BIN504 - Lecture IX

Support Vector Machines and the Kernel Trick

References:
Lee, Chp 6

Rev. 1.1 (Build 20130531231600)

Outline

- Boundary Classifiers
- Linear Support Vector Machines
- Soft Margins
- The Kernel Trick
  - Mercer's Theorem
  - Mercer Kernels
- Kernel PCA

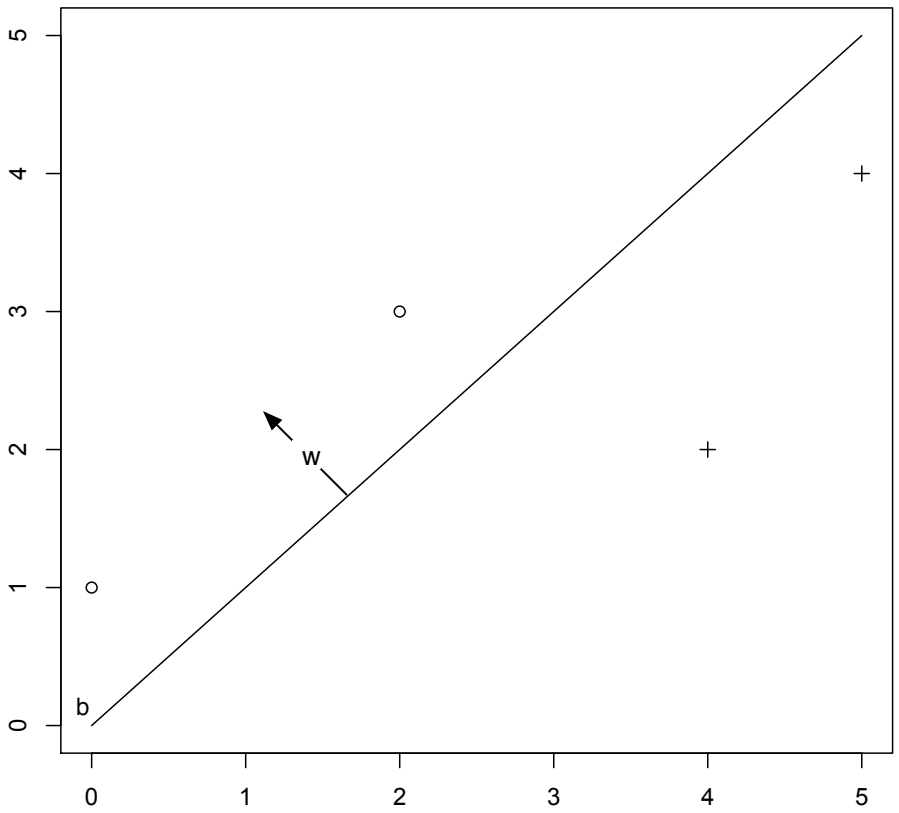
Supervised Learning

- In clustering we looked at unsupervised learning.
- Remember that in classification, there is a training set consisting of n samples:
  - data points $\mathbf{x}^{(i)}$ $i = 1 \ldots n$
    - Each $\mathbf{x}^{(i)}$ is vector of attribute values $[x_1^{(i)}, x_2^{(i)}, \ldots, x_m p^{(i)}]$
    - p being the number of dimensions
  - class labels $y^{(i)}$ $i = 1 \ldots n$
    - $y^{(i)}$ denote the class label.
    - For this lecture, we only consider binary classes.
    - $y^{(i)} = -1$ or 1
- This is supervised learning
  - The model is created to using previous data, i.e. the training set
- We generate a model h which predicts y given $\mathbf{x}$
- For simplicity we assume real attributes only:
  - $h : \mathbb{R}^p \mapsto \{-1, 1\}$

Classifiers

- Many classifiers exist
  - k-Nearest Neighbor
  - Decision Trees
  - Linear Discriminant Analysis
  - Bayesian, naïve and otherwise
  - ⋮
- Margin classifiers, particularly Support Vector Machines
  - Less affected by the curse of dimensionality
  - Fast training and operation
  - Good *one-size-fits-all* classifier

Decision Boundary



Define the decision boundary as:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

For example, $\mathbf{w} = [-1, 1]$:

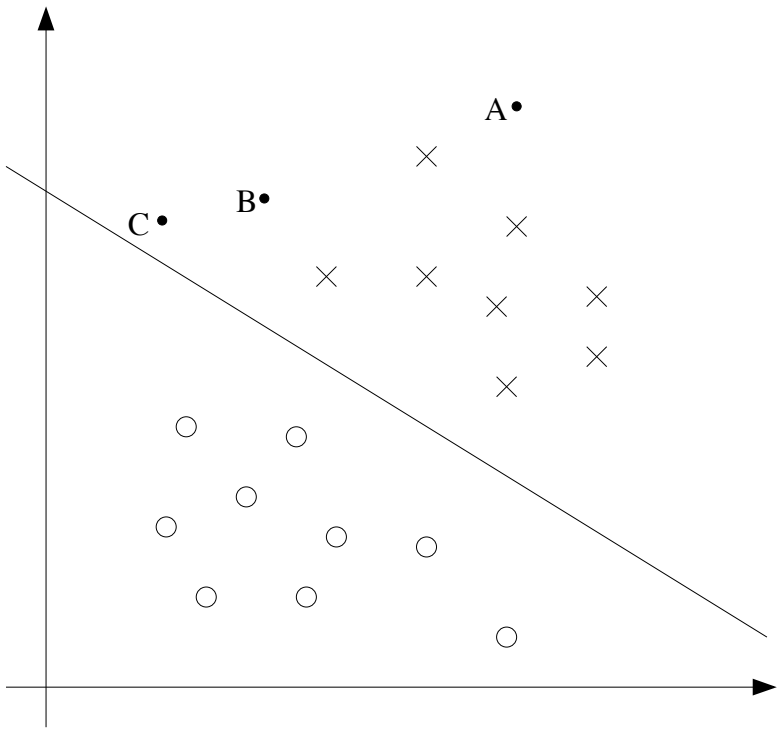
- $x^{(1)} = [0, 1]$
  $\mathbf{w}^T\mathbf{x}^{(1)} = 1$
- $x^{(2)} = [2, 3]$
  $\mathbf{w}^T\mathbf{x}^{(2)} = 1$
- $x^{(3)} = [4, 2]$
  $\mathbf{w}^T\mathbf{x}^{(2)} = -2$
- $x^{(4)} = [5, 4]$
  $\mathbf{w}^T\mathbf{x}^{(2)} = -1$

So, the model:

$$h(\mathbf{x}) = \begin{cases} -1, & \text{if } \mathbf{w}^T\mathbf{x} + b < 0 \\ 1, & \text{if } \mathbf{w}^T\mathbf{x} + b \geq 0 \end{cases}$$

Intuition

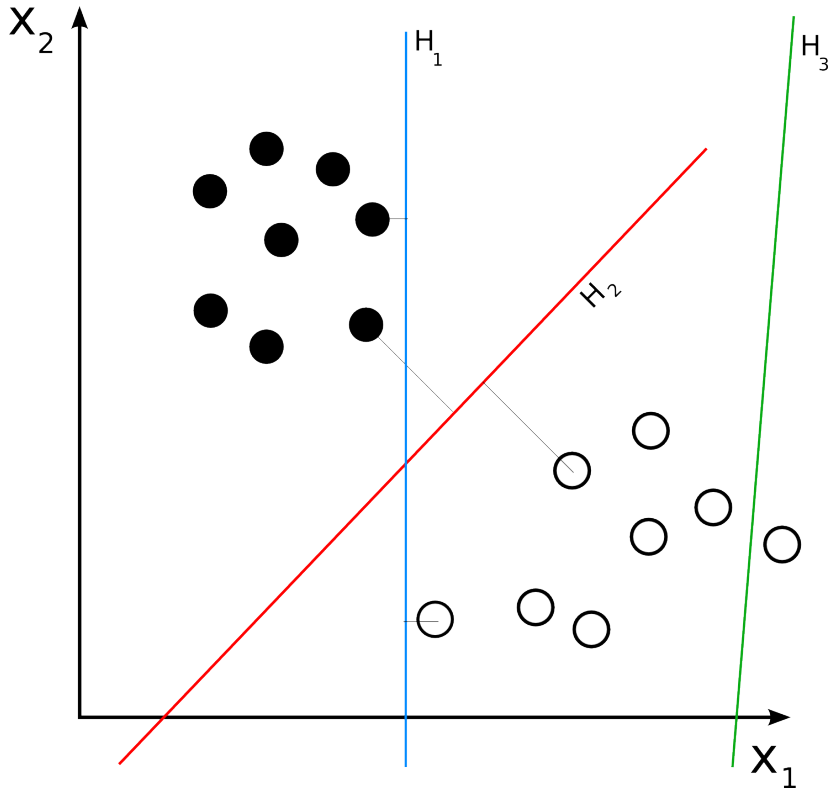
Consider the following case with the decision boundary shown:



