Support Vector Machine Approximation using Kernel PCA

Narayanan Sundaram

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-94
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-94.html

June 18, 2009

Support Vector Machine Approximation using Kernel PCA

Narayanan Sundaram
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley, CA, USA
narayans@eecs.berkeley.edu

Abstract

Support Vector Machine is a very important technique used for classification and regression. Although very accurate, the speed of SVM classification decreases with increase in the number of support vectors. This paper describes one method of reducing the number of support vectors through the application of Kernel PCA. This method is different from other proposed methods as we show that the exact choice of the reduced support vectors is not important as long as the vectors span a fixed subspace. This method reduces the number of support vectors by upto 90% without any significant degradation in performance. We also propose a heuristic to determine the reducibility of an SVM.

1 Introduction

Support Vector Machine(SVM) is a very successful method used for classification and regression problems extensively today. SVMs are known to be quite accurate and generalize very well to include non-linear classification using the "Kernel trick". However, they are known to be slow and the time taken for classification increases linearly with the number of support vectors. Reduced set methods try to speed up the SVM classification by reducing the number of support vectors [1], [2], [3].

This paper proposes a technique to reduce the number of support vectors through Kernel Principal Component Analysis (KPCA). Unlike other techniques which use an unconstrained optimization problem (using SVM regression or reformulation [3] or conjugate gradient [2]) or doing clustering in the kernel space [1], our method randomly picks points in the space of principal components. We claim that reduced set methods are an instance of dimensionality reduction and could be solved using a dimensionality reduction technique. We use PCA in the kernel space for dimensionality reduction.

This paper is organized as follows. Section 2 describes the technique of Support Vector Machine classification. Section 3 explains the technique of principal component analysis in the Kernel space. Section 2 introduces the preimage problem in kernel methods. The proposed method is described in detail in Section 5 and the results are discussed in Section 6. We conclude in Section 7.

2 Support Vector Machines

Support Vector Machines are an implementation of Structural Risk Minimization, proposed by Vapnik [4]. For a two-class classification problem, where the classes are labeled as +1 and -1, the SVM problem (known as the C-SVM formluation) can be written as

$$Minimize \ \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{l}\xi_i \tag{1}$$

such that

$$y_i(\mathbf{w}\cdot\mathbf{x_i}+b) \geq 1-\xi_i \ , \ i=1,2\ldots l$$
$$\xi_i \geq 0 \ , \ i=1,2\ldots l$$

where $\mathbf{x_i}$ are the input vectors and y_i are the class labels.

The problem is more commonly solved as the dual formulation, which also generalizes to the non-linear

classification case.

$$Maximize \ \ -\frac{1}{2}\alpha^T Q\alpha + 1^T\alpha \tag{2}$$

such that

$$y^T\alpha = 0$$
$$0 \leq \alpha_i \leq C \ , \ i = 1, 2 \ldots l$$

where K is the kernel matrix $Q_{ij} = y_i y_j K(\mathbf{x_i}, \mathbf{x_j}) = y_i y_j \Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x_j})$ in some high dimensional space Φ.

The output of the classifier for an unknown $\mathbf{x}$ is given by

$$y = sign(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x_i}, \mathbf{x}) + b) \tag{3}$$

where N is the number of non-zero α's in the result of the previous optimization problem. The set $\{\mathbf{x_i} | \alpha_i \neq 0\}$is the set of support vectors.

The reduced set methods therefore try to approximate $\sum_{j=1}^{N} \alpha_i K(\mathbf{x_j}, \mathbf{x}) \approx \sum_{k=1}^{M} \beta_k K(\mathbf{z_k}, \mathbf{x})$ by choosing appropriate $\{\beta_k, \mathbf{z_k}\}$ $(M << N)$[1].

3 Kernel PCA

Principal Component Analysis in the kernel space can be solved by doing an eigen value decomposition on the centered Kernel matrix K^c.

$$K^c(i,j) = \widetilde{\Phi}(\mathbf{x_i}) \cdot \widetilde{\Phi}(\mathbf{x_j}) \tag{4}$$

$$\widetilde{\Phi}(\mathbf{x_i}) = \Phi(\mathbf{x_i}) - \frac{1}{N}\sum_{j=1}^{N} \Phi(\mathbf{x_j}) \tag{5}$$

The eigen values of K^c, $\lambda_k = N\sigma_k^2$ (where σ_k^2 is the varaince along the k^{th} principal direction in the kernel space) and the eigen vectors are unnormalized principal directions.

More details about kernel PCA can be found in [5].

