BIN 504
Probabilistic and Statistical Modeling for Bioinformatics

Fall 2010-2011

Week 12
Principal Component Analysis
(slides on PCA from Babak Rasolzadeh and Jieping Ye)

Kernel Methods
(some SVM slides from Raymond J. Mooney)
Data Presentation

- Example: 53 Blood and urine measurements (wet chemistry) from 65 people (33 alcoholics, 32 non-alcoholics).
- Matrix Format

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>8.0000</td>
<td>4.8200</td>
<td>14.1000</td>
<td>41.0000</td>
<td>85.0000</td>
<td>29.0000</td>
<td>34.0000</td>
</tr>
<tr>
<td>A2</td>
<td>7.3000</td>
<td>5.0200</td>
<td>14.7000</td>
<td>43.0000</td>
<td>86.0000</td>
<td>29.0000</td>
<td>34.0000</td>
</tr>
<tr>
<td>A3</td>
<td>4.3000</td>
<td>4.4800</td>
<td>14.1000</td>
<td>41.0000</td>
<td>91.0000</td>
<td>32.0000</td>
<td>35.0000</td>
</tr>
<tr>
<td>A4</td>
<td>7.5000</td>
<td>4.4700</td>
<td>14.9000</td>
<td>45.0000</td>
<td>101.0000</td>
<td>33.0000</td>
<td>33.0000</td>
</tr>
<tr>
<td>A5</td>
<td>7.3000</td>
<td>5.5200</td>
<td>15.4000</td>
<td>46.0000</td>
<td>84.0000</td>
<td>28.0000</td>
<td>33.0000</td>
</tr>
<tr>
<td>A6</td>
<td>6.9000</td>
<td>4.8600</td>
<td>16.0000</td>
<td>47.0000</td>
<td>97.0000</td>
<td>33.0000</td>
<td>34.0000</td>
</tr>
<tr>
<td>A7</td>
<td>7.8000</td>
<td>4.6800</td>
<td>14.7000</td>
<td>43.0000</td>
<td>92.0000</td>
<td>31.0000</td>
<td>34.0000</td>
</tr>
<tr>
<td>A8</td>
<td>8.6000</td>
<td>4.8200</td>
<td>15.8000</td>
<td>42.0000</td>
<td>88.0000</td>
<td>33.0000</td>
<td>37.0000</td>
</tr>
<tr>
<td>A9</td>
<td>5.1000</td>
<td>4.7100</td>
<td>14.0000</td>
<td>43.0000</td>
<td>92.0000</td>
<td>30.0000</td>
<td>32.0000</td>
</tr>
</tbody>
</table>

- Spectral Format
Data Presentation

Univariate

Trivariate

Bivariate
Data Presentation

- Better presentation than ordinate axes?
- Do we need a 53 dimension space to view data?
- How to find the ‘best’ low dimension space that conveys maximum useful information?
- One answer: Find “Principal Components”
Why feature reduction?

- **Visualization**: projection of high-dimensional data onto 2D or 3D.

- **Data compression**: efficient storage and retrieval.

- **Noise removal**: positive effect on query accuracy.
What is Principal Component Analysis?

- Principal component analysis (PCA)
  - Reduce the dimensionality of a data set by finding a new set of variables, smaller than the original set of variables
  - Retains most of the sample's information.
  - Useful for the compression and classification of data.

- By information we mean the variation present in the sample, given by the correlations between the original variables.
  - The new variables, called principal components (PCs), are uncorrelated, and are ordered by the fraction of the total information each retains.
Geometric picture of principal components (PCs)

- the 1\textsuperscript{st} PC $\mathbf{z}_1$ is a minimum distance fit to a line in $\mathbf{X}$ space
- the 2\textsuperscript{nd} PC $\mathbf{z}_2$ is a minimum distance fit to a line in the plane perpendicular to the 1\textsuperscript{st} PC

PCS are a series of linear least squares fits to a sample, each orthogonal to all the previous.
Principal Components

- All principal components (PCs) start at the origin of the ordinate axes.
- First PC is direction of maximum variance from origin.
- Subsequent PCs are orthogonal to 1st PC and describe maximum residual variance.
Given \( m \) points in a \( n \) dimensional space, for large \( n \), how does one project on to a 1 dimensional space?

