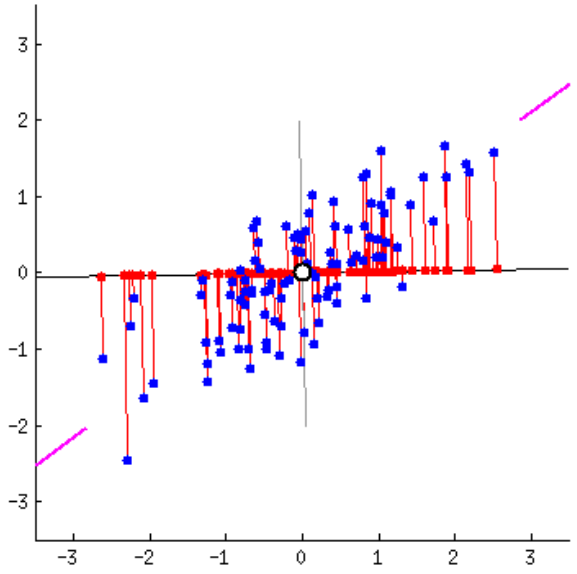
COMS20011 – Data-Driven Computer Science

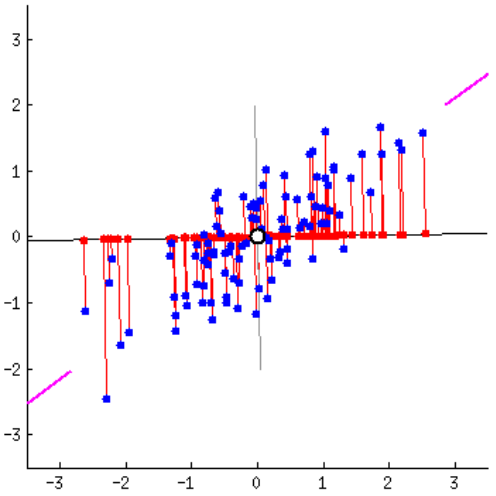


April 2024
Principal Component Analysis

Majid Mirmehdi

Lecture MM-07

Next in DDCS



Feature Selection and Extraction

- Signal basics and Fourier Series
- 1D and 2D Fourier Transform
- Another look at features
- **PCA for dimensionality reduction**
- Convolutions

Dimensionality Reduction

- benefit = reduce number of variables you have to worry about

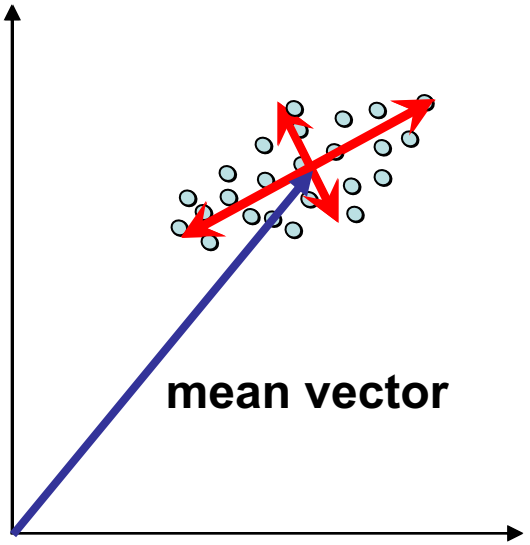
- Two typical approaches:
  - Selecting a subset of a given set of features → FS
  - Selecting a subset after transformation of a set of features → FE

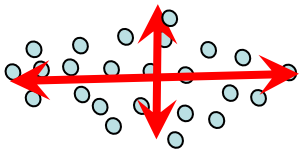
An example of transformation for Feature Extraction:

- Principal Component Analysis - The goal of PCA is to reduce the dimensionality of the data while retaining as much of the variation present in the dataset as possible.

Principal Component Analysis

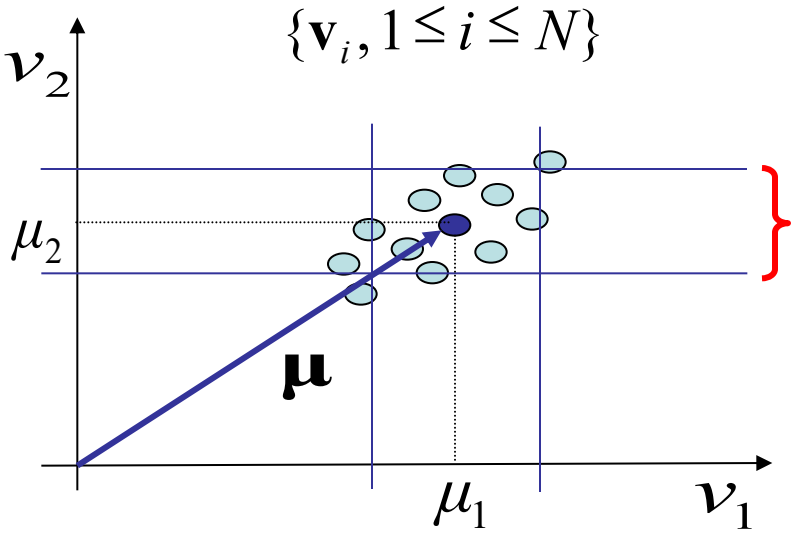
A geometrical view:





PCA decorrelates our data, i.e. it keeps the variance and removes the covariance.

