

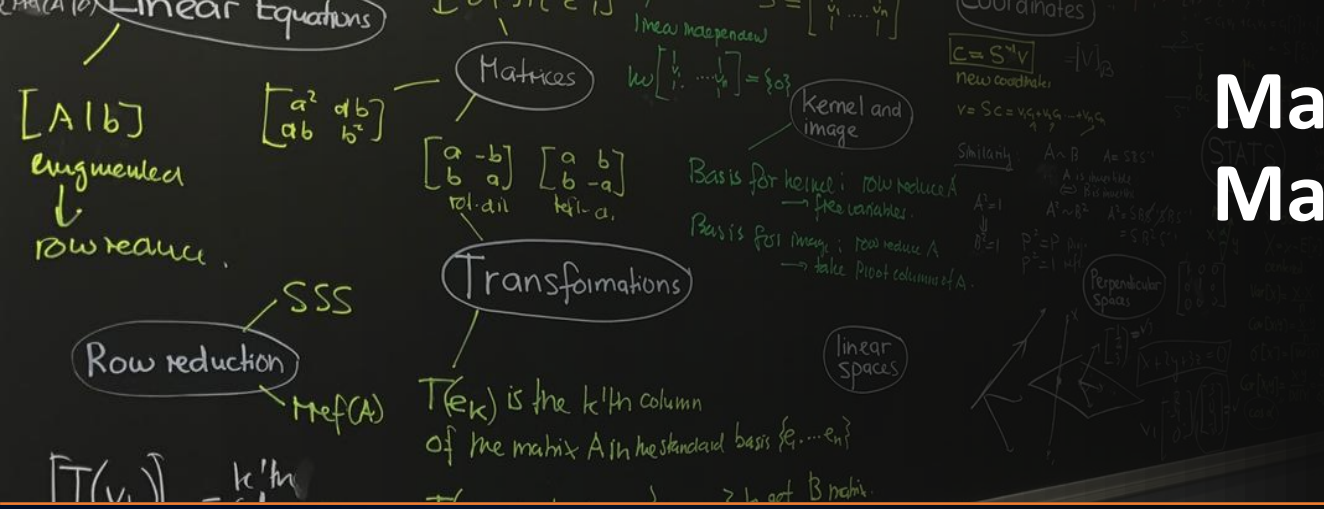
# Mathematical Foundations for Machine Learning and Data Science

## The Curse of Dimensionality and Dimensionality Reduction using PCA

Dr. Zubair Khalid

Department of Electrical Engineering  
School of Science and Engineering  
Lahore University of Management Sciences

[https://www.zubairkhalid.org/ee212\\_2021.html](https://www.zubairkhalid.org/ee212_2021.html)



# Outline

- *The Curse of Dimensionality*
- *Principal Component Analysis*

# Preamble

## Notation for Data:

# The Curse of Dimensionality

## Concept:

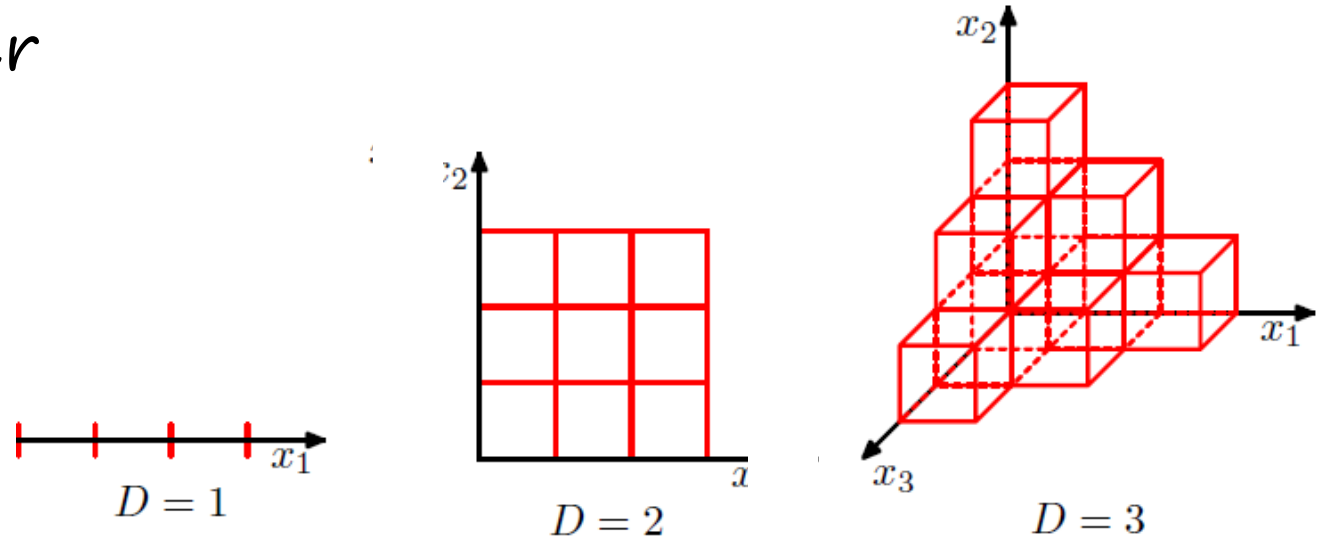
- Refers to the problems or phenomena associated with classifying, analyzing and organizing the data in high-dimensional spaces that do not arise in low-dimensional settings.
- For high-dimensional datasets, the size of data space is huge.
- In other words, the size of the feature space grows exponentially with the number of dimensions ( $d$ ) of the data sets.
- To ensure the points stay close to each other, the size ( $n$ ) of the data set must also have exponential growth. That means, we need a very large dataset to maintain the density of points in the high dimensional space.