Choose a line that fits the data so the points are spread out well along the line.
PCA: General

From $k$ original variables: $x_1, x_2, \ldots, x_k$:

Produce $k$ new variables: $y_1, y_2, \ldots, y_k$:

\[
\begin{align*}
    y_1 &= a_{11} x_1 + a_{12} x_2 + \ldots + a_{1k} x_k \\
    y_2 &= a_{21} x_1 + a_{22} x_2 + \ldots + a_{2k} x_k \\
    \vdots \\
    y_k &= a_{k1} x_1 + a_{k2} x_2 + \ldots + a_{kk} x_k
\end{align*}
\]
PCA: General

From \( k \) original variables: \( x_1, x_2, \ldots, x_k \):

Produce \( k \) new variables: \( y_1, y_2, \ldots, y_k \):

\[
y_1 = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1k}x_k
\]
\[
y_2 = a_{21}x_1 + a_{22}x_2 + \ldots + a_{2k}x_k
\]

\[
\ldots
\]
\[
y_k = a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kk}x_k
\]

such that:

\( y_k \)'s are uncorrelated (orthogonal)
\( y_1 \) explains as much as possible of original variance in data set
\( y_2 \) explains as much as possible of remaining variance etc.
2nd Principal Component, $y_2$

1st Principal Component, $y_1$
PCA coordinates
PCA Eigenvalues
From $k$ original variables: $x_1, x_2, ..., x_k$:

Produce $k$ new variables: $y_1, y_2, ..., y_k$:

$y_1 = a_{11}x_1 + a_{12}x_2 + ... + a_{1k}x_k$

$y_2 = a_{21}x_1 + a_{22}x_2 + ... + a_{2k}x_k$

...  

$y_k = a_{k1}x_1 + a_{k2}x_2 + ... + a_{kk}x_k$

such that:

$y_k$'s are uncorrelated (orthogonal)

$y_1$ explains as much as possible of original variance in data set

$y_2$ explains as much as possible of remaining variance

etc.
{a_{11},a_{12},...,a_{1k}} is 1st Eigenvector of correlation/covariance matrix, and coefficients of first principal component

{a_{21},a_{22},...,a_{2k}} is 2nd Eigenvector of correlation/covariance matrix, and coefficients of 2nd principal component

... 

{a_{k1},a_{k2},...,a_{kk}} is kth Eigenvector of correlation/covariance matrix, and coefficients of kth principal component
A 2D Numerical Example
PCA Example – STEP 1

• Subtract the mean from each of the data dimensions. All the x values have $\bar{x}$ subtracted and y values have $\bar{y}$ subtracted from them. This produces a data set whose mean is zero.

Subtracting the mean makes variance and covariance calculation easier by simplifying their equations. The variance and co-variance values are not affected by the mean value.
### PCA Example – STEP 1

<table>
<thead>
<tr>
<th>DATA:</th>
<th>ZERO MEAN DATA:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$y$</td>
</tr>
<tr>
<td>2.5</td>
<td>2.4</td>
</tr>
<tr>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>2.2</td>
<td>2.9</td>
</tr>
<tr>
<td>1.9</td>
<td>2.2</td>
</tr>
<tr>
<td>3.1</td>
<td>3.0</td>
</tr>
<tr>
<td>2.3</td>
<td>2.7</td>
</tr>
<tr>
<td>2</td>
<td>1.6</td>
</tr>
<tr>
<td>1</td>
<td>1.1</td>
</tr>
<tr>
<td>1.5</td>
<td>1.6</td>
</tr>
<tr>
<td>1.1</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>.69</td>
<td>.49</td>
</tr>
<tr>
<td>-1.31</td>
<td>-1.21</td>
</tr>
<tr>
<td>.39</td>
<td>.99</td>
</tr>
<tr>
<td>.09</td>
<td>.29</td>
</tr>
<tr>
<td>1.29</td>
<td>1.09</td>
</tr>
<tr>
<td>.49</td>
<td>.79</td>
</tr>
<tr>
<td>.19</td>
<td>-.31</td>
</tr>
<tr>
<td>-.81</td>
<td>-.81</td>
</tr>
<tr>
<td>-.31</td>
<td>-.31</td>
</tr>
<tr>
<td>-.71</td>
<td>-1.01</td>
</tr>
</tbody>
</table>
PCA Example – STEP 1

