DEPARTMENT OF STATISTICS
North Carolina State University
2501 Founders Drive, Campus Box 8203
Raleigh, NC 27695-8203

Institute of Statistics Mimeo Series No. 2570

July 2004

Compactly Supported Radial Basis Function Kernels

Hao Helen Zhang   Marc Geoton   Peng Liu

Department of Statistics, North Carolina State University, Raleigh, NC

hzhang, genton, pliu4@stat.ncsu.edu

Supported in part by National Science Foundation grants DMS-0405913.

Compactly Supported Radial Basis Function Kernels

Hao H. Zhang HZHANG@STAT.NCSU.EDU
Department of Statistics
North Carolina State University
Raleigh, NC 27695-8203, USA

Marc G. Genton GENTON@STAT.TAMU.EDU
Department of Statistics
Texas A&M University
College Station, TX 77843-3143, USA

Peng Liu PLIU3@STAT.NCSU.EDU
Department of Statistics
North Carolina State University
Raleigh, NC 27695-8203, USA

Editor:

Abstract

The use of kernels is a key factor in the success of many classification algorithms by allowing nonlinear decision surfaces. Radial basis function (RBF) kernels are commonly used but generally associated with dense Gram matrices. We consider a mathematical operator to sparsify any RBF kernel systematically. It yields a kernel with a compact support and sparse Gram matrix. Having many zero elements in Gram matrices can greatly reduce computer storage requirements and the number of floating point operations needed in computation. This paper develops a unified framework to study the efficiency gain and information loss due to the sparsifying operation. We propose two quantitative measures: similarity and sparsity, and investigate their mathematical properties. Furthermore, the similarity-sparsity tradeoff is suggested to tune the thresholding parameter adaptively. We then implement compactly supported RBF kernels with tuning in support vector machines (SVM), least squares SVM, and kernel principal component analysis. Simulations demonstrate that properly-tuned compactly supported kernels give favorable performances while enjoying efficient algorithms for computation.

Keywords: Radial Basis Function, Sparse Gram Matrix, Support Vector Machines, Kernel Principal Component Analysis.

1. Introduction

Kernel based methods are theoretically well-founded and extremely useful in many learning algorithms such as support vector machines (see Boser et al. (1992); Vapnik (1995); Cristianini and Shawe-Taylor (2000); Schölkopf et al. (1999); Schölkopf and Smola (2002)) and kernel principal component analysis in Schölkopf and Smola (2002). The idea is to embed the data into some feature space (usually high dimensional) and then apply linear algorithms to detect patterns in the feature space. The resulting solution is generally nonlinear in the original input domain, thus assuring great flexibility in the learning process.

For any input space $\mathcal{X} \subset \mathbb{R}^d$, a kernel is a positive definite function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, satisfying $\sum_{i,j=1}^n a_i a_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ for any $a_1, ..., a_n \in \mathbb{R}$ and $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathcal{X}$. A kernel can be regarded as a similarity measure between any two inputs. Gaussian kernels and polynomial kernels are commonly used; see Genton (2001) for a variety of kernels.

Given a training set of input vectors $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathcal{X}$, the Gram matrix associated with a kernel k is an $n \times n$ matrix K with elements $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j = 1, ..., n$. K is symmetric, positive definite, and generally dense in terms of not having any zero entries. A dense Gram matrix can be highly ill-conditioned especially for large datasets, thus posing severe numerical problems in solution feasibility and stability. Various methods have been proposed to speed up kernel machines, such as decomposition methods for SVM by Osuna et al. (1997), speedup methods for kernel PCA (Smola and Schölkopf (2000), Smola et al. (2000)), and other kernel methods (Burges (1996), Williams and Seeger (2001)).

The motivation of our work is to construct kernels which have local support and sparse Gram matrices. A closely related work was developed in Achlioptas et al. (2002) which sparsifies the Gram matrix K by randomly omitting its entries with some sampling scheme. Our method is based on certain thresholding techniques. One straightforward idea is to replace all small entries (smaller than some cutoff value) in K directly by zeros. However, this naive thresholding often destroys positive definiteness of the Gram matrix. In this paper, we study a mathematical operation which sparsifies K systematically while maintaining its positive definiteness. We will focus on radial basis function (RBF) kernels. Section 2 introduces compactly supported RBF kernels and discusses the efficiency gain in storage and computation with sparse Gram matrices. Section 3 proposes two quantitative measures, similarity and sparsity, to characterize sparsified Gram matrices, and shows how the similarity-sparsity tradeoff can be used to tune the thresholding parameter adaptively. A practical solution for selecting the thresholding parameter is further provided for the special case when the memory required to store K is beyond computer capacity. Section 4 illustrates the performances of compactly supported RBF kernels in three different contexts: support vector machines (SVM), least squares SVM, and kernel principal component analysis (PCA). Section 5 shows two real world examples, one of which has a high dimensional input space. A discussion is given in Section 6.

2. Compactly Supported Kernels and Gram Matrices

In this section, we define compactly supported RBF kernels and study the sparse property of their Gram matrices.

2.1 Compactly Supported RBF Kernels

A radial basis function (RBF) kernel, also known as an isotropic stationary kernel, is defined by a function $\psi : [0, \infty) \to \mathbb{R}$ such that

$$k(\mathbf{x}, \mathbf{x}') = \psi(\|\mathbf{x} - \mathbf{x}'\|),$$

where $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ and $\|\cdot\|$ denotes the Euclidean norm. A Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{\sigma^2})$ is one example of RBF kernels. Most RBF kernels are globally supported and therefore yield dense Gram matrices, which can be highly ill-conditioned for large datasets. Genton (2001) pointed out that for any RBF kernel k, a class of compactly supported kernels can be constructed by multiplying $k(\cdot,\cdot)$ with some compactly supported radial basis function (see Schaback (1995); Wendland (1995); Floater and Iske (1996)). In particular, we consider the following construction

$$k_C(\mathbf{x},\mathbf{x}') = \phi_C(\|\mathbf{x}-\mathbf{x}'\|)k(\mathbf{x},\mathbf{x}'), \tag{1}$$

and

$$\phi_C(\|\mathbf{x}-\mathbf{x}'\|) = \big[(1-\frac{\|\mathbf{x}-\mathbf{x}'\|}{C})_+\big]^\nu,$$

where $C > 0$, $\nu \geq \frac{d+1}{2}$ is an integer, and $(\cdot)_+$ is the positive part. The function $\phi_C(\cdot)$ is a sparsifying operator, which thresholds all the entries with distance $\|\mathbf{x}-\mathbf{x}'\|$ larger than C to zeros in the Gram matrix. The new kernel resulting from this construction preserves positive definiteness; see Gneiting (2002). For any input $\mathbf{x}$, define its C-neighborhood set

$$\delta_C(\mathbf{x}) = \{\mathbf{x}' : \|\mathbf{x}-\mathbf{x}'\| \leq C\}.$$

Then k_C is "local" and has compact support $[0, C]$ since

$$k_C(\mathbf{x},\mathbf{x}') = 0 \quad \text{if} \quad \mathbf{x}' \notin \delta_C(\mathbf{x}).$$

This important fact leads to the desired sparse structure in the associated Gram matrix K_C. The constant C is called the thresholding or truncation parameter, which controls the support size of the kernel k_C, or the degree of the sparsity in K_C. The power term ν decides the degree of smoothness (differentiability) of ϕ_C.