Reminder: Covariance Matrix



$$\boldsymbol{\mu} = \frac{1}{N}\sum_i \mathbf{v}_i$$

Covariance between components

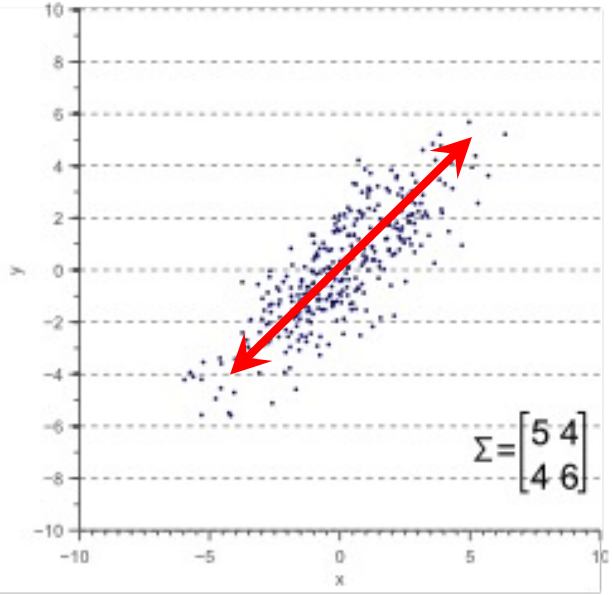
Variance along 1st dimension

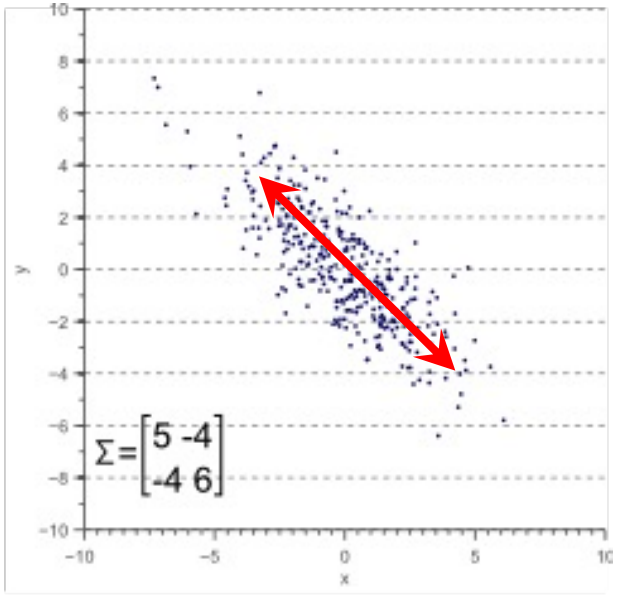
Variance along 2nd dimension

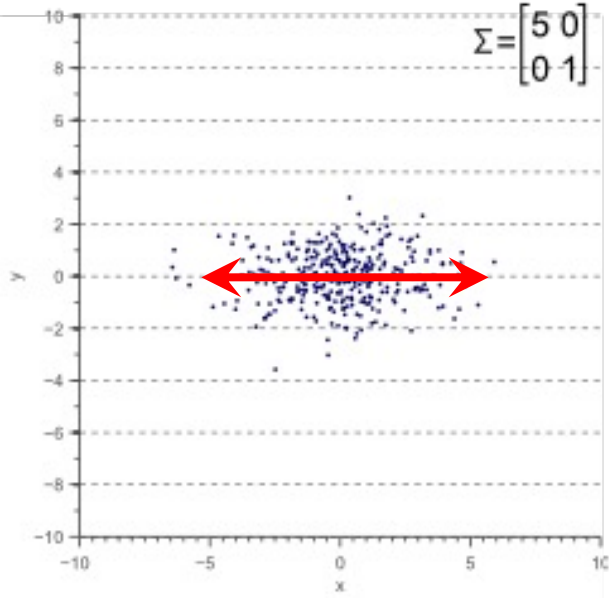
$$\mathbf{C} = \frac{1}{N}\sum_i \begin{bmatrix} (v_{i1}-\mu_1)^2 & (v_{i1}-\mu_1)(v_{i2}-\mu_2) \\ (v_{i1}-\mu_1)(v_{i2}-\mu_2) & (v_{i2}-\mu_2)^2 \end{bmatrix}$$

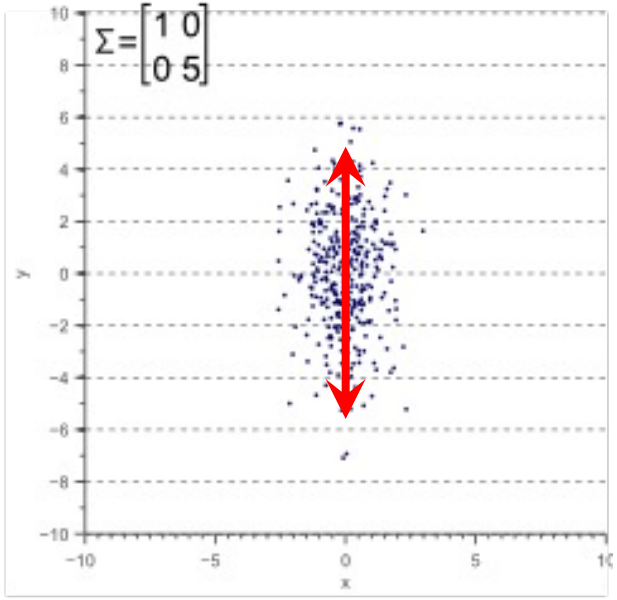
Spread and Covariance

- The shape of the data is defined by the covariance matrix.
- Diagonal spread is captured by the covariance, while axis-aligned spread is captured by the variance.