Figure 3.1: PCA example data, original data on the left, data with the means subtracted on the right, and a plot of the data
• **Calculate the covariance matrix**

\[
\begin{bmatrix}
0.616555556 & 0.615444444 \\
0.615444444 & 0.716555556
\end{bmatrix}
\]

• since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y variable increase together.
PCA Example – STEP 3

• Calculate the eigenvectors and eigenvalues of the covariance matrix

\[
\begin{align*}
eigenvalues &= \begin{bmatrix} .0490833989 \\ 1.28402771 \end{bmatrix} \\
eigenvectors &= \begin{bmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{bmatrix}
\end{align*}
\]
PCA Example – STEP 3

- Eigenvectors are plotted as diagonal dotted lines on the plot.
- Note they are perpendicular to each other.
- Note one of the eigenvectors goes through the middle of the points, like drawing a line of best fit.
- The second eigenvector gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.
PCA Example – STEP 4

- Reduce dimensionality and form *feature vector*
  the eigenvector with the *highest* eigenvalue is the *principle component* of the data set.

  In our example, the eigenvector with the largest eigenvalue was the one that pointed down the middle of the data.

  Once eigenvectors are found from the covariance matrix, the next step is to order them by *eigenvalue*, highest to lowest. This gives you the components in order of significance.
Now, if you like, you can decide to ignore the components of lesser significance.

You do lose some information, but if the eigenvalues are small, you don’t lose much.

- $n$ dimensions in your data
- calculate $n$ eigenvectors and eigenvalues
- choose only the first $p$ eigenvectors
- final data set has only $p$ dimensions.
PCA Example – STEP 4

• Feature Vector

Feature Vector = \((eig_1 \ eig_2 \ eig_3 \ldots \ eign)\)

We can either form a feature vector with both of the eigenvectors:

\[
\begin{pmatrix}
-.677873399 & -.735178656 \\
-.735178656 & .677873399
\end{pmatrix}
\]

or, we can choose to leave out the smaller, less significant component and only have a single column:

\[
\begin{pmatrix}
- .677873399 \\
- .735178656
\end{pmatrix}
\]
### Reconstruction of original Data

<table>
<thead>
<tr>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>-.827970186</td>
</tr>
<tr>
<td>1.77758033</td>
</tr>
<tr>
<td>-.992197494</td>
</tr>
<tr>
<td>-.274210416</td>
</tr>
<tr>
<td>-1.67580142</td>
</tr>
<tr>
<td>-.912949103</td>
</tr>
<tr>
<td>.0991094375</td>
</tr>
<tr>
<td>1.14457216</td>
</tr>
<tr>
<td>.438046137</td>
</tr>
<tr>
<td>1.22382056</td>
</tr>
</tbody>
</table>

![Graph](image): Original data restored using only a single eigenvector

**Figure 3.5:** The reconstruction from the data that was derived using only a single eigenvector.
Algebraic definition of PCs

Given a sample of \( n \) observations on a vector of \( p \) variables

\[
\{x_1, x_2, \cdots, x_n\} \in \mathbb{R}^p
\]

define the first principal component of the sample by the linear transformation

\[
z_1 = a_1^T x_j = \sum_{i=1}^{p} a_{i1} x_{ij}, \quad j = 1, 2, \cdots, n.
\]

where the vector

\[
a_1 = (a_{11}, a_{21}, \cdots, a_{p1})
\]

\[
x_j = (x_{1j}, x_{2j}, \cdots, x_{pj})
\]

is chosen such that \( \text{var}[z_1] \) is maximum.
Algebraic derivation of PCs

To find $\mathbf{a}_1$ first note that

$$\text{var}[z_1] = E((z_1 - \bar{z}_1)^2) = \frac{1}{n} \sum_{i=1}^{n} (a_1^T x_i - a_1^T \bar{x})^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} a_1^T (x_i - \bar{x})(x_i - \bar{x})^T a_1 = a_1^T S a_1$$

where

$$S = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$$

is the covariance matrix. $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ is the mean.

