

The Numerical Stability of Kernel Methods

Shawn Martin
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0310
smartin@sandia.gov

November 3, 2005

Abstract

Kernel methods use kernel functions to provide nonlinear versions of different methods in machine learning and data mining, such as Principal Component Analysis and Support Vector Machines. These kernel functions require the calculation of some or all of the entries of a matrix of the form $X^T X$. The formation of this type of matrix is known to result in potential numerical instability in the case of least squares problems. How does the computation of the kernel matrix impact the stability of kernel methods? We investigate this question in detail in the case of kernel PCA and also provide some analysis of kernel use in Support Vector Machines.

Keywords: Kernel Methods, Principal Component Analysis, Support Vector Machines, Numerical Stability

1 Introduction

Kernel functions were originally used by James Mercer in 1909 in the context of integral equations [17]. It wasn't until 1964 that these kernels were interpreted as inner products in feature spaces for use in machine learning [1]. Since the early 1990s, kernels have been applied to Support Vector Machines (SVMs) [3], Principal Component Analysis (PCA) [23], Fisher's Linear Discriminant [18], and a host of other algorithms.

Up to now the numerical stability of kernel-based algorithms has not been addressed. In this paper we show that one such algorithm, kernel PCA [23], is numerically unstable in the case of small eigenvectors. We also provide an example

showing that SVMs can be unstable in the case of order of magnitude differences in vector components.

This paper is organized as follows: in Section 2 we discuss the meaning of numerical stability; in Section 3 we give some background on kernel PCA; in Section 4 we present a simple example which shows that kernel PCA is numerically unstable; in Section 5 we explain why kernel PCA is unstable in general; in Section 6 we discuss SVMs and provide an example showing how SVMs can be unstable; and in Section 7 we offer our conclusions.

2 Numerical Stability

Numerical stability is the study of algorithm performance in terms of accuracy. While a numerically unstable algorithm will produce incorrect results in certain cases, a numerically stable algorithm will produce nearly correct results even in the case of an ill-conditioned problem. We use the definition of numerical stability as described in [27]. Specifically, if we use the notation $g : X \rightarrow Y$ to designate a *problem*, and the notation $\tilde{g} : X \rightarrow Y$ to designate an *algorithm*, then a numerically stable algorithm is an algorithm \tilde{g} such that given $\mathbf{x} \in X$ there exists $\tilde{\mathbf{x}} \in X$ such that

$$\frac{\|\tilde{g}(\mathbf{x}) - g(\tilde{\mathbf{x}})\|}{\|g(\mathbf{x})\|} = O(\epsilon_{\text{machine}}) \quad \text{and} \quad \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} = O(\epsilon_{\text{machine}}), \quad (1)$$

where $O(\epsilon_{\text{machine}})$ decreases in proportion to $\epsilon_{\text{machine}}$.

Many classical algorithms, such as Gaussian elimination and Gram-Schmidt orthogonalization, are numerically unstable. Often these algorithms are avoided by using numerically stable variants such as modified Gram-Schmidt, but just as often the algorithms are used anyway (sometimes due to ignorance) and are unstable only in certain cases. Gaussian elimination, for example, can give completely inaccurate results for certain matrices, but gives accurate results in all known real applications [27]. It is also interesting to note that stability analysis of algorithms can be very difficult. To this day, the rarity of matrices for which Gaussian elimination fails is not fully understood, despite study by such giants as von Neumann, Turing, and Wilkinson [27].

3 Kernel PCA

Principal Component Analysis, closely related to the Singular Value Decomposition (SVD), is a data analysis technique whereby the most variance is captured in the least number of coordinates [12], [14], [27]. PCA was originally investigated

by Pearson in 1901 [20] and Hotelling in 1933 [10] and is now applied in almost every area of scientific investigation.

The widespread use of PCA has spurred interest in the investigation of different nonlinear variants and extensions of PCA, including Hebbian networks [19], multi-layer perceptrons [16], Principal Curves [9] and kernel PCA [23]. In addition, there are other methods in data analysis similar in spirit to PCA such as Projection Pursuit [8], Independent Component Analysis [13], [11], Isomap [26], and Locally Linear Embedding [21].