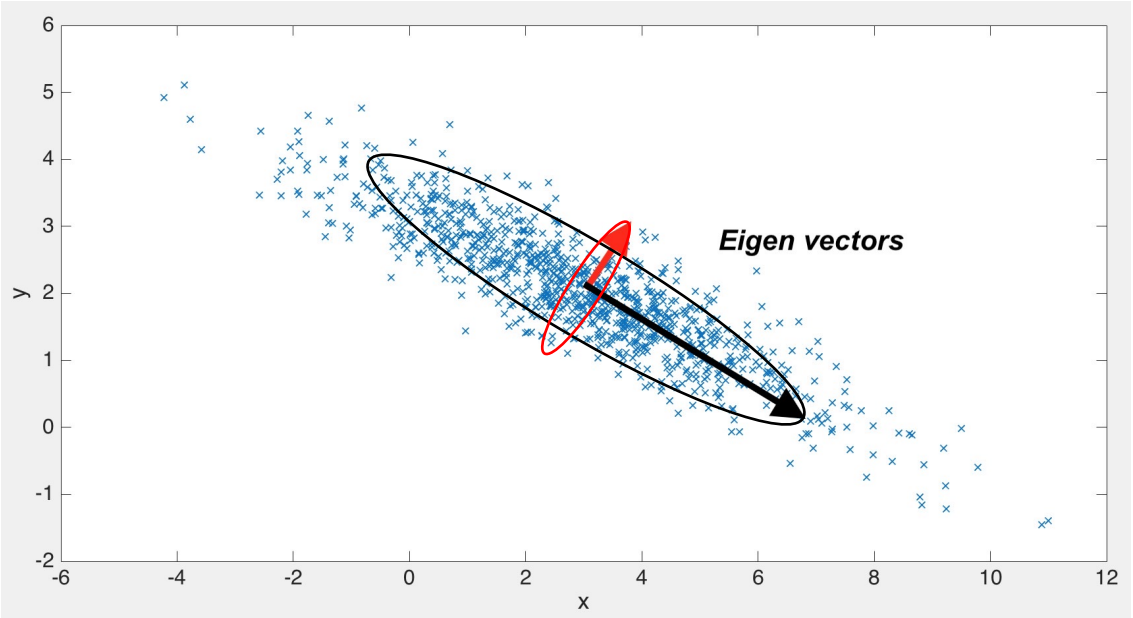






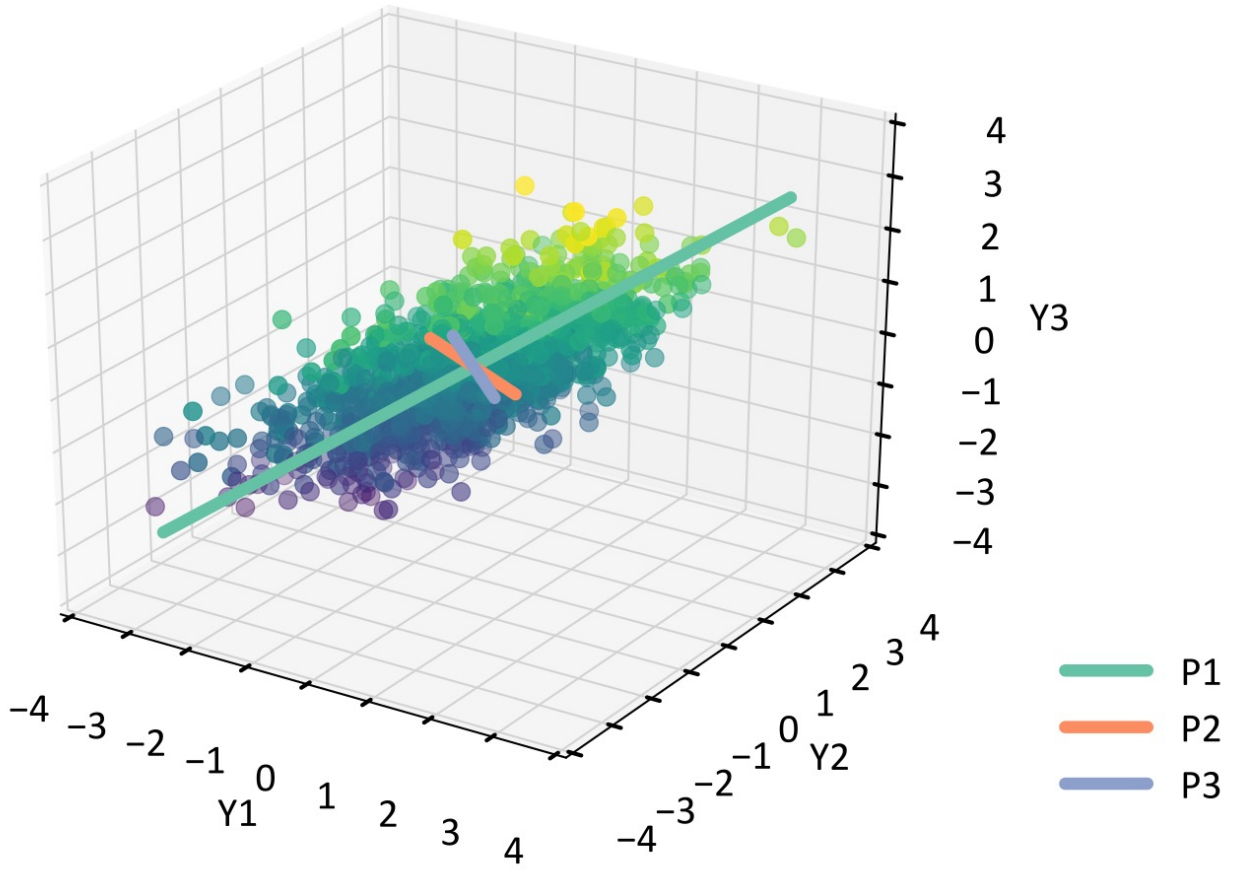
REMINDER – Covariance Matrix: Eigen analysis