4 Pre-image problem

The preimage problem in kernel methods is one of finding an approximate vector $\mathbf{z}$ in the input space given its image in the kernel space $\Phi(\mathbf{z})$. Generally $\Phi(\mathbf{z})$ is given as a sum of several known vectors in the kernel space $\Phi(\mathbf{z}) = \sum_{i=1}^{P} \gamma_i \Phi(\mathbf{x_i})$.

Pre-images for kernel points are not guaranteed to exist. Even if they exist, they need not be unique. However, several methods for finding approximate pre-images exist that give reasonably good results in practice. An iterative method for doing gradient descent to find the optimal preimage was given by [6]. [7] gives a one-step approximation of the method in [6] that is very good for our purposes. We use the method proposed by [7] in our experiments.

5 Proposed Method

Any reduced set method for reducing the number of support vectors must do an efficient approximation of the type

$$\sum_{i=1}^{N} \alpha_i \Phi(\mathbf{x_i}) \approx \sum_{k=1}^{M} \beta_k \Phi(\mathbf{z_k}) \tag{6}$$

This is a dimensionality reduction problem and hence should be treated as such. The exact choice of $\mathbf{z_k}$'s is not important as long as the set of all $\Phi(\mathbf{z_k})$'s constitute the space as all the $\Phi(\mathbf{x_i})$'s with minimal error. This is exactly what principal component analysis does. It finds the set of directions where the data has the most variance. Points should be chosen so that they span the directions as given by PCA in the kernel space. The exact choice of the points is not important.

The steps of the proposed method can be listed as follows:

1. Calculate the centered kernel matrix K_{xx}^c from the set of support vectors $\mathbf{x_i}$.

$$K_{xx}(i,j) = K(\mathbf{x_i}, \mathbf{x_j}) \tag{7}$$
$$K_{xx}^c = HK_{xx}H \tag{8}$$
$$\tag{9}$$

   where $H = I - \frac{1}{N}11^T$ is the centering matrix

2. Do Kernel PCA by doing an eigen value decomposition on K_{xx}^c.

$$K_{xx}^c = A\Lambda A^T \tag{10}$$

[1]The y_i in equation 3 can be combined into the α_i term and hence we use only α_i in the remainder of the paper

3. Discard eigen values smaller than a threshold. A value of 10^{-5} was used in our experiments. This was done to prevent numerical problems in the later stages of the algorithm.

4. Calculate the normalized principal directions.

$$\widetilde{V}_k = \frac{1}{\sqrt{\lambda_k}} \sum_{j=1}^{N} a_{jk} \widetilde{\Phi}(\mathbf{x_j}) \quad (11)$$

$$\text{where } \widetilde{\Phi}(\mathbf{x_j}) = \Phi(\mathbf{x_j}) - \frac{1}{N} \sum_{i=1}^{N} \Phi(\mathbf{x_i}) \quad (12)$$

In matrix form, this becomes,

$$\widetilde{V} = HA\Lambda^{-\frac{1}{2}} \quad (13)$$

5. Calculate new support vectors by choosing the projections on the principal directions from a uniform distribution $U[-\sigma_k, +\sigma_k]$ where $\sigma_k = \sqrt{\frac{\lambda_k}{N}}$ and adding it to the center of the distribution. In matrix form,

$$V = \widetilde{V}R + \frac{1}{N} \quad (14)$$

$$R = \frac{1}{\sqrt{N}} \Lambda^{\frac{1}{2}} U \quad (15)$$

where U is a matrix of points chosen from the uniform distribution $U[-1, +1]$.

6. Each column of V corresponds to a new support vector. Choose any M columns. (This step can be combined with the previous step by choosing a smaller R).

$$\Phi(\mathbf{z_k}) = \sum_{i=1}^{N} V_{ik} \Phi(\mathbf{x_i}) \quad (16)$$

7. Calculate the approximate pre-images of the points obtained in the previous step according to [7].

$$\mathbf{z_k} = \frac{\sum_{i=1}^{N} V_{ik} (\frac{1}{2}(1 - V_k^T K_{xx} V_k + 2V_k^T k_{\mathbf{x_i}})) \mathbf{x_i}}{\sum_{i=1}^{N} V_{ik} (\frac{1}{2}(1 - V_k^T K_{xx} V_k + 2V_k^T k_{\mathbf{x_i}}))} \quad (17)$$

$$k_{\mathbf{x_i}} = [K(\mathbf{x_i}, \mathbf{x_1}) \; K(\mathbf{x_i}, \mathbf{x_2}) \; \ldots \; K(\mathbf{x_i}, \mathbf{x_N})]^T \quad (18)$$

8. Calculate the new coefficients β by solving

$$K_{zz}\beta = K_{zx}\alpha \quad (19)$$

This ensures that both SVMs produce same results for all the $\mathbf{z_k}$'s [8]. We could instead solve the following system of equations (in a least squares sense) to get the β_k's.

$$K_{xz}\beta = K_{xx}\alpha \quad (20)$$

This seems to have marginal improvement in performance but was not considered further.

