%, 75% of the features respectively, and make comparison with using 100% of the features, that is, running SVM without any feature selection. Here we choose the percentage of matrices with relative condition number differences smaller than 10^2 as accuracy. Figure 4 illustrates the results obtained with a RBF kernel ($\gamma = 0.1$). Only feature selection with correlation has the property that with more features used the system becomes more accurate. For feature selection using w_{comb} and w_{SVM}, choosing 50% of the features seems to yield optimal results, with which they gain the highest accuracy. For feature selection using w_{cmp} it is an opposite story, choosing 50% of the features gives the worst results.

For linear kernel, all feature selection criteria except w_{cmp} seem to find their

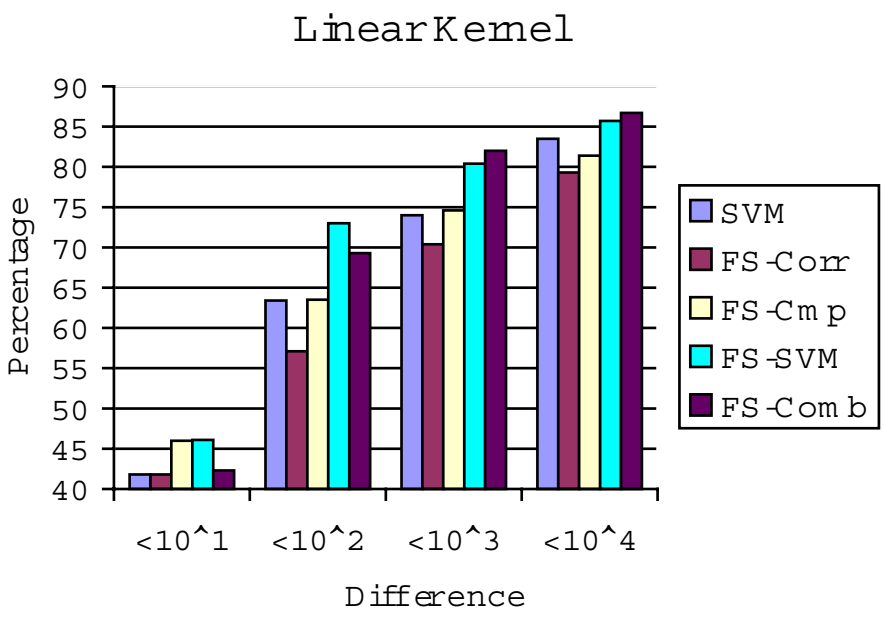

FIG. 2. *Comparison of accuracy with a linear kernel.*

optimized feature sets with 50% of the features. They get the best accuracy with 50% of the features. w_{cmp}, like using RBF kernel, performs the worst with 50% of the features (see Figure 5).

In Figure 6, nearly all the feature selection criteria get their best accuracy with 25% of the features. Even with the only exception w_{SVM}, the accuracy obtained with 25% of the features is very close to its best accuracy obtained with 50% of the features. Figure 6 explains why polynomial kernel ($d = 2$) does not work as well as RBF kernel and linear kernel in Figures 1 - 3. In these 3 figures, 50% of the features are used. Thus almost all the feature selection methods find their optimal feature sets for RBF kernel and linear kernel, but not for the polynomial kernel. The feature selection methods work well with 25% of the features for the polynomial kernel. Figure 6 also suggests that the polynomial kernel is worthwhile to try, as the fewer features used, the less the response time.

5.2. Response Time. Given a matrix, the time used to obtain the condition number is referred to as the response time. The response time for the LAPACK method is the time to compute the condition number using LAPACK routines. The response time for the prediction method includes the time to compute matrix features and the time for prediction. Here we also compare the response time for prediction using the whole matrix features and using half of the features selected based on w_{SVM}.

