



**Northeastern University**  
College *of* Engineering

## **IE6600-Workshop**

*Visualizing with PCA*

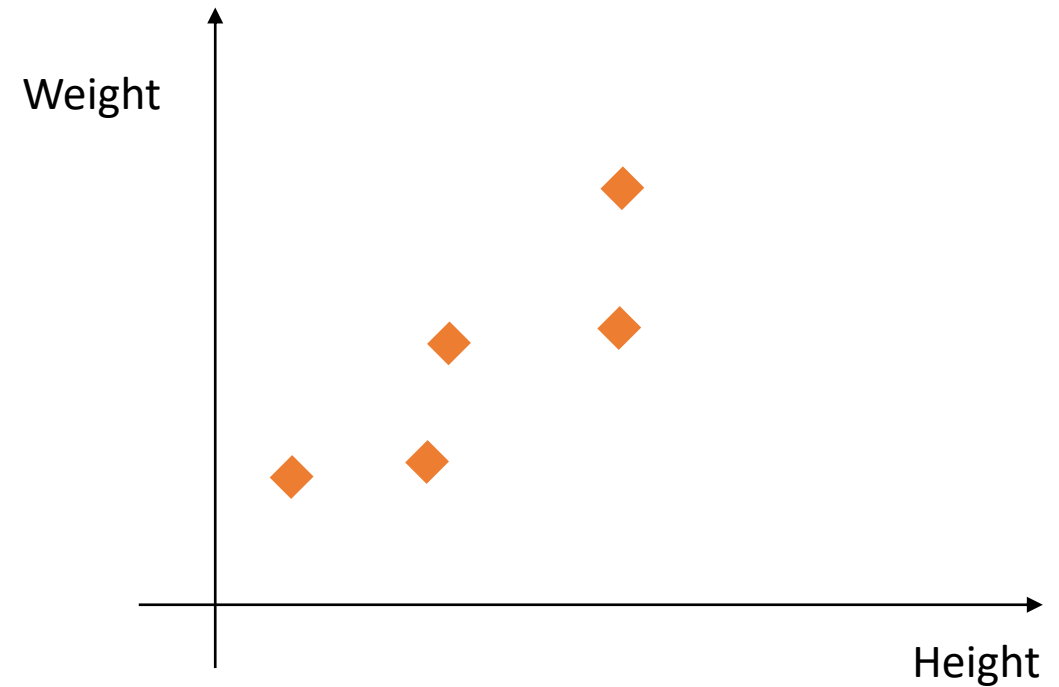
**Zhenyuan Lu**

# 1. Curse of dimensionality

# True vs. Observed Dimensionality

What is the dimensionality of this dataset?

ID	Observed Value ( $x_1$ ) - Height (cm)	Predictive Value ( $y$ ) - Weight (kg)
1	170	65
2	187	80
3	175	75
4	160	45
5	159	56

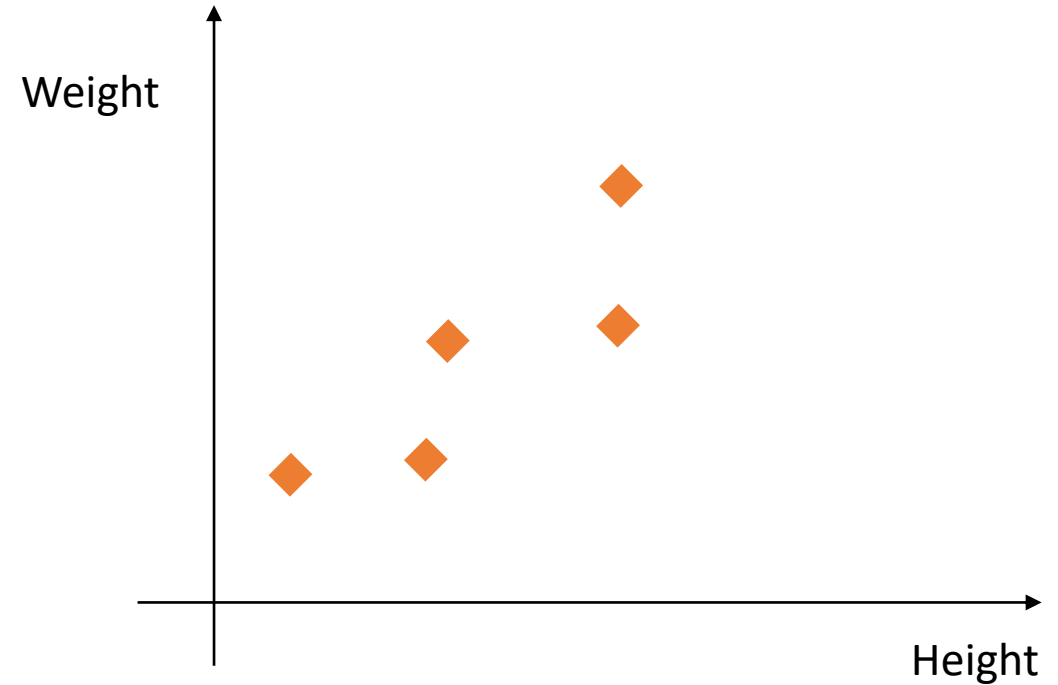


# True vs. Observed Dimensionality

What is the dimensionality of this dataset?

ID	Observed Value ( $x_1$ ) - Height (cm)	Predictive Value ( $y$ ) - Weight (kg)
1	170	65
2	187	80
3	175	75
4	160	45
5	159	56

Observed Dimensionality: 6



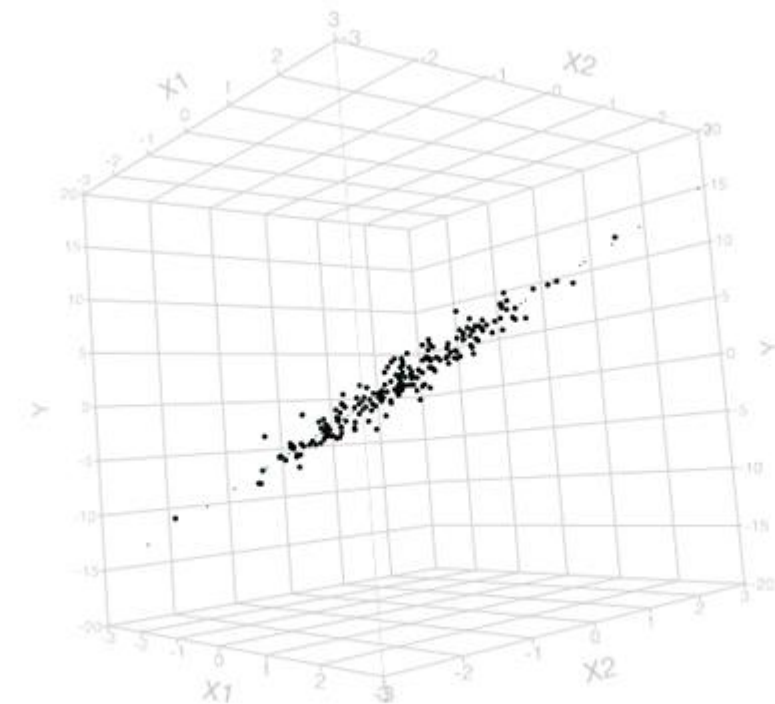
Features observed over time:

- $x_2$ : # of your waist (cm)
- $x_3$ : # of exams you have today
- $x_4$ : # of jokes you heard today
- $x_5$ : # of days left until summer

# True vs. Observed Dimensionality

True Dimensionality: 3

ID	Observed Value ( $x_1$ ) - Height (cm)	Observed value ( $x_2$ ) - Waist (cm)	Predictive Value ( $y$ ) - Weight (kg)
1	170	67	65
2	187	86	80
3	175	76	75
4	160	59	45
5	159	66	56



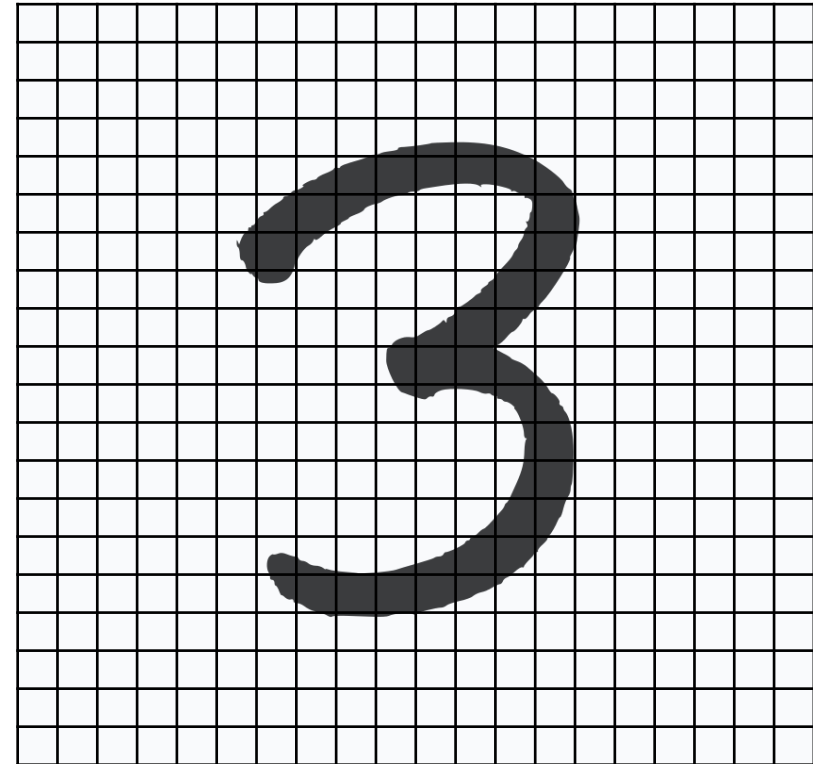
# Curse of Dimensionality

Example:

## True Dimensionality

- 20x20 bitmap:  $\{0, 1\}^{400}$  potential events
- The handwritten number **THREE** may be only  $2^{40}$  events

