

Matrix Condition Number Prediction with SVM Regression and Feature Selection

Shuting Xu, * Jun Zhang †

Department of Computer Science, University of Kentucky,
Lexington, KY 40506-0046, USA

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 design a feature selection criterion which combines the weights from SVR with the weights from comparison of matrices with their preconditioned counterparts. Our preliminary experiments show that the results are encouraging.

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 $\|A\| \cdot \|A^{-1}\|$. In this paper, we only stress on the condition number corresponding to 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*. Condition number is a widely used matrix feature in many areas, such as in numerical analysis and linear algebra. There are several ways to obtain the condition number of a matrix. The direct method is to compute A^{-1} first and then multiply its norm with the norm of A . However, computing $k(A)$

for even the simplest matrix norms is three times as expensive as solving $Ax = b$ in the first place. Another method is using a less expensive algorithm to estimate the condition number. There are some estimation algorithms in literature [2]. 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.

We propose a new approach to estimating the condition number of a *sparse* matrix - predicting condition number from matrix features using data mining techniques. The predictor used is SVM regression (SVR) [9, 10]. We also apply some feature selection methods [1, 7] to further reduce the time cost and improve precision. The method proposed is especially suitable for building an online condition number query system. The training of SVR can be done in background, thus the response time just includes the time to compute matrix features and predict the result through trained model, which is much faster than using the traditional methods mentioned above.

2 SVM Regression

SVM regression is an approach to predicting real-valued outputs. In ε -SV regression [10], 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 [9]. The problem can be transformed to the following convex optimization problem:

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

*The research work of S. Xu was supported by NSF under grant ACR-0234270. E-mail: sxu2@uky.edu, URL: <http://www.csr.uky.edu/~sxu2>.

†Correspondent. 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. E-mail: jzhang@cs.uky.edu, URL: <http://www.cs.uky.edu/~jzhang>.

$$\text{subject to } \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i, \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^*, \\ \xi_i, \xi_i^* \geq 0, \\ C > 0, \end{cases}$$

where C determines the trade-off between the flatness of $f(x)$ and the amount up to which deviation larger than ε is tolerated. ξ_i and ξ_i^* are slack variables accounting for errors. Kernel functions are applied to map input space into some feature space. The commonly used kernel functions are: Polynomial kernel: $K(x, x_i) = (\langle x, x_i \rangle + c)^d$, RBF kernel: $K(x, x_i) = e^{-\frac{\|x - x_i\|^2}{2\sigma^2}}$, and Neural Network kernel: $K(x, x_i) = \tanh(\eta \langle x, x_i \rangle + \vartheta)$.

3 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 condition number. We have extracted about 60 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 into the feature space. For more detailed description on the matrix features, please see [11].

4 Feature Selection

Our experiments show that the accuracy of the condition number predicted based on all the features seems to be good. But such features may contain some redundant information. We apply three feature selection methods to remove such redundancy. Feature selection may also bring other benefits [1, 3, 7]: reduce the computation time, save memory space, remove noise and possibly optimize the prediction accuracy. For an on-line condition number prediction system, it is crucial to lower the response time and improve precision.

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 . Correlation criteria can only detect linear dependencies between variables and target [1].

4.2 Weights from SVR There have been some feature selection methods based on the weights from the SVM classification model [3, 6]. Using the weights from SV-regression works in the same way. Like in neural

networks, the output prediction is of the form:

$$\text{predict}(x) = G\left(\sum_j w_j x_j + b\right),$$

where $G(x)$ is an activation function. A feature j with a larger weight w_j has more effect on the prediction than the feature with a smaller weight [6]. Shih *et al.* justified in [8] 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. 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. Such 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 i th original matrix, likewise, v_i^M is the vector of features for the i th preconditioned matrix. Then we can obtain the weight vector w_{cmp} to rank the features:

$$(4.1) \quad w_{cmp} = \sum_{i=1}^l (|k_i^A - k_i^M| * |v_i^A - v_i^M|).$$

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

$$(4.2) \quad w_{comb} = \text{nml}(w_{cmp}) + \text{nml}(w_{SVM}),$$

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}* [4] for SVM regression. There are 277 matrices from Matrix Market [5] tested in the experiments. We use altogether 60 matrix features. 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 5-fold cross validation. 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.

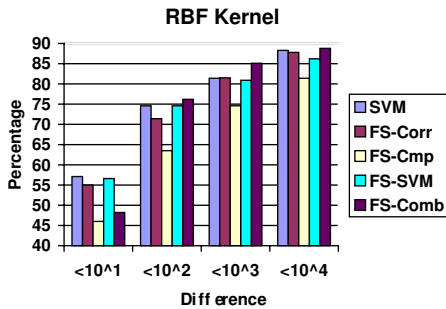


Figure 1: Comparison of accuracy with a RBF kernel.

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 10^1 , 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.

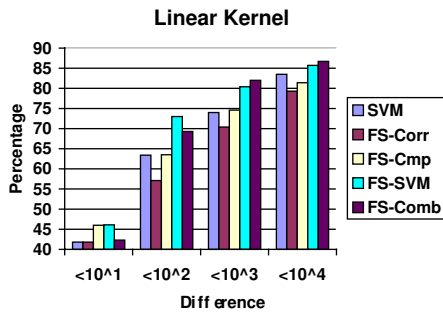


Figure 2: Comparison of accuracy with a linear kernel.

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.

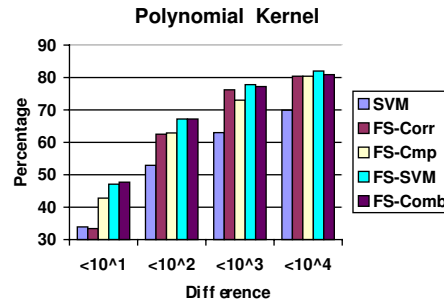


Figure 3: Comparison of accuracy with a polynomial kernel.

