CENG 463
Machine Learning

Lecture 17
Dimensionality Reduction

Dimensionality Reduction

One of the major components in analysis of multivariate data is dimensionality reduction.

This means creating a low-dimensional **projection** of the original data.

Low-dimensional means we have less number of features.

Dimensionality Reduction

One of the motivations is that our computation is faster with less number of features.

Check out these examples:

- Image data: each pixel of an image
  For a 64x64 gray-scale image there are 4096 features
- Genomic data: expressible pieces of the genes
  Several thousand features
- Text categorization: frequencies of phrases in a document
  More than ten thousand (phrases) features

Dimensionality Reduction

Another motivation is data visualization. Two or three dimensional projections help us

-To visually represent our data.
-To screen data for obvious outliers.
-To observe cluster tendency.

E.g. $x^{(i)} = \{x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)}, x_5^{(i)}\}$
A data set with five features projected to a 2D space:

$$x'^{(i)} = \{x'^{(i)}_1, x'^{(i)}_2\}$$

Note: $x_1^{(i)} \neq x'^{(i)}_1$

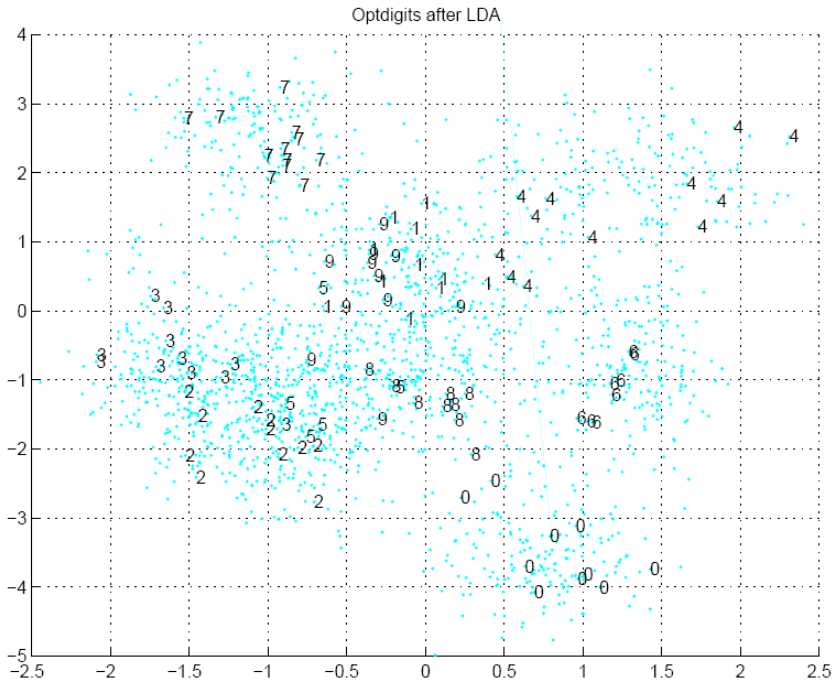


Figure from Introduction to Machine Learning 2ed., E Alpaydın, 2010.

Dimensionality Reduction

Inevitably, there will be some data loss.

Thus, the features that we use should be the most distinctive / representative ones.

How can we come up with the most distinctive features?

A popular technique is **Principal Component Analysis (PCA)**.

Principal Component Analysis

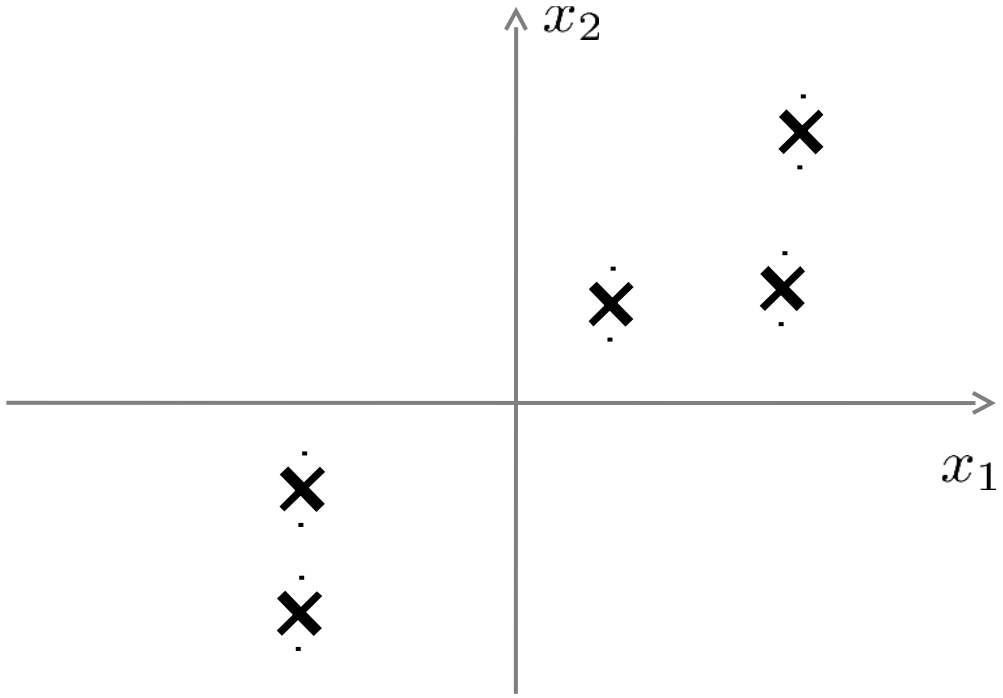
PCA uses linear algebra.

Specifically it is based on **eigen-decomposition** of data covariance matrix.

It changes the basis vectors and finds the best “subspace” that captures as much data variance as possible.