6 Results

We show the results of our method on two datasets - the adult dataset and the web dataset. The versions considered have been used in [9]. The adult dataset is a part of UCI Machine Learning Repository [10]. Details of the datasets are given in table 1.

Dataset	# Training examples	# Support vectors	# Dimensions	# Test Vectors
adult	1605	653	123	30956
web	4912	663	300	44837

Table 1: Details of the datasets used

The SVMs were trained using LibSVM [11]. RBF kernel $K(\mathbf{x}, \mathbf{y}) = e^{\gamma ||\mathbf{x}-\mathbf{y}||^2}$ was used in our experiments. The value of γ and C were determined using 10-fold crossvalidation.

6.1 Data distribution in the kernel space

Figure 1 shows the distribution of eigen values for the adult and web datasets. The graph represents the variance ignored by neglecting the principal directions that have smaller variance than the direction considered i.e. $y_i = \sum_{j=i}^{N} \lambda_j$, assuming the eigen values are sorted in descending order.

From the figure, it is clear that very few directions contribute to most of the variance in the data distribution. In the case of adult dataset, ignoring 90% of the principal components removes just 10% of the total variance.

It is clear that dimensionality reduction would work as long as the area under the curve in Figure 1 is less than 0.5 (the area under a straight line from (1,0) to (0,1)).

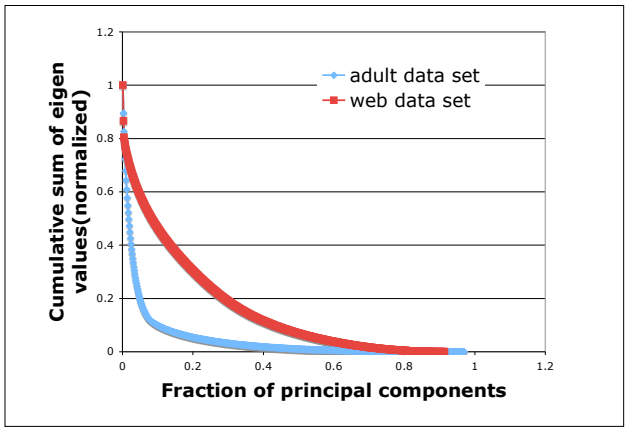

Figure 1: Eigen value distribution in Kernel space for adult and web datasets

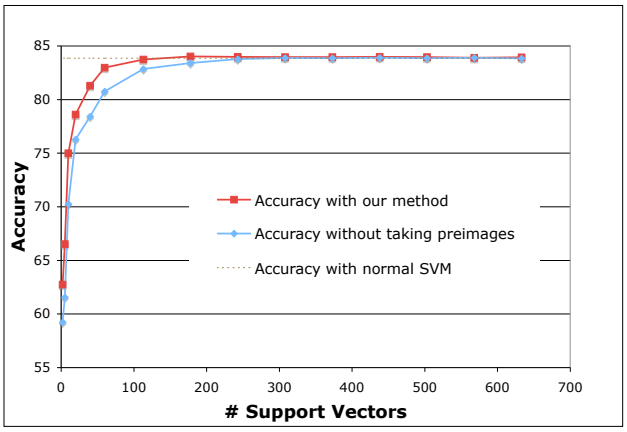

Figure 2: Effect of the pre-image approximation

6.2 Effect of the preimage approximation

Figure 2 shows the effect of the pre-image approximation on the accuracy of the final SVM for the adult dataset. Equation 19 is applied to points obtained in step 6 of the algorithm in Section 5 to get the optimal β for the points in the kernel space. From the figure, it is clear that although the preimage computation is approximate, it does not lead to any loss in performance. In fact, for this dataset, the performance after computing the preimages is better than the actual points in Kernel space. This might be because points in the kernel space do not correspond to any "real" data, whereas the preimages are points in the input space and hence the coefficients β are more accurate. The accuracy results are the average of 10 instances (to remove the effect of the random selection of points in our algorithm).