20x20 bitmap



1x400



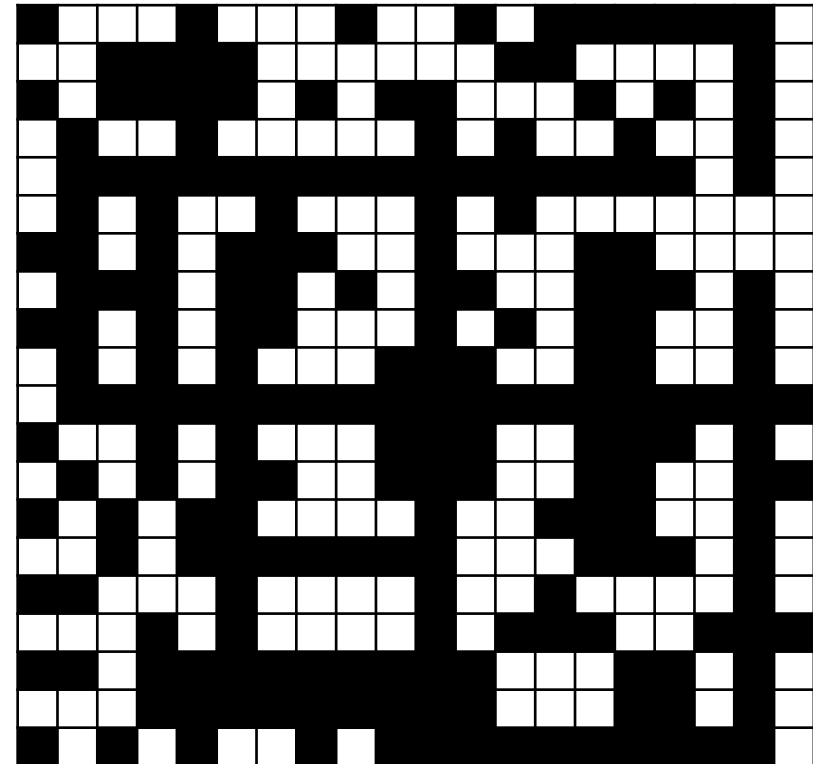
⋮

# Curse of Dimensionality

Example:

- 20x20 bitmap:  $\{0, 1\}^{400}$  potential events
- Randomly sampling  $2^{400}$  events

20x20 bitmap



1x400



⋮

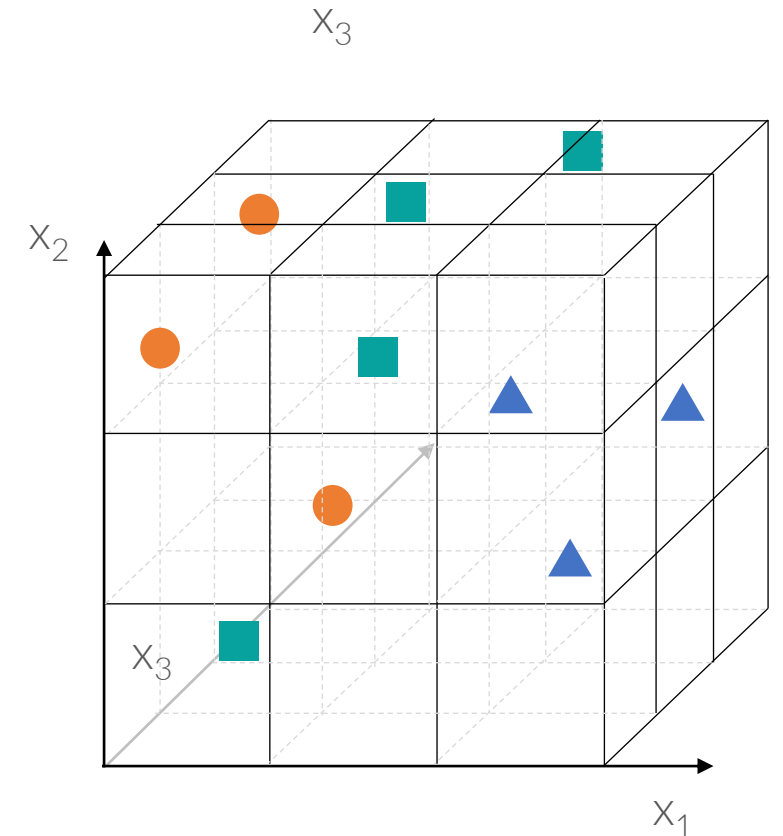
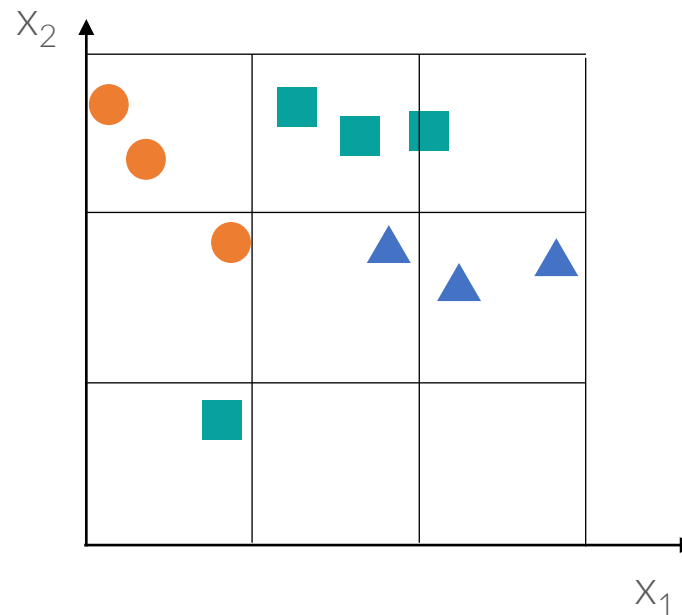
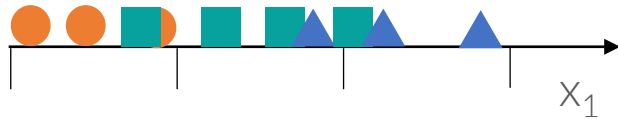
# Curse of Dimensionality

Data mining/Machine learning methods are statistical

- Use numbers to build up the predictor  $f(x)$
- Use categorical variables to classify:  $\{0, 1\}$

Dimensionality ( $d$ ) increases, and fewer observations ( $n$ ) per region  
In the case of  $d \gg n$

- 1d: 3 regions
- 2d:  $3^2$  regions
- 100d – well...





## 2.Methods of dealing with high dimensionality

# Methods *Dealing with high dimensionality*

## Use domain knowledge

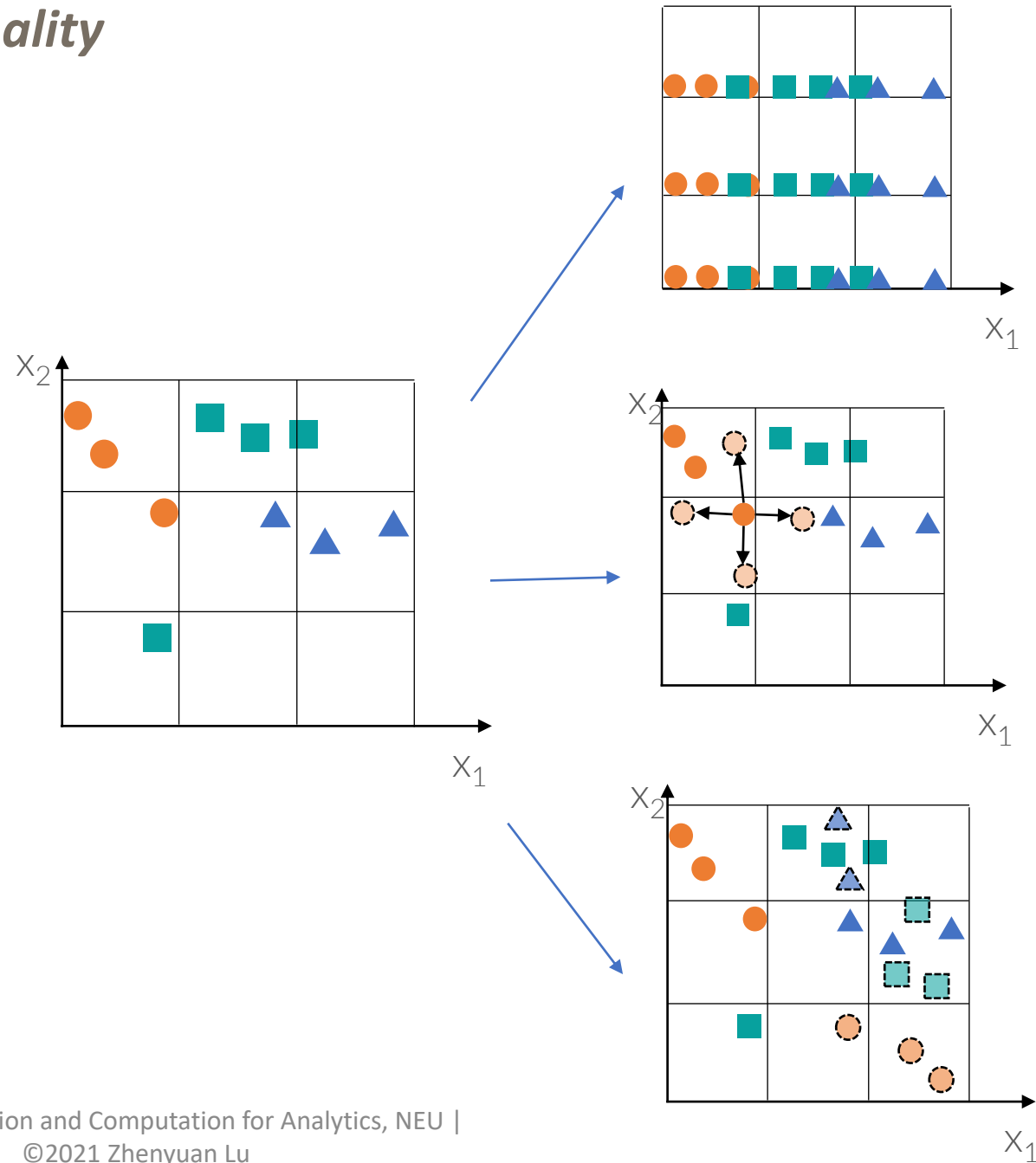
- Feature engineering
- e.g. Testosterone -> Muscle building

## Assumption on dimensions

- **Independence**: count along each dimension, e.g. Naïve bayes. When counting the frequency of  $x_1$  ignore the  $x_2$
- **Smoothness**: nearby region should have similar distribution of classes
- **Symmetry**: e.g. invariance to order of the dimensions: order doesn't matter

## Reduce the dimensionality

- Create a new set of variables



# Dimensionality Reduction

## Feature selection

- Select a subset of the original dimension:

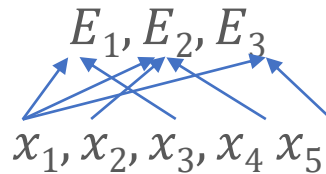
$$x_1, x_2, x_3, x_4, x_5$$

- Discriminative: Select class as predictor

## Feature extraction

- Construct a new set of dimensions:

$$E_i = f(x_1, x_2, x_3, x_4, x_5)$$



- e.g. Linear combination of original dimensions: PCA

# 3.Principal components analysis (PCA)

One of the most beautiful ALGORITHMs

# Principal Components Analysis (PCA)

The goal of PCA is to find a new set of dimensions (attributes) that better captures the variability of the data.