3.1 Kernels

Kernel PCA is based on the use of Mercer kernel functions. Kernel functions are functions of the form $k : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ such that there exists an accompanying map $\Phi : \mathbb{R}^n \rightarrow F$ with

$$k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}), \Phi(\mathbf{y})), \quad (2)$$

where F is a separable Hilbert space with inner product (\bullet, \bullet) . The map Φ is typically nonlinear and the relation (2) is used as a computational “trick” to avoid explicit computation of $\Phi(\mathbf{x})$. Some simple kernels are the linear kernel $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{y})$, the polynomial kernel $k(\mathbf{x}, \mathbf{y}) = ((\mathbf{x}, \mathbf{y}) + c)^d$, $d \in \mathbb{Z}$ and the gaussian radial basis function kernel $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/2\sigma^2)$, $\sigma \neq 0$.

Kernels are useful in machine learning because they allow the application of linear methods to nonlinear problems. In principle, an appropriate map Φ can be used to change a nonlinear problem in \mathbb{R}^n into a linear problem in F . Once the problem has undergone this transformation, a linear method can be applied.

Kernels come into play because a given nonlinear map Φ usually results in a large and sometimes infinite increase in the dimension of the original problem. Kernels are used to avoid this dimensional increase by replacing the inner products in a linear method with kernels. This substitution yields a nonlinear method which never explicitly uses the map Φ . By replacing inner products $(\mathbf{x}_i, \mathbf{x}_j)$ with kernels $k(\mathbf{x}_i, \mathbf{x}_j) = (\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j))$ we effectively remap our problem using Φ before applying an inner product in a higher dimensional space.

Additional information on kernels and their use in machine learning can be found in [4] and [24].

3.2 PCA

Kernel PCA combines kernel functions with the so-called snapshot method [15], [14] for computing principal components. To describe the snapshot method, suppose we have m data points $\{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subset \mathbb{R}^n$. If we write $p = \min\{m, n\}$

and record our data points in an $m \times n$ matrix $X = (\mathbf{x}_1, \dots, \mathbf{x}_m)$ then PCA is performed by computing the SVD¹

$$X = U\Sigma V^T, \quad (3)$$

where U is an $m \times m$ orthogonal matrix, V is an $n \times n$ orthogonal matrix and Σ is an $m \times n$ matrix with diagonal entries $\sigma_1 \geq \dots \geq \sigma_p \geq 0$. In this case the principal components are usually considered to be the columns of U and the singular values $\sigma_1, \dots, \sigma_p$ give a measure of the variance captured by the corresponding principal components. Typically the projections $U^T X = \Sigma V^T$ are considered most informative.

The snapshot method, originally considered for image analysis [15], refers to computing the SVD by performing an eigenvalue decomposition of

$$X^T X = V\Sigma^2 V^T = Q\Lambda Q^T, \quad (4)$$

in which case the eigenvalues $\lambda_1, \dots, \lambda_p$ correspond to the squares $\sigma_1^2, \dots, \sigma_p^2$ of the singular values and the eigenvectors Q correspond to the right singular vectors V . (If some of the singular values are equal this becomes a correspondence between subspaces.)

3.3 Formulation of Kernel PCA

Kernel PCA is a kernel version of the snapshot method for computing principal components. In other words the inner products in (4) are replaced by kernels to yield a nonlinear version of PCA. To describe kernel PCA more fully, let us consider a parallel development of the snapshot method in the Section 3.2. Suppose we denote $(\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_m))$ by \tilde{X} . We want to decompose \tilde{X} using some type of SVD

$$\tilde{X} = \tilde{U}\tilde{\Sigma}\tilde{V}^T, \quad (5)$$

where \tilde{U} , $\tilde{\Sigma}$, and \tilde{V} have properties similar to the properties possessed by U , Σ , and V in (3). Unfortunately $\Phi(X) \subset F$, where F is either a very high or infinite dimensional vector space so that (5) is either very difficult or impossible to compute.