Table 1 shows the average response time for the 277 matrices used in our tests. The prediction methods are 15 times faster than using LAPACK on average. 6 seconds

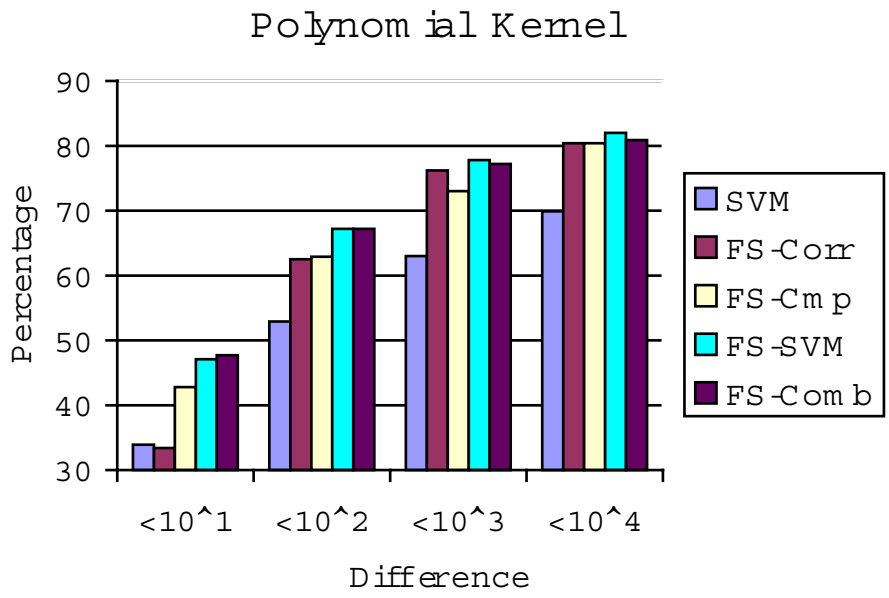

FIG. 3. *Comparison of accuracy with a polynomial kernel.*

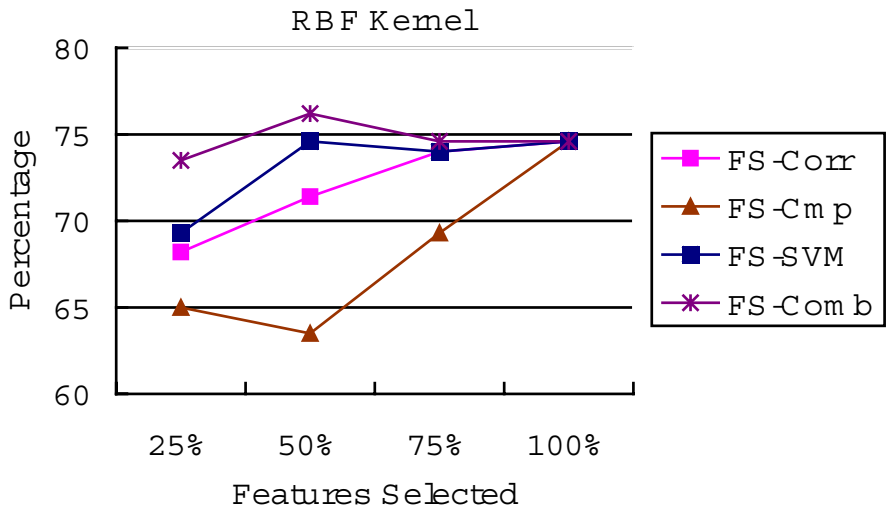

FIG. 4. *Comparison of accuracy with a RBF kernel with different percentage of features selected.*

is also an acceptable time for an online query system. Prediction with feature selection is only slightly faster than without any feature selection. Using half of the features does not mean reducing the time cost by half. In our system, we usually compute a group of features in a function. Thus the time cost for calculating one feature using the function is the same as calculating all the features provided by the function. We will consider code optimization to make feature selection more beneficial in response