- For which point (A,B,C) would you be most confident in your decision?
- Larger the margin the better.

Intuition (2)

Consider the following alternative models:



- Which model (H_1, H_2, H_3) is preferable?
- One with the highest accuracy and largest margin

Optimal Margin Classification

- A margin classifier constructs a boundary hyperplane
  - 2D $\rightarrow$ line, 3D $\rightarrow$ plane, 4+D $\rightarrow$ hyperplane
- The boundary hyperplane has at least as many dimensions as the data
  - Possibly more or even infinite dimensional
  - Later, in kernels...
- The boundary that maximizes the margin is found
  - Better generalization

Linear SVM

- The boundary being described as:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

- The margins may be:

$$\mathbf{w}^T\mathbf{x} + b = \pm 1$$

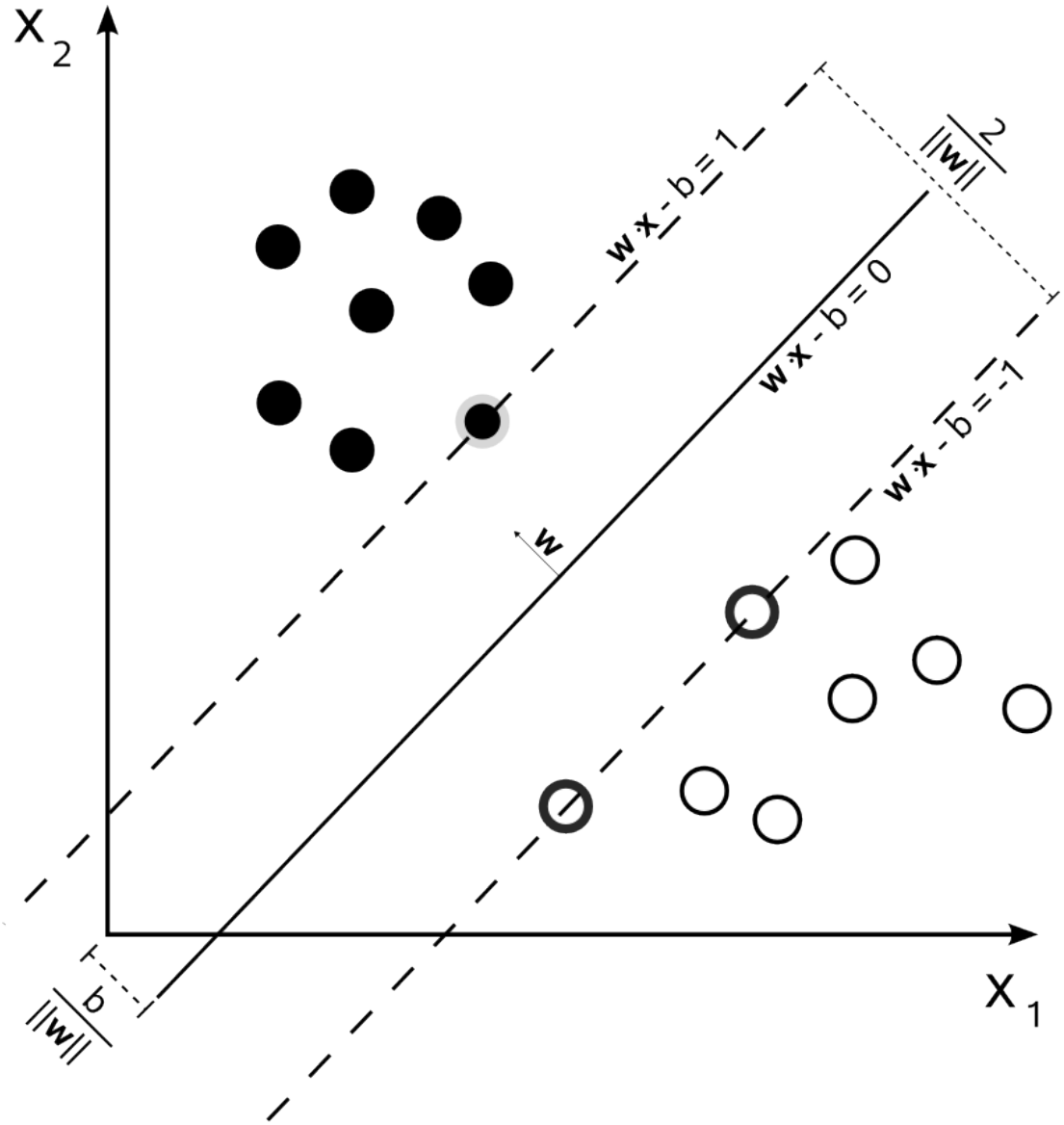
so for some sample $\mathbf{x}^{(i)}$:

$$h(\mathbf{x}^{(i)}) = \begin{cases} 1, & \text{if } \mathbf{w}^T\mathbf{x}^{(i)} + b \geq 1 \\ -1, & \text{if } \mathbf{w}^T\mathbf{x}^{(i)} + b \leq -1 \end{cases}$$

in other words:

$$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1$$

Linear SVM



Training the SVM

- We want to maximize $\frac{2}{||\mathbf{w}||}$
- So, minimize $||\mathbf{w}||$
  - Except, $||\mathbf{w}|| = \sqrt{\mathbf{w}^T\mathbf{w}}$
  - The square root makes it a nasty optimization problem.
  - Minimize $||\mathbf{w}||^2$ instead...
- Minimize

$$\frac{1}{2}||\mathbf{w}||^2$$

  s.t. for all points $i = 1, \ldots, n$,

$$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1$$

- So, again we have a constrained optimization problem.
  - Just like PCA...
  - Lagrange to the rescue!

Training the SVM (2)

Introducing Lagrange multipliers α_i the problem becomes:

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha} \geq 0} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) - 1] \right\}$$

- We are essentially looking for a saddle point.
- Find the αs that maximize within the minimum.
  - In the solution, most points will have $\alpha_i = 0$
    - These are the 'internal' points, far from the boundary
  - Non-zero α_i's correspond to $\mathbf{x}^{(i)}$'s on the margins (i.e. support vectors)
- The problem can now be easily solved with quadratic programming.

Optimization Results

- It turns out that the solution is:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y^{(i)} \mathbf{x}^{(i)}$$

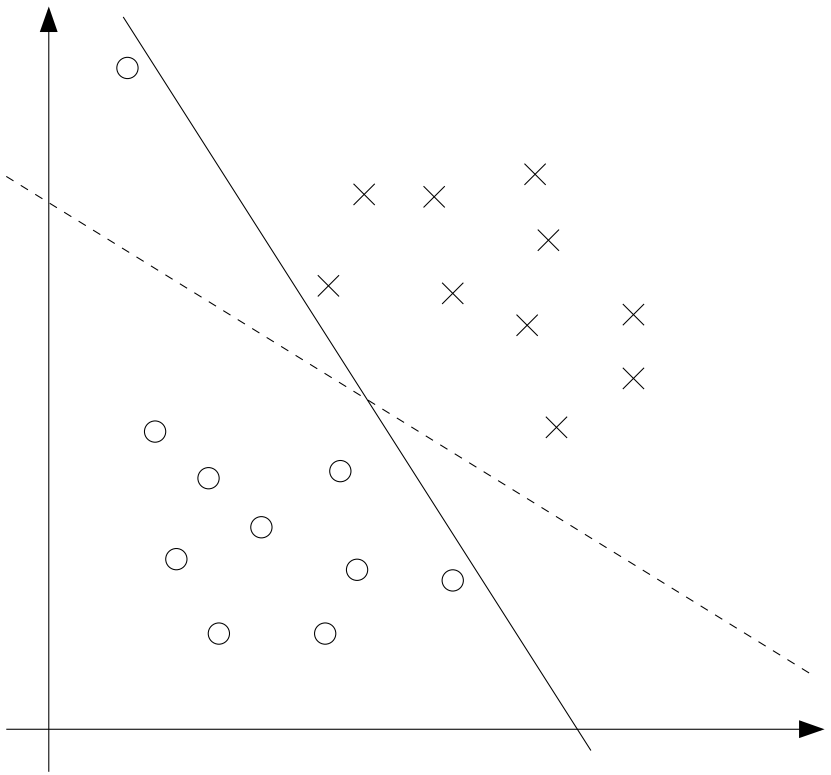
- Although the summation terms would only need to include $\mathbf{x}^{(i)}$ that are support vectors, since other points will have $\alpha_i = 0$
- The intercept b can then be found by:

$$b = \frac{1}{|SV|} \sum_{i \in SV} (\mathbf{w} \cdot \mathbf{x}^{(i)} - y^{(i)})$$

where SV is the set of i's such that $\alpha_i > 0$

Outliers

Consider the following case:



- The boundary that does not misclassify is not necessarily the best one.
  - Might have problems with generalization later...

Soft Margins

- To solve the problem, one can use soft margins.
  - Compromise between misclassification and maximum margin
- Achieved by adding slack variables to the system.

$$y^{(i)}(\mathbf{w} \cdot \mathbf{x^{(i)}} + b) \geq 1 - \xi_i$$

- The optimization problem now becomes:

$$\min_{\mathbf{w},\xi,b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \right\}$$

such that

$$y_{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0$$

for all $i = 1, \ldots, n$

- C is the soft margin parameter
  - Higher C means less misclassification allowed.

The Dual Form

- Using the fact that $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$ and $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y^{(i)}\mathbf{x}^{(i)}$ the original objective function: $\frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i[y^{(i)}(\mathbf{w}\cdot\mathbf{x}^{(i)} + b) - 1]$ can be converted to:

$$\tilde{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y^{(i)}y^{(j)}(\mathbf{x}^{(i)}\cdot\mathbf{x}^{(j)})$$

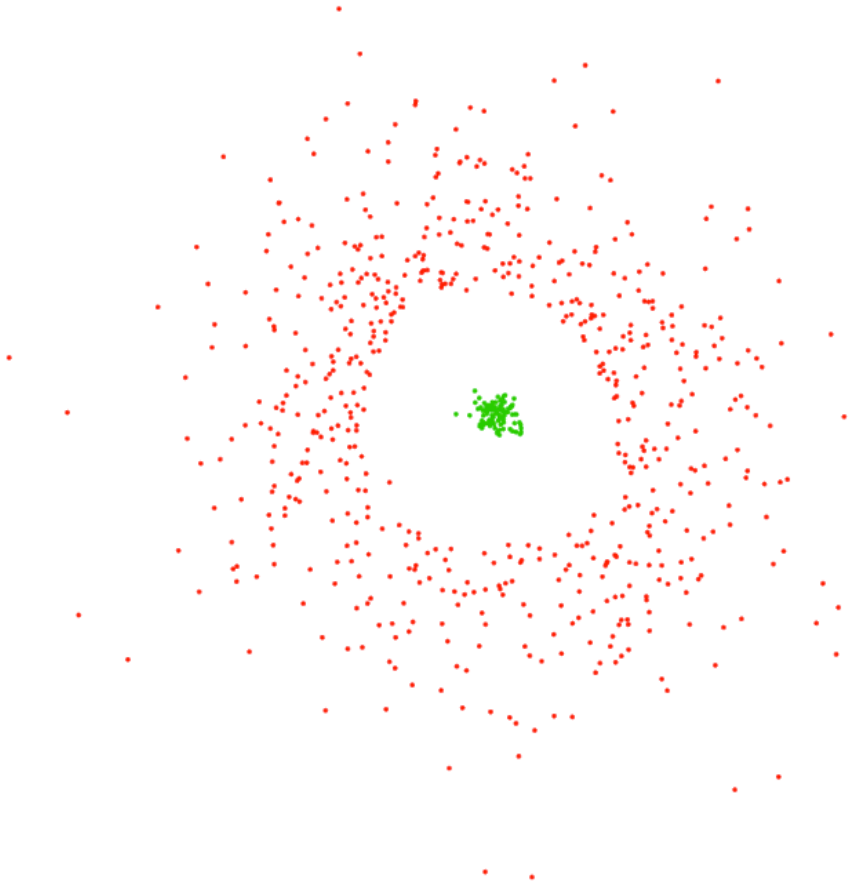
  to be maximized with the constraints $\alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha_i y_{(i)} = 0$
- To predict the class of a new point $\mathbf{z}$