We can, however, form the matrix $\tilde{X}^T \tilde{X}$ by computing the kernel matrix K , where the entries K_{ij} of K are given by $k(\mathbf{x}_i, \mathbf{x}_j)$. Then we can use the eigenvalue decomposition of K to get

$$\tilde{X}^T \tilde{X} = K = \tilde{V}\tilde{\Sigma}^2\tilde{V}^T = \tilde{Q}\tilde{\Lambda}\tilde{Q}^T. \quad (6)$$

¹Technically PCA is performed by mean subtracting the data X then diagonalizing the covariance matrix $\frac{1}{m} X X^T$. To provide a cleaner exposition, we assume without loss of generality that our data is mean zero and unit covariance.

Thus (6) is the kernel version of the snapshot relation (4) in Section 3.2. The eigenvalues $\tilde{\lambda}_1, \dots, \tilde{\lambda}_p$ correspond to the squares $\tilde{\sigma}_1^2, \dots, \tilde{\sigma}_p^2$ of the singular values and the eigenvectors \tilde{Q} correspond to the right singular vectors \tilde{V} . In particular we can compute the principal component projections $\tilde{U}^T \tilde{X} = \tilde{\Sigma} \tilde{V}^T$ of our remapped data. (These statements will be made precise in Section 5.)

This is not how kernel PCA was presented in [23], but the relations in (5) and (6) motivate the algorithm well: compute the eigenvalues and eigenvectors of K .² As an added benefit, the relations in (5) and (6) make clear why the algorithm is numerically unstable.

4 Example of Kernel PCA Instability

One of the interesting things about kernel PCA is that it is a direct extension of standard PCA. In other words, kernel PCA with a linear kernel $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{y})$ is standard PCA. Since standard PCA has been studied before, the easiest place to consider the numerical stability of kernel PCA is in the linear case.

It is known that PCA, computed using the SVD, is numerically stable. It is also known that computing PCA using an eigenvalue decomposition of the covariance matrix, or of $X^T X$, is numerically unstable [27].

A good example for analyzing numerical stability can be found in [27] on page 65-66. In this example, a matrix $X = U \Sigma V^T$ is considered, where U and V are random orthogonal square matrices of size 80, and Σ is a diagonal matrix with entries $2^{-1}, 2^{-2}, \dots, 2^{-80}$. We use X to show that computing the singular values by diagonalizing $X^T X$ is unstable. As seen in Figure 1(a), computing singular values by diagonalizing $X^T X$ is roughly half as accurate as computing singular values using the SVD.

It might be conjectured that the computation of the right singular vectors by diagonalizing $X^T X$ is stable despite the instability in the computation of the singular values. This is not the case. We see in Figure 1(b) that computation of the right singular vectors breaks down at exactly the same point that the singular value computation breaks down.

5 Explanation of Kernel PCA Instability

We can explain the instability of kernel PCA in the linear case using arguments in [27], and in the nonlinear case by extending those arguments.

²We have omitted the steps of kernel PCA involving mean subtraction and normalization.