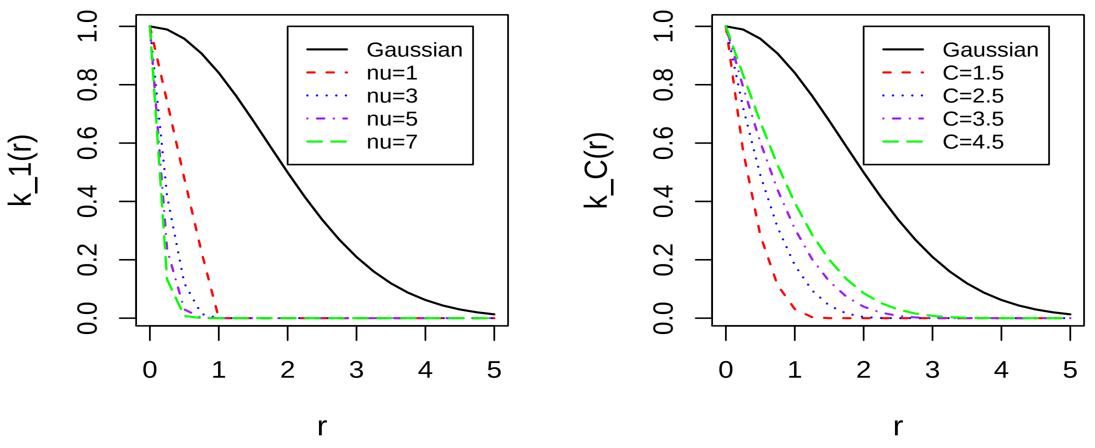

Figure 1: Gaussian kernel and the compactly supported kernels with varying ν and C. All the kernels are plotted as a function of $r = \|\mathbf{x}-\mathbf{x}'\|$.

Different choices of C and ν give different compactly supported kernels. Figure 1 plots the Gaussian kernel and the derived compactly supported kernels with various values of C and ν. All the kernels are depicted as a function of $r = \|\mathbf{x} - \mathbf{x}'\|$. Solid lines in both panels represent the Gaussian kernel with $\sigma = 2.4$. In the left panel, $C = 1$ and ν varies by taking values $1, 3, 5, 7$, thus the resulting kernels have a common support $[0, 1]$. In the right panel, $\nu = 3$ but C takes different values $1.5, 2.5, 3.5, 4.5$, leading to compactly supported kernels with different supports. Since ν is irrelevant to the sparsity of K_C, it is generally fixed at some value. We use $\nu = 3$ in this paper.

The following lemma shows that for a pair of inputs $\mathbf{x}$ and $\mathbf{x}'$, the smaller C is, the more shrinkage imposed on the function value $k(\mathbf{x}, \mathbf{x}')$. This implies that the Gram matrices K and K_C can be either similar or different, depending on the choice of C.

Lemma 1 *With ν fixed, for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$,*

$$\begin{aligned} \lim_{C \to 0} \quad k_C(\mathbf{x}, \mathbf{x}') &= 0, \\ \lim_{C \to \infty} \quad k_C(\mathbf{x}, \mathbf{x}') &= k(\mathbf{x}, \mathbf{x}'). \end{aligned}$$

Proof: The equations are easily established by the facts: $\lim_{C\to 0} \phi_C(\|\mathbf{x} - \mathbf{x}'\|) = 0$ and $\lim_{C\to\infty} \phi_C(\|\mathbf{x} - \mathbf{x}'\|) = 1$ for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. ■

The use of compactly supported kernels can make predictions more efficient. For example, given the training points $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)$, the nonlinear support vector machine (SVM) classifier trained with a kernel k is generally expressed as

$$f(\mathbf{x}) = b + \sum_{i=1}^{n} a_i k(\mathbf{x}, \mathbf{x}_i).$$

To predict the label for a future observation $\mathbf{x}_{new}$ using the classifier trained with the compactly supported kernel k_C, we only need to visit those $\mathbf{x}_i$'s in the set $\delta_C(\mathbf{x}_{new})$ since $f(\mathbf{x}_{new}) = b + \sum_{i=1}^{n} a_i k_C(\mathbf{x}_{new}, \mathbf{x}_i) = b + \sum_{i:\mathbf{x}_i \in \delta_C(\mathbf{x}_{new})} a_i k_C(\mathbf{x}_{new}, \mathbf{x}_i)$. That is, the representation of $f(\mathbf{x})$ is sparse in the sense that $k_C(\mathbf{x}, \mathbf{x}_i)$ might be zero. Trying to determine the set $\delta_C(\mathbf{x}_{new})$ efficiently might be challenging though. One simple idea is to discard all observations whose first component are not within $\pm C$ around the first component of $\mathbf{x}_{new}$, and compute the distance from the remaining observations to $\mathbf{x}_{new}$, but more clever algorithms might be constructed.

2.2 Sparse Gram Matrices

Compactly supported RBF kernels have sparse Gram matrices, which can produce crucial advantages for certain applications dealing with massive datasets. Firstly, sparse matrices give great savings in storage and computation memory by not storing many zero elements. For example, the storage requirement for an $n \times n$ full matrix is $8n^2$ bytes, if we use 8-bytes for reals and 4-bytes for integers. A sparse matrix is stored as a list of its nonzero elements together with the row and column index, in the column major order. Therefore, the storage requirement of an $n \times n$ sparse matrix with N nonzero entries is $12N + 4n$ bytes, which can be much less than $8n^2$ if N/n^2 is small. The sparsity of a matrix here (more elaborations in Section 3) is defined as the percentage of zero entries in the matrix.

sparsity	95%	75%	50%	0
bandwidth (bw)	325	824	1,122	2,000

Table 1: Bandwidths after applying the reverse Cuthill-McKee algorithm.

Secondly, sparse matrix algorithms require much less computation time than standard algorithms by avoiding arithmetic operations on zero elements. For example, the complexity of solving a linear equation system is $O(n^3)$ for a full $n \times n$ matrix and is approximately $O(N)$ for a sparse matrix with certain structures. Linear systems involving sparse coefficient matrices can generally be solved very efficiently using sparse linear algebra techniques; see Gilbert et al. (1992) with an emphasis on MATLAB commands.

The entries of K_C can be reordered to have certain matrix structures convenient for operations. Two commonly used techniques are reverse Cuthill-McKee algorithm (Gibbs et al. (1976), George and Liu (1981)) and the minimum degree reordering algorithm (George and Liu (1989)). The reverse Cuthill-McKee algorithm produces a matrix with a narrow bandwidth and generally makes nonzero elements display along the main diagonal line. For any symmetric matrix A, the bandwidth of its ith row, b_i, is defined as the difference between i and the smallest column number j such that $A_{ij} \neq 0$. The bandwidth of A is defined