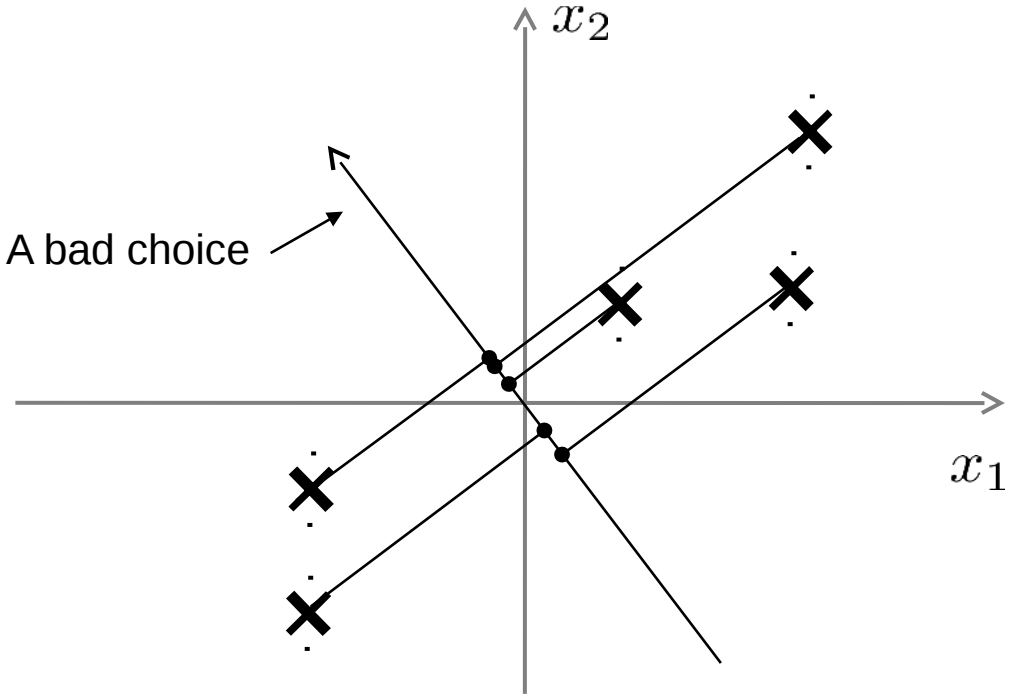
Principal Component Analysis

Example:
Reduce from 2-D to 1-D: Find **a direction** (a vector) onto which to project the data minimizes the projection error (and maximally represents the variance).



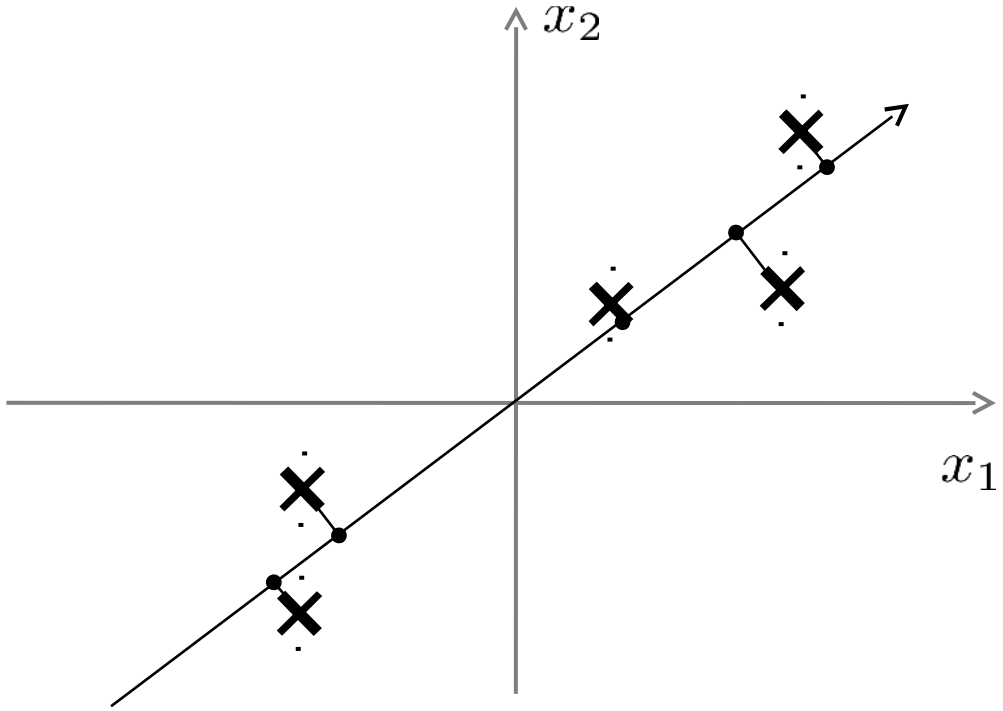
Principal Component Analysis

Example:
Reduce from 2-D to 1-D: Find **a direction** (a vector) onto which to project the data minimizes the projection error (and maximally represents the variance).



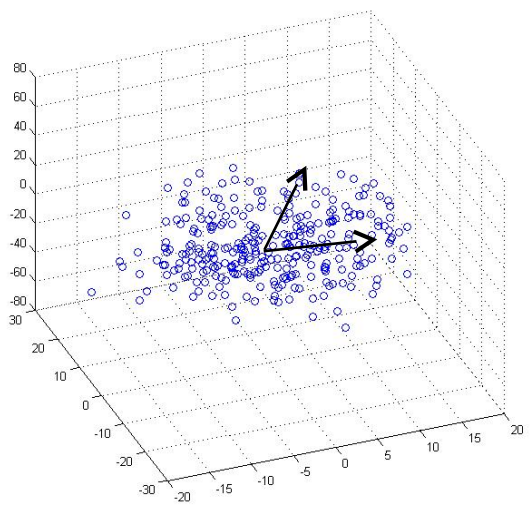
Principal Component Analysis

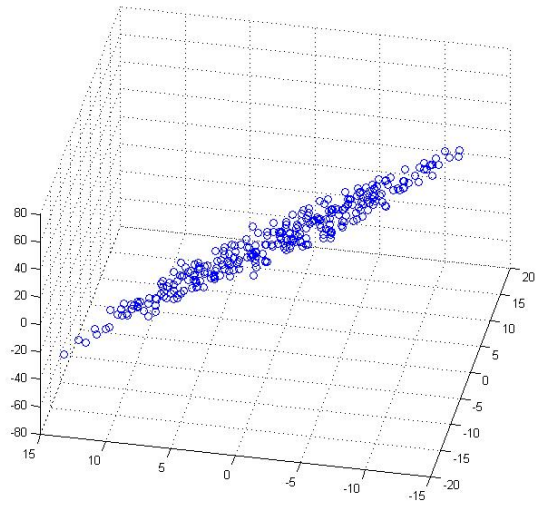
Example:
Reduce from 2-D to 1-D: Find **a direction** (a vector) onto which to project the data minimizes the projection error (and maximally represents the variance).