6.3 Reducibility of an SVM

Reduced set methods that do approximations of SVMs like [8],[3],[1] do not give us any insight into how reducible a particular instance of SVM is. We have no idea if a reduced set method would work, and if it works, what the performance-speed tradeoff curve is. We propose a heuristic for determining how reducible an SVM instance is. The area under the curve in Figure 1 is a measure of the reducibility of an SVM.

$$\text{Area under the curve} = \frac{\sum_{i=1}^{N} i\lambda_i}{N\sum_{i=1}^{N}\lambda_i} \qquad (21)$$

where the eigen values are sorted in descending order. A smaller area implies better reducibility.

Using this measure, the adult dataset has a reducibility of 0.0458 and the web dataset has a reducibility of 0.1558. The adult dataset is clearly more reducible than the web dataset.

6.4 Performance trade-off

Figure 3 shows the reduction in accuracy of the reduced SVM obtained using our method versus the number of support vectors used. The adult dataset clearly shows a trend that echos the eigen value distribution in Frigure 1. The web dataset does not show any significant difference from the original SVM, even when the number of support vectors used is very small. This can be attributed to the nature of the test dataset that is particularly biased towards one class and so does not indicate the errors due to the approximations clearly.

To see the error trends more clearly, we plot the normalized sum of squared errors for the test data

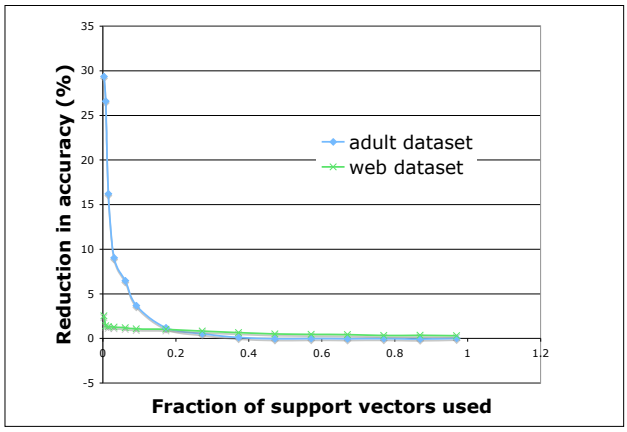

Figure 3: Reduction in SVM Accuracy Versus Fraction of original support vectors

$(E[||\sum_{i=1}^{N} \alpha_i K(\mathbf{x_i}, \mathbf{x}) - \sum_{k=1}^{M} \beta_k K(\mathbf{z_k}, \mathbf{x})||^2]$ - not just the output of the SVM) in Figure 4. The error trends closely resemble those of Figure 1.

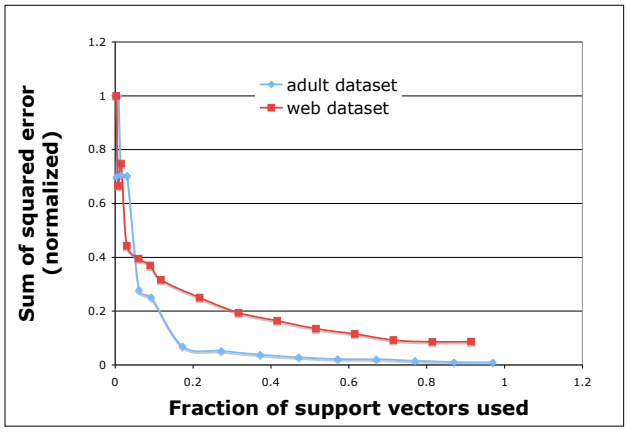

Figure 4: Sum of squared errors(normalized) Versus Fraction of original support vectors

6.5 Comparison to existing methods

Reduced set methods for SVMs have been studied before. [8],[3],[2] and [1] use techniques for reducing the number of support vectors. While all of them provide ways to do SVM reduction, none of them give a method for determining the reducibility of a given SVM.

[2] takes a fixed value of M to do an unconstrained optimization using conjugate gradient in the space of $\{\beta_k, \mathbf{z_k}\}$. [3] discuss two methods to do SVM reduction - one using SVM regression and one using a primal reformulation. These methods are limited by the fact the new SVM is constrained to contain only original data points as support vectors. The bigger problem, however, is that they give the user little handle on the accuracy-speed tradeoff. The SVM regression method, gives some handle with the ϵ parameter, but there is no direct relation between ϵ and the number of support vectors obtained.