- Eigenvectors and eigenvalues define **principal axes** and spread of points along directions, respectively.
- **Major axis** - eigenvector corresponding to larger eigenvalue (i.e. larger variance)
- **Minor axis** - eigenvector corresponding to smaller eigenvalue (i.e. smaller variance)
- These can be represented using major and minor axes of ellipses



Example in 3 Dimensions

- Eigenvectors and eigenvalues define **principal axes** and spread of points along directions



From: https://geostatisticslessons.com/lessons/sphereingmaf

Principal Component Analysis

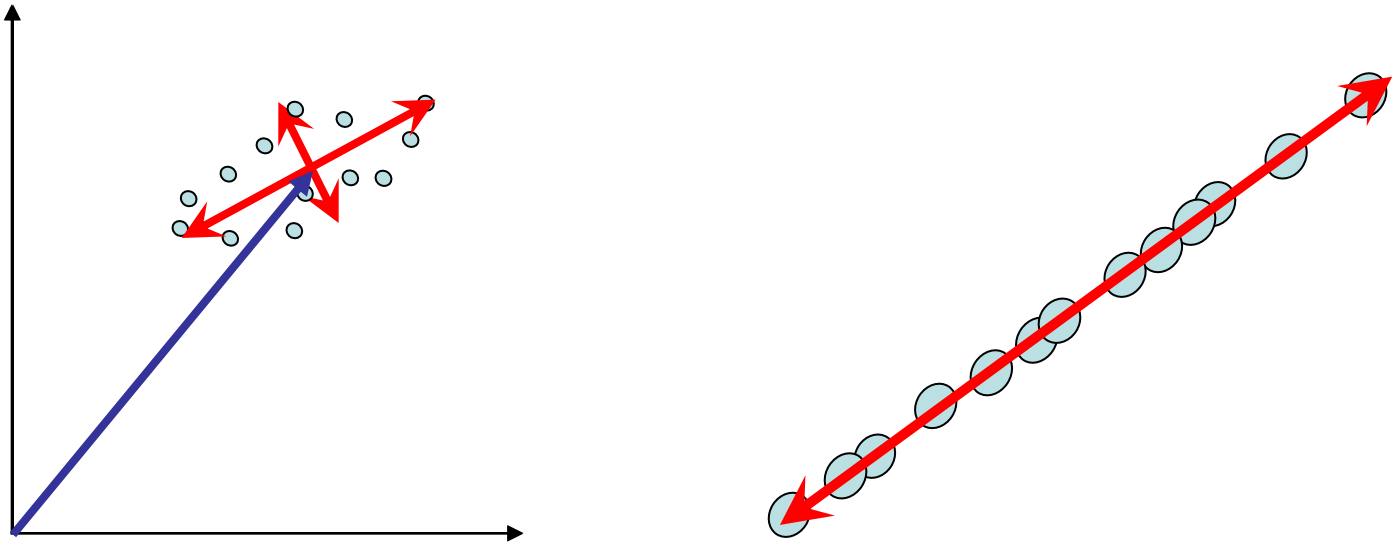
- PCA involves the transformation of a no. of correlated variables into a no. of new uncorrelated variables called → independent features.

- **Principal Axes**: the first direction that accounts for as much of the variance as possible (→ i.e. variance is maximum); then the direction orthogonal to the first for which the variance is maximum, and so on…

- Given N data vectors from p dimensions, find orthogonal vectors from d dimensions $(\text{where } d \leq p)$ that can be best used to represent the N data vectors.

Principal Component Analysis

A geometrical view:



PCA also allows us to represent our data using fewer dimensions by linearly projecting the data onto a lower-dimensional space, in a *least squares* sense.

Eigenvalues & Eigenvectors

If $\mathbf{C}$ is an nxn matrix, do there exist non-zero vectors $\mathbf{u}$ in R^n, such that $\mathbf{Cu}$ is a scalar multiple of $\mathbf{u}$?

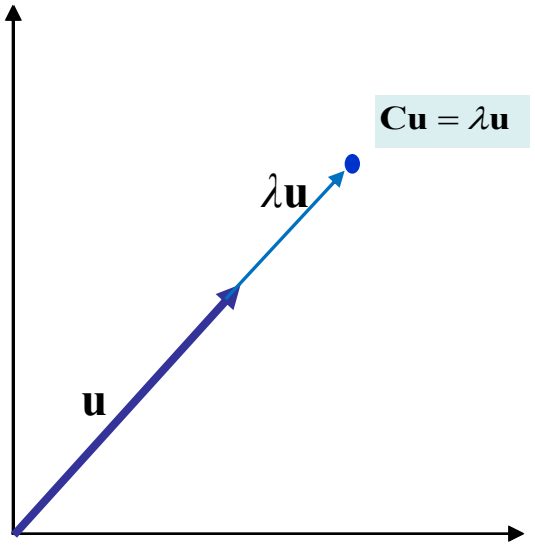
an nxn matrix

Eigenvalue – a scalar

$$\mathbf{Cu} = \lambda\mathbf{u}$$

Eigenvector – a non-zero vector in R^n

- A geometrical view:



Eigenvalue and Eigenvector Example

$$\mathbf{Cu} = \lambda \mathbf{u}$$

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \mathrm{x} \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix} \neq \lambda \mathrm{x} \begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

Not an eigenvector

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \mathrm{x} \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \mathrm{x} \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

Eigenvalue and eigenvector

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \mathrm{x} \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 24 \\ 16 \end{pmatrix} = 4 \mathrm{x} \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

Scaled eigenvector.
Still in the same direction.
Still the same eigenvalue.