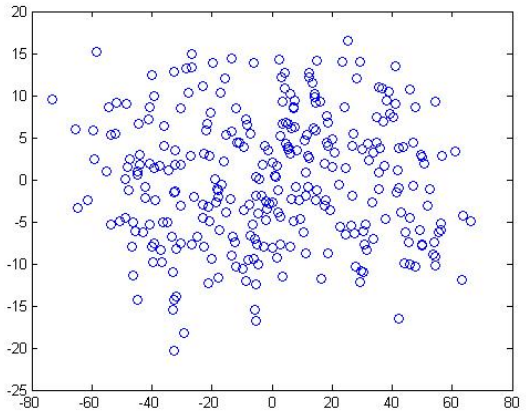


Principal Component Analysis

Example:
Reduce from 3-D to 2-D: Find two vectors ('best subspace') onto which to project the data minimizes the projection error.



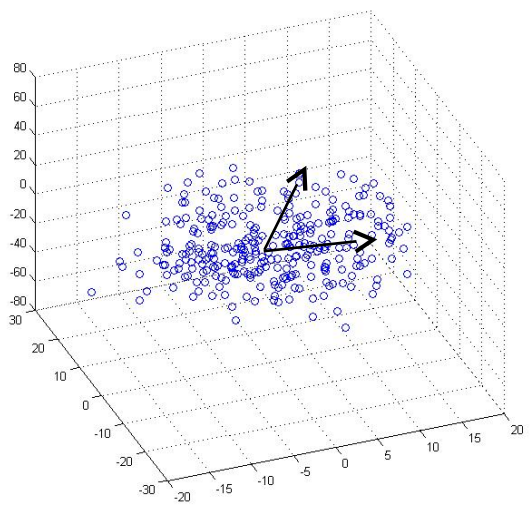


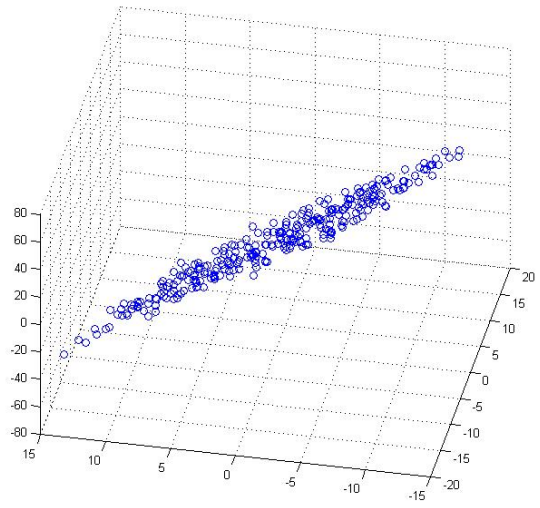


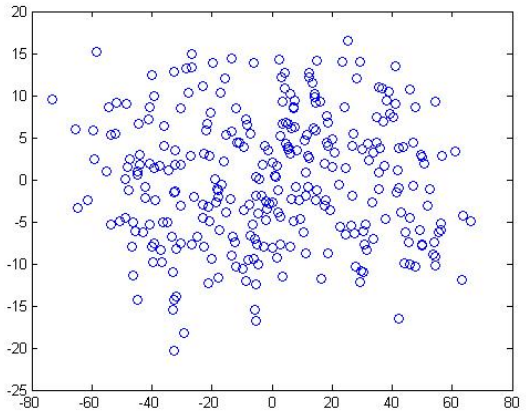
Figures from Stanford's Machine Learning Course given by Andrew Ng.

Principal Component Analysis

What we did here can be called as ***feature extraction***, because we created/extracted two new dimensions.
If we had chosen two of the original features (like x and y axes in the 3D figure), it would be ***feature selection***.







Figures from Stanford's Machine Learning Course given by Andrew Ng.

Covariance

Remember the mean and variance:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^i \qquad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x^i - \mu)^2$$

where x^i is the feature vector for sample i and μ and σ^2 are mean and variance vectors. For one feature:

$$\mu_1 = \frac{1}{m}\sum_{i=1}^{m} x_1^{\,i} \qquad \sigma_1^{\,2} = \frac{1}{m}\sum_{i=1}^{m} (x_1^{\,i} - \mu_1)^2$$

Covariance is similar to variance, but it measures how much the dimensions vary from the mean with respect to each other.