# Some Math

## Standard Basis Vector

d-dimensional Cartesian coordinate space is specified via the d unit vectors, called the standard basis vectors, along each of the axes. The j-th standard basis vector  $e_j$  is the d-dimensional unit vector whose j-th component is 1 and the rest of the components are 0

$$e_j = (0, 1_j, \dots, 0)^T$$

Any other vector in  $\mathbb{R}^d$  can be written as linear combination of the standard basis vectors. For example, each of the points  $x_i$  can be written as the linear combination

$$x_i = x_{i1}e_1 + x_{i2}e_2 + \dots + x_{id}e_d = \sum_{j=1}^d x_{ij}e_j$$

where the scalar value  $x_{ij}$  is the coordinate value along the j-th axis or attribute



# Standard Basis Vector

For example:

Consider the Iris data

$$x_1 = (5.9, 3.0, 4.2)$$

	sepal length $X_1$	sepal width $X_2$	petal length $X_3$	petal width $X_4$	class $X_5$
$x_1$	5.9	3.0	4.2	1.5	Iris-versicolor
$x_2$	6.9	3.1	4.9	1.5	Iris-versicolor
$x_3$	6.6	2.9	4.6	1.3	Iris-versicolor
$x_4$	4.6	3.2	1.4	0.2	Iris-setosa
$x_5$	6.0	2.2	4.0	1.0	Iris-versicolor
$x_6$	4.7	3.2	1.3	0.2	Iris-setosa
$x_7$	6.5	3.0	5.8	2.2	Iris-virginica
$x_8$	5.8	2.7	5.1	1.9	Iris-virginica
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x_{149}$	7.7	3.8	6.7	2.2	Iris-virginica
$x_{150}$	5.1	3.4	1.5	0.2	Iris-setosa

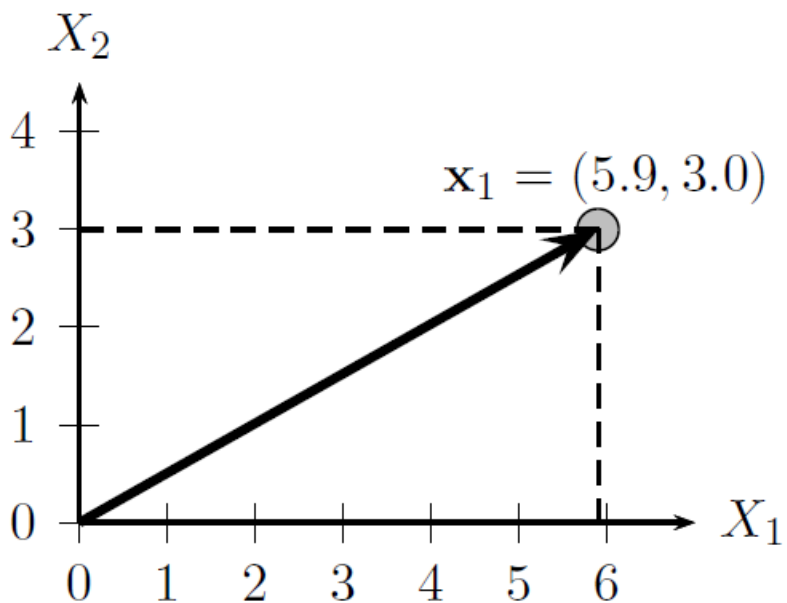
$$x_1 = 5.9e_1 + 3.0e_2 + 4.2e_3 = 5.9 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + 3.0 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + 4.2 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 5.9 \\ 3.0 \\ 4.2 \end{pmatrix}$$

# Geometric View

For example:

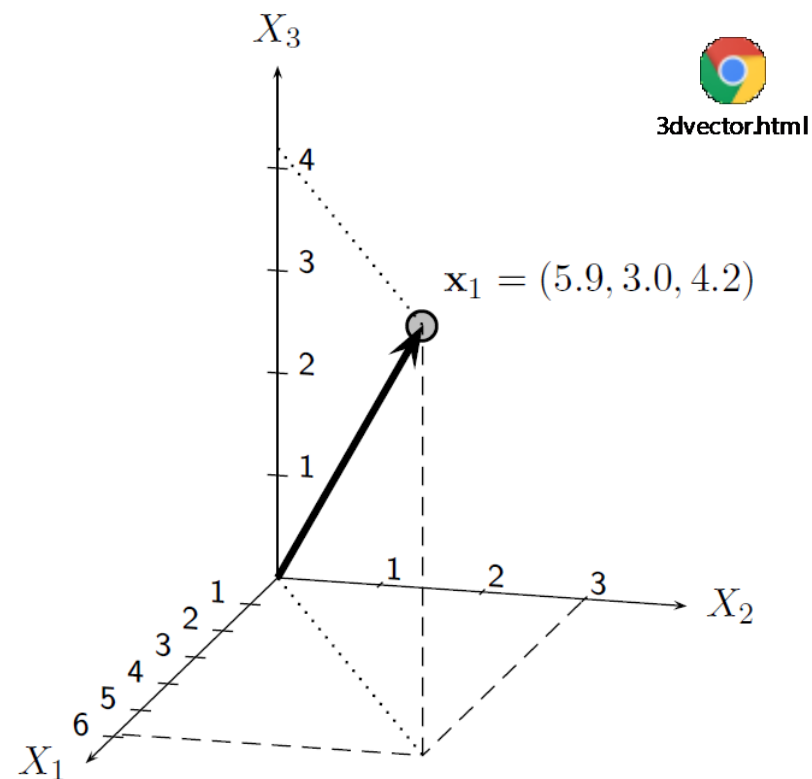
$$x_1 = (5.9, 3.0)$$

$$x_1 \in \mathbb{R}^2$$



$$x_1 = (5.9, 3.0, 4.2)$$

$$x_1 \in \mathbb{R}^3$$



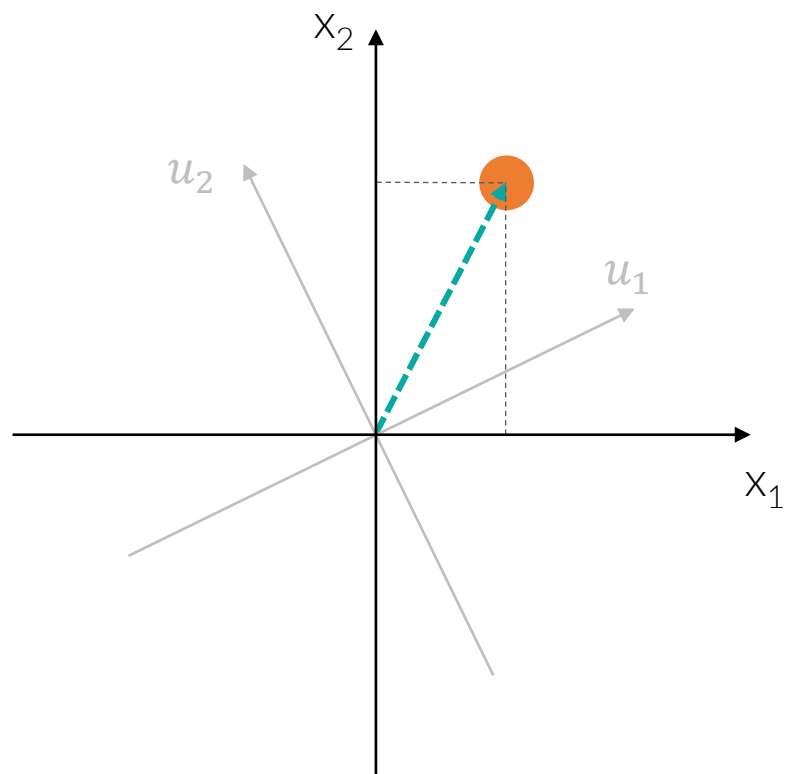
# Original Points to New Coordinates

Let the data  $\mathbf{D}$  consist of  $n$  points over  $d$  attributes, i.e., it is an  $n \times d$  matrix, given as