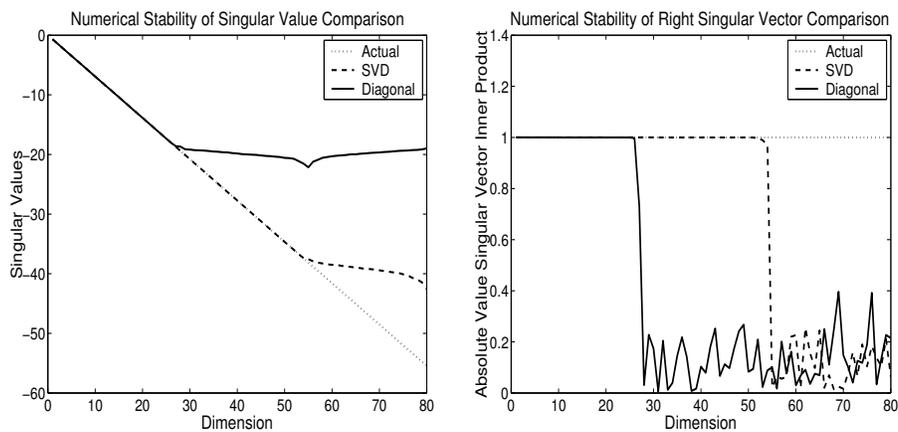


Figure 1: Numerical Stability Comparisons. On the left (a) we compare the ability of SVD versus diagonalization of $X^T X$ for computing singular values. The dotted line shows the actual singular values $2^{-1}, \dots, 2^{-80}$ versus dimension (1-80); the dashed line shows the singular values computed using a numerically stable implementation of the SVD; and the solid line shows the results when the singular values are computed by diagonalizing $X^T X$. On the right (b) we compare abilities for computing right singular vectors. In (b) we display absolute values of inner products (which correspond to angles) between computed and actual right singular vectors. We use the same key for the dotted, dashed, and solid lines as in (a).

5.1 Linear Case

To explain why computing singular values by diagonalizing $K = X^T X$ is unstable we adopt an argument from page 235 of [27]. This argument, presented in Section 5.1.2, uses a simplified version of the Bauer-Fike Theorem [2], which we state in Section 5.1.1.

5.1.1 Bounds

Theorem 1 (Bauer-Fike) *Suppose K is a real symmetric matrix with $K = Q\Lambda Q^T$ and δK is a perturbation of K . If $\bar{\lambda}_j$ is an eigenvalue of $K + \delta K$ then there exists an eigenvalue λ_j of K such that*

$$|\bar{\lambda}_j - \lambda_j| \leq \|\delta K\|_2. \quad (7)$$

Lemma 1 (Extension to Singular Values) *By applying the Bauer-Fike theorem (1) to the matrix*

$$\begin{pmatrix} 0 & X^T \\ X & 0 \end{pmatrix}$$

we get a similar bound on the singular values of X . In particular, if $\bar{\sigma}_j$ is a singular value of $X + \delta X$ then there exists a singular value σ_j of X with

$$|\bar{\sigma}_j - \sigma_j| \leq \|\delta X\|_2. \quad (8)$$

5.1.2 Singular Values versus Eigenvalues

We use the bounds in Section 5.1.1 to compare the results of a stable computation of the singular values of X with a computation of the singular values of X via a stable computation of the eigenvalues of $X^T X$.

Using definition (1), a stable algorithm for computing a singular value σ_j of X has the property that there is a perturbation δX of X and a singular value $\bar{\sigma}_j$ of $X + \delta X$ such that

$$\frac{|\sigma_j - \bar{\sigma}_j|}{|\bar{\sigma}_j|} = O(\epsilon_{\text{machine}}) \quad \text{and} \quad \frac{\|\delta X\|}{\|X\|} = O(\epsilon_{\text{machine}}), \quad (9)$$

where $|f| = O(\epsilon_{\text{machine}})$ means that f decreases in proportion to $\epsilon_{\text{machine}}$ (see [27] for details). Applying (8) to (9) we see that

$$\frac{|\sigma_j - \bar{\sigma}_j|}{|\bar{\sigma}_j|} \leq \frac{\|\delta X\|}{|\bar{\sigma}_j|} = O\left(\frac{\epsilon_{\text{machine}} \|X\|}{|\bar{\sigma}_j|}\right),$$

which implies that

$$|\sigma_j - \bar{\sigma}_j| = O(\epsilon_{\text{machine}} \|X\|). \quad (10)$$

In the case of computing an eigenvalue λ_j of $K = X^T X$ using a stable algorithm we get a similar bound

$$|\lambda_j - \bar{\lambda}_j| = O(\epsilon_{\text{machine}} \|K\|). \quad (11)$$

In order to compare the accuracy of computing singular values directly versus as square roots of eigenvalues of $X^T X$ we compare the bounds in (10) and (11). From (11) we get

$$|\lambda_j - \bar{\lambda}_j| = |\sigma_j^2 - \bar{\sigma}_j^2| = |\sigma_j + \bar{\sigma}_j| |\sigma_j - \bar{\sigma}_j|,$$

which gives

$$|\sigma_j - \bar{\sigma}_j| = O\left(\frac{\epsilon_{\text{machine}} \|K\|}{\sigma_j}\right). \quad (12)$$

Since $\|K\| = \|X^T X\| = \|X\|^2$ we see that (12) is less accurate than (10) by the factor $\frac{\|X\|}{\sigma_j} = \frac{\sigma_1}{\sigma_j}$. For singular values similar in magnitude to σ_1 this is not a problem but for small singular values σ_j this computation becomes unreliable.

