# The Numerical Stability of Kernel Methods

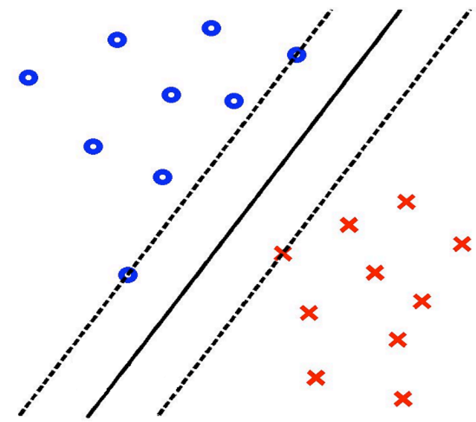
Shawn Martin

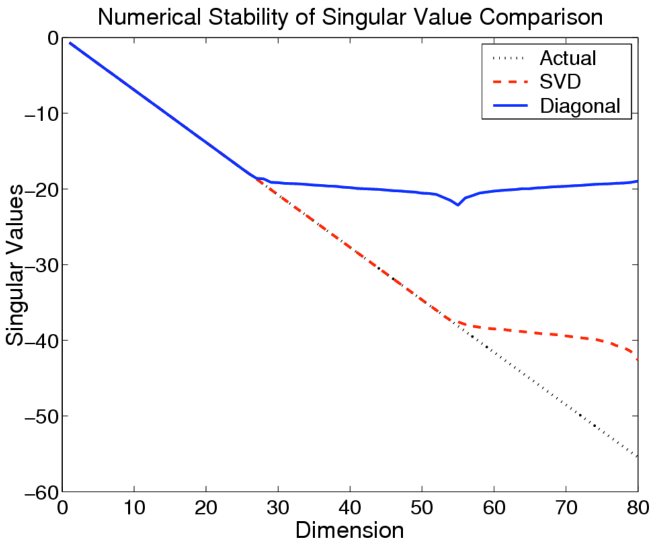
Sandia National Laboratories
Albuquerque, NM, USA

Dec. 15th, 2005

# Outline of Talk

- Kernel Methods
  - Background/Examples
- Numerical Stability
  - Background/Examples
- Stability Analysis
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Support Vector Machines
- Conclusions





# Support Vector Machines (SVMs)

A Support Vector Machine is the prototypical example of a kernel method in machine learning.

Given a dataset $\{(\mathbf{x}_i, y_i)\} \subseteq \mathbb{R}^n \times \{\pm 1\}$

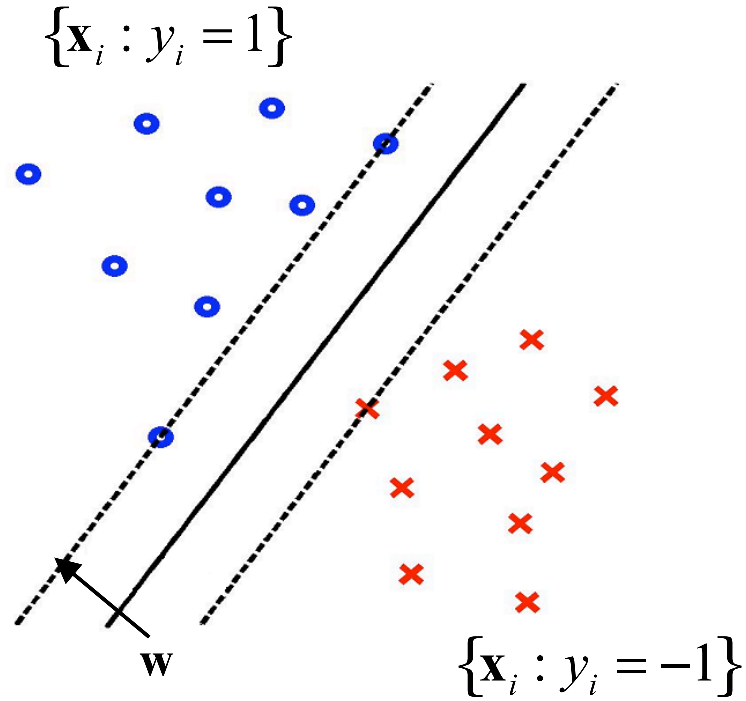
We solve the quadratic problem

$$\max \sum_i \alpha_i - \tfrac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \; \sum_i y_i \alpha_i = 0$$