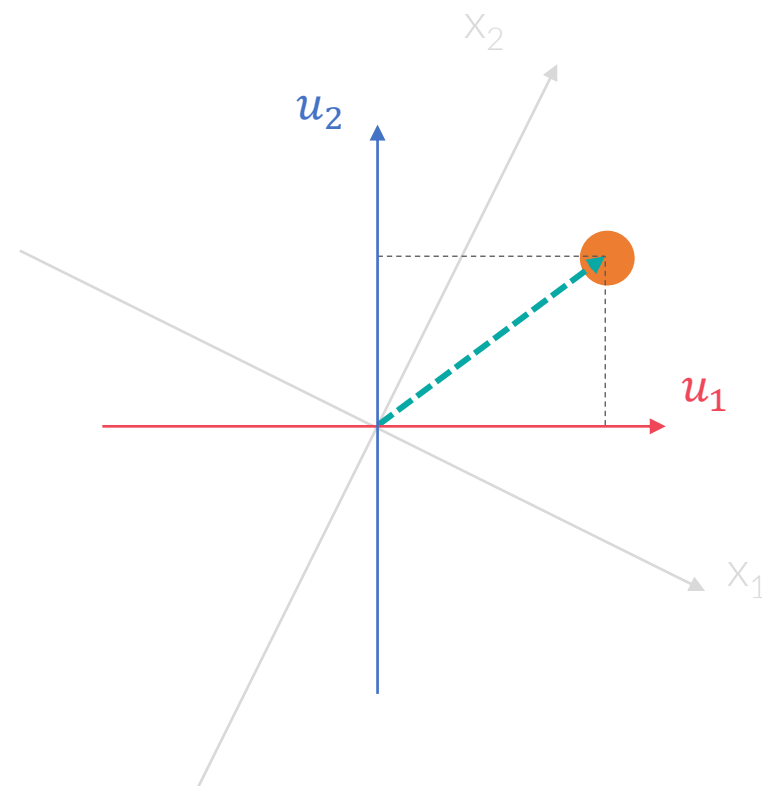
$$\mathbf{D} = \begin{pmatrix} & X_1 & X_2 & \cdots & X_d \\ \mathbf{x}_1 & x_{11} & x_{12} & \cdots & x_{1d} \\ \mathbf{x}_2 & x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_n & x_{n1} & x_{n2} & \cdots & x_{nd} \end{pmatrix}$$

Each point  $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{id})^T$  is a vector in the ambient  $d$ -dimensional vector space spanned by the  $d$  standard basis vectors  $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_d$ , where  $\mathbf{e}_i$  corresponds to the  $i$ -th attribute  $X_i$ . Recall that the standard basis is an orthonormal basis for the data space, i.e., the basis vectors are pair-wise orthogonal,  $\mathbf{e}_i^T \mathbf{e}_j = 0$ , and have unit length  $\|\mathbf{e}_i\| = 1$ .

# Original Points to New Coordinates



Original Basis



Optimal Basis

## Original Points to New Coordinates

As such, given any other set of  $d$  orthonormal vectors  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d$ , with  $\mathbf{u}_i^T \mathbf{u}_j = 0$  and  $\|\mathbf{u}_i\| = 1$  (or  $\mathbf{u}_i^T \mathbf{u}_i = 1$ ), we can re-express each point  $\mathbf{x}$  as the linear combination

$$\mathbf{x} = a_1 \mathbf{u}_1 + a_2 \mathbf{u}_2 + \dots + a_d \mathbf{u}_d$$

where the vector  $\mathbf{a} = (a_1, a_2, \dots, a_d)^T$  represents the coordinates of  $\mathbf{x}$  in the new basis. The above linear combination can also be expressed as a matrix multiplication

$$\mathbf{x} = \mathbf{U} \mathbf{a}$$

where  $\mathbf{U}$  is the  $d \times d$  matrix, whose  $i$ -th column comprises the  $i$ -th basis vector  $\mathbf{u}_i$

$$\mathbf{U} = \begin{pmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_d \\ | & | & & | \end{pmatrix}$$

# Original Points to New Coordinates

The matrix  $\mathbf{U}$  is an *orthogonal* matrix, whose columns, the basis vectors, are *orthonormal*, i.e., they are pairwise orthogonal and have unit length

$$\mathbf{u}_i^T \mathbf{u}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Since  $\mathbf{U}$  is orthogonal, this means that its inverse equals its transpose

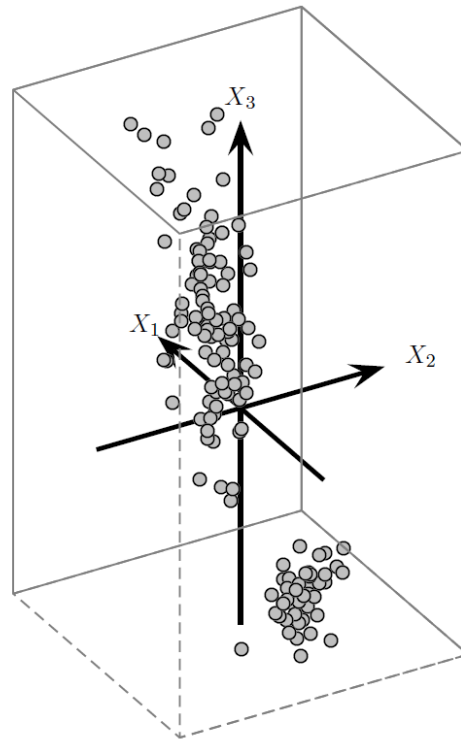
$$\mathbf{U}^{-1} = \mathbf{U}^T$$

which implies that  $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ , where  $\mathbf{I}$  is the  $d \times d$  identity matrix.

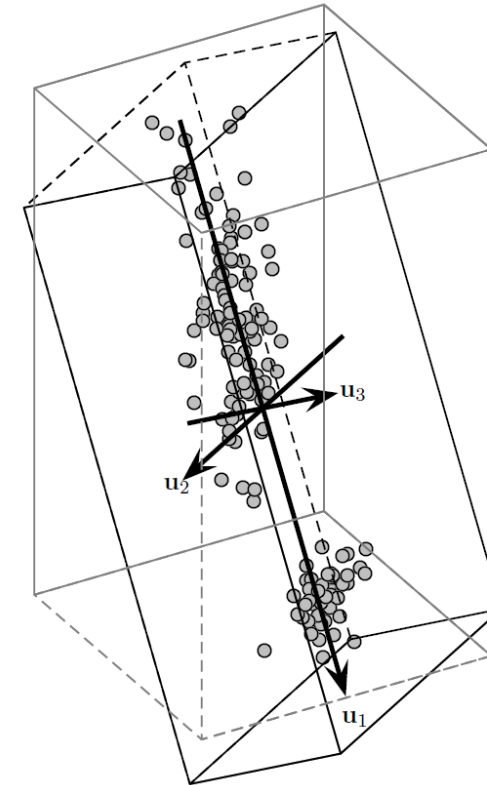
$$\begin{aligned} x &= Ua \\ U^T x &= U^T Ua \\ a &= U^T x \end{aligned}$$

# Original Points to New Coordinates

For example:



Original Basis



Optimal Basis

Mohammed J. Zaki, Wagner Meira, Jr., 2014

# Original Points to New Coordinates

For example:

$$u_1 = \begin{pmatrix} -0.390 \\ 0.089 \\ -0.916 \end{pmatrix} \quad u_2 = \begin{pmatrix} -0.639 \\ -0.742 \\ 0.200 \end{pmatrix} \quad u_3 = \begin{pmatrix} -0.663 \\ 0.664 \\ 0.346 \end{pmatrix}$$

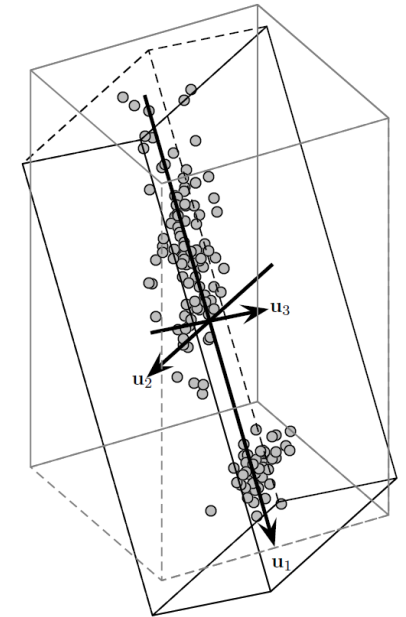
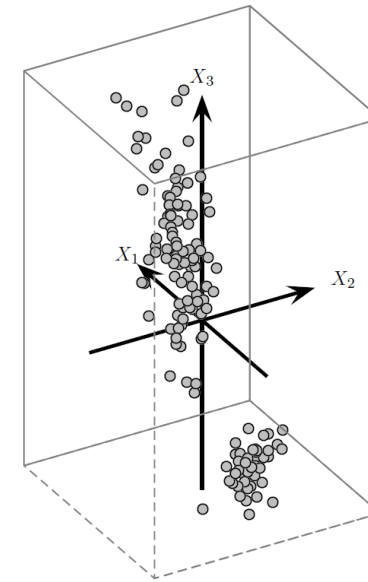
The new coordinates of the centered point

$x = (-0.343, -0.754, 0.241)^T$  can be computed as:

$$a = U^T x = \begin{pmatrix} -0.390 & 0.089 & -0.916 \\ -0.639 & -0.742 & 0.200 \\ -0.663 & 0.664 & 0.346 \end{pmatrix} \begin{pmatrix} -0.390 \\ 0.089 \\ -0.916 \end{pmatrix} = \begin{pmatrix} -0.154 \\ 0.828 \\ -0.190 \end{pmatrix}$$