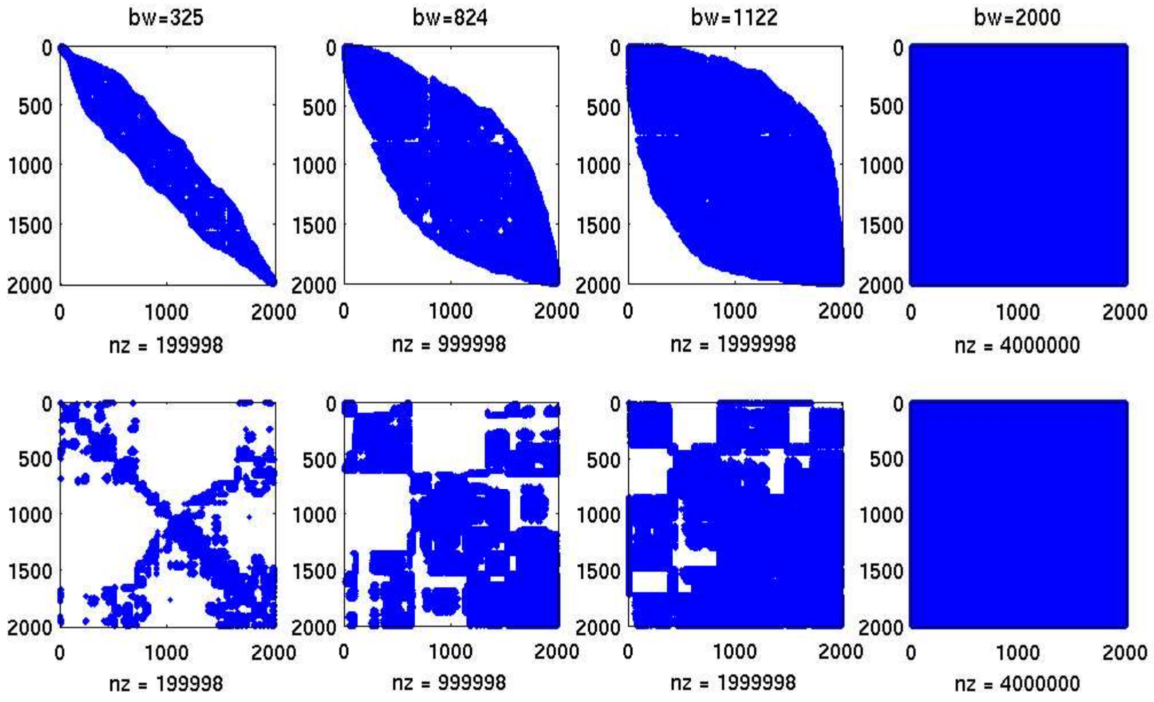

Figure 2: Structures of K_C after reordering (first row for reverse Cuthill-McKee, second row for minimum degree reordering). The sparsity degree is respectively $95\%, 75\%, 50\%, 0$ (from left to right). The "nz" is the number of nonzeros in K_C.

by $bw = 2\max_i\{b_i\} + 1$. Bandwidth of a banded matrix is often used as a measure of its sparsity. As an illustrative example, consider a $2,000 \times 2,000$ Gram matrix K (obtained from the green-red dataset described in Section 4) and the sparsified Gram matrix K_C with different degrees of sparsity: $95\%, 75\%, 50\%$ and 0. Table 1 shows the bandwidth of the reordered matrix K_C after applying the reverse Cuthill-McKee algorithm. The matrix structures after reordering are plotted in the first row of Figure 2. The minimum degree procedure produces a structure with large blocks of continuous zeros, as shown in the second row of Figure 2.

3. Tuning C

It is well known that the Gram matrix contains all the information about the input data. Lemma 1 shows that the smaller C, the sparser K_C is; and vice versa. Though a high degree of sparsity is desired for computational reasons, much information may have been lost due to the sparsifying process if K_C is too sparse, and the learning error may become unacceptable. On the other hand, if K_C is almost as dense as K, the efficiency gain in computation will be limited. Therefore we need to select the thresholding parameter C properly to balance the "information loss" and the "efficiency gain". This is analogous to controlling the bias-variance tradeoff of estimators in statistical modeling.

3.1 Similarity and Alignment

Define the matrix Φ_C by $(\Phi_C)_{ij} = \left[(1 - \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{C})_+\right]^\nu$. We will use $\circ$ for the Hadamard product of two matrices (element-wise product), see Horn and Johnson (1985). Then we have

$$K_C = \Phi_C \circ K. \tag{2}$$

In order to assess the similarity between K and K_C, we consider the empirical alignment of two Gram matrices

$$A(C) = \frac{< K, K_C >_F}{\sqrt{< K, K >_F, < K_C, K_C >_F}}, \tag{3}$$

where $< K_1, K_2 >_F = \sum_{i,j=1}^n K_1(\mathbf{x}_i, \mathbf{x}_j) K_2(\mathbf{x}_i, \mathbf{x}_j)$ is the Frobenius inner product between two matrices. This alignment was first proposed in Cristianini et al. (2001) for learning kernels. It can be viewed as the Pearson correlation coefficient of two bi-dimensional vectors defined by K and K_C. A convenient property of $A(C)$ is that it can be efficiently computed before any training of the kernel machine takes place, only based on the input information in the training set. The following theorem shows that, for the Gram matrix K_C defined in (2), the alignment score $A(C)$ has some nice mathematical properties as a function of C. We take the convention $0/0 = 0$ throughout this paper.

Theorem 1 *Given the input set $\{\mathbf{x}_1, ..., \mathbf{x}_n\} \subset \mathcal{X}$, the empirical alignment between K and K_C is*

$$A(C) = \frac{\sum_{i,j=1}^n (\Phi_C)_{ij} K_{ij}^2}{\sqrt{\sum_{i,j=1}^n K_{ij}^2}\sqrt{\sum_{i,j=1}^n (\Phi_C)_{ij}^2 K_{ij}^2}},$$

and has the following properties:

(i) $\lim_{C\to 0} A(C) = 0$*,*

(ii) $\lim_{C\to\infty} A(C) = 1$*,*

(iii) $A(C)$ *is an increasing function of* C*.*

Proof. (i) and (ii) can be easily verified using the definition of $A(C)$ in (3) and Lemma 1. Define $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|, i, j = 1, ..., n$. In order to prove (iii), it is sufficient to show that, if $r_{ij} \le C$ for all $i, j = 1, ..., n$, then

$$\tilde{A}(C) = \frac{\sum_{i,j=1}^{n}(1-\frac{r_{ij}}{C})^{\nu}K_{ij}^2}{\sqrt{\sum_{i,j=1}^{n}(1-\frac{r_{ij}}{C})^{2\nu}K_{ij}^2}},$$

is increasing in C. It is equivalent to show the derivative of $\ln[\tilde{A}(C)]$ with respect to C

$$D(C) = \left(\ln[\tilde{A}(C)]\right)' \ge 0. \tag{4}$$

Since

$$D(C) = \frac{\nu\sum_{i,j=1}^{n}(1-\frac{r_{ij}}{C})^{\nu-1}\frac{r_{ij}K_{ij}^2}{C^2}}{\sum_{i,j=1}^{n}(1-\frac{r_{ij}}{C})^{\nu}K_{ij}^2} - \frac{\nu\sum_{i,j=1}^{n}(1-\frac{r_{ij}}{C})^{2\nu-1}\frac{r_{ij}K_{ij}^2}{C^2}}{\sum_{i,j=1}^{n}(1-\frac{r_{ij}}{C})^{2\nu}K_{ij}^2},$$

the inequality in (4) becomes

$$[\sum_{i,j=1}^{n}(1-\frac{r_{ij}}{C})^{\nu-1}\frac{r_{ij}K_{ij}^2}{C^2}][\sum_{i,j=1}^{n}(1-\frac{r_{ij}}{C})^{2\nu}K_{ij}^2] \ge [\sum_{i,j=1}^{n}(1-\frac{r_{ij}}{C})^{2\nu-1}\frac{r_{ij}K_{ij}^2}{C^2}][\sum_{i,j=1}^{n}(1-\frac{r_{ij}}{C})^{\nu}K_{ij}^2]. \tag{5}$$