to obtain the normal to the separating hyperplane

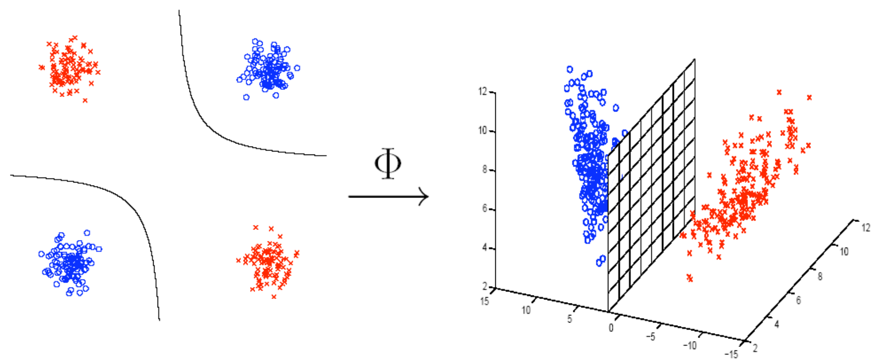
$$\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i$$




(Support Vectors are $\mathbf{x}_i$ such that $\alpha_i \neq 0$, shown as lying on dashed lines.)

# The Kernel "Trick"

If the data is not linearly separable we can map the dataset into a higher dimensional space using a nonlinear map $\Phi : \mathbb{R}^n \rightarrow F$ before solving the linear problem.



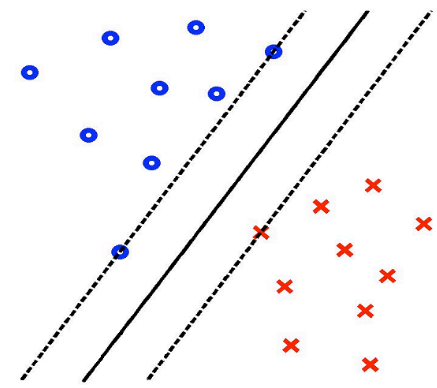

$$\Phi(x, y) = (x^2, \sqrt{2}xy, y^2)$$

This is accomplished by replacing the inner products $(\mathbf{x}_i, \mathbf{x}_j)$ in the SVM problem with a kernel function, where a kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ such that

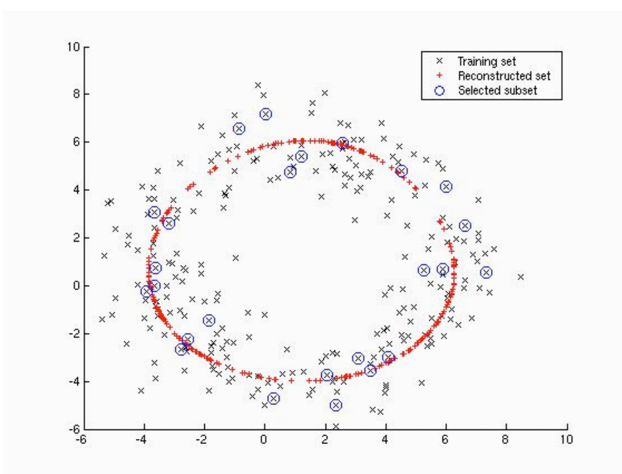
$$k(\mathbf{x}_i, \mathbf{x}_j) = (\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)).$$

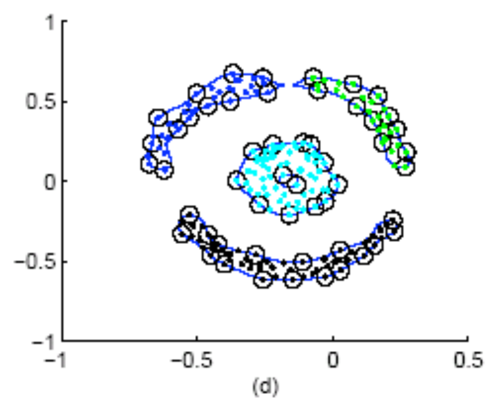
Any method which can be written in terms of $(\mathbf{x}_i, \mathbf{x}_j)$ so that kernel functions can be used is called a kernel method.