$x$  can be written as the linear combination

$$x = -0.154u_1 + 0.828u_2 - 0.190u_3$$

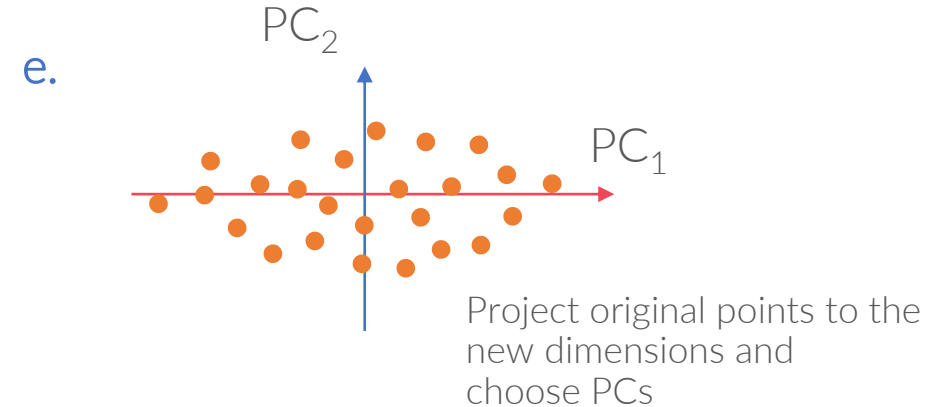
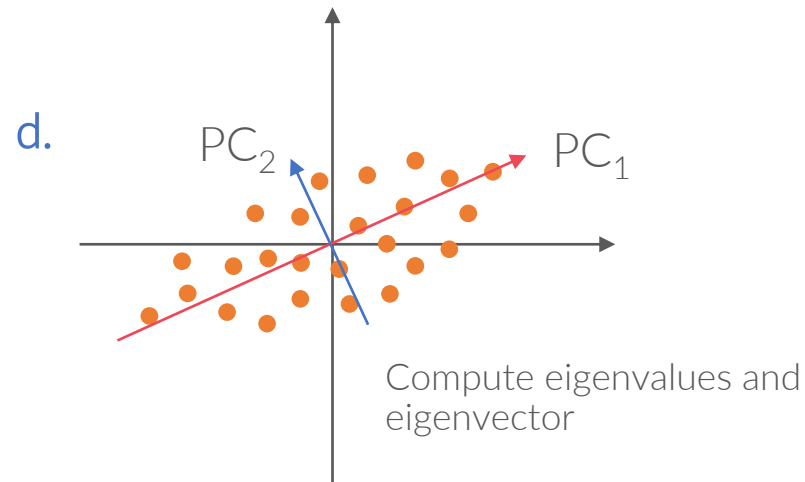
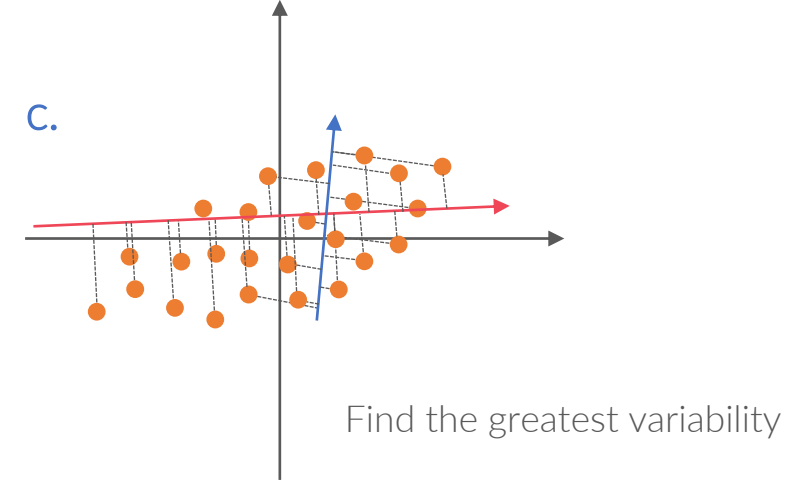
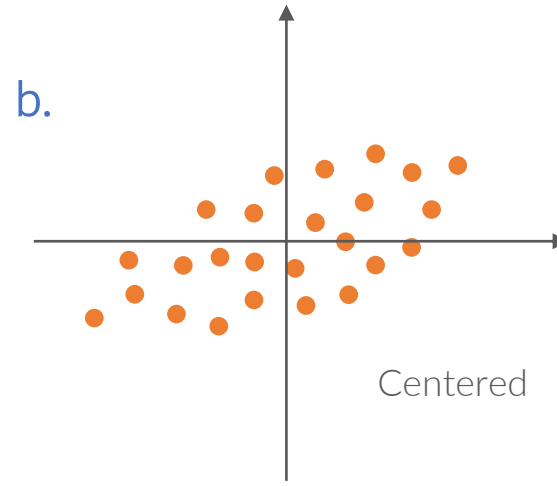
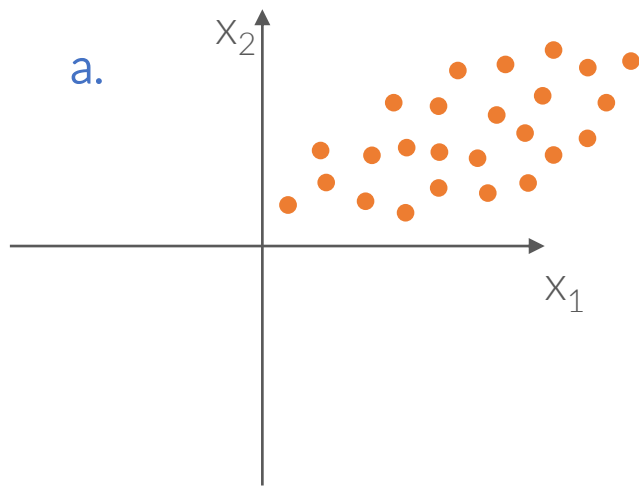


$$x = Ua$$
$$a = U^T x$$



# PCA

1. Find the **first dimension** to capture **as much of the variability** as possible
2. The **second dimension** is orthogonal to the first, and subject to that constraint, captures as much of the remaining variability.
3. And so on...until the **d<sup>th</sup> dimension**.



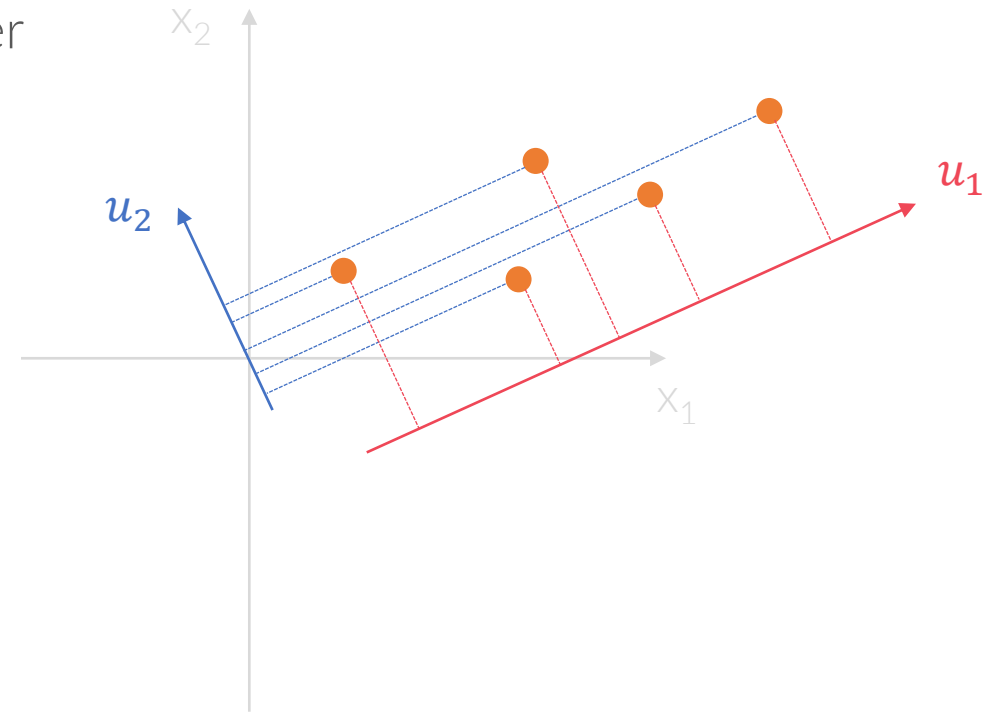
## PCA Why find greatest variability?

### Example:

We have a 2 dimensional data here to project all the data points to 1 dimension axis  $u_1, u_2 : \{x_1, x_2\} \rightarrow u_1, u_2$

The data points in  $u_1$ -space are more expanded (greater variability) than in  $u_2$ -space.

1. Points are close in  $u_2$ -space but far in  $(x_1, x_2)$ -space which is not so ideal to represent the original dataset
2. The overall distances in  $u_1$ -space, with the highest variability, can represent the original distribution and variability



## PCA *Find mean and center the data*

1. Center the data at zero:  $Z = D - \mathbf{1} \cdot \mu^T$

$$\mathbf{Z} = \mathbf{D} - \mathbf{1} \cdot \mu^T = \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{pmatrix} - \begin{pmatrix} \mu^T \\ \mu^T \\ \vdots \\ \mu^T \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^T - \mu^T \\ \mathbf{x}_2^T - \mu^T \\ \vdots \\ \mathbf{x}_n^T - \mu^T \end{pmatrix} = \begin{pmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_n^T \end{pmatrix}$$

Mohammed J. Zaki, Wagner Meira, Jr., Data Mining and Analysis: Fundamental Concepts and Algorithms, Cambridge University Press

## PCA Direction

1. Center the data at zero:  $Z = D - 1 \cdot \mu^T$
2. Covariance matrix  $\Sigma = \frac{1}{n} (Z^T Z)$ 
  - covariance of  $x_1, x_2$ :
  - $x_1, x_2$  increase or decrease together or when one decreases the other one increases

### Example

$$\text{cov}(x_i, x_j) = \frac{(x_i - \mu_i)^T (x_j - \mu_j)}{n}, \text{ After data centered: } \mu = 0, \text{cov}(x_i, x_j) = \frac{x_i^T x_j}{n}$$