$$\mathbf{w}^T\mathbf{z} + b = \sum_{i\in SV} \alpha_i y^{(i)}(\mathbf{x}^{(i)}\cdot\mathbf{z}) + b$$

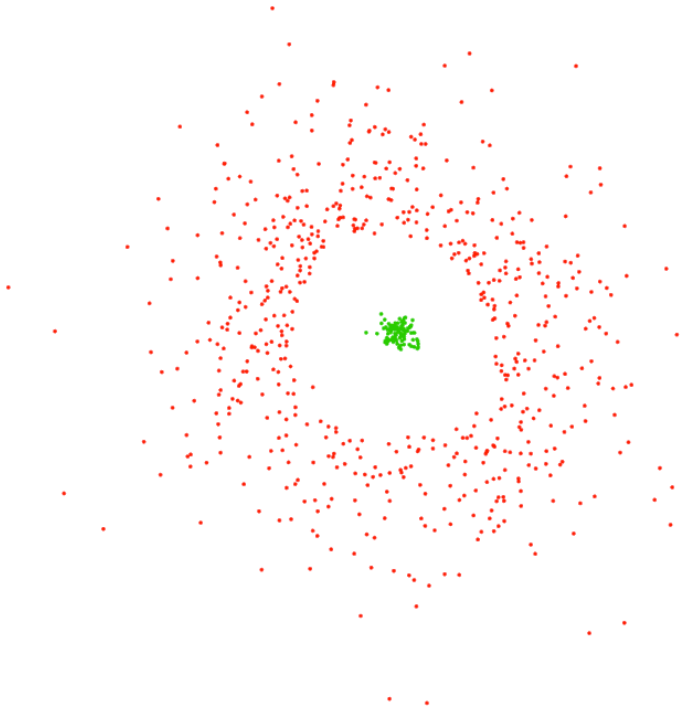
- The fact that we can express these in dot products of points will become very important in a few minutes!

Linear Separability

Can we separate this data with SVM?



Feature Mapping



- Our original attribute space is two dimensional: $\mathbf{x} = [x_1, x_2]$
- What if we map to another space ϕ s.t.

$$\phi(\mathbf{x}) = [x_1, x_2, x_1^2 + x_2^2]$$

- What used to be linearly inseparable is now separable.

Kernels

Definition

A Kernel Function is defined as the dot product of two vectors mapped with some feature mapping ϕ:

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z}) = < \phi(\mathbf{x}), \phi(\mathbf{z}) >$$

- Remember the dual form:

$$\tilde{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} (\mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)})$$

- The linear kernel is $K(\mathbf{x}, \mathbf{z}) = \mathbf{x} \cdot \mathbf{z}$:

$$\tilde{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

- Substituting a higher dimensional kernel in place of the dot product is called the kernel trick.

Advantage of Kernels

- We generally do not use the mappings ϕ themselves but rather use their dot products directly
- Very frequently, calculating $K(\mathbf{x}, \mathbf{z})$ directly is much cheaper than first calculating $\phi(\mathbf{x})$ and $\phi(\mathbf{z})$ and then taking the dot product.

Example

Consider the mapping $\phi(\mathbf{x})$ that consists of pairwise products of all the dimensions of $\mathbf{x}$. E.g. for a 3-dimensional vector $\mathbf{x}$:

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix}$$

This would take $O(n^2)$ time to compute for n dimensions. Whereas the kernel of the same function:

$$K(x, z) = \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) = \left(\sum_{i=1}^{n} x_i z_i \right)^2$$

Would take only $O(n)$ time to compute.