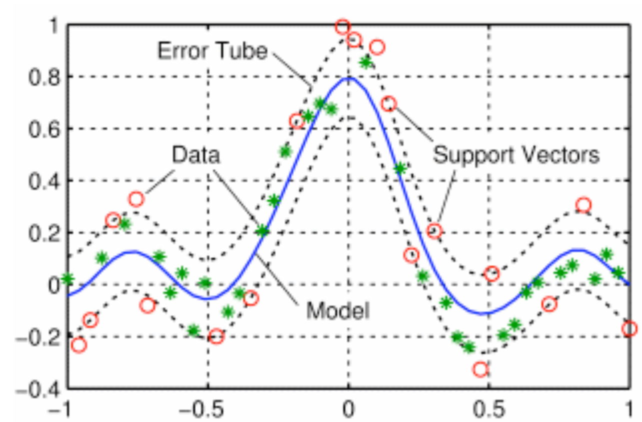
# Examples of Kernel Methods



- Support Vector Machines
  - SV Classification
  - SV Regression
  - SV Clustering
- Kernel Principal Component Analysis
- Kernel Fisher's Discriminant Analysis



# Numerical Stability

We use the definition of numerical stability from the field of Scientific Computing/Numerical Analysis.

**Definition**: If $g : X \to Y$ is a problem and $\tilde{g} : X \to Y$ is an algorithm, then $\tilde{g}$ is *numerically stable* if for every $\mathbf{x} \in X$ there exists $\tilde{\mathbf{x}} \in X$ such that

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

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

> *"A stable algorithm gives nearly the right answer to nearly the right question."* (Trefethen & Bau, 1997).

# Stability vs. Conditioning

- An algorithm can be stable or unstable.
  - Stable: small changes in input result in small changes in output.
  - Unstable: small changes in input can result in large changes in output.
- Similarly, a problem can be well- or ill-conditioned.
  - Well-conditioned: small changes in problem give small changes in solution.
  - Ill-conditioned: small changes in problem can give large changes in solution.

<u>Worst Case Scenarios</u>

| | stable | unstable |
|---|---|---|
| well-conditioned | good | bad |
| ill-conditioned | bad | bad |

# Example of Numerical Instability

Suppose we are solving $A\mathbf{x} = \mathbf{b}$, where

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow \mathbf{x} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

This problem is well-conditioned ($\kappa \approx 2.6$). If we perturb $A$ we get

$$\tilde{A} = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow \tilde{\mathbf{x}} = \begin{bmatrix} -1/ 1-10^{-20} \\ 1/ 1-10^{-20} \end{bmatrix} \approx \mathbf{x}$$

If we use Gaussian elimination with Pivoting (stable) we get

$$\tilde{P}\tilde{A} = \tilde{L}\tilde{U} = \begin{bmatrix} 1 & 0 \\ 10^{-20} & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \Rightarrow \tilde{\mathbf{x}} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \mathbf{x}$$

If we use Gaussian elimination without pivoting (unstable) we get

$$\tilde{A} = \tilde{L}\tilde{U} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix} \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1-10^{-20} \approx -10^{-20} \end{bmatrix} \Rightarrow \tilde{\mathbf{x}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \neq \mathbf{x}$$

# Some Unstable Algorithms

- Matrix inversion using determinants.
- Gaussian elimination w/o pivoting.
- Least squares by normal equations

$$A^T A\mathbf{x} = A^T \mathbf{b}.$$

- Eigenvalues as roots of the characteristic polynomial.
- Principal Component Analysis by diagonalizing

$$X^T X.$$

# Basic Idea of this Work

- When an algorithm uses $M^T M$ it tends to be unstable.
  - Least squares by $A^T A\mathbf{x} = A^T \mathbf{b}$.
  - PCA by $X^T X$.
- Kernel methods use kernel function evaluation $k\left(\mathbf{x}_i, \mathbf{x}_j\right)$, equivalent to $X^T X$ in kernel space.
- Are kernel methods unstable?