Eigenvalue and Eigenvectors

Given the data covariance matrix $\mathbf{C}$, then :

$$\mathbf{C}\mathbf{u}_i = \lambda_i \mathbf{u}_i \rightarrow \mathbf{C}\mathbf{u}_i - \lambda_i \mathbf{u}_i = 0 \rightarrow \mathbf{u}_i(\mathbf{C} - \lambda_i \mathbf{I}) = 0$$

Solving this *characteristic equation* leads to the eigenvalues and eigenvectors:

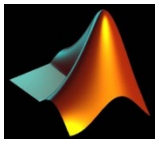
$$|\mathbf{C} - \lambda_i \mathbf{I}| = 0$$

Quite easy in 2 dimensions, just bearable in 3, but not easy as we move into higher dimensions. Enter Matlab/Python…

⇨ Orientation given by eigenvector of covariance matrix

⇨ Spread given by eigenvalue of covariance matrix

Matlab: eigenvalues & eigenvectors



```
%% example to demonstrate the computation of eigenvalues and eigenvectors

disp('This is the example data set:')
V = [2.8 2.2 2.2 1.6 2.5 1.4 1.8 1.2 2.1 1.3;
     3.0 2.0 2.8 1.6 2.7 1.2 2.1 1.5 2.3 1.4
     7.0 7.4 6.2 6.4 6.6 7.0 6.9 7.1 6.5 7.1];
disp(V');
m1 = mean(V(1,:)); m2 = mean(V(2,:)); m3 = mean(V(3,:));
disp('The mean vector is:'); disp([m1 m2 m3]);

disp('Press a key to continue and see the covariance C:'); pause;
kov = cov(V')

disp('press a key to continue and show the eigenvectors and eigenvalues...'); pause;
[eigvec,eigval] = eig(kov)

disp('And finally, just to prove the equation: C u = lambda u')
disp('For example, take the 2nd eigenvalue and eigenvector'); pause;

disp('First C u');   kov*eigvec(:,2)
disp('then lambda u');   eigval(2,2)*eigvec(:,2)
```

See unit web page for Python code

Principal Component Analysis

Consider a data set of N p-dimensional samples $\mathbf{v}_1, \mathbf{v}_2 \ldots, \mathbf{v}_N$.

Let the mean of the samples be at $\mathbf{m}$. Then we can get for example a 1D representation by projecting the data onto a line running through the sample mean:

$$\mathbf{v}_k = \mathbf{m} + a_k \mathbf{u}$$

where $\mathbf{u}$ is a unit vector in the direction of the line, and the scalar a_k is the distance of the point $\mathbf{v}_k$ from the mean $\mathbf{m}$.

- Thus, we find an optimal set of coefficients a_k, $k=1,...,N$, such that:

$$a_k = \mathbf{u}^t (\mathbf{v}_k - \mathbf{m})$$

- The result is a least-squares solution which projects the vectors $\mathbf{v}_k$ onto the line in the direction $\mathbf{u}$ that passes through the sample mean.

Principal Component Analysis

- We can represent the data using a combination of other significant eigenvectors in higher dimensions.
- Thus, we can *approximate* any $\mathbf{v} \in \mathrm{R}^p$ as a linear combination of an orthonormal set of basis vectors $< u_1, u_2, \ldots, u_d >$ where $d \leq p$.

$$\hat{\mathbf{v}} = \mathbf{m} + \sum_{i=1}^{d} \mathbf{a}_i \mathbf{u}_i$$

i.e. $\hat{\mathbf{v}} = \mathbf{m} + a_1 u_1 + a_2 u_2 + \ldots + a_d u_d$

- The eigenvectors are a set of basis vectors for representing any feature vector $\mathbf{v}$ such that $\| \mathbf{v} - \hat{\mathbf{v}} \|$ is minimised → the principal components
- d characterises a lossy or lossless representation of the data $(d \leq p)$

Recipe for PCA

1 - Adjust the data to zero-mean

2.8	3.0
2.2	2.0
2.2	2.8
1.6	1.6
2.5	2.7
1.4	1.2
1.8	2.1
1.2	1.5
2.1	2.3
1.3	1.4

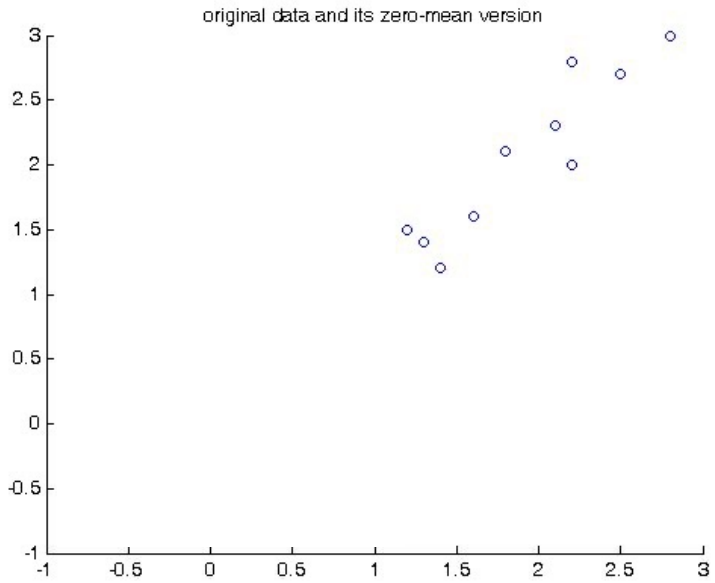
v

mean:

1.91 2.06

0.89	0.94
0.29	-0.06
0.29	0.74
-0.31	-0.46
0.59	0.64
-0.51	-0.86
-0.11	0.04
-0.71	-0.56
0.19	0.24
-0.61	-0.66

$(\mathbf{v} - \mathbf{m})$



Recipe for PCA

2 - Find the Covariance Matrix

$$\mathbf{C} = \begin{pmatrix} 0.2887 & 0.3149 \\ 0.3149 & 0.4004 \end{pmatrix}$$

3 – Compute the eigenvalues and eigenvectors of $\mathbf{C}$

$$\boldsymbol{\lambda} = \begin{pmatrix} 0.0242 \\ 0.6640 \end{pmatrix} \qquad \mathbf{u} = \begin{pmatrix} -0.7669 & 0.6418 \\ 0.6418 & 0.7669 \end{pmatrix}$$

Note $\mathbf{u}^t\mathbf{u} = \| \mathbf{u} \| = 1.$

Recipe for PCA

4 – Order eigenvalues from highest to lowest value

- Eigenvector with the *highest* eigenvalue → 1st principal axis
- Eigenvector with the next *highest* eigenvalue → 2nd principal axis
- and so on (if there were more dimensions!)

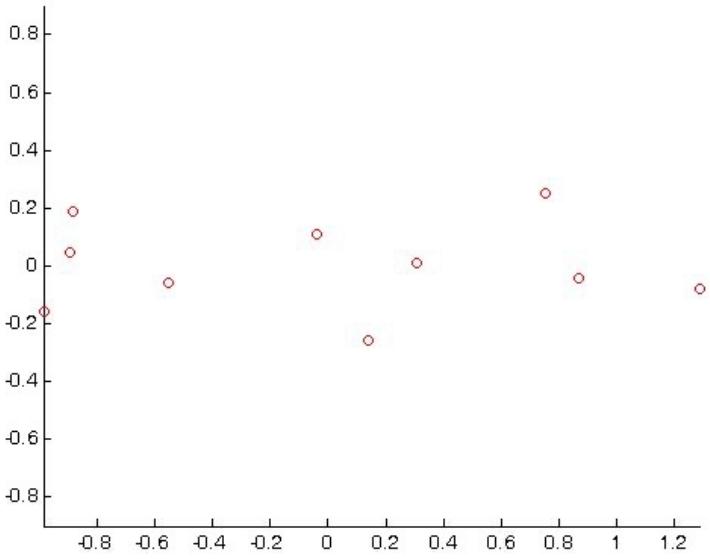
$$\text{Reordered} \quad \boldsymbol{\lambda} = \begin{pmatrix} 0.6640 \\ 0.0242 \end{pmatrix}$$

$$\mathbf{u} = \begin{pmatrix} 0.6418 & -0.7669 \\ 0.7669 & 0.6418 \end{pmatrix}$$

Recipe for PCA

5 – Generate the new representation of the data

1.2921	-0.0793
0.1401	-0.2609
0.7536	0.2525
-0.5517	-0.0575
0.8695	-0.0417
-0.9868	-0.1608
-0.0399	0.1100
-0.8851	0.1851
0.3060	0.0083
-0.8976	0.0442



Note also our data is now totally uncorrelated, i.e. its covariance matrix is diagonal

$$\mathbf{C}_{new} = \begin{pmatrix} 0.6640 & 0 \\ 0 & 0.0242 \end{pmatrix}$$

This step relates to $\mathbf{a} = \mathbf{u}^t(\mathbf{v} - \mathbf{m})$

Recipe for PCA

2.8	3.0
2.2	2.0
2.2	2.8
1.6	1.6
2.5	2.7
1.4	1.2
1.8	2.1
1.2	1.5
2.1	2.3
1.3	1.4

$\mathbf{V}$

6 – Get the old data back (lossless or lossy):

Both principal components to get lossless data back.

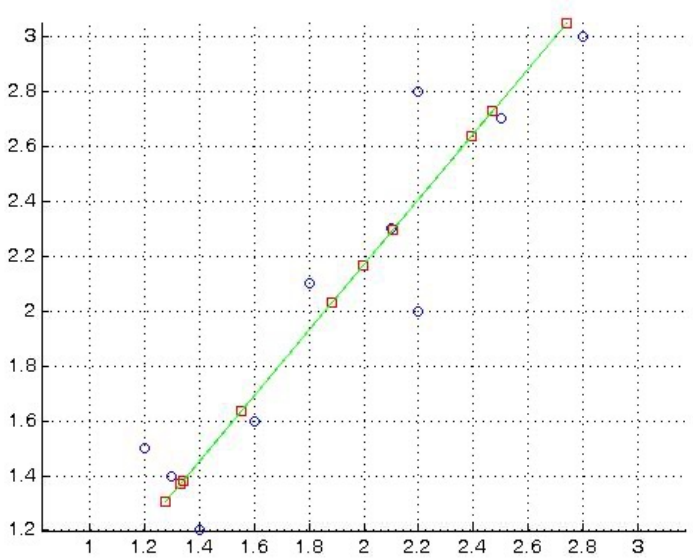
One principal component to get approximate data back (as shown here).

$$\begin{pmatrix} 1.2921 \\ 0.1401 \\ 0.7536 \\ -0.5517 \\ 0.8695 \\ -0.9868 \\ -0.0399 \\ -0.8851 \\ 0.3060 \\ -0.8976 \end{pmatrix} * \begin{pmatrix} 0.6418 & 0.7669 \end{pmatrix} + \begin{pmatrix} 1.91 & 2.06 \end{pmatrix} = \begin{pmatrix} 2.73 & 3.05 \\ 1.99 & 2.16 \\ 2.39 & 2.63 \\ 1.55 & 1.63 \\ 2.46 & 2.72 \\ 1.27 & 1.30 \\ 1.88 & 2.02 \\ 1.34 & 1.38 \\ 2.10 & 2.29 \\ 1.33 & 1.37 \end{pmatrix}$$

Add mean

The new reduced dimensionality representation of our original data →our feature vector

$\hat{\mathbf{V}}$



This step relates to $\mathbf{v} = \mathbf{m} + \sum_{i=1}^{d} \mathbf{a}_i \mathbf{u}_i$

Dimensionality Reduction

- Importance of PCA lies in dimensionality reduction while maintaining as much of the variance as possible!