Define $p_{ij} = \frac{1-r_{ij}}{C}$ for $i, j = 1, ..., n$. Then (5) is expressed as

$$[\sum_{i,j=1}^{n} p_{ij}^{\nu-1}(1-p_{ij})K_{ij}^2][\sum_{i,j=1}^{n} p_{ij}^{2\nu}K_{ij}^2] \ge [\sum_{i,j=1}^{n} p_{ij}^{2\nu-1}(1-p_{ij})K_{ij}^2][\sum_{i,j=1}^{n} p_{ij}^{\nu}K_{ij}^2],$$

which can be further simplified to

$$[\sum_{i,j=1}^{n} p_{ij}^{\nu-1}K_{ij}^2][\sum_{i,j=1}^{n} p_{ij}^{2\nu}K_{ij}^2] \ge [\sum_{i,j=1}^{n} p_{ij}^{2\nu-1}K_{ij}^2][\sum_{i,j=1}^{n} p_{ij}^{\nu}K_{ij}^2]. \tag{6}$$

Finally we can verify (6) by the following two Hölder inequalities:

$$\sum_{i,j=1}^{n} p_{ij}^{2\nu-1}K_{ij}^2 = \sum_{i,j=1}^{n}\big(p_{ij}^{\frac{2v^2}{v+1}}K_{ij}^{2\frac{v}{v+1}}\big)\big(p_{ij}^{\frac{v-1}{v+1}}K_{ij}^{2\frac{1}{v+1}}\big) \le [\sum_{i,j=1}^{n} p_{ij}^{2v}K_{ij}^2]^{\frac{v}{v+1}}[\sum_{i,j=1}^{n} p_{ij}^{v-1}K_{ij}^2]^{\frac{1}{v+1}}$$

$$\sum_{i,j=1}^{n} p_{ij}^{\nu}K_{ij}^2 = \sum_{i,j=1}^{n}\big(p_{ij}^{\frac{v(v-1)}{v+1}}K_{ij}^{2\frac{v}{v+1}}\big)\big(p_{ij}^{\frac{2v}{v+1}}K_{ij}^{2\frac{1}{v+1}}\big) \le [\sum_{i,j=1}^{n} p_{ij}^{v-1}K_{ij}^2]^{\frac{v}{v+1}}[\sum_{i,j=1}^{n} p_{ij}^{2v}K_{ij}^2]^{\frac{1}{v+1}}.$$

■

3.2 Sparsity of K_C

The Gram matrix K is dense in terms of not having any zero entries. Given a training set of size n, we define the sparsity of K_C relative to K by

$$S(C) = \frac{\text{the number of zero entries in } K_C}{n^2}. \tag{7}$$

The following theorem describes the property of $S(C)$ as a function of C.

Theorem 2 *Given the input set $\{\mathbf{x}_1, ..., \mathbf{x}_n\} \subset \mathcal{X}$, the sparsity $S(C)$ satisfies:*

(i) $\lim_{C\to 0} S(C) = 1$*,*

(ii) $\lim_{C\to\infty} S(C) = 0$*,*

(iii) $S(C)$ is a decreasing function of C.

Proof. Lemma 1 implies that $\lim_{C\to 0}(K_C)_{ij} = 0$ and $\lim_{C\to\infty}(K_C)_{ij} = K_{ij}$ for $i, j = 1, ..., n$, thus (i) and (ii) hold. As C increases, the entries of K_C become closer to those of K and the number of zeros in K_C decreases. Therefore (iii) is true by the definition of $S(C)$. ■

3.3 Similarity-Sparsity Tradeoff

For any compactly supported kernel k_C, a large empirical alignment $A(C)$ indicates little information loss due to the sparsifying process, and a large sparsity $S(C)$ implies a high degree of sparsity. Theorems 1 and 2 show that both $A(C)$ and $S(C)$ take values in the interval $[0, 1]$; as C increases, $A(C)$ increases but $S(C)$ decreases. Therefore it is impossible to maximize both scores simultaneously. This similarity-sparsity tradeoff is analogous to the bias-variance tradeoff in statistical modeling to avoid overfitting. We propose three procedures to tune C adaptively, and they are all convenient to implement before any training of the kernel machine takes place.

The first procedure is to maximize a linear combination of similarity and sparsity,

$$\max_C \; A(C) + \gamma S(C). \tag{8}$$

Here $\gamma > 0$ is the tuning parameter. The smaller γ is, the more weight is put on the similarity; the larger γ is, the more weight is put on the sparsity. When $\gamma = 1$, the similarity and the sparsity are equally weighted. The choice of γ depends on the nature of problems and the practical need of researchers. In Figure 3, we plot the linear combination curve with $\gamma = 0.5$ and $\gamma = 1$ in the green-red example (Example 1 in Section 4).

The second tuning procedure is to control the lower bound for information loss while maximizing the sparsity of the Gram matrix,

$$\max_C \; S(C), \quad \text{subject to} \quad A(C) \geq \mu, \tag{9}$$

where $\mu \in [0, 1]$ is the desired degree of similarity. Alternatively, we can maximize the similarity $A(C)$ conditional on a desired degree of sparsity,

$$\max_C \; A(C), \quad \text{subject to} \quad S(C) \geq \tau. \tag{10}$$

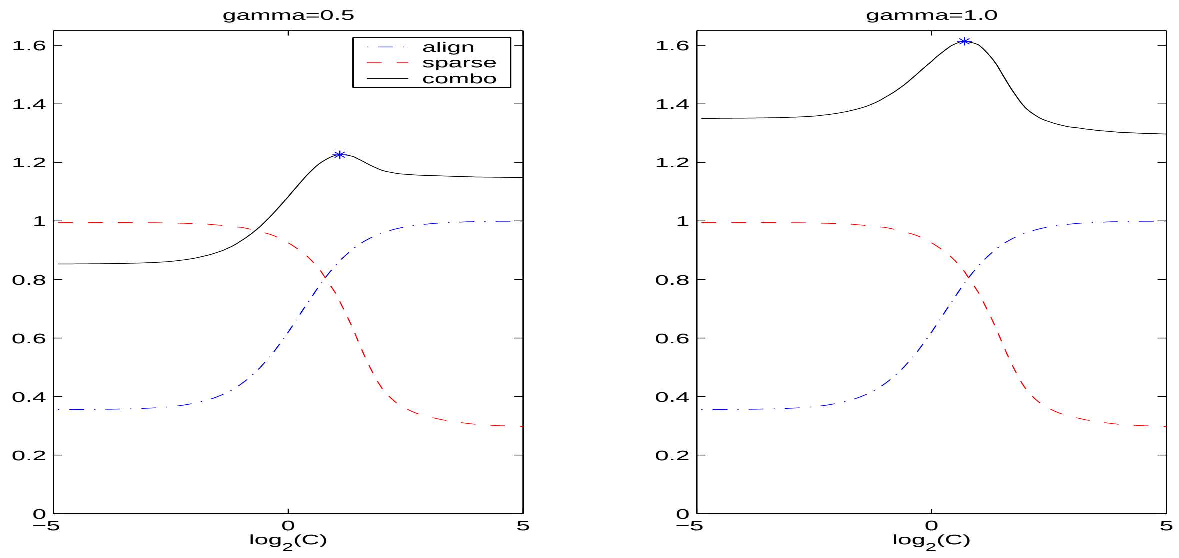

Figure 3: Alignment (dashed-dotted lines), sparsity (dashed lines), and their linear combination score (solid lines) with $\gamma = 0.5$ and $\gamma = 1$. The symbol * marks the location of the optimal C where the combination score is maximized.

This third procedure is especially useful in massive dataset situations when the size of the Gram matrix is beyond computer storage limit. In practice, we can pre-evaluate the maximal number of nonzero entries that can be stored based on computer capacity, and then determine the lower bound for the degree of sparsity. For example, if only 25% of the nonzero entries in a matrix can be stored, we may choose $\tau = 75\%$ and C to be the first quartile of pairwise distances $\{\|\mathbf{x}_i - \mathbf{x}_{i'}\|, i, i' = 1, ..., n\}$.