[8] is closely related to our method. While [8] does use Kernel PCA to explain the reducibility of SVMs, they do not use the principal directions obtained in any way. They allow only original data points to enter into the new solution. This is achieved by a number of techniques - iterative optimization, quadratic programming and iterative kernel PCA. These methods are computationally expensive and provide only as much performance as our simple scheme. [1] uses kernel clustering and distance based preimage computations to reduce SVMs. This is also a very compute-intensive method, and does not provide a direct relation between the maximum-distance parameter(R) used and the number of support vectors obtained.

The USPS handwritten digit recognition dataset is used by [8] and [1]. To see why this dataset is attractive for SVM reduction, see Figure 5 for the eigen value distribution for the SVM classifying digit 5 from the rest of the digits. The reducibility is 0.0237, even better than the adult dataset and hence, is highly reducible.

For this particular instance of SVM, for a reduced set of size 96 support vectors, our method gives 50 errors, while [8] gives 47, 54 and 26 errors for 3 different methods. [1] gives 31 errors with 83 vectors. Our method is only worse than compute intensive methods like iterative optimization and kernel clustering, while

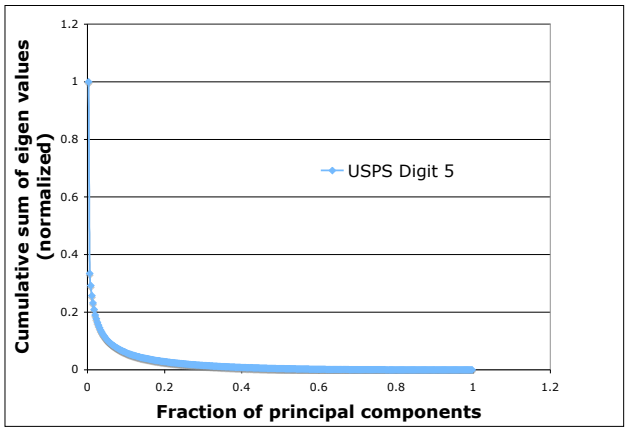

Figure 5: Eigen value distribution for the SVM classifying digit 5 from the rest

comparable to methods based on quadratic programming and iterative kernel PCA, but is much faster than those.

It should be noted that we get these results inspite of not taking into account the different α's as part of dimensionality reduction. Doing this will improve the results. Naïve uniform sampling gives us a decent performance. Better sampling heuristics will lead to better results. A simple extension would be to restrict the new support vectors to lie in the convex hull of the original support vectors.

Our results could be also be interpreted in a different way. We do not need accurate eigen value decomposition for the kernel PCA. As long as the errors in the eigen vectors are within the limits of $\pm\sqrt{\frac{\lambda_k}{N}}$ for all vectors V_k, we can use them to compute the new vectors. This approximation could lead to significant speedups.

7 Conclusion

A reduced set method based on kernel PCA is proposed in this paper. It is conceptually simple and easy to implement. The advantage of the method is that it gives comparable reduction performance to other complicated methods based on quadratic programming and iterative kernel PCA. We get a 90% reduction in support vectors with less than 1% degradation in the accuracy for the adult and web datasets by this method. The proposed method is also faster than other complicated techniques for the purpose.

References

[1] Z.-Q. Zeng, J. Gao, and H. Guo, "Simplified support vector machines via kernel-based clustering," *AI 2006: Advances in Artificial Intelligence*, vol. 4304, pp. 1189–1195, Oct 2006.

[2] C. J. C. Burges and B. Scholkopf, "Improving the accuracy and speed of support vector machines," *NIPS*, Dec 1996.

[3] E. Osuna and F. Girosi, "Reducing the run-time complexity of support vector machines," *ICPR*, Aug 1998.

[4] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

[5] B. Scholkopf, A. Smola, and K.-R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, Jul 1998.

[6] S. Mika, B. Scholkopf, A. Smola, K.-R. Muller, M. Scholz, and G. Ratsch, "Kernel pca and de-noising in feature spaces," *Advances in Neural Information Processing Systems*, Aug 1999.

[7] Y. Rathi, S. Dambreville, and A. Tannenbaum, "Statistical shape analysis using kernel PCA," *IST/SPIE Symposium on Electronic Imaging*, Dec 2006.

[8] B. Scholkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Muller, G. Ratsch, and A. Smola, "Input space vs. feature space in kernel-based methods," 1999.

[9] J. C. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," *Technical Report- Microsoft research*, Apr 1998.

[10] A. Asuncion and D. Newman, "UCI machine learning repository," 2007.

[11] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.