In particular, it is interesting to revisit the example in Section 4 in the context of our newly derived bounds. If we assume that $\epsilon_{\text{machine}} \approx 2^{-50}$ for double precision then we see that our expected accuracy for computing singular values directly is $O(\epsilon_{\text{machine}} \|X\|) = O(\epsilon_{\text{machine}} \sigma_1) = O(2^{-51})$. This agrees well with Figure 1(a). Our expected accuracy for computing singular values by diagonalizing $X^T X$, on the other hand, is $O(\epsilon_{\text{machine}} \frac{\sigma_1^2}{\sigma_j}) = O(2^{j-52})$. Thus for $\sigma_j = 2^{-26}$ we have an expected accuracy of $O(2^{-26})$. We can't do any better than this and in fact the worst case $\sigma_j = 2^{-80}$ has an expected accuracy of $O(2^{28})!$ From this perspective our eigenvalue decomposition in Figure 1(a) does a remarkably good job.

5.1.3 Singular Vectors and Eigenvectors

In Figure 1(b) we saw that computing the right singular vectors by diagonalizing $X^T X$ was unstable. This is true in general because the computation of eigenvectors is ill-conditioned [7], [22]. In particular an eigenvector calculation is very sensitive to the gaps between eigenvalues: a small gap yields inaccurate computations. In our case this means that our eigenvector calculations will become inaccurate when the eigenvalues reach maximum effective precision. Thus the instability in computing the singular values by diagonalizing $X^T X$ carries over into the computation of the right singular vectors.

5.2 Nonlinear Case

In this section we show that kernel PCA is unstable in the nonlinear case. We accomplish this by straightforward direct extension of the results in Sections 3.2 and 5.1.

We first define $\tilde{X} : \mathbb{R}^m \rightarrow F$ by $\tilde{X}\mathbf{v} = v_1\Phi(\mathbf{x}_1) + \dots + v_m\Phi(\mathbf{x}_m)$ and $\tilde{X}^T : F \rightarrow \mathbb{R}^m$ by $\tilde{X}^T\mathbf{u} = ((\Phi(\mathbf{x}_1), \mathbf{u}), \dots, (\Phi(\mathbf{x}_m), \mathbf{u}))$. Both \tilde{X} and \tilde{X}^T are linear and the adjoint \tilde{X}^* of \tilde{X} is \tilde{X}^T .

Using these definitions and the standard induction proof (see e.g. [27]) we get the kernel version of the SVD previously mentioned in Section 3.3 as Equation (5).

Theorem 2 (Kernel SVD) *Suppose \tilde{X} is defined as above and p is the dimension of the range of \tilde{X} . Then there exists a decomposition*

$$\tilde{X} = \tilde{U}\tilde{\Sigma}\tilde{V}^T \quad (13)$$

such that \tilde{V} is an orthogonal $m \times m$ matrix, $\tilde{\Sigma}$ is a $p \times m$ diagonal matrix with entries $\sigma_1 \geq \dots \geq \sigma_p \geq 0$, and $\tilde{U} : \mathbb{R}^p \rightarrow F$ is a unitary operator. Furthermore the singular values $\sigma_1, \dots, \sigma_p$ are uniquely determined as are the subspaces associated with equal singular values. In particular a distinct singular value determines (up to sign) the associated singular vectors.

The Bauer-Fike bound on singular values in (8) can also be extended. By applying the standard Bauer-Fike theorem to the matrix

$$\begin{pmatrix} 0 & \tilde{X}^T\tilde{U} \\ \tilde{U}^T\tilde{X} & 0 \end{pmatrix}$$

we get a bound on the singular values of \tilde{X} . In particular if $\bar{\sigma}_j$ is a singular value of $\tilde{X} + \delta\tilde{X}$ then there is a singular value σ_j of \tilde{X} with

$$|\bar{\sigma}_j - \sigma_j| \leq \|\delta\tilde{X}\|. \quad (14)$$

Using kernel SVD (13) and the Bauer-Fike bound (8) the argument in Section 5.1.2 carries over directly to the nonlinear case.