Covariance

Covariance is measured between two dimensions:

$$Cov(x_1, x_2) = \sigma_{12} = \frac{1}{m}\sum_{i=1}^{m}(x_1^i - \mu_1)(x_2^i - \mu_2)$$

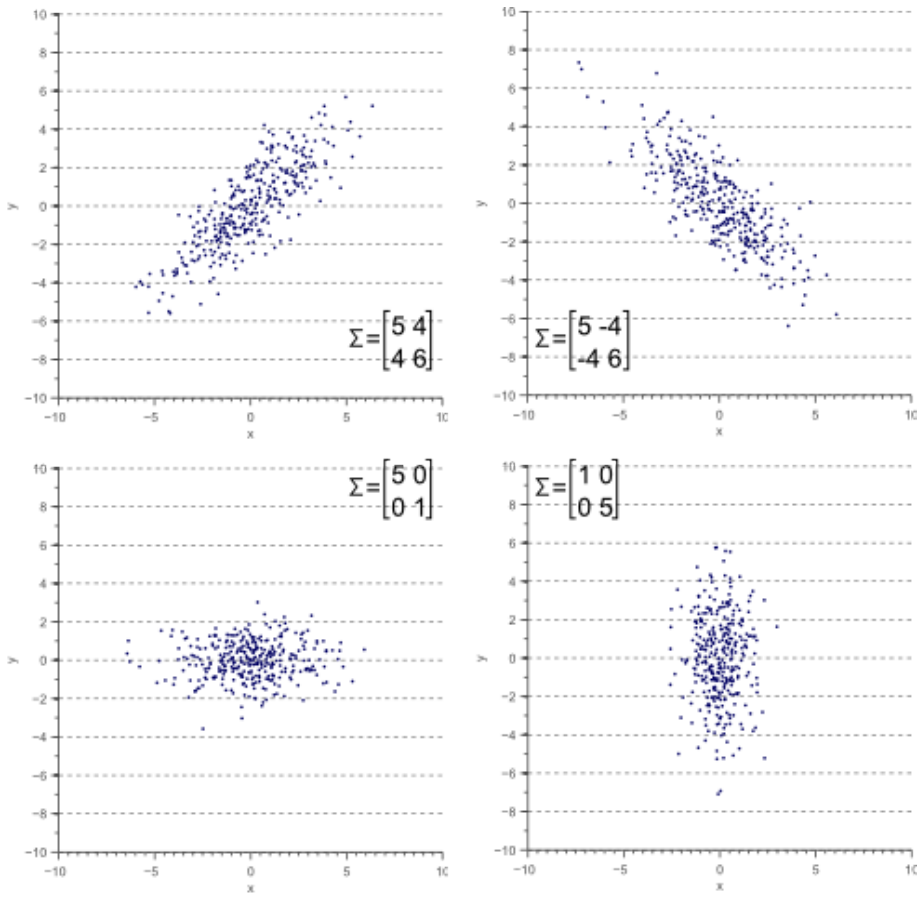
Note: $Cov(x_1, x_1) = \sigma_1^2 = \frac{1}{m}\sum_{i=1}^{m}(x_1^i - \mu_1)^2$

If you had a 3-D data set (x_1, x_2, x_3), then you need to measure the covariance between x_1 and x_2, x_2 and x_3, x_1 and x_3 separately.

Covariance

What does the covariance tell us?

- If Cov is positive, it means that both dimensions increase together.
- If Cov is negative, then as one dimension increases, the other decreases.
- If Cov is small, there is not much correlation.
- If Cov is close to zero, it indicates that the two dimensions are independent of each other.



Covariance

Exercise: Find the covariance between x_1 and x_2 dimensions in the following data set:

$$\sigma_{12} = \frac{1}{m}\sum_{i=1}^{m}(x_1^i - \mu_1)(x_2^i - \mu_2)$$

	1	2	3	4	5
x_1	10	40	20	22	28
x_2	41	12	32	20	20

What the result indicates?

Covariance

Exercise: Find the covariance between x_1 and x_2 dimensions in the following data set:

$$\sigma_{12} = \frac{1}{m}\sum_{i=1}^{m}(x_1^i - \mu_1)(x_2^i - \mu_2)$$

	1	2	3	4	5
x_1	10	40	20	22	28
x_2	41	12	32	20	20

$$\mu_1 = 24$$

$$\mu_2 = 25$$

What the result indicates?

Covariance

Exercise: Find the covariance between x_1 and x_2 dimensions in the following data set:

$$\sigma_{12} = \frac{1}{m}\sum_{i=1}^{m}(x_1^i - \mu_1)(x_2^i - \mu_2)$$

	1	2	3	4	5
x_1	10	40	20	22	28
x_2	41	12	32	20	20

$$\mu_1 = 24$$

$$\mu_2 = 25$$

$$\sigma_{12} = \frac{1}{5}(-14*16 - 16*13 - 4*7 + 2*5 - 4*5) = -94$$

$$\sigma_1^2 = 97.6$$

$$\sigma_2^2 = 104.8$$

What the result indicates?

Covariance

Exercise: Find the covariance between x_1 and x_2 dimensions in the following data set:

$$\sigma_{12} = \frac{1}{m}\sum_{i=1}^{m}(x_1^i - \mu_1)(x_2^i - \mu_2)$$

	1	2	3	4	5
x_1	10	40	20	22	28
x_2	41	12	32	20	20

$$\mu_1 = 24$$

$$\mu_2 = 25$$

$$\sigma_{12} = \frac{1}{5}(-14*16 - 16*13 - 4*7 + 2*5 - 4*5) = -94$$

$$\sigma_1^2 = 97.6$$

$$\sigma_2^2 = 104.8$$

What the result indicates?

It indicates a negative correlation. One dimension decreases as the other one increases.

Covariance Matrix

Covariances between all dimensions are represented with a matrix:

$$\Sigma \equiv \mathrm{Cov}(x) = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & & & \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

To vectorize the computation, one can use:

$$\Sigma = \frac{1}{m}(x - \mu)^{\mathrm{T}}_{d\mathrm{x}m} \cdot (x - \mu)_{m\mathrm{x}d}$$

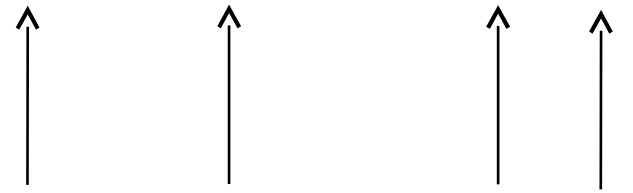
Eigenvectors and Eigenvalues

Vector u is an eigenvector of matrix A if there exists a constant $\gamma \neq 0$ such that $Au = \gamma u$

Then, γ is called an eigenvalue of A (associated with u)

Numerical example:

$$\begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 12 \\ 8 \end{bmatrix} = 4 \times \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$



$A \qquad u \qquad = \gamma \quad u$

Eigenvectors and Eigenvalues

Only square matrices may have eigenvectors.