# The Curse of Dimensionality

## Illustration 1:

- For high-dimensional datasets, the size of data space is huge.

For an exponentially large number of cells, we need an exponentially large amount of training data to ensure that the cells are not empty.



Ref: CB

# The Curse of Dimensionality

## Illustration 2:

Consider a ball of radius  $r$  defined as

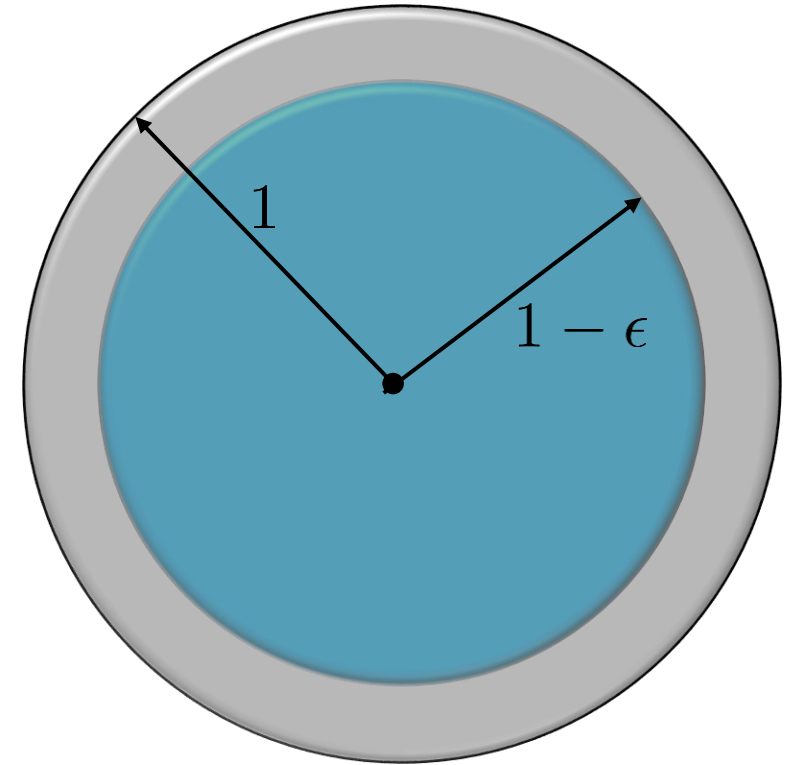
$$B(r) = \{\|\mathbf{x}\|_2 \leq r \mid \mathbf{x} \in \mathbf{R}^d\}$$