4. Applications of Compactly Supported RBF Kernels

In this section, we illustrate the performances of compactly supported RBF kernels in three different contexts: Support Vector Machines (SVM), least squares SVM, and kernel principal component analysis. We implement all three tuning procedures proposed in Section 3.3.to select the thresholding parameter C.

4.1 Support Vector Machines

Support vector machine (SVM) is a binary classifier algorithm proposed in Boser et al. (1992), Vapnik (1995), and Vapnik (1998). Cristianini and Shawe-Taylor (2000) gave a comprehensive introduction on the SVMs. Given a training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$ where $y_i \in \{+1, -1\}$ is the label specifying the class that the example belongs to, the task is to learn a classification rule $f(\mathbf{x}) \rightarrow \{+1, -1\}$ which will be used to make predictions for future inputs. The nonlinear SVM can be cast as a regularization problem in a reproducing kernel

Hilbert space (RKHS) associated with the kernel k and Gram matrix K. It solves

$$\min_{b,\mathbf{a}} \frac{1}{n}\sum_{i=1}^{n}[1-y_i f(\mathbf{x}_i)]_+ + \lambda \mathbf{a}^T K \mathbf{a}, \tag{11}$$

where $f(\mathbf{x}) = b + \sum_{i=1}^{n} a_i k(\mathbf{x}, \mathbf{x}_i)$ and $\mathbf{a} = (a_1, ..., a_n)^T$. The function $[1-yf]_+$ is the so-called hinge loss function, acting as a convex upper bound of the misclassification error. The parameter $\lambda > 0$ is a smoothing parameter. Define $\mathbf{y} = (y_1, ..., y_n)^T$, $\mathbf{Y} = \mathrm{diag}[y_1, ..., y_n]$, $\mathbf{1}_n = (1, ..., 1)^T \in \mathbb{R}^n$, and I_q to be the identity matrix of size $q \times q$. In general, the following dual problem of (11) is solved with respect to the dual variables $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n)^T$:

$$\max_{\boldsymbol{\alpha}} \quad W(\boldsymbol{\alpha}) = -\frac{1}{2}(\boldsymbol{\alpha}\mathbf{Y})^T K(\boldsymbol{\alpha}\mathbf{Y}) + \mathbf{1}_n^T\boldsymbol{\alpha}, \quad \text{subject to} \;\; \mathbf{0} \le \boldsymbol{\alpha} \le \lambda \mathbf{1}_n, \;\; \boldsymbol{\alpha}^T\mathbf{y} = 0. \tag{12}$$

This is a quadratic programming with linear constraints. With the compactly supported RBF kernel, we need to solve (12) with K replaced by K_C. Some software packages can take advantage of the sparse structure of K_C, for example the TOMLAB (`http://tomlab.biz`) and the SPRNLP (`http://tomlab.biz/products/sprnlp`), and optimize the quadratic objective more efficiently with K_C than with K. For large datasets, this efficiency gain in computation can be very substantial.

Example 1: Let us consider the green-red example used in Hastie et al. (2001). Two classes are labeled as "green" and "red", each consisting of a mixture of ten Gaussian clusters. Ten centers m_k's of the green class are generated from the bivariate Gaussian distribution $N_2((1,0)^T, I_2)$, and those of the red class are from $N_2((0,1)^T, I_2)$. Figure 4 plots the centers of two classes (large circles). For each class, 100 observations are generated as follows: for each observation, we pick an m_k at random with probability 1/10 and generate a data from an $N_2(m_k, I_2/5)$. To evaluate the classification accuracy of the SVM solutions, we also generate $1,000$ test data points.

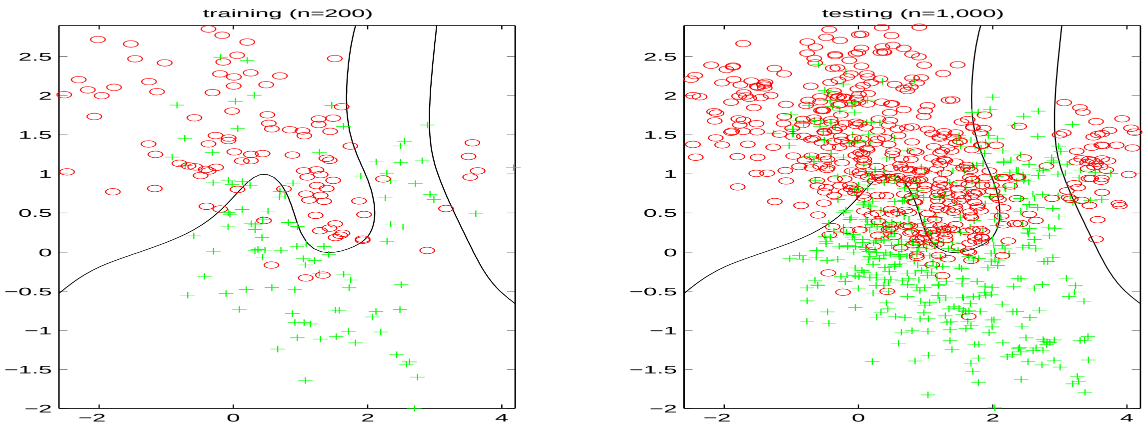

Figure 4: Scatter plots of 200 training examples (left panel) and $1,000$ testing examples (right panel) for the green-red example. Large circles are the class centers. The solid curve it the optimal Bayes rule decision boundary.

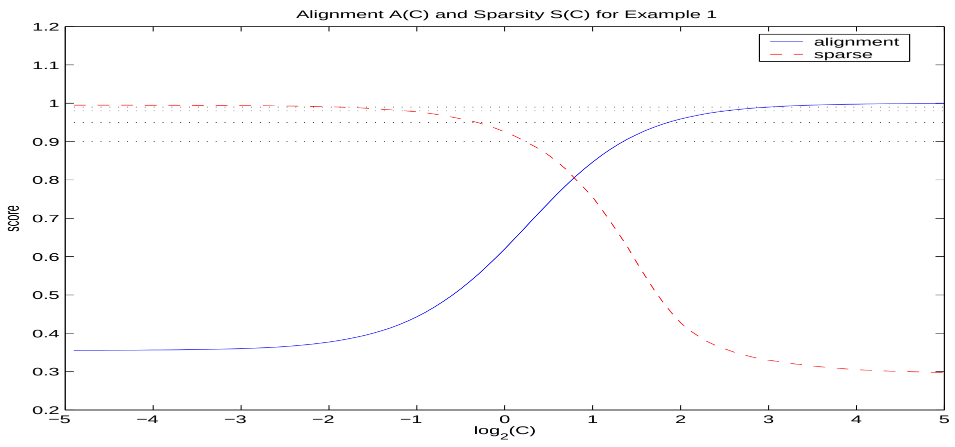

Figure 5: Alignment $A(C)$ (solid line) and sparsity $S(C)$ (dashed line) as a function of C. The horizontal dotted lines correspond to different $\mu : 0.90, 0.95, 0.98,$ and 0.99.