A nxn matrix has n eigenvectors, each with its own eigenvalue.

Eigenvectors of a matrix are linearly independent of (orthogonal to) each other.

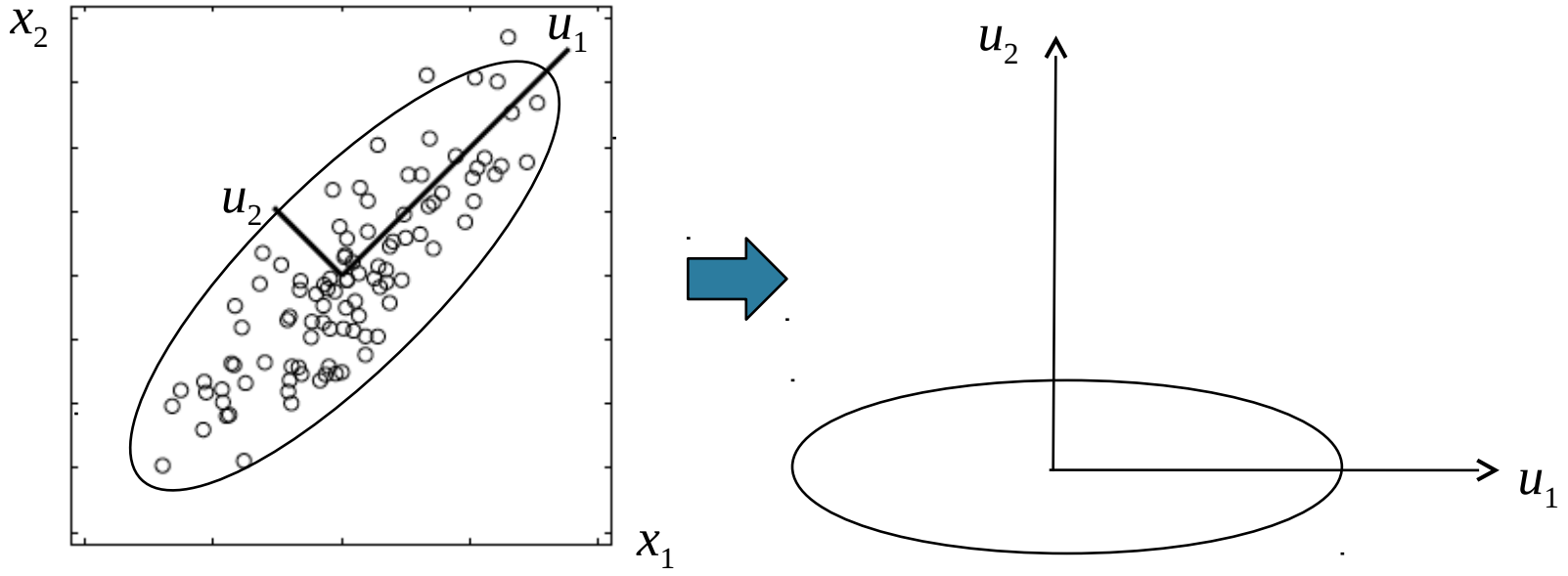
Note: ‘Eigen’ originates in the German language and can be loosely translated as “of itself”.

Eigenvalue of a matrix can be referred to as “value of itself”. Turkish translation is “özdeğer”.

Changing Basis Vectors

Is there another basis (principal components) that best re-expresses our data set?

Extracting eigenvectors and projecting our data on these vectors is a re-expression in another basis.



Putting All Together

Solving PCA: Eigenvectors of Covariance Matrix

SVD (Singular Value Decomposition) is a general method of extracting eigenvectors.

In PYTHON, *numpy.linalg.svd()* can be used to calculate the eigenvectors of the covariance matrix using SVD.

U, s, V = np.linalg.svd(Σ)

Putting All Together

```
U, s, V = np.linalg.svd(Σ)
```

$$U = \begin{bmatrix} | & | & | & | \\ u_1 & u_2 & \dots & u_n \\ | & | & | & | \end{bmatrix}_{nxn} \qquad S = \begin{bmatrix} \gamma_1 & \gamma_2 & \dots & \gamma_n \end{bmatrix}_{nx1}$$

U contains the eigenvectors,
s is a vector of eigenvalues.

Reducing the Number of Dimensions using Eigenvectors

To reduce data from n-dimensions to k-dimensions, We take k number of eigenvectors.

$$\mathrm{U}_{\text{reduce}} = \begin{bmatrix} | & | & | & | \\ u_1 & u_2 & \dots & u_k \\ | & | & | & | \end{bmatrix}_{n\text{x}k}$$