Volume of a ball of radius  $r$

$$V(d) = K_D r^D$$

Fraction of a volume between the balls of radius 1  
and radius  $1 - \epsilon$

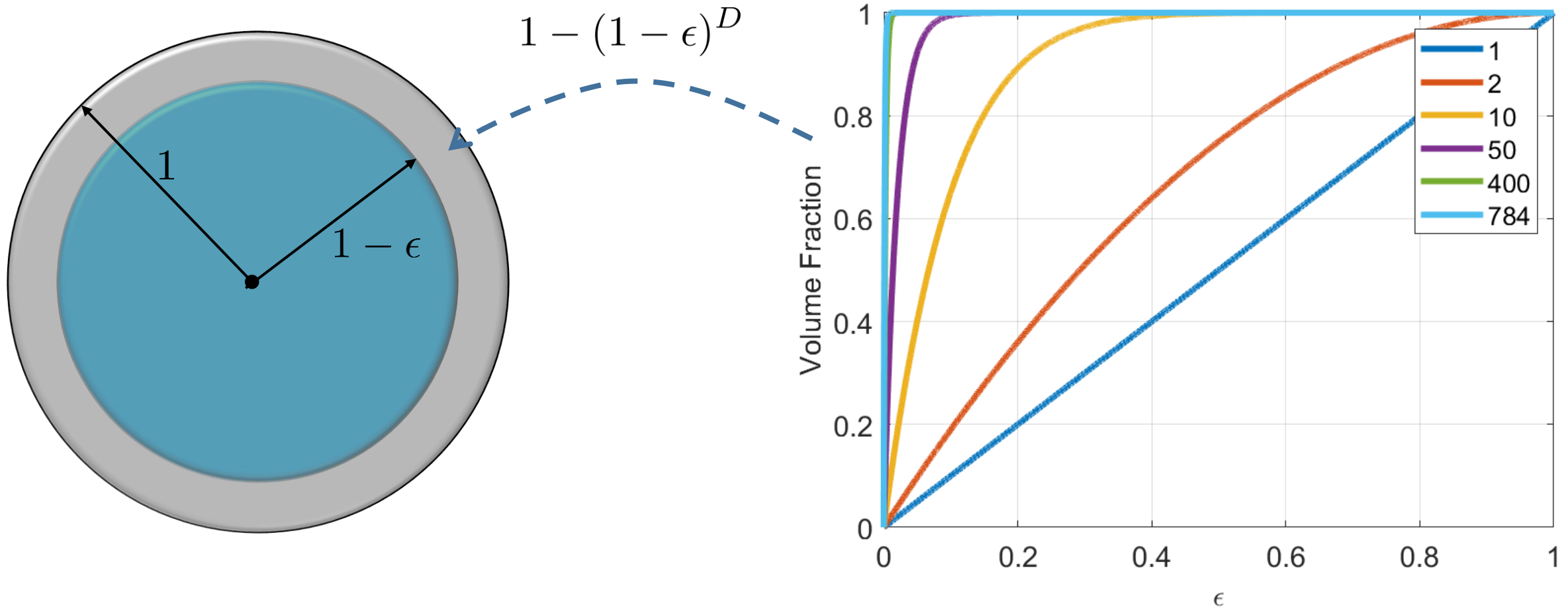
$$\frac{V(1) - V(1 - \epsilon)}{V(1)} = 1 - (1 - \epsilon)^D$$



$$K_D = \frac{\pi^{D/2}}{\Gamma(\frac{n}{2} + 1)}$$

# The Curse of Dimensionality

## Illustration 2:



# The Curse of Dimensionality

## Illustration 2 (Another viewpoint):

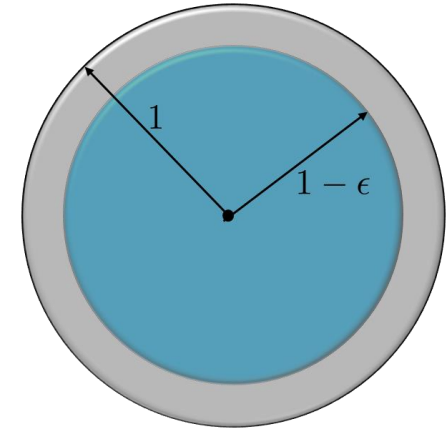
Calculate Probabilities that a uniformly distributed point is inside

$$\epsilon = 0.1$$

the shell:  $1 - (1 - \epsilon)^D$

the inner ball:  $(1 - \epsilon)^D$

D = 1	2	10	50	400	784
0.1	0.19	0.65	0.995	1.000	1.000
0.9	0.81	0.35	0.005	0.000	0.000



For  $D = 50$ , 5 out of 1000 data-points would be inside the inner ball.

For  $D = 400$ ,  $(1 - \epsilon)^D = 4.9774e - 19$ ; almost all points lie on the surface of the ball.

If you take a test point on the origin and  $D = 400$ , (almost) every point is at the same (Euclidean) distance from the origin.

# The Curse of Dimensionality

## Illustration 2 (Another viewpoint):

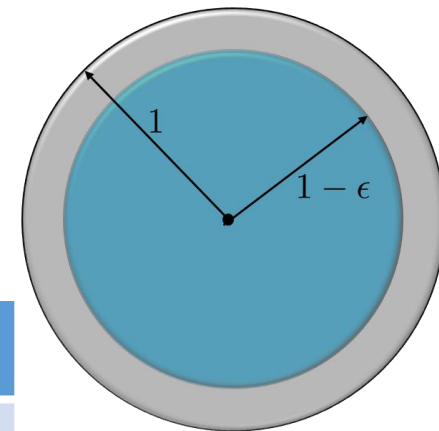
Calculate Probabilities that a uniformly distributed point is inside

$$\epsilon = 0.01$$

the shell:  $1 - (1 - \epsilon)^D$

the inner ball:  $(1 - \epsilon)^D$

D = 1	2	10	50	400	784
0.01	0.02	0.096	0.395	0.982	0.999
0.99	0.98	0.904	0.605	0.018	0.0004



# Practical Datasets


- With the increase in the number of features or number of dimensions of the feature space, data-points are never near to one another.
- Real-world data in the higher dimensional space is confined to a region with effective lower dimensionality.
  - **Dimensionality Reduction**
- Real-world data exhibits smoothness that enables us to make predictions exploiting interpolation techniques.
- For example,
  - Data along a line or a plane in higher dimensional space
  - detection of orientation of object in an image; data lies on effectively 1 dimensional manifold in probably 1million dimensional space.
  - Face recognition in an image (50 or 71 features)