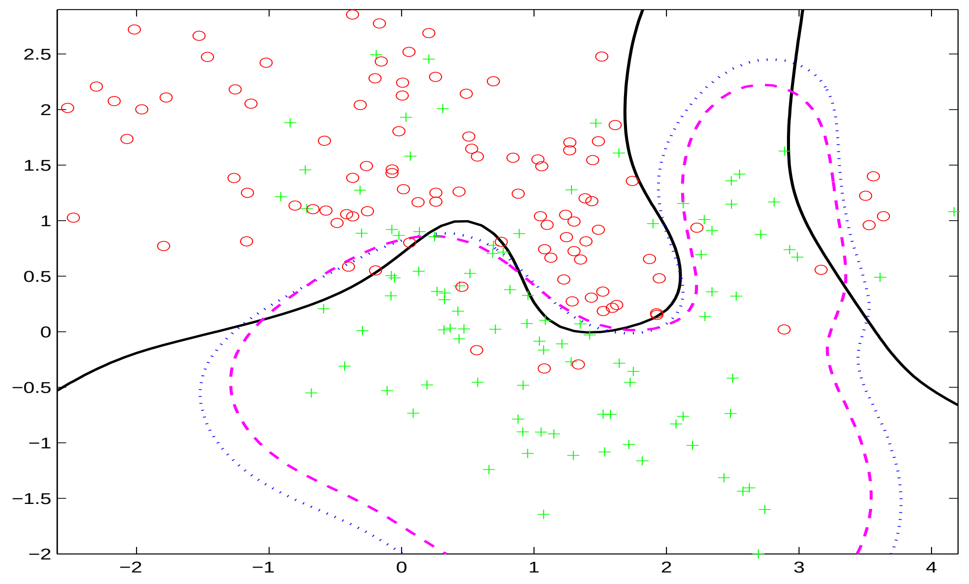

Figure 6: Classification boundaries given by the Bayes rule (solid line), SVM with Gaussian kernel (dotted line), and SVM with compactly supported kernel (dashed line).

We use $\sigma = 0.6$ in the Gaussian kernel. For the compactly supported kernels, the optimal C is chosen over the range $\log_2(C) \in [-5, 5]$ using the second procedure in Section 3.3, which maximizes $S(C)$ subject to $A(C) \geq \mu$. Various values of μ are considered. Figure 5 plots the $A(C)$ and $S(C)$ scores and the dotted lines correspond to $\mu = 0.90, 0.95,\ 0.98,$ and 0.99. We then fit the SVM with Gaussian kernel k and the compactly supported kernel

μ	$\hat{C}$	$A(\hat{C})$	$S(\hat{C})$	test error
0.90	1.583	0.912	0.482	0.236
0.95	2.239	0.956	0.301	0.229
0.98	3.394	0.981	0.209	0.229
0.99	4.800	0.991	0.185	0.228
Gaussian kernel	∞	1	0	0.228

Table 2: SVM fitting with Gaussian kernel and compactly supported kernels.

$k_{\hat{C}}$. The smoothing parameter λ in the SVM is tuned with 10-fold cross validation. Table 2 summarizes the optimal value $\hat{C}$, $A(\hat{C})$, and $S(\hat{C})$ for each choice of μ. When $\mu = 0.99$, the SVM solution obtained with $k_{\hat{C}}$ gives the same accuracy as the solution with k, but the Gram matrix $K_{\hat{C}}$ has the sparsity 18.5%. When $\mu = 0.95$, though the test error of the solution with $k_{\hat{C}}$ increases slightly from 0.228 to 0.229, the sparsity of the associated Gram matrix increases to 30.1%. When $\mu = 0.90$, the test error of the solution with $k_{\hat{C}}$ is 3.5% higher than the solution with k (0.236 vs 0.228), but 48.2% of the entries in $K_{\hat{C}}$ are zeros. Figure 6 plots the classification boundaries given by the Bayes rule, the SVM solution with k, and the SVM solution with $k_{\hat{C}}$ corresponding to $\mu = 0.98$. The Bayes rule has the test error 0.218, which is the optimal rate using the true distribution of the data.

4.2 Least Squares Support Vector Machines

Regularized least squares classification was suggested in Poggio and Girosi (1990) and investigated in Rifkin et al. (2003). Recently the least squares support vector machine (LS-SVM) was used as a variant of SVM by Suykens and Vandewalle (1999) and Suykens et al. (1999). The advantages of the LS-SVM include its good classification accuracy and easy implementation by solving a linear equation system. This can make a dramatic difference in computation cost for large datasets compared to the standard SVM, where a quadratic programming has to be employed. In theory both LS-SVM and SVM target on the Bayes rule asymptotically; see Lin (2002).

The LS-SVM seeks a separating hyperplane $f(\mathbf{x}) = b + \sum_{i=1}^{n} a_i k(\mathbf{x}, \mathbf{x}_i)$ by solving

$$\min_{b,\mathbf{a}} \frac{1}{n} \sum_{i=1}^{n} \left[1 - y_i f(\mathbf{x}_i)\right]^2 + \lambda \mathbf{a}^T K \mathbf{a}. \tag{13}$$

The coefficients b and $\mathbf{a} = (a_1, ..., a_n)^T$ are the solution of the following linear system:

$$\begin{pmatrix} 0 & \mathbf{1}_n^T \\ \mathbf{1}_n & K + \frac{1}{\lambda} I_n \end{pmatrix} \begin{pmatrix} b \\ \mathbf{a} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{y} \end{pmatrix}. \tag{14}$$

One standard technique for solving a linear system is the Cholesky factorization. Sparse linear systems arise by replacing K with K_C in (14). The advantage of using K_C is that it gives a higher degree of sparsity in the Cholesky factor and thus reduces the computational cost. Hamers et al. (2002) implemented the compactly supported RBF kernels in LS-SVM without tuning C. The following example shows that compactly supported RBF kernels with properly selected C have favorable performances. The computation time reported (in seconds) is the average time of solving the linear equation system (14).

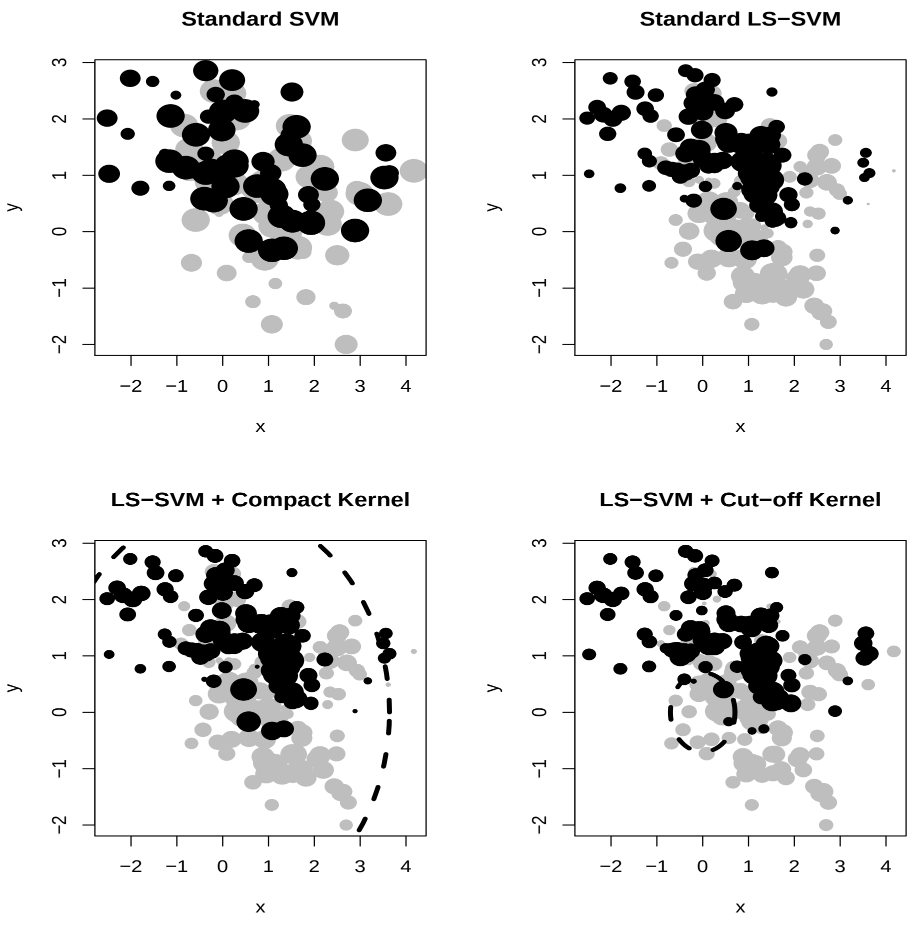

Figure 7: Bubble plots for absolute values of a_i (the larger the a_i the larger the bubble): Standard SVM+RBF (top left); Standard LS-SVM +RBF (top right); LS-SVM + compactly supported RBF with $\nu = 3$ and C tuned by similarity-sparsity tradeoff (bottom left); LS-SVM + 10% cut-off RBF (bottom right).