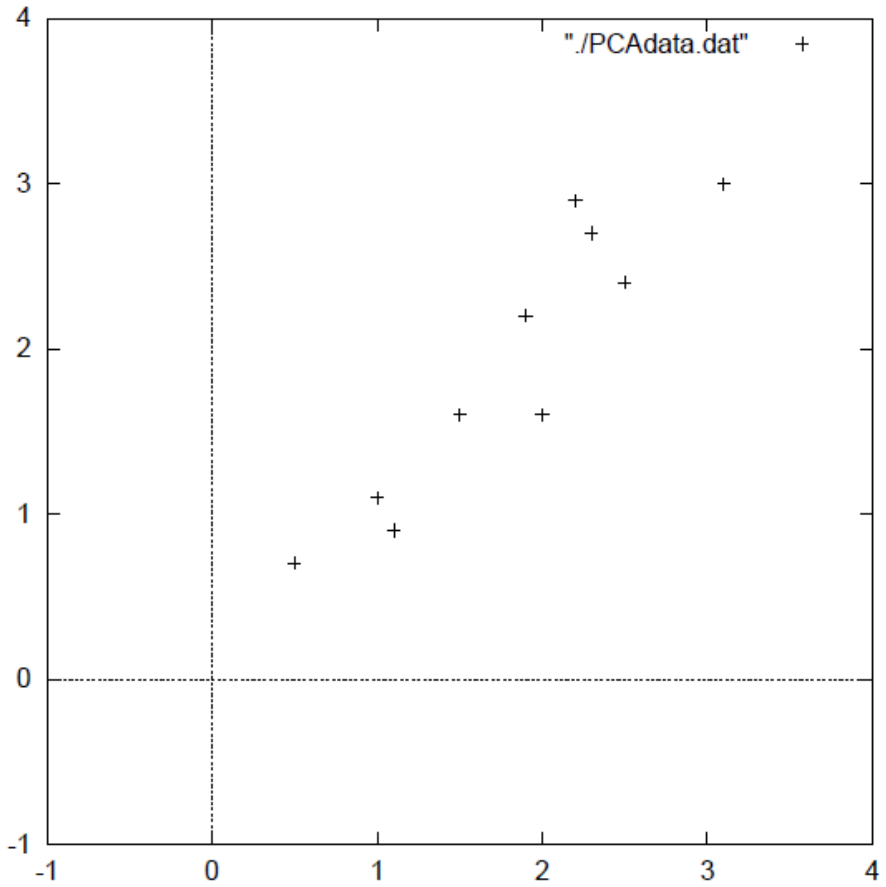
To project our data on this k-dimensional basis:

$$z = \mathrm{U}_{\text{reduce}}^{\mathrm{T}} \cdot x$$

$$z_{k\text{x}1}^{(i)} = \begin{bmatrix} - & u_1 & - \\ : & : & : \\ - & u_k & - \end{bmatrix}_{k\text{x}n} \cdot x_{n\text{x}1}^{(i)}$$

Numerical Example

Input Data:



$$\text{Data} = \begin{array}{c|c} x_1 & x_2 \\ \hline 2.5 & 2.4 \\ 0.5 & 0.7 \\ 2.2 & 2.9 \\ 1.9 & 2.2 \\ 3.1 & 3.0 \\ 2.3 & 2.7 \\ 2 & 1.6 \\ 1 & 1.1 \\ 1.5 & 1.6 \\ 1.1 & 0.9 \end{array} \quad x$$

Example from ‘A tutorial on Principal Components Analysis’ by Lindsay Smith.

Numerical Example

Preparing Covariance:

Data =

x	
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

DataAdjust =

x-μ	
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

$$\Sigma \equiv \mathrm{Cov}(x) = \begin{bmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{bmatrix}$$

Example from ‘A tutorial on Principal Components Analysis’ by Lindsay Smith.

Numerical Example

Calculate eigenvectors and eigenvalues:

$$[U, S, V] = \text{svd}(\Sigma)$$

$$U = \begin{bmatrix} -0.6779 & -0.7352 \\ -0.7352 & 0.6779 \end{bmatrix} \qquad S = \begin{bmatrix} 1.28 & 0 \\ 0 & 0.049 \end{bmatrix}$$

Example from ‘A tutorial on Principal Components Analysis’ by Lindsay Smith.

Numerical Example

Calculate eigenvectors and eigenvalues:

$$[U, S, V] = \text{svd}(\Sigma)$$

$$U = \begin{bmatrix} -0.6779 & -0.7352 \\ -0.7352 & 0.6779 \end{bmatrix} \qquad S = \begin{bmatrix} 1.28 & 0 \\ 0 & 0.049 \end{bmatrix}$$

u_1 u_2 γ_1 γ_2

Numerical Example

Calculate eigenvectors and eigenvalues:

$$[U, S, V] = \text{svd}(\Sigma)$$

$$U = \begin{bmatrix} -0.6779 & -0.7352 \\ -0.7352 & 0.6779 \end{bmatrix} \qquad S = \begin{bmatrix} 1.28 & 0 \\ 0 & 0.049 \end{bmatrix}$$

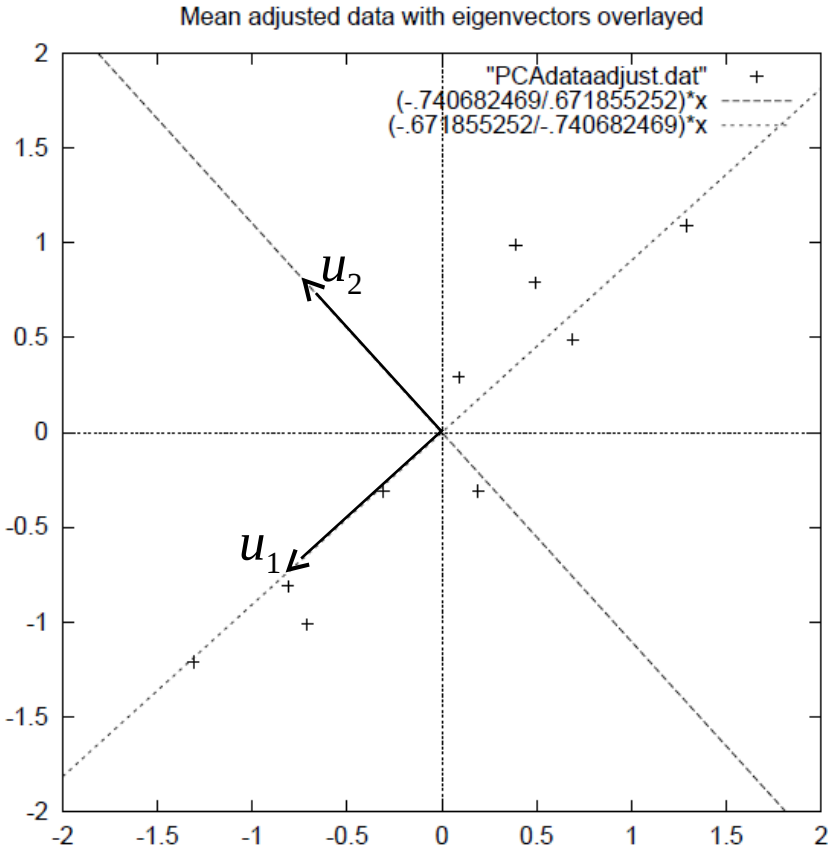
u_1 u_2 γ_1 γ_2

$\gamma_1 >> \gamma_2$ indicates that first eigenvactor is much more effective to represent the variance in our data.