# Dimensionality Reduction

## Feature Extraction:

Transform *existing* features to obtain a set of *new* features using some mapping function.

$$\mathbf{x} = [x_1, x_2, \dots, x_d]$$



$$\mathbf{z} = [z_1, z_2, \dots, z_k]$$

- The mapping function  $\mathbf{z}=f(\mathbf{x})$  can be linear or non-linear.
- Can be interpreted as projection or mapping of the data in the higher dimensional space to the lower dimensional space.
- Mathematically, we want to find an *optimum* mapping  $\mathbf{z}=f(\mathbf{x})$  that preserves the desired information as much as possible.

# Dimensionality Reduction

## Feature Extraction:

### Idea:

- Finding optimum mapping is equivalent to optimizing an *objective function*.
- We use different objective functions in different methods;
  - **Minimize Information Loss**: Mapping that represent the data as accurately as possible in the lower-dimensional space, e.g., Principal Components Analysis (PCA).
  - **Maximize Discriminatory Information**: Mapping that best discriminates the data in the lower-dimensional space, e.g., Linear Discriminant Analysis (LDA).
- Here we focus on PCA, that is, a linear mapping.
- Why Linear: Simpler to Compute and Analytically Tractable.

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

- Given features in  $d$ -dimensional space
- Project into lower dimensional space using the following linear transformation

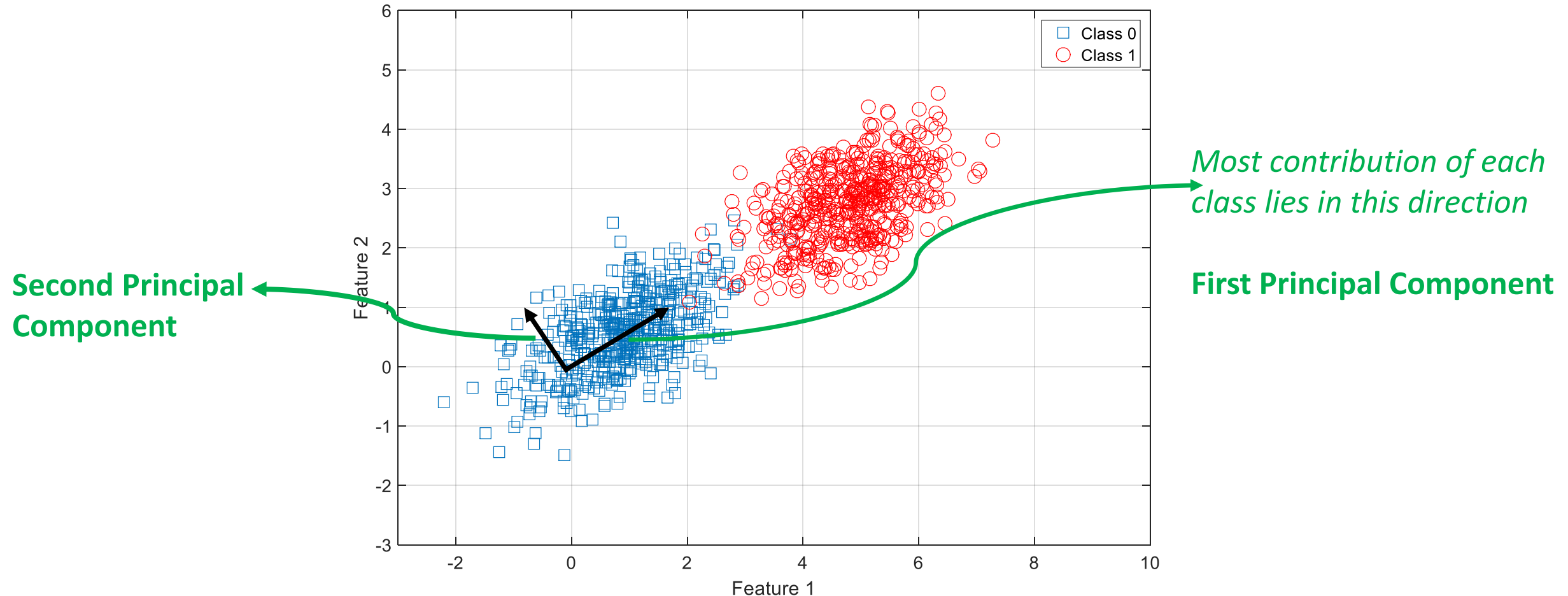
$$\mathbf{z} = \mathbf{W}^T \mathbf{x}$$

- For example (can you tell me size of matrix  $\mathbf{W}$  for the following cases),
  - find best planar approximation to 4D data
  - find best planar approximation to 100D data
- We want to find this mapping while preserving as much information as possible, and ensuring
  - **Objective 1:** the features after mapping are uncorrelated; cannot be reduced further
  - **Objective 2:** the features after mapping have large variance