In the following, we assume the Data is centered. $\bar{x} = 0$
Algebraic derivation of PCs

Assume \( \bar{x} = 0 \)

Form the matrix:
\[
X = [x_1, x_2, \cdots, x_n] \in \mathbb{R}^{p \times n}
\]

Then
\[
S = \frac{1}{n} XX^T
\]

Obtain eigenvectors of S by computing the SVD of X:
\[
X = U\Sigma V^T
\]
To find \( \mathbf{a}_1 \) that maximizes \( \text{var}[z_1] \) subject to \( \mathbf{a}_1^T \mathbf{a}_1 = 1 \)

Let \( \lambda \) be a Lagrange multiplier

\[
L = \mathbf{a}_1^T S \mathbf{a}_1 - \lambda(\mathbf{a}_1^T \mathbf{a}_1 - 1)
\]

\[
\frac{\partial}{\partial \mathbf{a}_1} L = S \mathbf{a}_1 - \lambda \mathbf{a}_1 = 0
\]

\[
\Rightarrow (S - \lambda I_p) \mathbf{a}_1 = 0
\]

therefore \( \mathbf{a}_1 \) is an eigenvector of \( S \)

corresponding to the largest eigenvalue \( \lambda = \lambda_1 \).
To find the next coefficient vector $a_2$ maximizing $\text{var}[z_2]$

subject to $\text{cov}[z_2, z_1] = 0$

and to $a_2^T a_2 = 1$

First note that $\text{cov}[z_2, z_1] = a_1^T S a_2 = \lambda_1 a_1^T a_2$

then let $\lambda$ and $\phi$ be Lagrange multipliers, and maximize

$$L = a_2^T S a_2 - \lambda (a_2^T a_2 - 1) - \phi a_2^T a_1$$
Algebraic derivation of PCs

\[ L = a_2^T S a_2 - \lambda (a_2^T a_2 - 1) - \phi a_2^T a_1 \]

\[ \frac{\partial}{\partial a_2} L = S a_2 - \lambda a_2 - \phi a_1 = 0 \Rightarrow \phi = 0 \]

\[ S a_2 = \lambda a_2 \quad \text{and} \quad \lambda = a_2^T S a_2 \]
Algebraic derivation of PCs

We find that $a_2$ is also an eigenvector of $S$ whose eigenvalue $\lambda = \lambda_2$ is the second largest.

In general

$$\text{var}[z_k] = a_k^T S a_k = \lambda_k$$

- The $k^{\text{th}}$ largest eigenvalue of $S$ is the variance of the $k^{\text{th}}$ PC.
- The $k^{\text{th}}$ PC $z_k$ retains the $k^{\text{th}}$ greatest fraction of the variation in the sample.
Algebraic derivation of PCs

• Main steps for computing PCs
  – Form the covariance matrix $S$.
  – Compute its eigenvectors: $\{a_i\}_{i=1}^p$
  – Use the first $d$ eigenvectors $\{a_i\}_{i=1}^d$ to form the $d$ PCs.

$$G \leftarrow [a_1, a_2, \cdots, a_d]$$

– The transformation $G$ is given by

A test point $x \in \mathbb{R}^p \rightarrow G^T x \in \mathbb{R}^d$. 
Optimality property of PCA

Dimension reduction

Original data

$X \in \mathbb{R}^{p \times n} \rightarrow G^T X \in \mathbb{R}^{d \times n}$

Reconstruction

$G^T X \in \mathbb{R}^{d \times n} \rightarrow \overline{X} = G(G^T X) \in \mathbb{R}^{p \times n}$

$Y = G^T X \in \mathbb{R}^{d \times n}$
Optimality property of PCA

Main theoretical result:

The matrix $G$ consisting of the first $d$ eigenvectors of the covariance matrix $S$ solves the following min problem:

$$\min_{G \in \mathbb{R}^{p \times d}} \|X - G(G^T X)\|_F^2 \text{ subject to } G^T G = I_d$$

PCA projection minimizes the reconstruction error among all linear projections of size $d$. The reconstruction error is given by:

$$\|X - \overline{X}\|_F^2$$
Applications of PCA


• *Probabilistic Disease Classification of Expression-Dependent Proteomic Data from Mass Spectrometry of Human Serum*. Lilien. 2003.
PCA for image compression

Original Image

d=1
d=2
d=4
d=8
d=16
d=32
d=64
d=100
Introduction to Kernel Methods

• What do we do in data analysis?
  – Represent our data in some form of abstraction: as vectors, as graphs, as sequence of letters, as images, etc.