6 Support Vector Machines

We have shown that kernel PCA is numerically unstable, and that the instability arises from the computation of the singular values of \tilde{X} as the square roots of the eigenvalues of the kernel matrix $K = \tilde{X}^T\tilde{X}$. Unfortunately, the formation of the

kernel matrix is central to kernel PCA, and a stable alternative is not immediately apparent. In fact, the formation of the kernel matrix is central to all kernel methods. What does this imply about the stability of these kernel methods, including the stability of SVMs? Although we do not answer this question in general, we present here a simple example with which we investigate the stability of SVMs.

Support Vector Machines are classifiers (there are also SVMs which perform regression [25]) which were originally developed in [28], [3], [5], among others. SVMs are based on a maximal margin hyperplane, which was originally presented in [28], and use nonlinear kernel functions, originally suggested in [3]. SVMs are able to handle errors using the soft margin generalization, first proposed in [5]. Due to their use of nonlinear kernel functions, SVMs are very adaptable (able to assume polynomial, radial basis function, and neural network forms) and have been applied successfully to a wide variety of problems. For an introduction to SVMs see [4], [6], [29].

In the case of a SVM, we have a dataset $\{\mathbf{x}_1, \dots, \mathbf{x}_m\} \subset \mathbb{R}^n$ with class labels $\{y_1, \dots, y_m\} \subset \{-1, +1\}$. If we assume that the two classes are linearly separable, then our SVM assumes the form $f(\mathbf{x}) = \text{sign}((\mathbf{w}, \mathbf{x}) - b)$, where \mathbf{w} and b are computed by solving the quadratic programming problem

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m y_i y_j \alpha_i \alpha_j (\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^m \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^m y_i \alpha_i = 0 \\ & \alpha_i \geq 0 \text{ for } i = 1, \dots, m. \end{aligned} \tag{15}$$

If $\alpha_1^*, \dots, \alpha_m^*$ is the solution to (15) then $\mathbf{w} = \sum_{i=1}^m y_i \alpha_i^* \mathbf{x}_i$ and $b = (\mathbf{w}, \mathbf{x}_i) - y_i$, independent of i , assuming $\alpha_i^* > 0$.

To illustrate the potential instability of the SVM calculation, we now consider the dataset $\{\mathbf{x}_0 = \mathbf{0}, \mathbf{x}_1 = \sigma_1 \mathbf{e}_1, \dots, \mathbf{x}_m = \sigma_m \mathbf{e}_m\} \subset \mathbb{R}^m$ with class labels $\{y_0 = -1, y_1 = \dots = y_m = 1\}$, where $\mathbf{e}_1, \dots, \mathbf{e}_m$ is the standard basis for \mathbb{R}^m , and $\sigma_1, \dots, \sigma_m$ are chosen to be positive. For this dataset, the SVM quadratic program in (15) reduces to

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^m \alpha_i^2 \sigma_i^2 - 2 \sum_{i=1}^m \alpha_i \\ \text{s.t.} \quad & \alpha_i \geq 0 \text{ for } i = 1, \dots, m, \end{aligned} \tag{16}$$

where $\alpha_0 = \sum_{i=1}^m \alpha_i$. The reduced program in (16) has solution $(\alpha_0^*, \frac{2}{\sigma_1^2}, \dots, \frac{2}{\sigma_m^2})$ with $\mathbf{w} = (\frac{2}{\sigma_1}, \dots, \frac{2}{\sigma_m})$ and $b = 1$.

Now let $\sigma_i = 2^{-i}$ for $i = 1, \dots, 80$. In this case, the solution to (15) is given by $\alpha_0^* = 2 \sum_{i=1}^{80} (2^i)^2$ and $\alpha_i^* = 2 \times (2^i)^2$ for $i = 1, \dots, 80$ with $\mathbf{w} = 2(2^1, \dots, 2^{80})$

and $b = 1$. Because both α_0 and $\alpha_1, \dots, \alpha_{80}$ are included in the calculation of (15), we observe that the constraint $\alpha_0 = \sum_{i=1}^{80} \alpha_i$ implies a strict limit on the precision of the results. In particular, $\alpha_0^* = 2 \sum_{i=1}^{80} (2^i)^2$ implies that we are limited to approximately half of the available machine precision, exactly analogous to our example in Section 4, when we used diagonalization to compute the eigenvalues of $X^T X$. Furthermore, we emphasize that this observation holds regardless of the particular algorithm used to solve (15).