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

### Geometric Intuition:

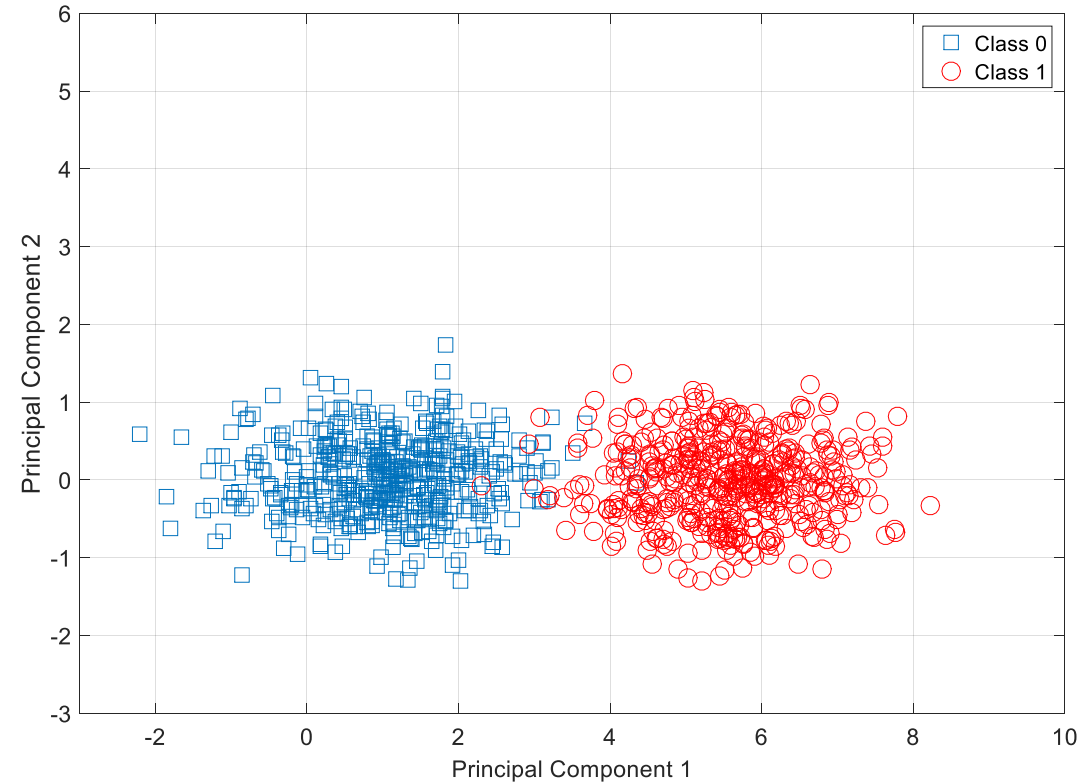


Toy Illustration in two dimensions

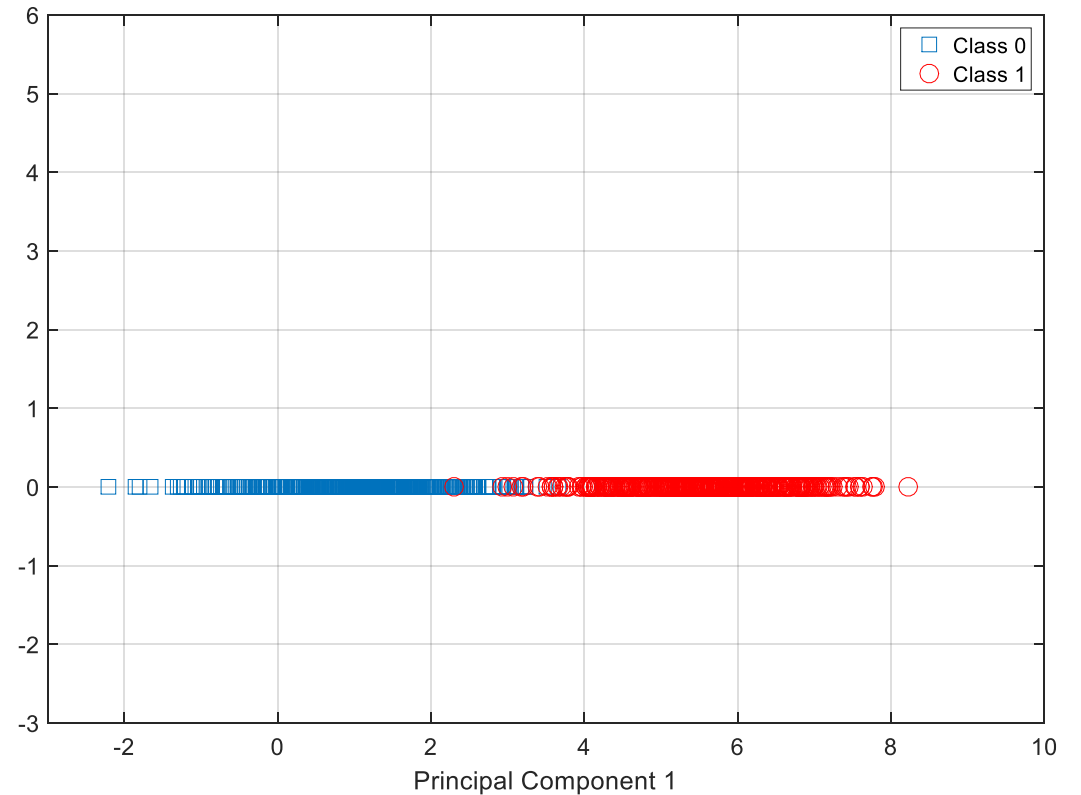
# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