• Then we design algorithms to process that type of abstraction

• We denote the set of objects we want analyze as \( S = (x_1, \ldots, x_n) \) where \( x_i \in \mathcal{X} \)

• We denote the representation of \( x \) as \( \phi(x) \in \mathcal{F} \)
Example

• Let $\mathcal{X}$ be the set of all protein sequences.
• We choose to represent the protein sequences by their sequence of one letter amino acid codes.
• $\phi(x) = \text{MVILARCLLIMKVM}$
• Pairwise similarity, multiple sequence alignment, Blast, etc. runs on this representation.
Kernel Methods

- Kernel methods bypass this data abstraction step.
- The data are not represented individually.
- They are represented through a set of pairwise comparisons.
- We use the kernel function \( k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \) which is a real-valued comparison function, to represent our data.
- For a set of \( n \) objects we have a \( n \times n \) square matrix to represent the data.
- Kernel methods are designed to process such square matrices
Example

\[ \mathcal{X} \]

\[ \phi(S) = (\text{ILAVM, LCLRIM, VMK}) \]

\[ K = \begin{pmatrix}
1 & 0.5 & 0.3 \\
0.5 & 1 & 0.6 \\
0.3 & 0.6 & 1
\end{pmatrix} \]
Kernel restriction

• Not every comparison function is a valid kernel. A function $k$ is a kernel function only if it is symmetric and positive definite, that is,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$$

for any $n > 0$, any choice of $n$ objects, and any choice of $n$ real numbers $c_1, \ldots, c_n$. 

Exercise

• What does positive definite mean intuitively?
• Assume we have a \( n \times n \) square matrix. Can we check whether it is a valid kernel matrix?
• Is this matrix a valid kernel matrix?

\[
K = \begin{pmatrix}
1 & 0.5 & 0.3 \\
0.5 & 1 & 0.6 \\
0.3 & 0.6 & 1 \\
\end{pmatrix}
\]
• What about this matrix?

\[
K = \begin{pmatrix}
1 & 2.5 & 0.3 \\
2.5 & 1 & 0.6 \\
0.3 & 0.6 & 1
\end{pmatrix}
\]
Inner product as a kernel function

• If we have a vector representation of the objects we want to analyze, then we can use the inner product between their vector representations as a valid kernel function.

• This is called linear kernel.

\[ k_L(x, x') = x^T x' = \sum_{i=1}^{p} x_i x'_i \]
Other kernel functions

• Commonly used kernel functions are Gaussian radial basis function (RBF) kernel and polynomial kernel.

• However, interestingly, all different kernel functions can be proven to be equal to the linear kernel function at some vector feature space $\mathcal{F}$, if infinite dimensional vectors are also allowed.

• Any kernel on a space $\mathcal{X}$ can be represented as an inner product after the space $\mathcal{X}$ is mapped to a Hilbert space $\mathcal{F}$, called the feature space.
Kernel strength

• The most important benefit of kernel methods is that we do not need to represent our objects in some feature space.

• What we need to provide is to provide a pairwise comparison function.

• Example: Nodes on a graph
  – How to represent them?
    • Difficult
  – A kernel can be the designed based on the shortest path between two nodes.
Kernel Methods

• The algorithms designed to process the square kernel matrices are called kernel methods.

• Example tasks:
  – Clustering
  – Classification
  – Regression
  – Computing certain properties of data

• Support Vector Machines is a kernel method designed for classification of objects.
The Kernel Trick

• Existing algorithms that work using the inner products of vector representations of objects can be modified easily by replacing the dot product by a kernel function.
  – This is called kernelization.

• Huge benefit:
  – Methods that were previously identified as linear methods such as linear discriminant analysis or PCA can be transformed into non-linear methods by replacing the dot product by a kernel that has non-linear capabilities (like RBF kernel).
Nonlinear PCA

Linear projections will not detect the pattern.
Another example
Nonlinear PCA using Kernels

- Traditional PCA applies linear transformation
  - May not be effective for nonlinear data

- Solution: apply nonlinear transformation to potentially very high-dimensional space.

\[ \phi : x \rightarrow \phi(x) \]

- Computational efficiency: apply the kernel trick.
  - Require PCA to be rewritten in terms of dot product.

\[ K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \]
Nonlinear PCA using Kernels

Rewrite PCA in terms of dot product

Assume the data has been centered, i.e., $\sum_i x_i = 0$.