Example 2: Generate a large dataset of size $n = 5,000$ from the green-red example used in Example 1. We fit the least squares SVM with the standard Gaussian kernel and compactly supported kernels for which the optimal C is chosen with the third procedure in Section 3.3. We allow τ to take different values $90\%, 80\%, 70\%$ and 50%, and the corresponding $\hat{C}$ is thus the 10th, 20th, 30th, and 50th percentile of the pairwise distances $\{\|\mathbf{x}_i - \mathbf{x}_{i'}\|, i, i' = 1, ..., n\}$. The smoothing parameter λ is selected with an extra tuning set of size 3,000 generated from the same distribution. Table 3 shows that the compactly supported kernels may give comparable accuracy as the Gaussian kernel, but with substantial savings in computation. When $\tau = 0.5$, the LS-SVM solution with $k_{\hat{C}}$ gives the test error 21.5% which is even better than that with Gaussian kernel, but it saves the computation

τ	C	test error	time [s.]
0.9	0.717	0.272	8.95
0.8	1.060	0.230	9.73
0.7	1.060	0.218	9.65
0.5	1.927	0.215	12.73
Gaussian	∞	0.216	23.93

Table 3: Least squares SVM fitting with Gaussian and compactly supported kernels.

time by 47% (12.73 vs 23.93 seconds). When $\tau = 0.7$, the compactly supported kernel solution has the test error 21.8%, which is slightly worse than the Gaussian kernel, but saves the computation time by 60%.

To show the "local" property of the classifier given by the compactly supported kernel, we depict bubble plots for the a_i's of the least square SVM solutions in Figure 7. Here the small dataset of size $n = 200$ from Example 1 is used. The radius of a bubble denotes the corresponding a_i in absolute value. For evaluating the function value at some example $\mathbf{x}_{new}$, all the points falling outside its neighbor $\delta_C(\mathbf{x}_{new})$ need not to be accessed. In top panels, all the coefficients a_i's in the SVM and LS-SVM solutions obtained with the Gaussian kernel k are different from zero. Therefore, the classification of a new input is based on all the training points. However with compactly supported kernels, only training points located in a circle (dashed curves in bottom panels) centered at $\mathbf{x}_{new}$ and of radius C are necessary. The 10% cut-off RBF consists in defining C as the 10% percentile of the sorted set of all distances between the input points.

4.3 Kernel Principal Component Analysis

Kernel principal component analysis (Schölkopf and Smola (2002)) is a kernel-based method for performing a nonlinear form of principal component analysis, and it has been an effective approach for feature extraction. Assume a nonlinear map $\Phi : \mathcal{X} \to \mathcal{H}$ maps the data from the input space into some feature space $\mathcal{H}$ associated with the kernel k,

$$k(\mathbf{x}, \mathbf{x}') =< \Phi(\mathbf{x}), \Phi(\mathbf{x}') >_{\mathcal{H}}, \tag{15}$$

where $< \cdot, \cdot >_{\mathcal{H}}$ is the dot product defined in $\mathcal{H}$. The Gram matrix K is defined by $K_{ij} =< \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >_{\mathcal{H}}$. In general, the feature space $\mathcal{H}$ contains many nonlinear features of input variables or their high-order interactions, thus it can have a large or possibly infinite dimension. We assume the data in $\mathcal{H}$ is centered, that is, satisfying $\sum_{i=1}^{n} \Phi(\mathbf{x}_i) = 0$, and has the covariance matrix $\Sigma = \frac{1}{n} \sum_{i=1}^{n} \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)^T$. Σ is positive definite, and it has n non-negative eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$ and the corresponding normalized eigenvectors $\mathbf{v}^1, \cdots, \mathbf{v}^n$. Assume only the first p λ_i's are nonzero eigenvalues. Then for any point $\Phi(\mathbf{x})$ the principal components are given by the p projections $< \Phi(\mathbf{x}), \mathbf{v}^j >_{\mathcal{H}}$ for $j = 1, ..., p$. Schölkopf and Smola (2002) introduced an algorithm to perform the kernel PCA efficiently. They showed that the λ_i's are also eigenvalues of $\frac{1}{n}K$, with the corresponding normalized eigenvectors $\boldsymbol{\alpha}^1, \cdots, \boldsymbol{\alpha}^n$. In order to extract the principal components in $\mathcal{H}$ for any test

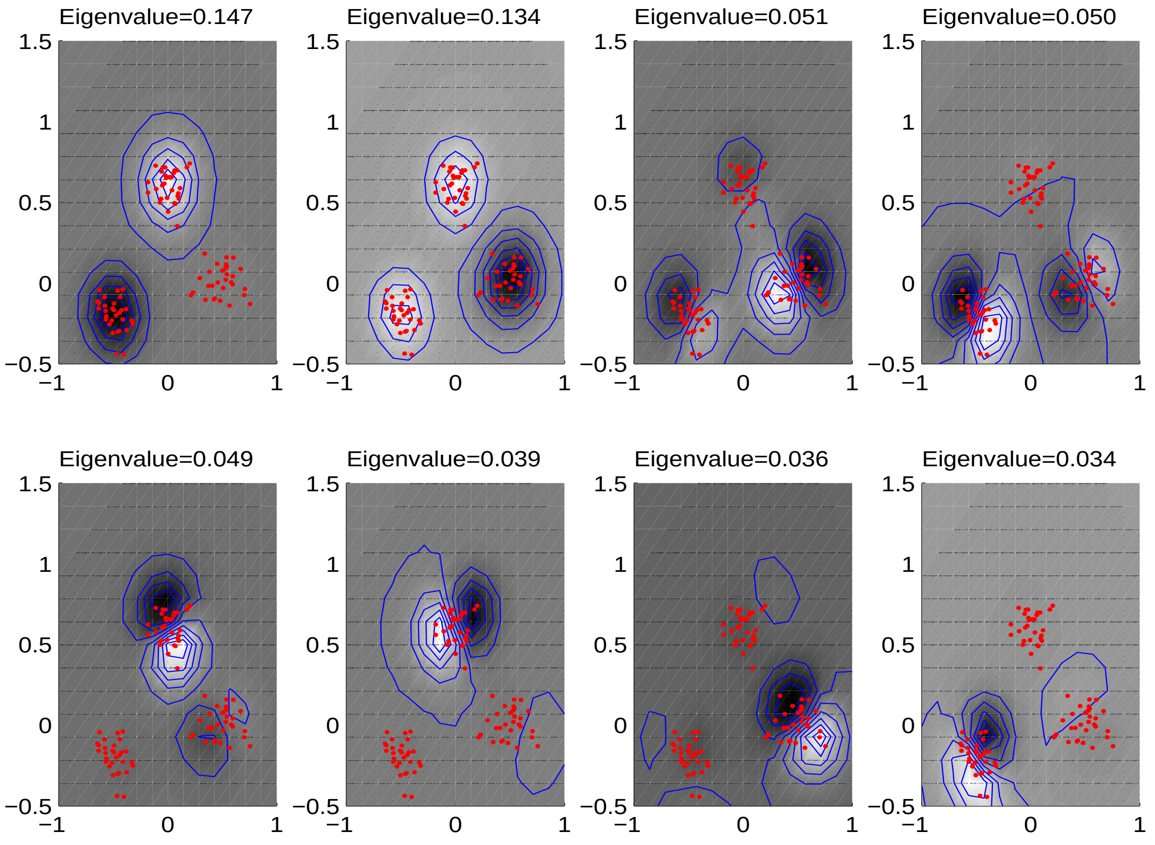

Figure 8: Kernel PCA using compactly supported RBF kernels.