### Geometric Intuition:



*Change of coordinates: Linear combinations of features*



*Ignoring the Second Component/Feature*

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

### Mathematical Formulation:

We have  $n$  feature vectors of the form  $\mathbf{x} \in \mathbf{R}^d$ .

Note  $d$  represents the number of features.

In PCA, we want to represent  $\mathbf{x}$  in a new space of lower dimensionality using only  $k$  basis vectors ( $k < d$ ), that is,

$$\hat{\mathbf{x}} = \sum_{i=1}^k z_i \mathbf{v}_i$$

such that

$$\|\mathbf{x} - \hat{\mathbf{x}}\|_2$$

is minimized.

Here  $\mathbf{v}_i \in \mathbf{R}^d$  for  $i = 1, 2, \dots, k$  represent the  $k$  number of orthogonal vectors that form the basis, referred to as principal components, of the subspace of dimensionality  $= k$ .

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

### Mathematical Formulation:

How do we find the basis vectors  $\mathbf{v}_i \in \mathbf{R}^d$  for  $i = 1, 2, \dots, k$ ?

### Steps to find Principal Components:

We have  $n$  feature vectors  $\mathbf{x}_i \in \mathbf{R}^d$ ,  $i = 1, 2, \dots, n$ .

#### Step 1: Compute Sample Mean:

Sample mean (note summation over the number of feature vectors  $n$ )

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$$

#### Step 2: Subtract Sample Mean:

Subtract sample mean from each feature vector  $\mathbf{x}_i$  to obtain  $\mathbf{s}_i$ , that is,

$$\mathbf{s}_i = \mathbf{x}_i - \bar{\mathbf{x}}$$

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

### Mathematical Formulation:

#### Step 3: Calculate the Covariance Matrix:

Now we have  $n$  feature vectors  $\mathbf{s}_i \in \mathbf{R}^d$ ,  $i = 1, 2, \dots, n$ .

Calculate the Covariance Matrix as follows

$$\Sigma = \frac{1}{n} \sum_{i=1}^n \mathbf{s}_i \mathbf{s}_i^T$$

This can also be expressed as

$$\Sigma = \frac{1}{n} \mathbf{S} \mathbf{S}^T$$

where

$$\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n]$$

*What is special about these vectors?*

*Zero mean; taken along all feature vectors*

*How do you interpret the entries of the matrix? Spend some time and try to understand this!*



For two vectors  $\mathbf{f}, \mathbf{g} \in \mathbf{R}^n$ , covariance is defined as

$$\sigma_{\mathbf{fg}} = \frac{1}{n} \sum_i^n (f_i - \text{avg}(\mathbf{f})) (g_i - \text{avg}(\mathbf{g}))$$

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

### Special about the Covariance Matrix:

The covariance matrix is symmetric, that is,  $\Sigma^T = \Sigma$ . (super easy to show)

The covariance matrix is positive semi-definite. (again, super easy)

Size of  $\Sigma$  is  $d \times d$ .

### Step 4: Carry out Eigenvalue Decomposition of Covariance Matrix:

Carry out eigenvalue decomposition of the covariance matrix as

$$\Sigma = \mathbf{V}\mathbf{D}\mathbf{V}^T$$

Here the matrix  $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d]$  contains  $d$  orthogonal eigenvectors  $\mathbf{v}_i \in \mathbf{R}^d$ , referred to as principal components, that serve as the basis of  $\mathbf{R}^d$ .

Here the matrix  $\mathbf{D}$  is a diagonal matrix with eigenvalues denoted by  $\lambda_1, \lambda_2, \dots, \lambda_d$ .

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

### Step 5: Dimensionality Reduction

We wanted to find the basis vectors  $\mathbf{v}_i \in \mathbf{R}^d$  for  $i = 1, 2, \dots, k$ .

We have  $\mathbf{v}_i \in \mathbf{R}^d$  for  $i = 1, 2, \dots, d$ .