The covariance matrix $S$ can be written as

$$ S = \frac{1}{n} \sum_i x_i x_i^T $$

Let $v$ be the eigenvector of $S$ corresponding to nonzero eigenvalue

$$ Sv = \frac{1}{n} \sum_i x_i x_i^T v = \lambda v \Rightarrow v = \frac{1}{n\lambda} \sum_i (x_i^T v)x_i $$

Eigenvectors of $S$ lie in the space spanned by all data points.
Nonlinear PCA using Kernels

\[ S_v = \frac{1}{n} \sum_{i} x_i x_i^T v = \lambda v \Rightarrow v = \frac{1}{n \lambda} \sum_{i} (x_i^T v) x_i \]

The covariance matrix can be written in matrix form:

\[ S = \frac{1}{n} X X^T, \text{ where } X = [x_1, x_2, \cdots, x_n]. \]

\[ v = \sum_i \alpha_i x_i = X \alpha \quad S_v = \frac{1}{n} X X^T X \alpha = \lambda X \alpha \]

\[ \frac{1}{n} (X^T X)(X^T X) \alpha = \lambda (X^T X) \alpha \]

\[ \frac{1}{n} (X^T X) \alpha = \lambda \alpha \]

Any benefits?
Next consider the feature space: \( \phi : x \rightarrow \phi(x) \)

\[
S^\phi = \frac{1}{n} X^\phi (X^\phi)^T,
\]

where \( X^\phi = [x_1^\phi, x_2^\phi, \ldots, x_n^\phi] \).

\[
v = \sum_{i} \alpha_i \phi(x_i) = X^\phi \alpha
\]

\[
\frac{1}{n} (X^\phi)^T X^\phi \alpha = \lambda \alpha
\]

The \((i,j)\)-th entry of \((X^\phi)^T X^\phi\) is \( \phi(x_i) \cdot \phi(x_j) \)

Apply the kernel trick:

\[
K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)
\]

K is called the kernel matrix.

\[
\frac{1}{n} K \alpha = \lambda \alpha
\]
Nonlinear PCA using Kernels

• Projection of a test point $x$ onto $v$:

$$\phi(x) \cdot v = \phi(x) \cdot \sum_i \alpha_i \phi(x_i)$$

$$= \sum_i \alpha_i \phi(x) \cdot \phi(x_i) = \sum_i \alpha_i K(x, x_i)$$

Explicit mapping is not required here.
• *Principal Component Analysis*. I.T. Jolliffe.

• *Kernel Principal Component Analysis*. Schölkopf, et al.

• *Geometric Methods for Feature Extraction and Dimensional Reduction*. Burges.
The Kernel Trick

• Another benefit:
  – The linear methods were applicable to objects that could be represented as vectors.
  – However, with the kernel function, the classic methods can be applicable to nonvectorial objects as long as a kernel is defined for those object.
  – Example: We can apply PCA to protein sequences.
Using the kernel function to compute distances between objects

- The objects are not represented by a vector representation. We only have a kernel function $k$ on these objects.
- How to define a distance measure between the objects?
- We can use the distance in the induced feature space
Computing the distance

- The distance between two vectors can be written in terms of inner products:
  \[ d(x, y)^2 = x \cdot x + y \cdot y - 2x \cdot y \]

- Therefore, we can use the kernel function (which is the inner product in the induced feature space) to compute distance between two objects.

\[
d(x_1, x_2) = \sqrt{k(x_1, x_1) + k(x_2, x_2) - 2k(x_1, x_2)}
\]
Support Vector Machines

- Suppose that the objects we analyze have some labels associated with them.
- Support Vector Machines are kernel methods to learn a label assignment function $f$, which can be used to assign labels to objects of unknown labels.
- SVMs are typically used in a binary classification setting.
Support Vector Machines
Support Vector Machines
Support Vector Machines
Support Vector Machines
Support Vector Machines
Support Vector Machines
Support Vector Machines

Support vectors

margin
Support Vector Machines

Support vectors

margin
Support Vector Machines

Support vectors

margin
Support Vector Machines

$w^T x + b > 0$

$w^T x + b = 0$

$w^T x + b < 0$

$f(x) = \text{sign}(w^T x + b)$
Linear Separators

• Which of the linear separators is optimal?
Classification Margin

- Distance from example $x_i$ to the separator is $r = \frac{w^T x_i + b}{\|w\|}$.
- Examples closest to the hyperplane are support vectors.
- Margin $\rho$ of the separator is the distance between support vectors.
Maximum Margin Classification