Say we have two sets of attributes  $\sigma_1^2 = 2, \sigma_2^2 = 0.6$

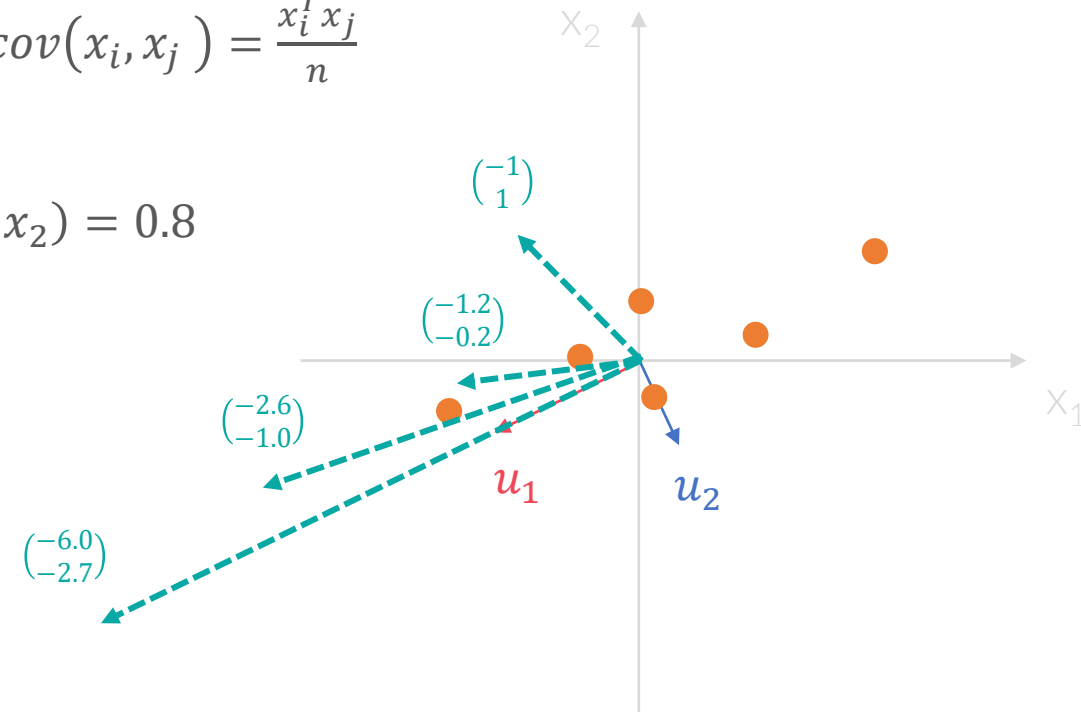
$$\begin{matrix} & x_1 & x_2 \\ x_1 & \begin{pmatrix} 2.0 & 0.8 \end{pmatrix} \\ x_2 & \begin{pmatrix} 0.8 & 0.6 \end{pmatrix} \end{matrix} \quad \text{cov}(x_1, x_2) = 0.8$$

3. Multiply a vector  $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$  by  $\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}$ :

$$\text{a. } \begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1.2 \\ -0.2 \end{pmatrix} \quad \text{b. } \begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} -1.2 \\ -0.2 \end{pmatrix} = \begin{pmatrix} -2.6 \\ -1.0 \end{pmatrix}$$

$$\text{c. } \begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} -2.6 \\ -1.0 \end{pmatrix} = \begin{pmatrix} -6.0 \\ -2.7 \end{pmatrix} \rightarrow \begin{pmatrix} -14.1 \\ -6.4 \end{pmatrix} \rightarrow \begin{pmatrix} -33.3 \\ -15.1 \end{pmatrix}$$

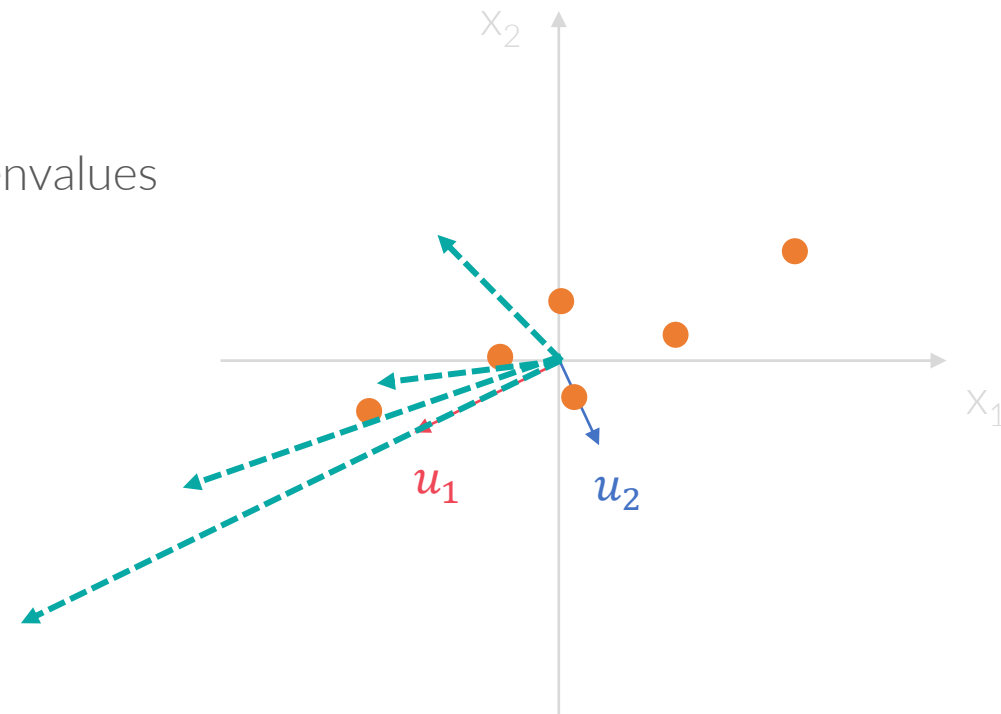
Slope: 0.45      0.454      0.454



Towards the greatest variance direction

## PCA Direction

4. Look for a vector always keep in the same direction:  $\Sigma u = \lambda u$
- $u$ : eigenvectors
  - $\Sigma$ : covariance matrix
  - $\lambda$ : scalar variable
  - Principal components = eigenvectors with largest eigenvalues



Towards the greatest variance direction

## PCA Find eigenvalues, eigenvector, and PCs

1. Find eigenvalues by solving :  $\Sigma \mathbf{u} = \lambda \mathbf{u} \rightarrow |\Sigma - \lambda I| = 0$ . (**NOTE: Determinant of matrix A:  $|A|$** )

$$- \begin{vmatrix} 2.0 - \lambda & 0.8 \\ 0.8 & 0.6 - \lambda \end{vmatrix} = (2 - \lambda)(0.6 - \lambda) - 0.8 * 0.8 = \lambda^2 - 2.6\lambda + 0.56 = 0$$

$$- \{\lambda_1, \lambda_2\} = \frac{1}{2}(2.6 \pm \sqrt{2.6^2 - 4 * 0.56}) = \{2.36, 0.23\}$$

2. Find  $i^{th}$  eigenvector by solving:  $\Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$

$$- \begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{12} \end{pmatrix} = 2.36 \begin{pmatrix} u_{11} \\ u_{12} \end{pmatrix}$$

$$\rightarrow \begin{matrix} 2u_{11} + 0.8u_{12} = 2.36u_{11} \\ 0.8u_{11} + 0.6u_{12} = 2.36u_{12} \end{matrix} \rightarrow u_{11} = 2.2u_{12} \rightarrow u_1 \sim \begin{pmatrix} 2.2 \\ 1 \end{pmatrix},$$

$$\rightarrow \text{make } ||u_1|| = 1, u_1 \frac{1}{\sqrt{2.2^2 + 1}}, \text{ then } u_1 = \begin{pmatrix} 0.91 \\ 0.41 \end{pmatrix}, \text{ slope} = 0.454$$

$$- \begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} u_{21} \\ u_{22} \end{pmatrix} = 0.23 \begin{pmatrix} u_{21} \\ u_{22} \end{pmatrix} \rightarrow \text{then } u_2 = \begin{pmatrix} -0.41 \\ 0.91 \end{pmatrix}$$

3. 1<sup>st</sup> PC:  $\begin{pmatrix} 0.91 \\ 0.41 \end{pmatrix}$ , 2<sup>nd</sup> PC:  $\begin{pmatrix} -0.41 \\ 0.91 \end{pmatrix}$

## PCA *Fraction of total variance, and choose dimensionality*

Often we may not know how many dimensions,  $r$ , to use for a good approximation. One criteria for choosing  $r$  is to compute the fraction of the total variance captured by the first  $r$  principal components, computed as

$$f(r) = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_r}{\lambda_1 + \lambda_2 + \cdots + \lambda_d} = \frac{\sum_{i=1}^r \lambda_i}{\sum_{i=1}^d \lambda_i} = \frac{\sum_{i=1}^r \lambda_i}{\text{var}(D)}$$

Given a certain desired variance threshold, say  $\alpha$ , starting from the first principal component, we keep on adding additional components, and stop at the smallest value  $r$ , for which  $f(r) \geq \alpha$  ( $\alpha$  can be 0.9, 0.95 as purposes).

In practice,  $\alpha$  is usually set to 0.9 or higher, so that the reduced dataset captures at least 90% of the total variance.

---

**Algorithm 7.1:** Principal Component Analysis

---

**PCA** ( $\mathbf{D}, \alpha$ ):

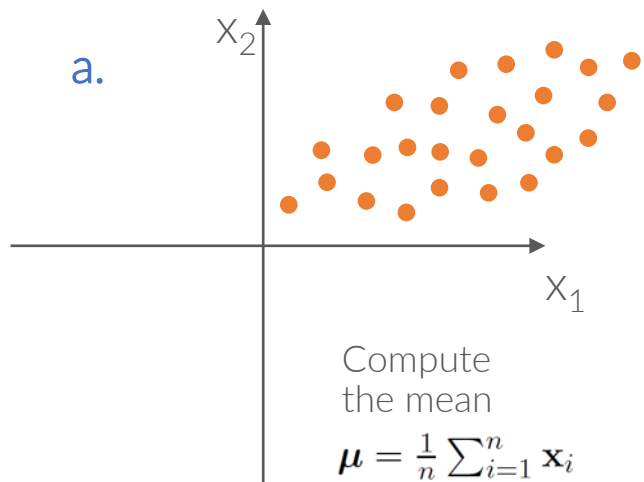
- 1  $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$  // compute mean
  - 2  $\mathbf{Z} = \mathbf{D} - \mathbf{1} \cdot \boldsymbol{\mu}^T$  // center the data
  - 3  $\boldsymbol{\Sigma} = \frac{1}{n} (\mathbf{Z}^T \mathbf{Z})$  // compute covariance matrix
  - 4  $(\lambda_1, \lambda_2, \dots, \lambda_d) = \text{eigenvalues}(\boldsymbol{\Sigma})$  // compute eigenvalues
  - 5  $\mathbf{U} = (\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_d) = \text{eigenvectors}(\boldsymbol{\Sigma})$  // compute eigenvectors
  - 6  $f(r) = \frac{\sum_{i=1}^r \lambda_i}{\sum_{i=1}^d \lambda_i}$ , for all  $r = 1, 2, \dots, d$  // fraction of total variance
  - 7 Choose smallest  $r$  so that  $f(r) \geq \alpha$  // choose dimensionality
  - 8  $\mathbf{U}_r = (\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_r)$  // reduced basis
  - 9  $\mathbf{A} = \{\mathbf{a}_i \mid \mathbf{a}_i = \mathbf{U}_r^T \mathbf{x}_i, \text{ for } i = 1, \dots, n\}$  // reduced dimensionality data
- 