- *Q: How to select  $k$  out of  $d$ ?*
- *A: Simple, select the ones corresponding to  $k$  largest eigenvalues.*

Construct the mapping matrix of size  $d \times k$  as

$$\mathbf{W} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k]$$

to reduce the dimensionality of the feature space from  $\mathbf{R}^d$  to  $\mathbf{R}^k$  as

$$\mathbf{z} = \mathbf{W}^T \mathbf{x}$$

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

Using  $\mathbf{z}$ , we can go back to  $\mathbf{R}^d$  to obtain approximation of  $\mathbf{x}$  as

$$\hat{\mathbf{x}} = \sum_{i=1}^k z_i \mathbf{v}_i = \mathbf{W}\mathbf{z}$$

## Connection with the Objectives:

- *Objective 1: the features after mapping are uncorrelated; cannot be reduced further*
  - *Enabled by orthogonality of the principal components*
- *Objective 2: the features after mapping have large variance*
  - *We have used covariance matrix to define the mapping and used eigenvectors with largest eigenvalues, that is, those dimensions capturing the variations in the data.*
  - *PCA maps the data along the directions where we have most of the variations in the data.*

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

### How do we choose k?

- It depends on the amount of information, that is variance, we want to preserve in the mapping process.
- We can define a variable  $T$  to quantify this preservation of information

$$\frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^d \lambda_i} > T$$

- $T=1$ , when  $k=d$ ; No reduction.
- $T=0.8$ , interpreted as that 80% variation in the data has been preserved.

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

**Example:**  $d = 2, n = 10, k = 1$

**Step 1: Compute sample mean:**

$$\bar{\mathbf{x}} = [1.81, 1.91]$$

$x_1$

$x_2$

2.5000	2.4000	$\mathbf{x}_1$
0.5000	0.7000	$\mathbf{x}_2$
2.2000	2.9000	
1.9000	2.2000	
3.1000	3.0000	
2.3000	2.7000	
2.0000	1.6000	
1.0000	1.1000	
1.5000	1.6000	
1.1000	0.9000	

**Step 2: Subtract Sample Mean:**

$$\mathbf{s}_i = \mathbf{x}_i - \bar{\mathbf{x}}$$

$s_1$

$s_2$

0.6900	0.4900	$\mathbf{s}_1$
-1.3100	-1.2100	$\mathbf{s}_2$
0.3900	0.9900	
0.0900	0.2900	
1.2900	1.0900	
0.4900	0.7900	
0.1900	-0.3100	
-0.8100	-0.8100	
-0.3100	-0.3100	
-0.7100	-1.0100	

**Step 3: Calculate the Covariance Matrix:**

$$\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n]$$

$$\Sigma = \frac{1}{n} \sum_{i=1}^n \mathbf{s}_i \mathbf{s}_i^T = \frac{1}{n} \mathbf{S} \mathbf{S}^T$$

$$\Sigma = \begin{bmatrix} 0.5549 & 0.5539 \\ 0.5539 & 0.6449 \end{bmatrix}$$

We have divided by  $n$ . Some authors divide by  $n-1$ . It won't change the principal components

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

### Example:

### Step 4: Carry out Eigenvalue Decomposition of Covariance Matrix:

$$\Sigma = \mathbf{V}\mathbf{D}\mathbf{V}^T \quad \mathbf{V} = \begin{bmatrix} -0.7352 & 0.6779 \\ 0.6779 & 0.7352 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 0.0442 & 0 \\ 0 & 1.1556 \end{bmatrix} \quad \mathbf{z}$$

### Step 5: Dimensionality Reduction

Use  $\mathbf{W} = [\mathbf{v}_2]$  (associated with the largest eigenvalue) to reduce the dimensionality of the feature space from  $\mathbf{R}^2$  to  $\mathbf{R}$  as