➢ Sum of the variances = sum of all eigenvalues = 100% of variance in original data

$$\sum_{i=1}^{p} \lambda_i$$

➢ The proportion of the variance that each eigenvector represents can be calculated by dividing the eigenvalue corresponding to that eigenvector by the sum of all eigenvalues.

Hence, the first d eigenvalues can be said to account for a fraction of the total variance in the data.

$$\frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{p} \lambda_i}$$

Example: how to account for a % of variance

- Around 2000 people were asked a set of questions about their *Internet use*. Let's say they asked each person 50 questions.

- There are therefore $p = \mathbf{50}$ **variables, making it a 50-dimensional dataset**. There will then be 50 eigenvectors and eigenvalues out of that dataset.

- Let's say the eigenvalues of the dataset were (in descending order):
  $39.8, 19.2, 17.0, 10.0, 3.2, 1.0, 0.4, 0.21, 0.0979, \ldots\ldots$ with a total sum of $\sum_{i=1}^{50} \lambda_i = 98.5$

- **Only 5 which have big enough values – indicating there is a lot of info (variance) along their corresponding five eigenvectors (directions)!**

- The dataset can thus be reduced from 50 dimensions to only 5 by ignoring all the eigenvectors that have insignificant eigenvalues. Nice way of simplifying the data!

- **Percentage of variance captured by the first 5 components:**

$$\frac{\sum_{i=1}^{5} \lambda_i}{\sum_{i=1}^{50} \lambda_i} \Rightarrow \frac{89.2}{98.5} \Rightarrow \sim 91\%$$

Example Application: Face Recognition using PCA

Set of normalized face images

Eigenfaces

Training:

1. Acquire initial set of N face images (training set) → $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N$
2. The image pixels are then the feature vectors $(S\ x\ 1)$
3. Compute the average image $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$
4. Adjust the data set → $\boldsymbol{\Phi}_i = \mathbf{x}_i - \bar{\mathbf{x}}$
5. Compute the covariance of the image set

Mean face $\bar{\mathbf{x}}$

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})\,(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\Phi}_i\,\boldsymbol{\Phi}_i^T = \frac{1}{N}\,\mathbf{A}\mathbf{A}^T$$

where $\mathbf{A} = [\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \ldots, \boldsymbol{\Phi}_N]$, i.e the columns of $\mathbf{A}$ are the $\boldsymbol{\Phi}_i$, a $S\ x\ N$ matrix.

Eigenfaces

Training:

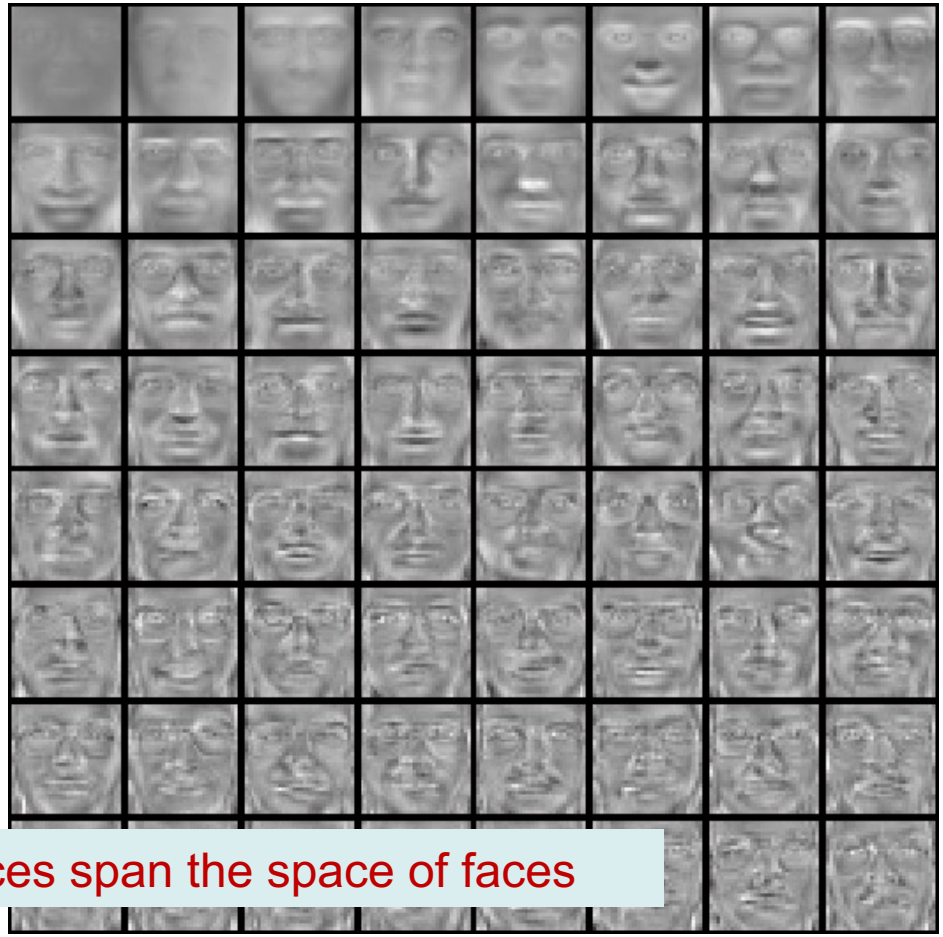
- Compute the eigenvalues and eigenvectors of this covariance matrix → *eigenfaces*

$$(\frac{1}{N}\mathbf{A}\mathbf{A}^T)\mathbf{u} = \boldsymbol{\lambda}\mathbf{u}$$