# Principal Component Analysis (PCA)

Principal Component Analysis is (roughly) a matrix factorization

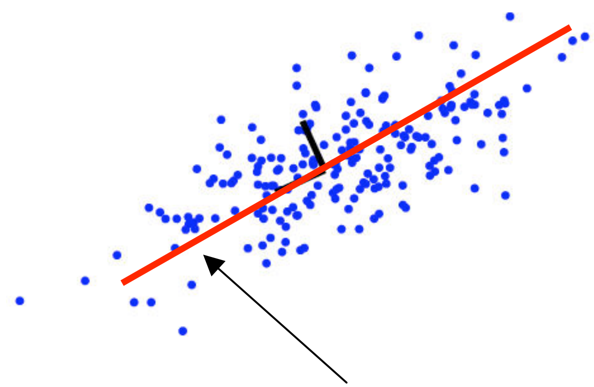
$$X = U \Sigma V^T,$$

where

- $X$ is the data matrix,
- $U, V$ are orthogonal matrices,
- $\Sigma$ is a diagonal matrix with

$$\sigma_1 \geq \ldots \geq \sigma_p \geq 0,$$

- the projections $U^T X = \Sigma V^T$ capture the most variance in the least number of coordinates.



# Snapshot Method for PCA

The snapshot method computes the decomposition

$$X = U\Sigma V^T,$$

by diagonalizing

$$X^T X = V\Sigma^2 V^T,$$

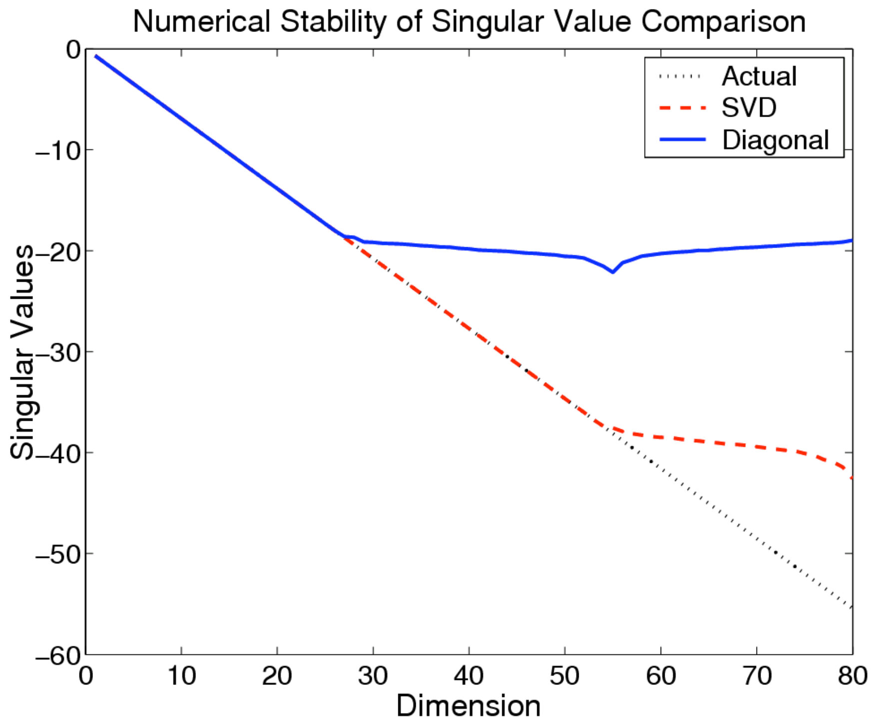
so that the eigenvectors $X^T X$ give $V$ and the eigenvalues of $X^T X$ give the squares of the singular values