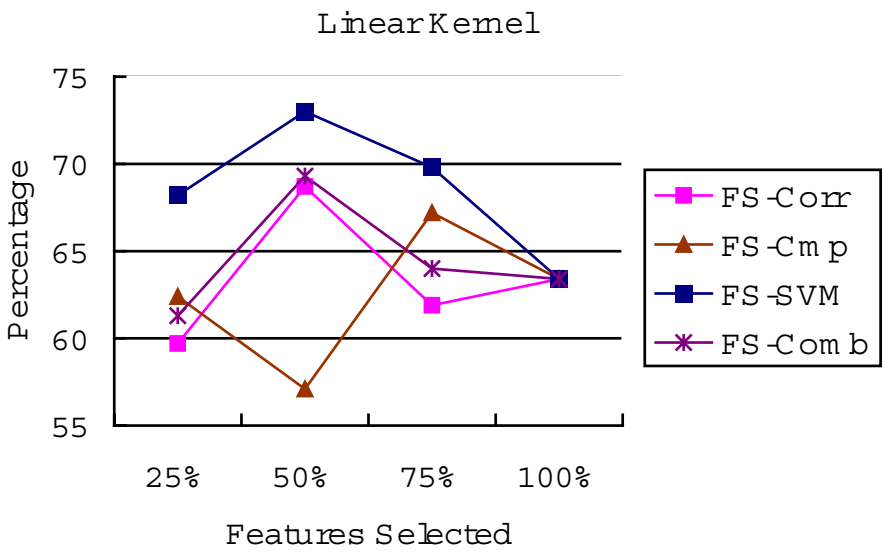

FIG. 5. *Comparison of accuracy with a linear kernel with different percentage of features selected.*

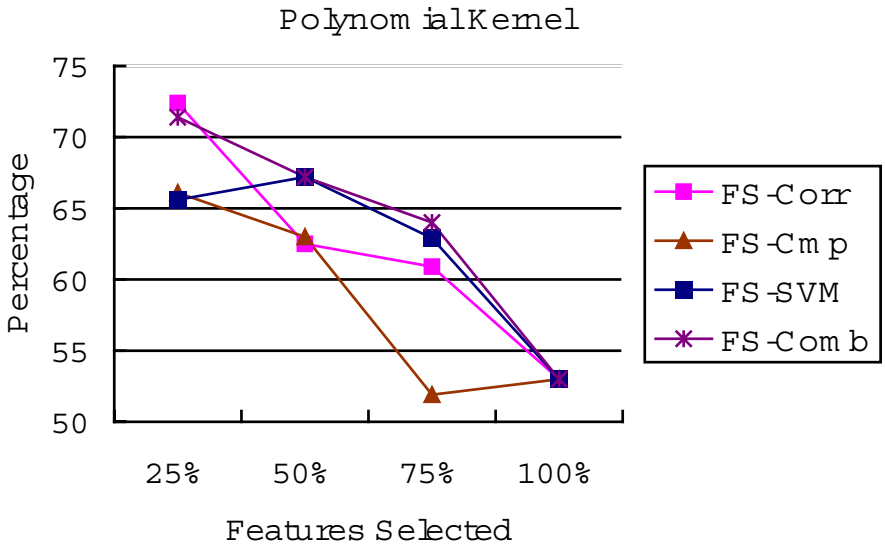

FIG. 6. *Comparison of accuracy with a polynomial kernel with different percentage of features selected.*

time, once we decide which selected features are to be computed.

The prediction method is especially advantageous in response time for large size matrices. For example, in Table 2 the average LAPACK response time for the 78 matrices with size larger than 2000 is around 6 minutes, while using the prediction methods, the response time is only about 20 seconds. A matrix with size greater than 1000 is large in our experiments though this may not be true in reality. As LAPACK

TABLE 1
Average response time (in seconds).

LAPACK	prediction(all)	prediction(FS)
99.23	6.56	6.32

TABLE 2
Average response time for larger size matrices (in seconds).

Size	NumMat	LAPACK	prediction(all)	prediction(FS)
≥ 1000	119	227.22	15.17	14.62
≥ 2000	78	340.74	22.81	21.99

will run out of memory on our computers with matrices of size larger than 20000, we can only test matrices under this size for comparison.

TABLE 3
Performance comparison for some large size matrices (in seconds).

Matrix name	Size	nnz	LAPACK	prediction(all)	prediction(FS)
ADD20	2395	13151	8206.7	0.94	0.81
CRY10000	10000	49699	2262.4	23.41	24.01
LNS_3937	3937	25407	2977.0	1.38	1.33
PSMIGR_1	3140	543160	2129.8	15.82	3.31

Table 3 gives some examples of how the prediction methods exceed the LAPACK method in response time. For instance, LAPACK uses about two and a half hours to compute the condition number of the matrix ADD20, the prediction methods only need less than one second. Although this may not be true for all the matrices, the significantly reduced response time is exactly our motivation for using prediction and feature selection.