• Implies that only support vectors matter; other training examples are ignorable.
Linear SVM Mathematically

- Let training set \( \{(x_i, y_i)\}_{i=1..n}, x_i \in \mathbb{R}^d, y_i \in \{-1, 1\} \) be separated by a hyperplane with margin \( \rho \). Then for each training example \((x_i, y_i)\):

  \[
  w^T x_i + b \leq -\rho/2 \quad \text{if } y_i = -1
  \]
  \[
  w^T x_i + b \geq \rho/2 \quad \text{if } y_i = 1 \quad \iff \quad y_i(w^T x_i + b) \geq \rho/2
  \]

- For every support vector \( x_s \) the above inequality is an equality. After rescaling \( w \) and \( b \) by \( \rho/2 \) in the equality, we obtain that distance between each \( x_s \) and the hyperplane is

  \[
  r = \frac{y_s(w^T x_s + b)}{\|w\|} = \frac{1}{\|w\|}
  \]

- Then the margin can be expressed through (rescaled) \( w \) and \( b \) as:

  \[
  \rho = 2r = \frac{2}{\|w\|}
  \]
Linear SVMs Mathematically (cont.)

- Then we can formulate the quadratic optimization problem:

  Find \( w \) and \( b \) such that
  \[
  \rho = \frac{2}{\|w\|} \text{ is maximized}
  \]
  and for all \((x_i, y_i), i=1..n:\)
  \[y_i(w^T x_i + b) \geq 1\]

Which can be reformulated as:

Find \( w \) and \( b \) such that
\[
\Phi(w) = \|w\|^2 = w^T w \text{ is minimized}
\]
and for all \((x_i, y_i), i=1..n:\)
\[y_i (w^T x_i + b) \geq 1\]
The Optimization Problem

Solution

• Given a solution $\alpha_1 \ldots \alpha_n$ to the dual problem, solution to the primal is:

$$w = \sum \alpha_i y_i x_i \quad b = y_k - \sum \alpha_i y_i x_i^T x_k \quad \text{for any } \alpha_k > 0$$

• Each non-zero $\alpha_i$ indicates that corresponding $x_i$ is a support vector.

• Then the classifying function is (note that we don’t need $w$ explicitly):

$$f(x) = \sum \alpha_i y_i x_i^T x + b$$
Soft Margin Classification

- What if the training set is not linearly separable?
- *Slack variables* $\xi_i$ can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.
Soft Margin Classification  
Mathematically

• The old formulation:

Find \( \mathbf{w} \) and \( b \) such that 
\[
\Phi(\mathbf{w}) = \mathbf{w}^T \mathbf{w} \text{ is minimized}
\]
and for all \((\mathbf{x}_i, y_i), i=1..n: y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1\)

• Modified formulation incorporates slack variables:

Find \( \mathbf{w} \) and \( b \) such that 
\[
\Phi(\mathbf{w}) = \mathbf{w}^T \mathbf{w} + C \sum \xi_i \text{ is minimized}
\]
and for all \((\mathbf{x}_i, y_i), i=1..n: y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0\)

• Parameter \( C \) can be viewed as a way to control overfitting: 
it “trades off” the relative importance of maximizing the margin and fitting the training data.
Linear SVMs: Overview

- The classifier is a *separating hyperplane*.
- Most “important” training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points \( x_i \) are support vectors with non-zero Lagrangian multipliers \( \alpha_i \).
- Both in the dual formulation of the problem and in the solution training points appear only inside inner products:

\[
\text{Find } \alpha_1 \ldots \alpha_N \text{ such that } \\
Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j x_i^T x_j \text{ is maximized and } \\
(1) \sum \alpha_i y_i = 0 \\
(2) \ 0 \leq \alpha_i \leq C \text{ for all } \alpha_i
\]

\[
f(x) = \sum \alpha_i y_i x_i^T x + b
\]
Non-linear Classification

• The problem
  – The maximal margin classifier is an important concept, but it cannot be used in many real-world problems
  – There will in general be no linear separation in the original feature (input) space.