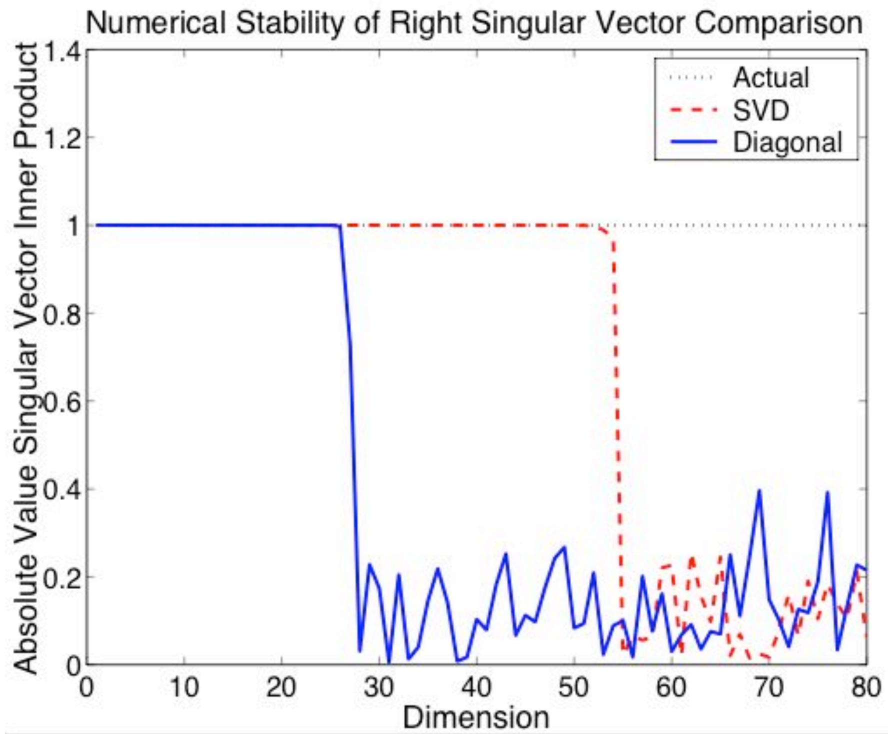
$$\sigma_1^2 \geq \ldots \geq \sigma_p^2,$$

(Name snapshot originates from image processing.)

# Stability of PCA

PCA is stable when computed using the SVD but is unstable when computed using the snapshot method (diagonalizing $X^TX$).



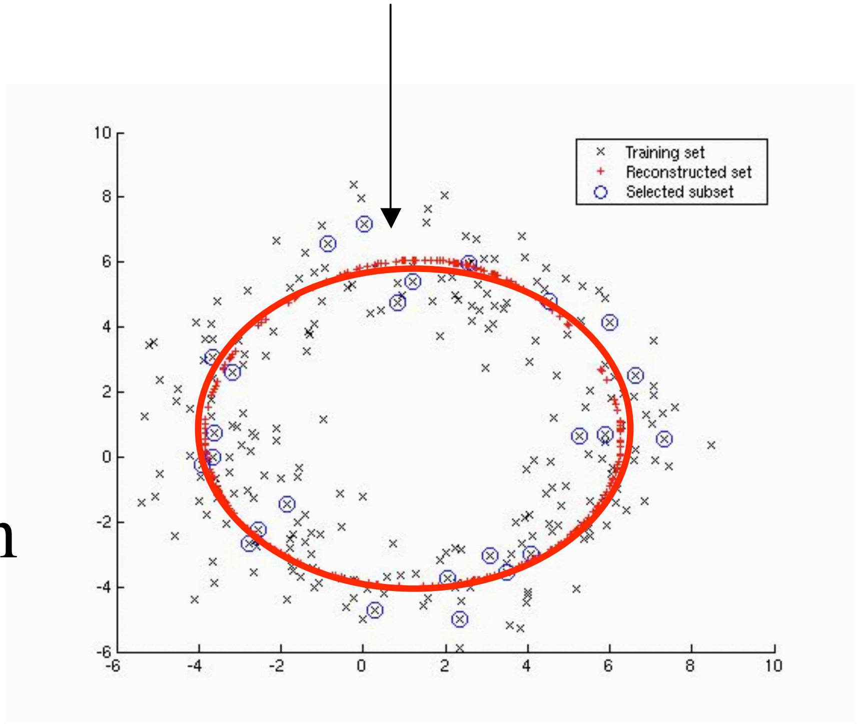

The instability boils down to the fact that $\|X^TX\| = \|X\|^2$, so that computing $\sigma_1^2, \ldots, \sigma_p^2$ instead of $\sigma_1, \ldots, \sigma_p$ results in a loss of accuracy.