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 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.

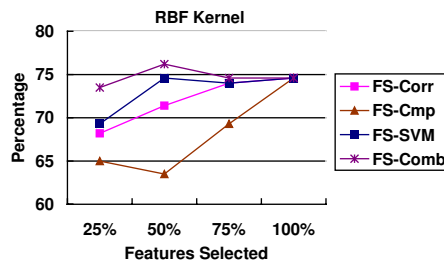


Figure 4: Comparison of accuracy with a RBF kernel with different percentage of features selected.

Next, we compare the performance of the feature selection methods with different amount of features selected. We test the accuracy using 25%, 50%, 75% of the features respectively, and make comparison with us-

ing 100% of the features, that is, running SVM without 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 be optimal, with which they gain the highest accuracy. For feature selection using w_{cmp} it is an opposite story, choosing 50% of the features yields the worst results.

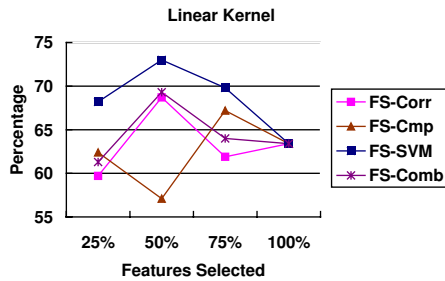


Figure 5: Comparison of accuracy with a linear kernel with different percentage of features selected.

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

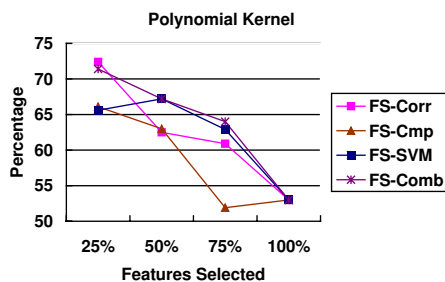


Figure 6: Comparison of accuracy with a polynomial kernel with different percentage of features selected.

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 optimized

feature sets for RBF kernel and linear kernel, but not for polynomial kernel. The feature selection methods work well with 25% of the features for polynomial kernel. Figure 6 also suggests that 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 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: Average response time (in seconds).

T_{LAPACK}	$T_{predict-all}$	$T_{predict-FS}$
99.23	6.56	6.32

Table 1 shows the average response time for the 277 matrices used in our tests. T_{LAPACK} and $T_{predict}$ denote the CPU time (in seconds) used by LAPACK and the prediction method respectively. The prediction methods are 15 times faster than using LAPACK on average. 6 seconds is also an acceptable time for an online query system. Prediction with feature selection is only slightly faster than without feature selection. Using half of the features does not mean reducing the time cost in 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 need to do code optimization to make feature selection more beneficial in response time.

The prediction method is especially advantageous in response time for large size matrices. For example, in Table 2 the average 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. $NumMat$ denotes the number of matrices. A matrix with size greater than 1000 is large in our experiments though this may not be true in reality. As LAPACK 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 gives some examples of how much the prediction methods may exceed the LAPACK method in computation time. $LgCN_{LAPACK}$ represents the 10-based logarithm of condition number computed by LAPACK while $LgCN_{predict}$ is the predicted 10-based

Table 2: Average response time for large size matrices (in seconds).

Size	NumMat	T_{LAPACK}	$T_{predict-all}$	$T_{predict-FS}$
≥ 1000	119	227.22	15.17	14.62
≥ 2000	78	340.74	22.81	21.99

Table 3: Performance comparison for some large size matrices.

Name	n	Non-zeros	T_{LAPACK}	$T_{predict-all}$	$LgCN_{LAPACK}$	$LgCN_{predict-all}$
GEMAT11	4929	33108	236.7	2.9	8.057436	8.1577207
LNS_3937	3937	25407	2977.0	1.4	14.417698	14.517881
PSMIGR_1	3140	543160	2129.8	15.8	9.858676	9.9595671

logarithm of condition number. For instance, LAPACK uses 2977 seconds to compute the condition number of the matrix LNS_3937, the prediction method only needs 1.4 seconds. The relative difference of condition numbers obtained by these two methods is only about $10^{0.1}$. Although this difference may not be expected for all matrices, it is exactly our motivation for using prediction to obtain the condition numbers for general sparse matrices.

6 Concluding Remarks

In this paper we proposed 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 may be sufficient for those people who just want to know whether the matrix 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 the intelligent preconditioner recommendation system (IPRS) system [11, 12]. In IPRS, the condition number is used as one of the matrix features to predict the solvability of a matrix, thus it is crucial to obtain it with a low time cost. We also tried several feature selection methods. We designed 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.

References

- [1] I. Guyon, A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(2003), 1157-1182.
- [2] 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.
- [3] R. Jin, H. Liu. Robust feature induction for support vector machines. In *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.
- [4] 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.
- [5] <http://math.nist.gov/MatrixMarket/>
- [6] D. Mladenović, J. Brank, M. Grobelnik, N. Milic-Frayling. Feature selection using linear classifier weights: interaction with classification models. In *Proceedings of SIGIR'04*, Sheffield, UK, July, 2004.
- [7] L. C. Molina, L. Belanche, 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.
- [8] L. Shih, Y. Chang, 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.
- [9] A. J. Smola, B. Schölkopf. A tutorial on support vector regression. *NeuroCOLT Technical Report Series*, NC2-TR-1998-030, 1998.
- [10] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.
- [11] S. Xu, E. Lee, 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.
- [12] S. Xu, E. Lee, 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.