6. Concluding Remarks. In this paper we propose a new approach to estimating the condition number of a matrix - predicting them from the matrix features. We use SVM regression with feature selection. The experiments show that around 75% of the matrices can be predicted with a relative difference from the computed condition number within 10^2. The accuracy is low compared with direct computation or estimation, but it is enough for those people who just want to know whether the matrix in question is *well-conditioned* or *ill-conditioned*. The advantage of the prediction method is that the response time is very low, especially for large size matrices. Thus it is desirable for an online condition number query. It is also fitted for our IPRS system. As we will use condition number as one of the matrix features to predict the solvability of a matrix, It is crucial to obtain it with a low time cost.

We also tried several feature selection methods. We proposed a combinational feature selection criterion which uses both the weights from SVR and from comparison of a matrix and its preconditioned counterpart. The experimental results show that using feature selection can reduce the time cost and improve or maintain the accuracy. The combinational feature selection criterion is one of the best methods tested.

Our future works include further improving the accuracy of the prediction method. We will also work out the error bound for the accuracy of the proposed method to increase the confidence of the user. We need to find more representative matrix features, as well as try other feature selection methods to better utilize these features. Then we will use the predicted condition number in our IPRS. Following the idea of this paper, we may also try to predict other important matrix attributes such as rank and eigenvalue distribution.

REFERENCES

[1] E. Anderson, Z. Bai, and C. Bischof, et al, *LAPACK Users' Guide.* SIAM, Philadelphia, PA, 3rd Edition, 1999.

[2] R. Barrett, M. Berry, and T. F. Chan, et al, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, PA, 2nd Edition, 1994.

[3] P. S. Bradley and O. L. Mangasarian, *Feature selection via concave minimization and support vector machines*, in: Proceedings of the Fifteenth International Conference on Machine Learning, 1998.

[4] K. P. Bennett and C. Campbell, *Support vector machines: Hype or Hallelujah?* SIGKDD Explorations, 2:2(2000), pp. 1–13.

[5] J. J. Dongarra, J. R. Bunch, and C. B. Moler, et al, *LINPACK Users' Guide.* SIAM, Philadelphia, PA, 1979.

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

[7] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, PA, 1997.

[8] I. Guyon and A. Elisseeff, *An introduction to variable and feature selection*, Journal of Machine Learning Research, 3(2003), pp. 1157–1182.

[9] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer-Verlag, New York, 2001.

[10] N. J. Higham, *Fortran codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation*, ACM Trans. Math. Soft., 14(1988), pp. 381–396.

[11] R. Jin and H. Liu, *Robust feature induction for support vector machines*, in: Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004.

[12] 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.

[13] S. Kaniel, *Estimates for some computational techniques in linear algebra*, Math. Comput., 20(1966), pp. 369–378.

[14] Y. Li, S. Gong, and H. Liddell, *Support vector regression and classification based multi-view face detection and recognition*, in: Proc. of the IEEE International Conference on Automatic Face and Gesture Recognition (FGR'00)., Grenoble, France, 2000.

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

[16] D. Mladenić, J. Brank, M. Grobelnik, and N. Milic-Frayling, *Feature selection using lin-*

ear classifier weights: interaction with classification models, in: Proceedings of SIGIR'04, Sheffield, UK, July, 2004.

[17] L. C. Molina, L. Belanche, and A. Nebot, *Feature selection algorithms: A survey and experimental evaluation*, in: Proceedings of 2002 IEEE International Conference on Data Mining (ICDM'02), Maebashi City, Japan, 2002.

[18] A. Rakotomamonjy, *Variable selection using SVM-based criteria*, Journal of Machine Learning Research, 3(2003), pp. 1357–1370.

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

[20] L. Shih, Y. Chang, and J. Rennie, et al, *Not too hot, not too cold: The Bundled-SVM is just right!* in: Workshop on Text Learning (TextML-2002), Sydney, Australia, 2002.

[21] A. J. Smola, B. Scholköpf, *A tutorial on support vector regression*, NeuroCOLT Technical Report Series, NC2-TR-1998-030, 1998.

[22] T. B. Trafalis and H. Ince, *Support vector machine for regression and applications to financial forecasting*, in: Proceedings of IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), Como, Italy, 2000.

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

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

[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, 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.

[27] H. Yang, L. Chan, and I. King, *Support vector machine regression for volatile stock market prediction*, in: Proceedings of the Third International Conference on Intelligent Data Engineering and Automated Learning, 2002.