Common Kernels

- Polynomial Kernel

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z} + c)^d$$

- Gaussian Kernel

$$K(\mathbf{x}, \mathbf{z}) = e^{\left(\frac{||(\mathbf{x}-\mathbf{z})||^2}{2\sigma^2}\right)}$$

  Also called the Radial Basis Function (RBF) Kernel

- Inverse Multiquadric Kernel

$$K(\mathbf{x}, \mathbf{z}) = \frac{1}{\sqrt{||(\mathbf{x} - \mathbf{z})||^2 - c}}$$

Can Any Function Be a Kernel?

- Intuitively, a kernel is a similarity measure between $\phi(\mathbf{x})$ and $\phi(\mathbf{z})$, and indirectly between $\mathbf{x}$ and $\mathbf{z}$
  - Larger the K, the more similar $\mathbf{x}$ and $\mathbf{z}$
- With some restrictions, you can come up with any kernel that you think will best separate your problem.
  - e.g. Edit distance as kernel for sequence data
- Note that any kernel $K(\mathbf{x}, \mathbf{z})$ must be the dot product $\phi(\mathbf{x}) \cdot \phi(\mathbf{z})$ for some defined mapping ϕ
- Let $\mathbf{K}$ be a matrix s.t.

$$K_{ij} = K(x^{(i)}, x^{(j)})$$

  - Since $\phi(\mathbf{x}) \cdot \phi(\mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z})$, $\mathbf{K}$ must be symmetric.
  - Due to self similarity, $\mathbf{v}^{\mathbf{T}} K \mathbf{v} \geq 0$ for any vector $\mathbf{v}$. In other words $\mathbf{K}$ must also be positive semi-definite

Mercer's Theorem

Theorem

Mercer's Theorem states that for K to be a valid kernel, it is **necessary and sufficient** *for the corresponding kernel matrix* $\mathbf{K}$ *to be* **symmetric positive semi definite.**

In other words

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \mathbf{K}_{ij} \geq 0$$

for any $n > 0$, any n data points, and any choice of real numbers $c_{1,\ldots,n}$

- Such kernels are called Mercer Kernels
- Any Mercer Kernel may be used in a kernel trick
- Only a Mercer Kernel is guaranteed to work in all cases during a kernel trick.
  - Non-Mercer kernels may work for limited data ranges/sets.

Exercises

Example

Does the following the kernel matrix belong to a Mercer Kernel?

$$\mathbf{K} = \begin{bmatrix} 1 & 0.5 & 0.3 \\ 0.5 & 1 & 0.6 \\ 0.3 & 0.6 & 1 \end{bmatrix}$$

YES

Example

How about this one?

$$\mathbf{K} = \begin{bmatrix} 1 & 2.5 & 0.3 \\ 2.5 & 1 & 0.6 \\ 0.3 & 0.6 & 1 \end{bmatrix}$$

NO. Try $c_1 = 1, c_2 = -1, c_3 = 1$

Kernels and PCA

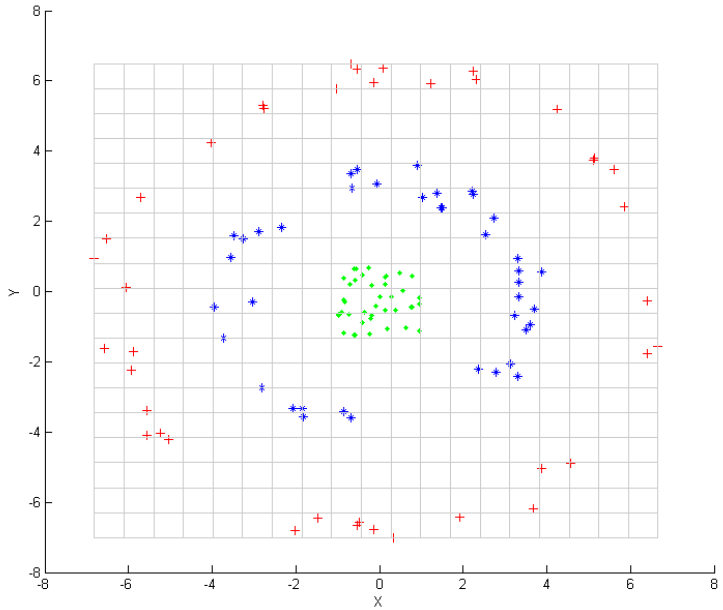
- Remember the final equation for principal components (eigenvectors) $\mathbf{W}$, and covariance matrix $\mathbf{\Sigma}$:

$$\mathbf{\Lambda} = \mathbf{W}^{-1}\mathbf{\Sigma}\mathbf{W}$$

- The covariance vector is $\mathbf{\Sigma} = \mathbf{X'}^T\mathbf{X'}$, where $\mathbf{X'}$ is the centered data matrix.
  - The covariance vector is built up from the dot products: $\Sigma_{ij} = x'^{(i)} \cdot x'^{(j)}$
- In essence, the covariance vector is a kernel.
- Thus, you could just replace it with another kernel K:

$$\mathbf{\Lambda} = \mathbf{W}^{-1}\mathbf{K}\mathbf{W}$$

Example: PCA with RBF Kernel



- Is there a set of linear PCs (rotations) that will cluster this data?
- Since, we want dimensional reduction, ideally you want to be able to cluster just on the first PC.

Figure by Petter Strandmark

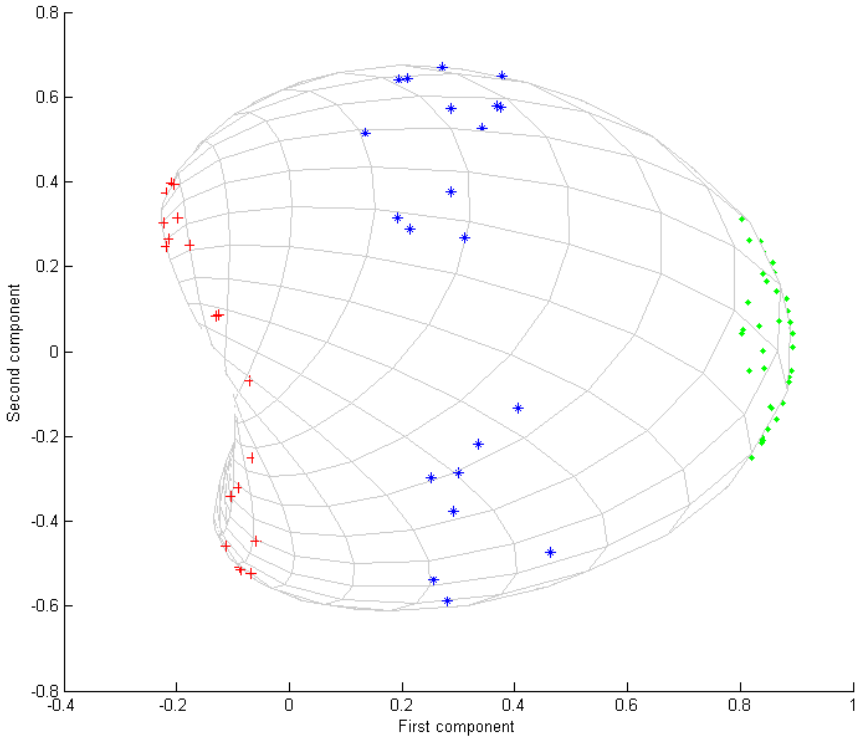
Example: PCA with RBF Kernel (2)

- Instead of linear PCA, let's use the RBF kernel:

$$k(\mathbf{x}, \mathbf{y}) = e^{\frac{-||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2}}$$

- A very common and powerful kernel
- Creates a small hyperball around an instance.
- The actual mappings $\phi(\mathbf{x}), \phi(\mathbf{x})$ since they represent vectors in an infinite dimensional Hilbert space.
  - But we can still calculate $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$ since it represents a convergent power series.
- σ is the measure of the radius of the hyperball around an instance.
  - Should be large enough that close instances are considered similar, but not too large that everything becomes similar.
  - Sigma Tuning

Example: PCA with RBF Kernel (3)



- Notice that just the first PC is enough to cluster.

Figure by Petter Strandmark

Kernel PCA and SVM in R

- SVMs are implemented in R under the e1071 package available from CRAN
  - the `svm` function
  - allows linear, RBF, polynomial and sigmoid kernels
- Kernel PCA is available in the kernlab package from CRAN
  - the `kpca` function
  - supports RBF, polynomial, hyperbolic, Laplacian, Bessel, and spline kernels.