Example from ‘A tutorial on Principal Components Analysis’ by Lindsay Smith.

Numerical Example

Eigenvectors are superimposed on the mean adjusted data:

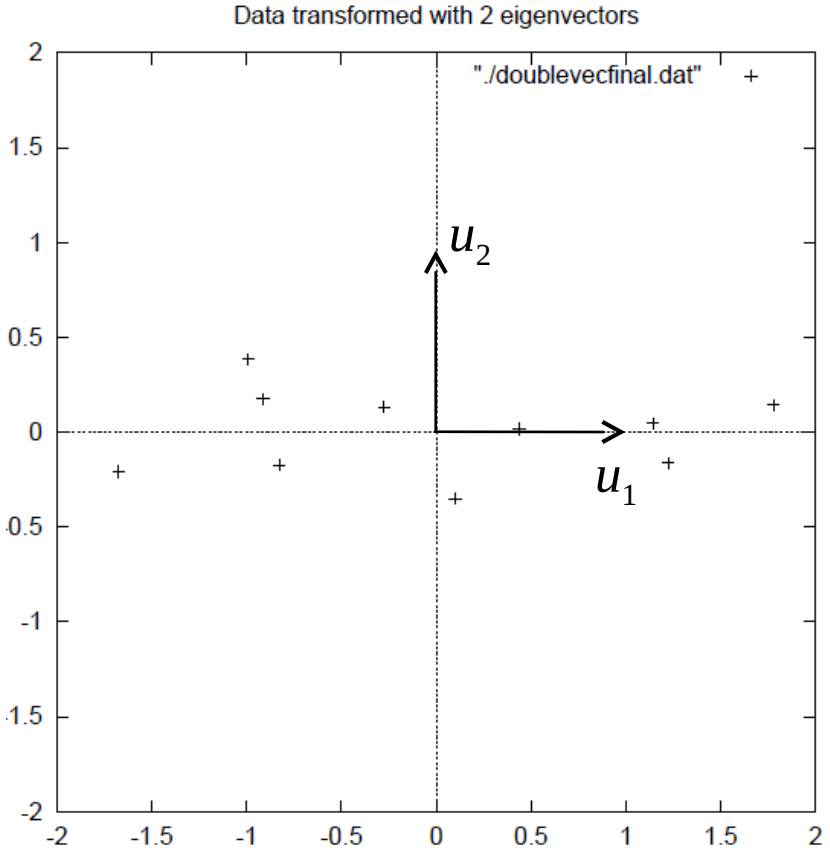


Example from ‘A tutorial on Principal Components Analysis’ by Lindsay Smith.

Numerical Example

Changing basis w/o dimension reduction: $z_{2xm} = \mathrm{U}^{\mathrm{T}}_{2x2} \cdot (x - \mu)_{2xm}$

Now, mean adjusted 2D data is on new coordinate system.



z

-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287

Example from 'A tutorial on Principal Components Analysis' by Lindsay Smith.

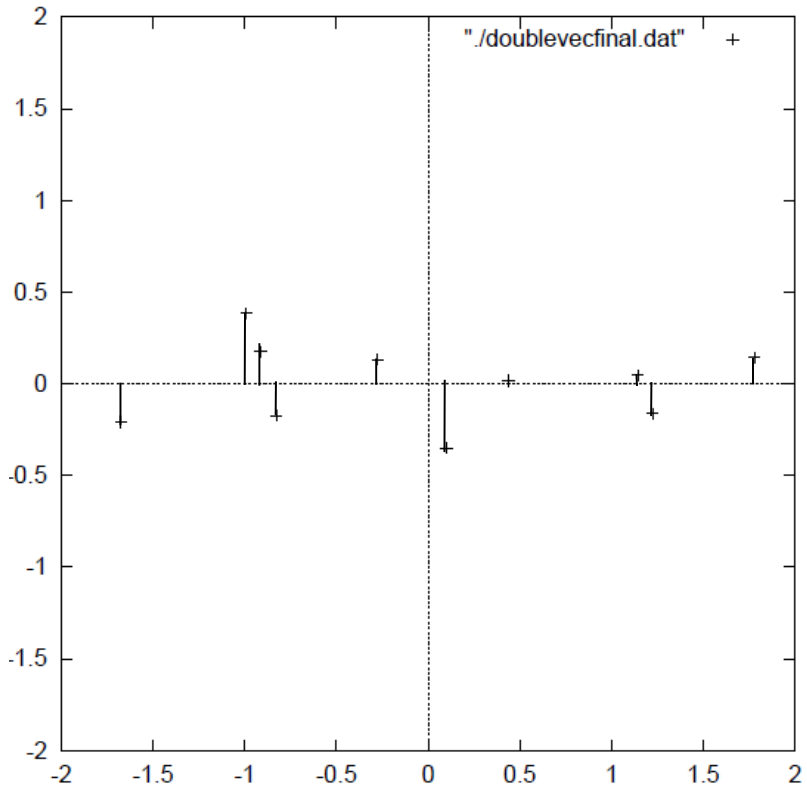
Numerical Example

Reducing from 2-D to 1-D: Use the first eigenvector since its representation capacity of our data is better.

$$U_{\text{reduce}} = \begin{bmatrix} -0.6779 \\ -0.7352 \end{bmatrix}$$

u_1

$$z'_{1xm} = U^{T}_{\text{reduce}} \cdot (x - \mu)_{2xm}$$



z'

-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056

Example from 'A tutorial on Principal Components Analysis' by Lindsay Smith.