- Then compute the coefficients, as in

$$\mathbf{a}_i = \mathbf{u}^{\mathrm{T}}(\mathbf{x}_i - \bar{\mathbf{x}})$$

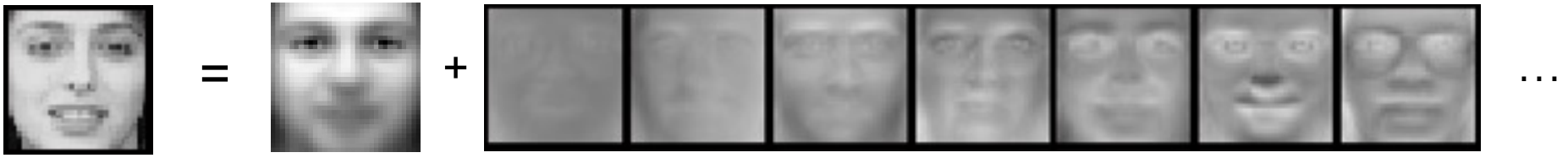
K largest eigenvectors: $\mathbf{u_1}, \mathbf{u_2}, \ldots \mathbf{u_k}$ (or basis vectors visualized as images)



The eigenfaces span the space of faces

Eigenfaces

We can reconstruct a representation of each known individual in face space using a weighted linear combination of the eigenfaces.



$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + a_1\mathbf{u}_1 + a_2\mathbf{u}_2 + a_3\mathbf{u}_3 + \cdots$$

Eigenfaces

Testing (for example, find a matching face)

Given a novel image $\mathbf{y}$,

- Project $\mathbf{y}$ into face space to obtain the eigen coefficients

$$\mathbf{b} = \mathbf{u}^{\mathrm{T}}(\mathbf{y} - \bar{\mathbf{x}})$$

- Find most likely candidate by distance computation between the feature vectors (distance in face space)

$$\operatorname{argmin} \| \mathbf{b} - \boldsymbol{a_i} \| \qquad i = 1,2, \ldots, N$$

Use Euclidean or Mahalanobis distance

PCA characteristics: a summary

PCA: a projection of data that best represents it in a least squares sense:

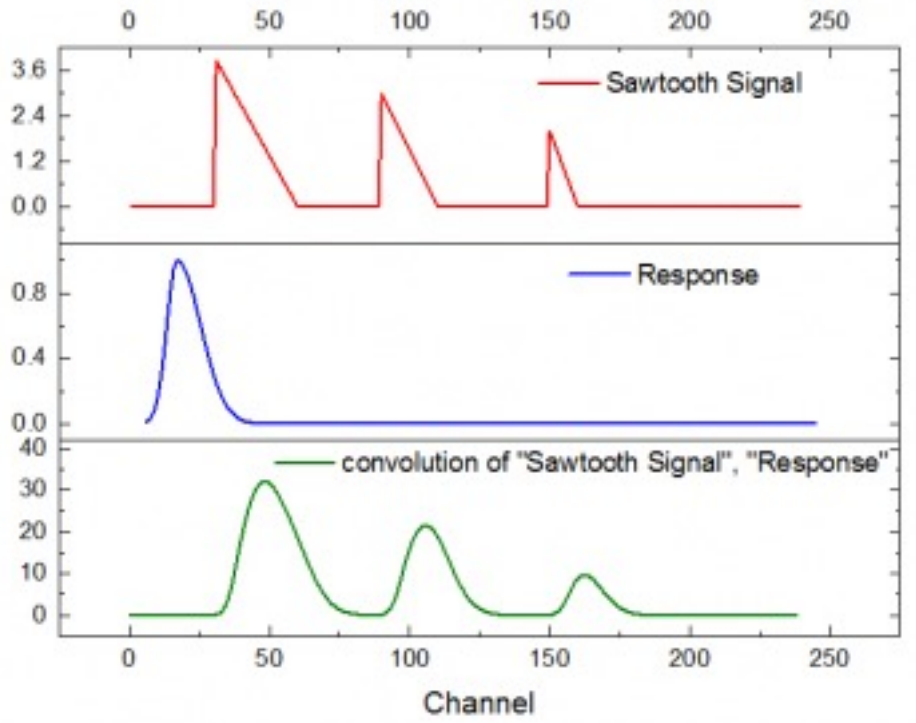
- ★ Reveals the structure in data.
- ★ Provides independent, uncorrelated features.
- ★ Provides reduced dimensionality and speeds up machine learning methods.
- ★ Reduced and uncorrelated feature set makes the process of clustering and classification *much easier.*


Need to reduce outliers to ensure better modelling of data


Low interpretability: PCs are linear combinations of the features from the original data, but they are not as easy to interpret, i.e. it is difficult to tell which are the most important features in the dataset after computing principal components.


The technique is linear, therefore any non-linear correlation between variables will not be captured.

Next in DDCS



Feature Selection and Extraction

- Signal basics and Fourier Series
- 1D and 2D Fourier Transform
- Another look at features
- PCA for dimensionality reduction
- **Convolutions**