Mohammed J. Zaki, Wagner Meira, Jr., Data Mining and Analysis: Fundamental Concepts and Algorithms

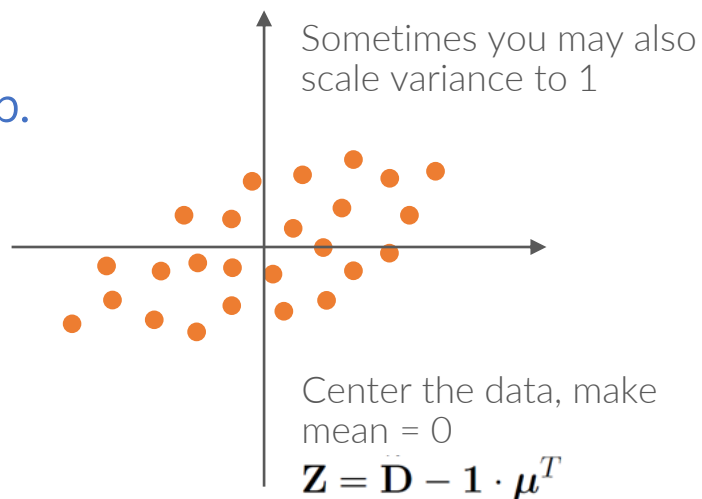


# PCA

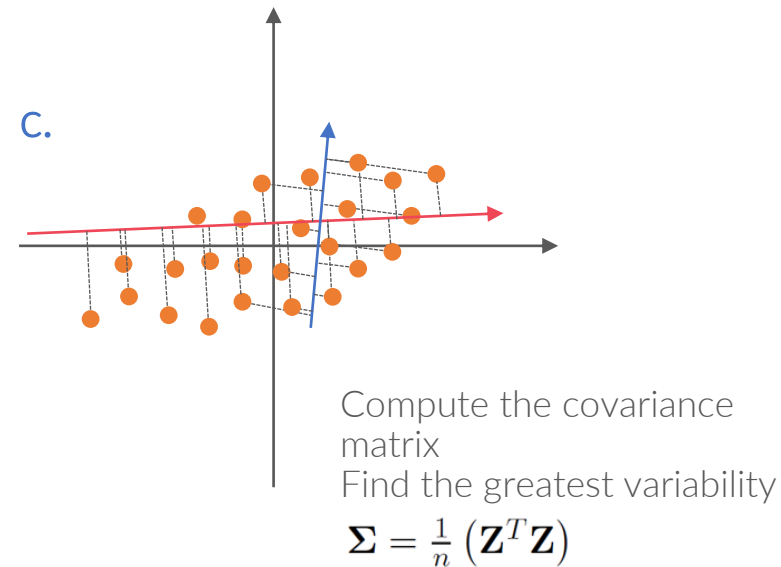
a.



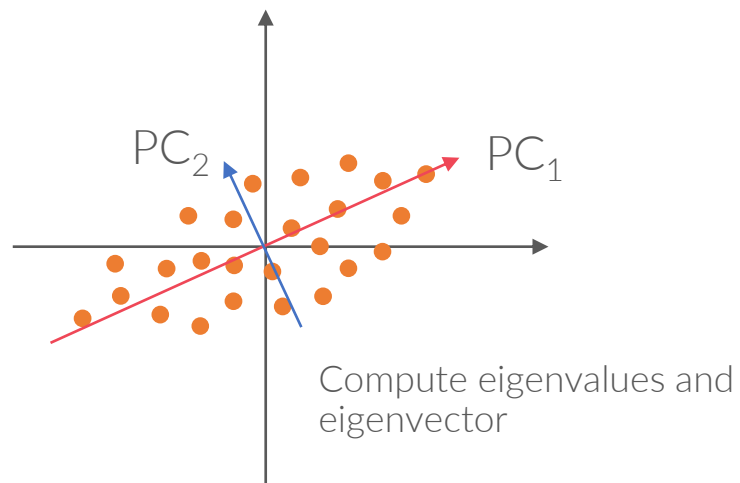
b.



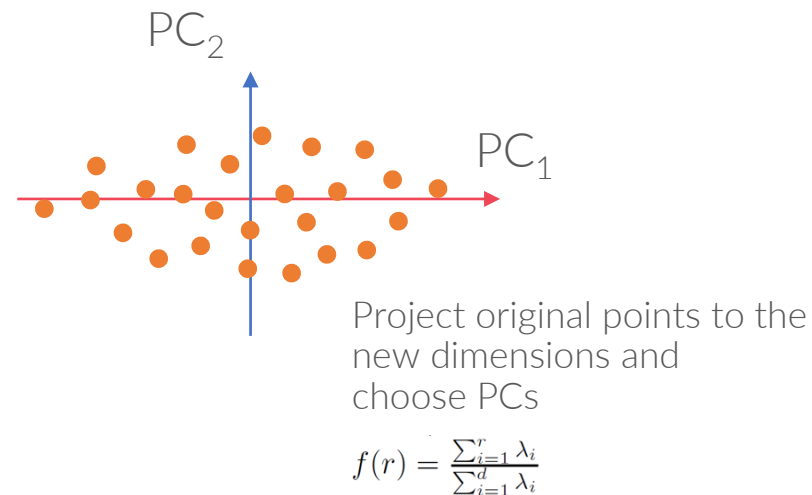
c.



d.



e.



$$\Sigma = \begin{pmatrix} 0.681 & -0.039 & 1.265 \\ -0.039 & 0.187 & -0.320 \\ 1.265 & -0.32 & 3.092 \end{pmatrix}$$

$$\lambda_1 = 3.662$$

$$\lambda_2 = 0.239$$

$$\lambda_3 = 0.059$$

## Question

$$u_1 = \begin{pmatrix} -0.39 \\ 0.089 \\ -0.916 \end{pmatrix} \quad u_2 = \begin{pmatrix} -0.639 \\ -0.742 \\ 0.200 \end{pmatrix} \quad u_3 = \begin{pmatrix} -0.663 \\ 0.664 \\ 0.346 \end{pmatrix}$$

What is the total variance?

What is the fraction of total variance for each PC?

If let  $\alpha = 0.95$  how many PCs we need to keep?

## 4. PCA Implementation in R

---

**Algorithm 7.1:** Principal Component Analysis

---

PCA ( $\mathbf{D}, \alpha$ ):

- 1  $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$  // compute mean
  - 2  $\mathbf{Z} = \mathbf{D} - \mathbf{1} \cdot \boldsymbol{\mu}^T$  // center the data
  - 3  $\boldsymbol{\Sigma} = \frac{1}{n} (\mathbf{Z}^T \mathbf{Z})$  // compute covariance matrix
  - 4  $(\lambda_1, \lambda_2, \dots, \lambda_d) = \text{eigenvalues}(\boldsymbol{\Sigma})$  // compute eigenvalues
  - 5  $\mathbf{U} = (\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_d) = \text{eigenvectors}(\boldsymbol{\Sigma})$  // compute eigenvectors
  - 6  $f(r) = \frac{\sum_{i=1}^r \lambda_i}{\sum_{i=1}^d \lambda_i}$ , for all  $r = 1, 2, \dots, d$  // fraction of total variance
  - 7 Choose smallest  $r$  so that  $f(r) \geq \alpha$  // choose dimensionality
  - 8  $\mathbf{U}_r = (\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_r)$  // reduced basis
  - 9  $\mathbf{A} = \{\mathbf{a}_i \mid \mathbf{a}_i = \mathbf{U}_r^T \mathbf{x}_i, \text{ for } i = 1, \dots, n\}$  // reduced dimensionality data
- 

```
1 data("iris")
2 head(iris)

1 irisCent <- iris %>% transmute(sL=Sepal.Length-mean(Sepal.Length),
2                               sW=Sepal.Width-mean(Sepal.Width),
3                               pL=Petal.Length-mean(Petal.Length),
4                               pW=Petal.Width-mean(Petal.Width))
```

```
1 ic <- cov(irisCent)
```

```
1 # use eigen() function to compute eigenvalues and eigenvectors
2 ieigen <- eigen(ic)
3 ie <- ieigen$values
4 iv <- ieigen$vectors
5 row.names(iv) <- names(iris %>% select(-Species))
6 colnames(iv) <- paste0(rep("PC", ncol(iv)), 1:ncol(iv)) # or
  sprintf("PC%d", 1:4)
```

```
1 # Fraction of the total variance
2 fr <- ie/sum(ie)
```

```
1 # Choose number of dimensionality
2 threshold <- function(x, th) {
3   sum <- 0
4   seq <- 0
5   for (i in 1:length(x)) {
6     sum <- sum + x[i]
7     if (sum >= th) {
8       seq <- i
9       break
10    }
11  }
12  return(seq)
13 }
14
15
16 threshold(x=fr, 0.95)
```