How to Choose k ?

Q. How can we choose k (no of principle components)?
A. By looking at eigenvalues.

Method: Sort the eigenvalues in descending order and aggregate them to find the percentage of variation explained by the first k components.

Percentage of Variance (PoV):

$$\frac{\gamma_1 + \gamma_2 + \cdots + \gamma_k}{\gamma_1 + \gamma_2 + \cdots + \gamma_k + \cdots + \gamma_n}$$

Typically, stop at PoV>0.9.

How to Choose k ?

As eigenvalues decrease, one can stop at "elbow".

PoV vs. k. 90% may be a good point to stop.

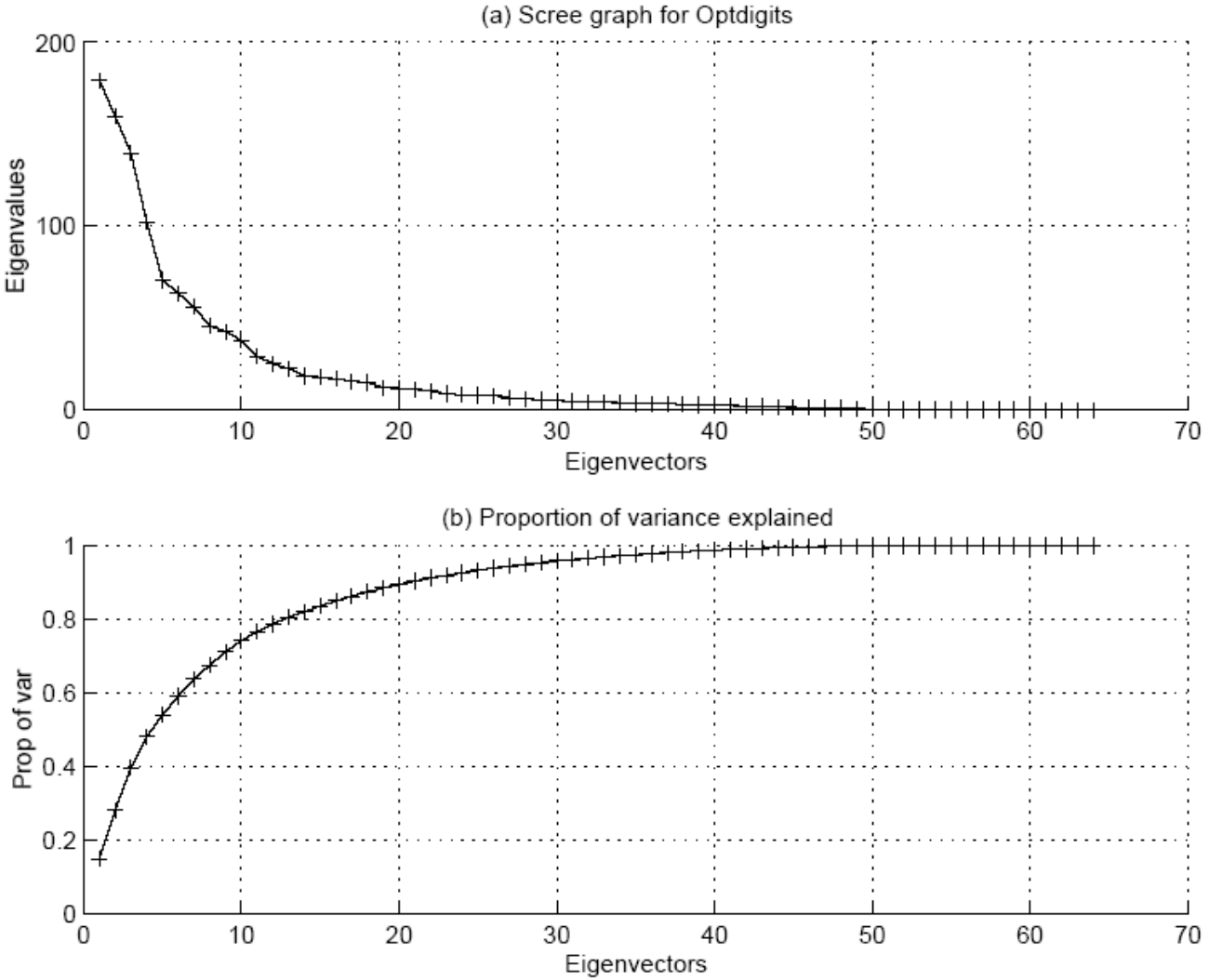


Figure from Introduction to Machine Learning 2ed., E Alpaydın, 2010.

An Example from Computer Vision

Let's see how PCA works on face images.

An image dataset that contains 32x32 grayscale images.

Each row of data matrix corresponds to one face image (a row vector of length 1024).

Example and figures from Stanford's Machine Learning Course given by Andrew Ng.

An Example from Computer Vision

With PCA, we obtain the principal components of the dataset. Each principal component (in U) is a vector of length 1024.

We can visualize these components by reshaping them into a 32x32 image matrix.

The first 36 principal components that describe the largest variations: ⟶

Example and figures from Stanford's Machine Learning Course given by Andrew Ng.

An Example from Computer Vision

One can use the principal components to reduce the dimension of the face dataset.

This allows to use the learning algorithm with a smaller input size (e.g., 36 or 100 dimensions) instead of the original 1024 dimensions.

This speeds up the learning algorithm.

An Example from Computer Vision

Original images of faces.

The faces reconstructed from only 100 principal components.

Example and figures from Stanford's Machine Learning Course given by Andrew Ng.

Summary

What we have seen is not a classification or a clustering algorithm. It is just a method that is used while working on multivariate data.

Low-dimensional projections of the original data is useful for any classification / clustering algorithm.

- It speeds up the algorithm.
- 2D and 3D projections enable us visually observe the data.