• The solution
  – Map the data into another feature space that can be separated linearly.
Non-linear SVMs

- Transform $x \rightarrow \phi(x)$
- The linear classifier depends only on $xx_i$, hence transformed algorithm depends only on $\phi(x)\phi(x_i)$
- Use kernel function $K(x_i, x_j)$ such that $K(x_i, x_j) = \phi(x)\phi(x_i)$
Kernel Methods: the mapping
Non-linear SVMs
Mathematically

• Dual problem formulation:

Find \( \alpha_1 \ldots \alpha_n \) such that
\[
Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j K(x_i, x_j)
\]
is maximized and

1. \( \sum \alpha_i y_i = 0 \)
2. \( \alpha_i \geq 0 \) for all \( \alpha_i \)

• The solution is:
\[
f(x) = \sum \alpha_i y_i K(x_i, x_j) + b
\]
Kernel Functions

• The kernel function: It is the function that represents the inner product of some space in ANOTHER space.

• Some spaces are known only by their kernel function (ie: their projection is UNKNOWN)

• It’s the case with these kernel functions:

  – Gaussian RBF kernel: \[ k(\bar{x}, \bar{y}) = e^{-\frac{\|\bar{x} - \bar{y}\|^2}{2\sigma^2}} \]

  – Sigmoid kernel: \[ k(\bar{x}, \bar{y}) = a \tanh\left(\langle \bar{x}, \bar{y} \rangle + b \right) \]
Gaussian RBF kernel

- The most popular kernel function (most powerful) is the Gaussian RBF kernel:

\[ k(\bar{x}, \bar{y}) = e^{-\frac{\|\bar{x} - \bar{y}\|^2}{2\sigma^2}} \]

Powerful kernel as its effect is to create a small classification “hyperball” around an instance. This kernel doesn’t have a projection formula since its dimension is infinite (you can create as many “balls” as you want).

Where \( \sigma \) is a measure of the radius of the “hyperball” around an instance.

You want this ball to be big enough so “hyperballs” connect with each other (pattern recognition) but not too big to overlap the other class.
\[ K(X,Y) = \exp \left( \frac{-||X - Y||^2}{2\sigma^2} \right) \]
Characteristics of SVMs

• Scales well to high-dimensional problems

• Feature selection, choosing the best kernel function are critical issues for the success of an SVM.
Choosing the Kernel Function

- Probably the most tricky part of using SVM.
- Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, …)
- There are even research to estimate the kernel matrix from available information
- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try
Other Aspects of SVM

• How to use SVM for multi-class classification?
  – One can change the QP formulation to become multi-class
  – More often, multiple binary classifiers are combined
  – One can train multiple one-versus-all classifiers, or combine multiple pairwise classifiers “intelligently”

• How to interpret the SVM discriminant function value as probability?
  – By measuring how far a test object is from the separating hyperplane

• Some SVM software (like libsvm) have these features built-in
SVM applications

- SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.
- SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.
- SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.
- SVM techniques have been extended to a number of tasks such as regression [Vapnik et al. ’97], principal component analysis [Schölkopf et al. ’99], etc.
- Most popular optimization algorithms for SVMs use decomposition to hill-climb over a subset of $\alpha_i$’s at a time, e.g. SMO [Platt ’99] and [Joachims ’99]
- Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner.
libsvm demonstration

• libsvm applet
Kernels for Strings

• We may try to represent Strings as fixed dimensional feature vectors and then use the inner product between these vectors as a valid kernel.

• But this approach does not make use of the main utility of kernel methods which is:
  – We do not need to come up with an explicit vector representation for objects
  – We can go ahead and directly design kernels
Kernels for strings

• Example vector representation:
  – Count all $k$-tuples in a string, and use these counts to represent the string

• Other kernels for strings?

• Can we use the edit distance between two string to define a kernel, without representing the strings as vectors?
Combining kernels

• If you have defined many different kernels for a certain object domain, you can use a linear combination of these kernels as a valid kernel.

• This technique can be employed as an information integration approach.

\[ k = \sum_{i=1}^{c} \mu_i k_i \]
From similarity measures to kernels

• You may have a very good similarity measure between the objects under study, but it may not be a valid kernel.

• Can we transform the similarity measure into a valid kernel?

• There is no single answer.
  – But, a number of approaches exist.

• We can use similarity scores to a number of template objects to construct a vector.
Another way

- If we want to convert a similarity matrix into a valid kernel matrix
  - We can use eigendecomposition of the similarity matrix and remove negative eigenvalues to construct a symmetric positive definite kernel matrix.