# Kernel PCA (kPCA)

Kernel PCA uses the snapshot method in the re-mapped space.

If the re-mapped data $\Phi(X)$ is denoted $\tilde{X}$ then $\tilde{X}^T\tilde{X}$ is the kernel matrix, with entries $k(\mathbf{x}_i,\mathbf{x}_j)$ so that we can obtain $V^T\Sigma^2 V$ by diagonalization.

First kPC Projection



# Stability of kPCA

- Kernel PCA is computed using the snapshot methods so is unstable in the linear case.
  - Apply Bauer-Fike bound on eigenvalues

$$\left|\bar{\lambda}_j - \lambda_j\right| \le \|\delta K\|_2 .$$

  - Compare bounds on $X$ with bounds on $X^T X$.
- Kernel PCA is unstable in the nonlinear case by extension
  - Extend Bauer-Fike to $\tilde{X}$.
  - Compare bounds on $\tilde{X}$ with bounds on $\tilde{X}^T \tilde{X}$.

# Stability of SVMs (I)

Q. SVMs use the matrix $\tilde{X}^T \tilde{X}$. Are they unstable?

A. Yes. Suppose our dataset is given by

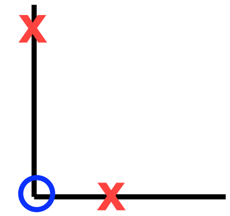
$$\{\mathbf{x}_0 = \mathbf{0}, \mathbf{x}_1 = \sigma_1 \mathbf{e}_1, \ldots, \mathbf{x}_m = \sigma_m \mathbf{e}_m\} \subseteq \mathbb{R}^m$$

$$\{y_0 = -1, y_1 = 1 = \cdots y_m = 1\}.$$

In this case the SVM problem becomes

$$\min_\alpha \ \tfrac{1}{2} \sum_{\mathrm{i}=1}^m \alpha_i^2 \sigma_i^2 \ - \ 2 \sum_i \alpha_i$$

$$\text{s.t.} \quad \alpha_i \geq 0 \text{ for } i = 1, \ldots, m,$$



where $\alpha_0 = \sum_{i=1}^m \alpha_i$.

# Stability of SVMs (II)

The reduced SVM problem has solution $\left(\alpha_0^*, \frac{2}{\sigma_1^2}, \ldots, \frac{2}{\sigma_m^2}\right)$

with $\alpha_0^* = \sum_{i=1}^{m} \frac{2}{\sigma_i^2}, \mathbf{w} = \left(\frac{2}{\sigma_1}, \ldots, \frac{2}{\sigma_m}\right)$, and $b = 1$.

Now let $\sigma_i = 2^{-i}$ for $i = 1,\ldots,80$. In this case, the solution is given by $= 2\sum_{i=1}^{80}\left(2^i\right)^2$, and $\alpha_i^* = 2\left(2^i\right)^2$ for $i = 1,\ldots,80$ with $\mathbf{w} = 2\left(2^1,\ldots,2^{80}\right)$ and $b = 1$.

The fact that $\alpha_0^* = \sum_{i=1}^{\mathrm{m}} \alpha_i^*$ implies a limit on the precision of the results.

# Conclusions

- Algorithms which use $X^T X$ are often numerically unstable
  - Least squares by normal equations,
  - PCA by solving eigenvalue problem.
- Kernel methods implicitly use $X^T X$. Are they unstable?
  - In two cases: kernel PCA, separable SVMs.
- On the other hand:
  - kPCA is only unstable for small singular vectors (often considered to be noise).
  - SVM example is artificial and does not use regularization.
- In practice, kernel methods have *potential* stability problems. However, further work needs to be done:
  - Are there any real applications where instability can be observed?
  - Does regularization/scaling fix these problems?