7 Conclusions

In this paper we showed that kernel PCA is numerically unstable. This instability arises from the computation of the singular values of \tilde{X} as the square roots of the eigenvalues of the kernel matrix $K = \tilde{X}^T \tilde{X}$. Unfortunately, the formation of the kernel matrix is central to kernel PCA and a stable alternative is not immediately apparent.

On the other hand, in most applications of kernel PCA we are only interested in the top few statistically significant singular vectors. Thus the fact that it is difficult to compute the smaller singular values is unimportant from a practical point of view.

We also discussed how the formation of the kernel matrix K affects the computation of a SVM. We provided an example where the formation of K results in numerical instability, in the case of linearly separable data.

Again, however, our example was artificial. In most real-world problems, it would probably not occur that one measurement was several orders of magnitude different than another. Further, if in fact there was such a difference between measurements, the practitioner would benefit by scaling the measurements so that they were of similar magnitudes. Finally, the practitioner would not use a separable SVM, but would instead use the soft margin generalization, which incorporates a regularization term. It may be that the soft margin generalization is stable.

Thus the practical implications of our analysis seem rather limited: it may be unwise to trust kernel methods beyond eight digits of accuracy, and only for artificial examples. As discussed in Section 2, however, numerically unstable methods can often be used successfully, and only occasionally propagate errors out of control, giving entirely incorrect solutions. To avoid this phenomenon, we need to better understand the stability of kernel methods. We have shown here that certain kernel methods can be unstable in certain cases. If kernel methods are unstable in general, and if and when these instabilities occur in practice, are more important but much more difficult questions.

8 Acknowledgements

This work was funded by the Office of Advanced Scientific Research Mathematics Information and Computer Science Division, and by Sandia National Laboratories Laboratory Directed Research and Development. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

References

- [1] M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.
- [2] F. L. Bauer and C. T. Fike. Norms and exclusion theorems. *Numerische Mathematik*, 2:137–141, 1960.
- [3] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.
- [4] C. J. C. Burges. Geometry and invariance in kernel based methods. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 89–116, Cambridge, MA, 1999. MIT Press.
- [5] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.
- [6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. Cambridge University Press, 2000.
- [7] J. Demmel. A brief tour of eigenproblems (chapter 2). In Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, Philadelphia, 2000.
- [8] J. H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82:249–266, 1987.
- [9] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84:502–516, 1989.

- [10] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441 and 498–520, 1933.
- [11] Aapo Hyvärinen. Survey on Independent Component Analysis. *Neural Computing Surveys*, 2:94–128, 1999.
- [12] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.
- [13] C. Jutten and J. Herault. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture, 1991.
- [14] M. Kirby. *Geometric Data Analysis*. John Wiley & Sons, 2001.
- [15] M. Kirby and L. Sirovich. Application of the Karhunen-Loève procedure for the characterization of human faces. *IEEE Trans. PAMI*, 12(1), January 1990.
- [16] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37:233–243, 1991.
- [17] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.
- [18] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.
- [19] E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.
- [20] K. Pearson. On lines and planes of closest fit to systems of points in space. *Phil. Mag.*, 2:559–572, 1901.
- [21] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 22 Dec. 2000.
- [22] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Halstead Press, 1992. See Y. Saad’s homepage at the University of Minnesota, Department of Computer Science and Engineering, <http://www-users.cs.umn.edu/saad>.
- [23] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 327–352. MIT Press, Cambridge, MA, 1999.

- [24] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
- [25] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.
- [26] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 22 December 2000.
- [27] L. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997.
- [28] V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).
- [29] V. Vapnik. *Statistical Learning Theory*. Wiley Interscience, New York, 1998.