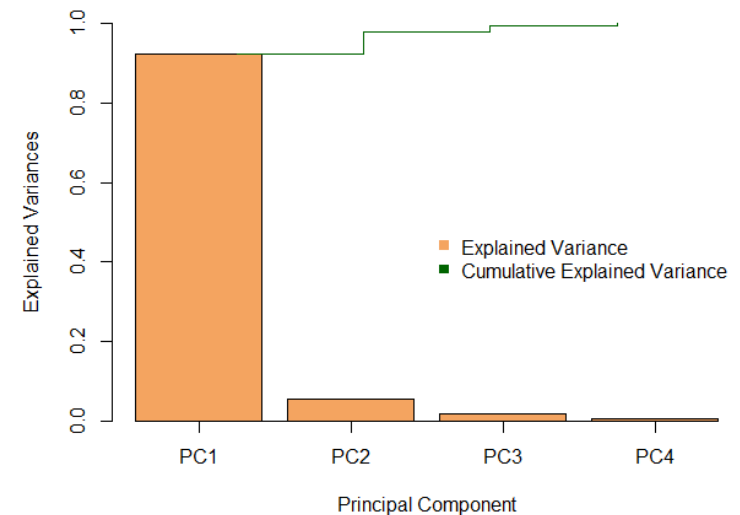
## Algorithm 7.1: Principal Component Analysis

PCA ( $\mathbf{D}, \alpha$ ):

- 1  $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$  // compute mean
- 2  $\mathbf{Z} = \mathbf{D} - \mathbf{1} \cdot \boldsymbol{\mu}^T$  // center the data
- 3  $\boldsymbol{\Sigma} = \frac{1}{n} (\mathbf{Z}^T \mathbf{Z})$  // compute covariance matrix
- 4  $(\lambda_1, \lambda_2, \dots, \lambda_d) = \text{eigenvalues}(\boldsymbol{\Sigma})$  // compute eigenvalues
- 5  $\mathbf{U} = (\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_d) = \text{eigenvectors}(\boldsymbol{\Sigma})$  // compute eigenvectors
- 6  $f(r) = \frac{\sum_{i=1}^r \lambda_i}{\sum_{i=1}^d \lambda_i}$ , for all  $r = 1, 2, \dots, d$  // fraction of total variance
- 7 Choose smallest  $r$  so that  $f(r) \geq \alpha$  // choose dimensionality
- 8  $\mathbf{U}_r = (\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_r)$  // reduced basis
- 9  $\mathbf{A} = \{\mathbf{a}_i \mid \mathbf{a}_i = \mathbf{U}_r^T \mathbf{x}_i, \text{ for } i = 1, \dots, n\}$  // reduced dimensionality data

```
1 biPCA <- prcomp(iris[1:4], scale = TRUE)
2 biPCA$sdev^2/sum(biPCA$sdev^2)
3 biPCA$rotation
```

```
1 barplot(
2   fr,
3   ylim = c(0, 1),
4   col = "sandybrown",
5   xlab = "Principal Component",
6   ylab = "Explained Variances",
7   axes = TRUE
8 )
9 axis(1, c(0.7, 1.9, 3.1, 4.3),
10      labels = sprintf("PC%d", 1:4))
11 lines(cumsum(fr), type = 's', col = "darkgreen")
12 legend(
13   x = 2.5,
14   y = 0.5,
15   legend = c("Explained Variance", "Cumulative
16 Explained Variance"),
17   pch = c(15, 15),
18   col = c("sandybrown", "darkgreen"),
19   bty = 'n'
20 )
```



---

**Algorithm 7.1:** Principal Component Analysis

---

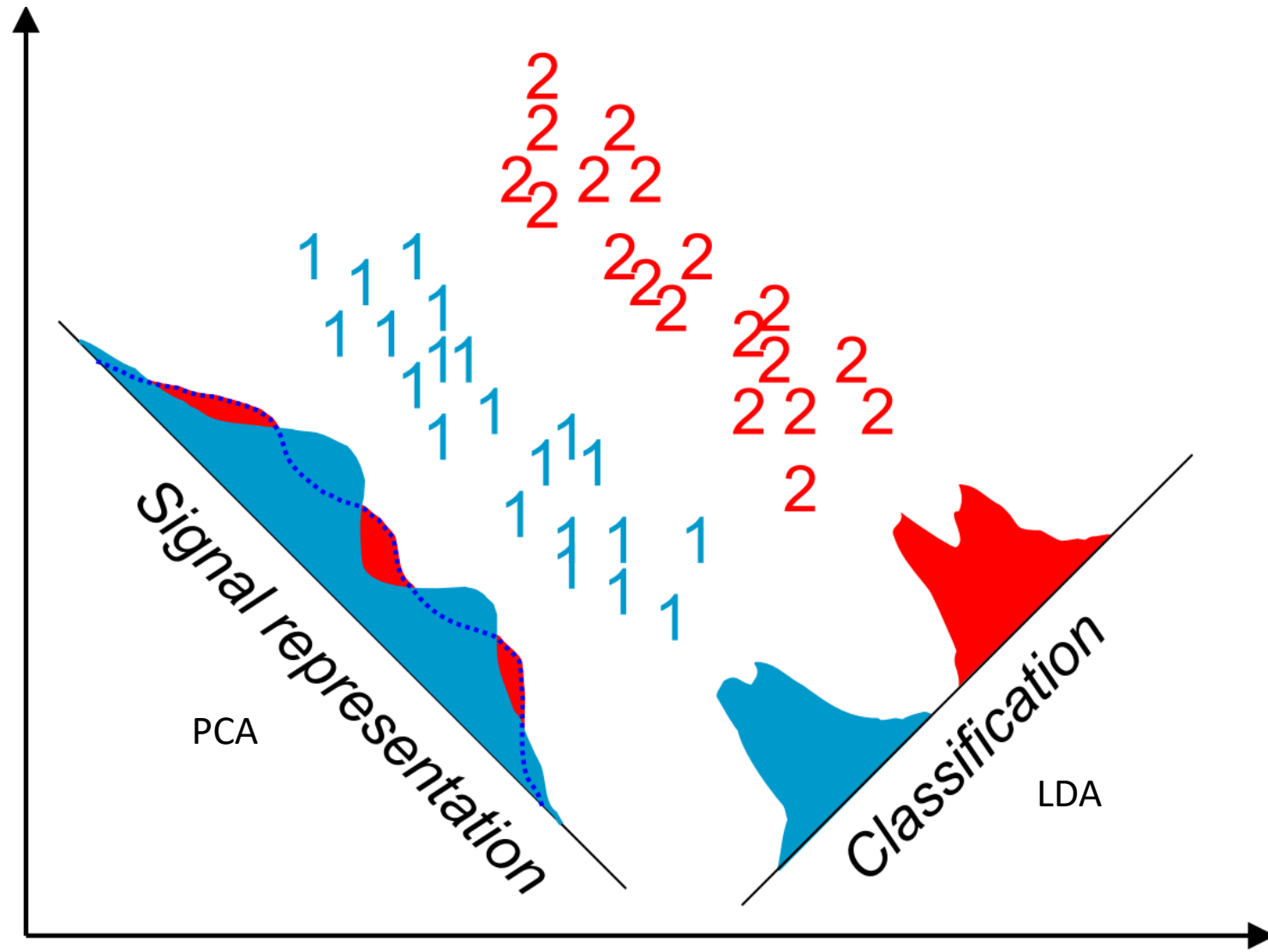
PCA ( $\mathbf{D}, \alpha$ ):

- 1  $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$  // compute mean
  - 2  $\mathbf{Z} = \mathbf{D} - \mathbf{1} \cdot \boldsymbol{\mu}^T$  // center the data
  - 3  $\boldsymbol{\Sigma} = \frac{1}{n} (\mathbf{Z}^T \mathbf{Z})$  // compute covariance matrix
  - 4  $(\lambda_1, \lambda_2, \dots, \lambda_d) = \text{eigenvalues}(\boldsymbol{\Sigma})$  // compute eigenvalues
  - 5  $\mathbf{U} = (\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_d) = \text{eigenvectors}(\boldsymbol{\Sigma})$  // compute eigenvectors
  - 6  $f(r) = \frac{\sum_{i=1}^r \lambda_i}{\sum_{i=1}^d \lambda_i}$ , for all  $r = 1, 2, \dots, d$  // fraction of total variance
  - 7 Choose smallest  $r$  so that  $f(r) \geq \alpha$  // choose dimensionality
  - 8  $\mathbf{U}_r = (\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_r)$  // reduced basis
  - 9  $\mathbf{A} = \{\mathbf{a}_i \mid \mathbf{a}_i = \mathbf{U}_r^T \mathbf{x}_i, \text{ for } i = 1, \dots, n\}$  // reduced dimensionality data
- 

### Exercise:

1. Try three datasets: mpg, BostonHousing (mlbench), BreastCancer (mlbench) on two scale methods: a) mean=0, b) mean=0, variance=1.
2. Compare to the built-in PCA function prcomp()

# PCA VS Linear Discriminant Analysis



© Ricardo Gutierrez-Osuna



## 5. PCA: Pros and Cons

# PCA *Pros and Cons*

## Pros

1. Good performance on processing speed
2. Reflects intuition on the data
3. Efficient reduction in size of data

## Cons

1. Doesn't consider class separability since it doesn't take into account the class labels
2. PCA simply performs a coordinate rotation that aligns the transformed axes with the directions of maximum variance
3. There is no guarantee that the directions of maximum variance will contain good features for discrimination
  - PCA cannot recognize the class labels

# Resources

# Resource

## Textbook:

Galit Shmueli, Peter C. Bruce, Inbal Yahav, Nitin R. Patel, Kenneth C. Lichtendahl Jr., Data Mining for Business Analytics: Concepts, Techniques, and Applications in R (DMBA), Wiley, 1st Edition, ISBN-10: 1118879368, ISBN-13: 978-1118879368.

## Additional Textbooks:

R For Data Science ([open license](#), R4DS), Wickham, Hadley, and Garrett Grolemund

R Markdown ([open license](#), RMD), Xie, Yihui, et al.

James, Gareth, et al. An Introduction to Statistical Learning: with Applications in R. Springer, 2017. ([open license](#), ISL)

Mohammed J. Zaki, Wagner Meira, Jr., Data Mining and Analysis: Fundamental Concepts and Algorithms, Cambridge University Press, May 2014. ISBN: 9780521766333.

David Hand, Heikki Mannila, Padhraic Smyth. Principles of Data Mining, The MIT Press, 2001, ISBN-10: 026208290X, ISBN-13: 978-0262082907.

Tan, Pang-Ning, et al. Introduction to Data Mining (DM). Pearson Education, 2006.

## Materials

@Victor Lavrenko