point $\mathbf{x}$, we can compute the projections onto $\mathbf{v}^1, ..., \mathbf{v}^p$ by

$$< \Phi(\mathbf{x}), \mathbf{v}^j >_{\mathcal{H}} = \sum_{i=1}^{n} \alpha_i^j k(\mathbf{x}_i, \mathbf{x}), \quad \text{for } j = 1, ..., p.$$

Example 3: We consider the three-cluster example in Schölkopf and Smola (2002) and conduct the kernel PCA with the compactly supported kernels. Each cluster consists of 30 points generated from a bivariate Gaussian, with the center $[-0.5 \;\; -0.2]^T, [0 \;\; 0.6]^T, [0.5 \;\; 0]^T$ respectively and the common covariance matrix $0.01 I_2$. The optimal cut-off value $\hat{C} = 0.758$ is chosen by maximizing the combination $A(C) + \gamma S(C)$ with $\gamma = 0.3$, and $K_{\hat{C}}$ has the sparsity 0.559 and alignment 0.936. Figure 8 plots the first eight nonlinear principal components extracted using the kernel $k_{\hat{C}}$. We observe that the first two principal components have the largest two eigenvalues and nicely separate the data into three clusters. The remaining six components have small eigenvalues and split the clusters into halves. This result shows similar patterns as in Schölkopf and Smola (2002) where the ordinary Gaussian kernel is used, except that the large eigenvalues of K_C are attenuated due to its sparsity structure.

5. Conclusion and Discussion

We propose a method for systematically sparsifying radial basis function kernels such that the resulting kernel is compactly supported, and hence has a sparse Gram matrix. The resulting sparse property of the Gram matrix can produce crucial advantages when dealing with massive datasets in terms of demanding much less computation time and storage space. We suggest a unified framework to balance the efficiency gain and information loss due to the sparsifying operation, which provides an adaptive tuning approach for the thresholding parameter.

In theory, it is required that the power term $\nu > \frac{d+1}{2}$ in equation (1) to ensure the positive definiteness of the sparsified Gram matrix. This condition is easy to satisfy when d is small or moderate, but can be very restrictive for high dimensional data as in our last example where $d = 256$. Surprisingly, even when we use a small value of $\nu = 3$ or 5 for such datasets, the compactly supported RBF kernel gives comparable classification performances as the traditional support vector machines while enjoying the sparsity advantage. The reason behind this observation will be investigated in the future.

For any positive definite matrix, its eigenvalues (spectrum) and eigenvectors contain much essential information about it. Some of our limited experience suggest that it is possible to describe the similarity between K and K_C based on their spectrum. It is our future work to study how this measure is related to the sparsity of K_C.

Acknowledgments

We would like to acknowledge support for this project from the National Science Foundation grant DMS-0405913 and the Statistical and Applied Mathematical Science Institute (SAMSI) program on Data Mining and Machine Learning during 2003-2004.

References

D. Achlioptas, F. McSherry, and B. Schölkopf. Sampling techniques for kernel methods. *Advances in Neural Information Processing Systems*, 14:335–342, 2002.

B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. *Fifth Annual ACM Workshop on Computational Learning Theory, Pittsburgh, PA*, pages 144–152, 1992.

C.J.C. Burges. Simplied support vector decision rules. In *Proceeding of the 13th International Conference on Machine Learning, Morgan Kaufmann*, pages 71–77, 1996.

N. Cristianini and J. Shawe-Taylor. *Introduction to Support Vector Machines.* Cambridge University Press, 2000.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel target alignment. *Neural Information Processing Systems*, pages 367–373, 2001.

M. S. Floater and A. Iske. Multistep scattered data interpolation using compactly supported radial basis functions. *Journal of Computational and Applied Mathematics*, 73:65–78, 1996.

M. G. Genton. Classes of kernels for machine learning: a statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2001.

A. George and J.W-H. Liu. *Computer Solution of Large Sparse Positive Definite Matrices.* Prentice Hall, 1981.

A. George and J.W-H. Liu. The evolution of the minimum degree algorithm. *SIAM Review*, 31:1–19, 1989.

N. E. Gibbs, W. G. Poole, and P. K. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal on Numerical Analysis*, 13:236–250, 1976.

J. R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in matlab: design and implementation. *SIAM Journal on Matrix Analysis*, 13(1):333–356, 1992.

T. Gneiting. Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83:493–508, 2002.

B. Hamers, J. A. K. Suykens, and B. De Moor. Compactly supported RBF kernels for sparsifying the gram matrix in LS-SVM regression models. *Proceedings ICANN*, pages 720–726, 2002.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, 2001.

R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, New York, NY, 1985.

Y. Lin. SVM and the Bayes rule in classification. *Data Mining and Knowledge Discovery*, 6:259–275, 2002.

E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural networks for signal processing VIII - Proceedings of the 1997 IEEE workshop*, pages 276–285. New York, IEEE, 1997.

T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of IEEE*, 78:1481–1497, 1990.

R. Rifkin, G. Yeo, and T. Poggio. Regularized Least-Squares Classification. In J.A.K. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, editors, *Advances in Learning Theory: Methods, Models and Applications.* IOS Press, Amsterdam, 2003.

R. Schaback. Creating surfaces from scattered data using radial basis functions. *Mathematical Methods in Computer Aided Geometric Design III, M. Dhlen, T. Lyche and L.L. Schumaker (eds.), Vanderbilt Univ. Press*, pages 477–496, 1995.

B. Schölkopf, C. J. C. Burges, and A. J. Smola. *Advances in Kernel Methods - Support Vector Learning.* MIT Press, Cambridge, MA, 1999.

B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.

A. J. Smola, O. L. Mangasarian, and B. Schöelkopf. Sparse kernel feature analysis. In *24th Annual Conference of Gesellschaft fr Klassifikation, University of Passau, Passau, Germany*, 2000.

A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceeding of the 17th International Conference on Machine Learning, Morgan Kaufmann*, pages 911–918, 2000.

J.A.K. Suykens, L. Lukas, P. Van Dooren, B. De Moor, and J. Vandewalle. Least squares support vector machine classifiers: a large scale algorithm. *EFFTD'99 European Conf. on Circuit Theory and Design*, pages 839–842, 1999.

J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Proceeding Letters*, 9(3):293–300, 1999.

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

V. N. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

H. Wendland. Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree. *Advances in Computational Mathematics*, 4:389–396, 1995.

C. K. I. Williams and M. Seeger. Using the Nystrom method to speed up kernel machines. In *Advances in Neural Information Processing Systems.* MIT Press, 2001.