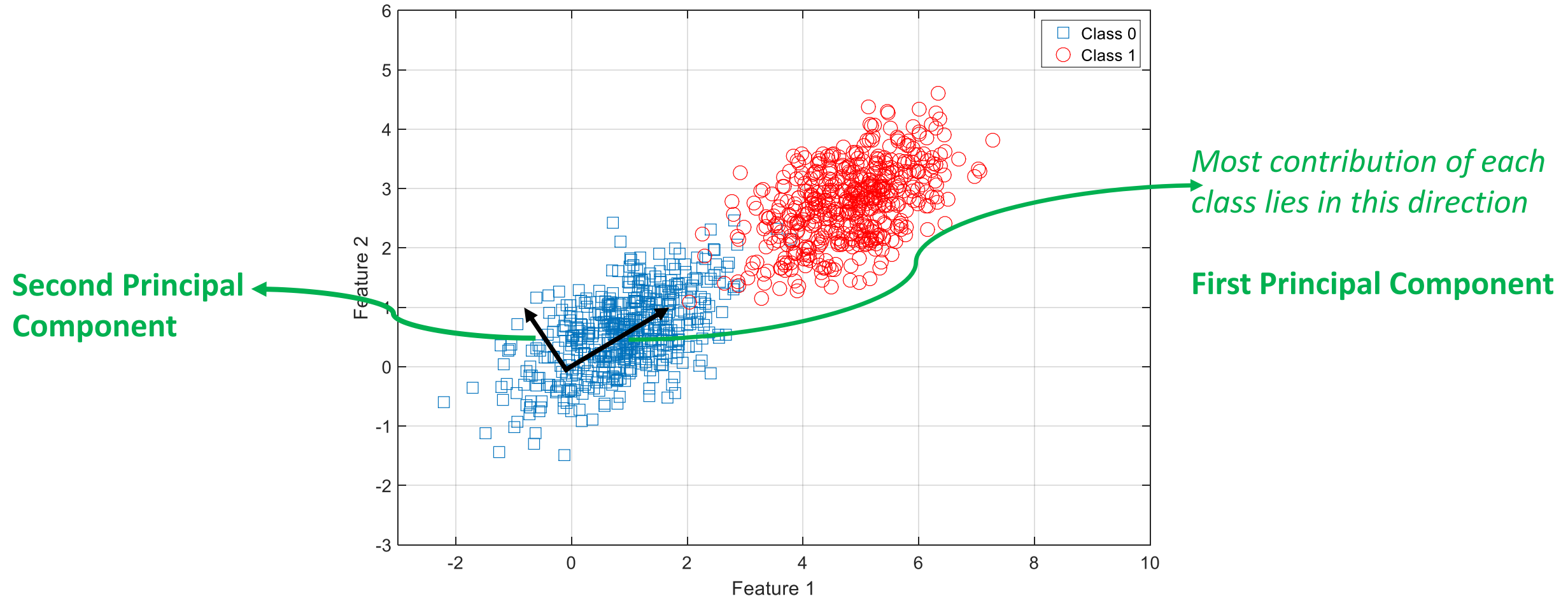
$$\mathbf{z} = \mathbf{W}^T \mathbf{x}$$

3.4591  
0.8536  
3.6233  
2.9054  
4.3069  
3.5441  
2.5320  
1.4866  
2.1931  
1.4073

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

### Geometric Intuition:

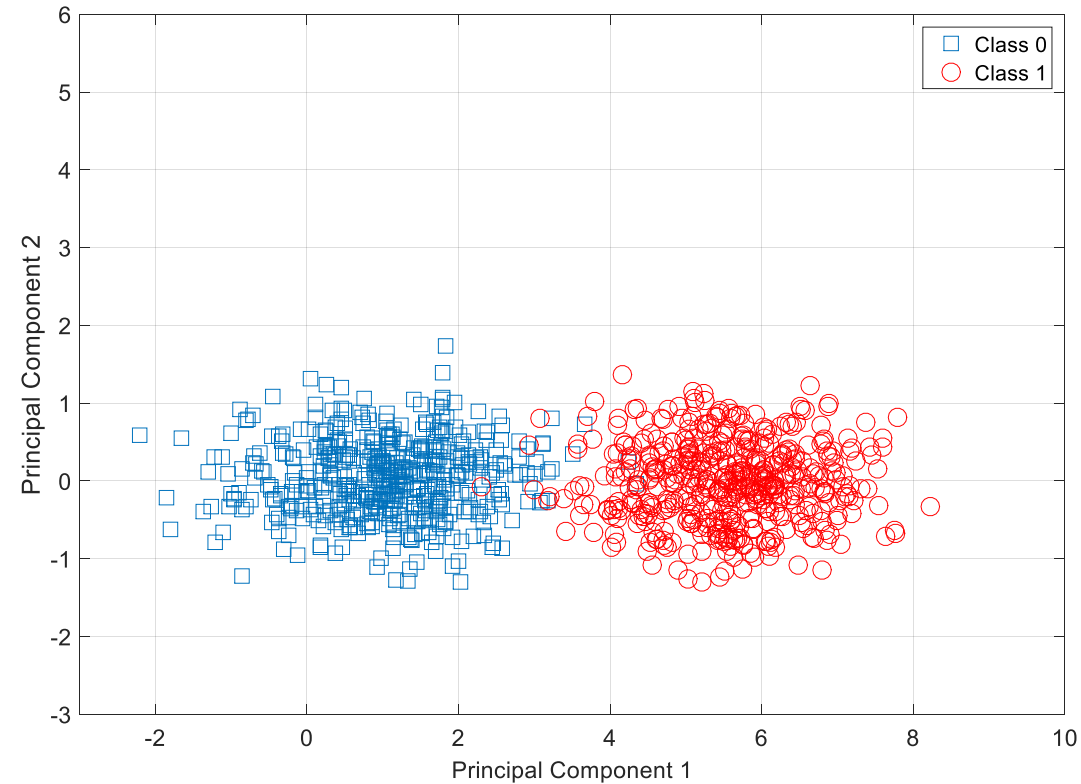


Toy Illustration in two dimensions

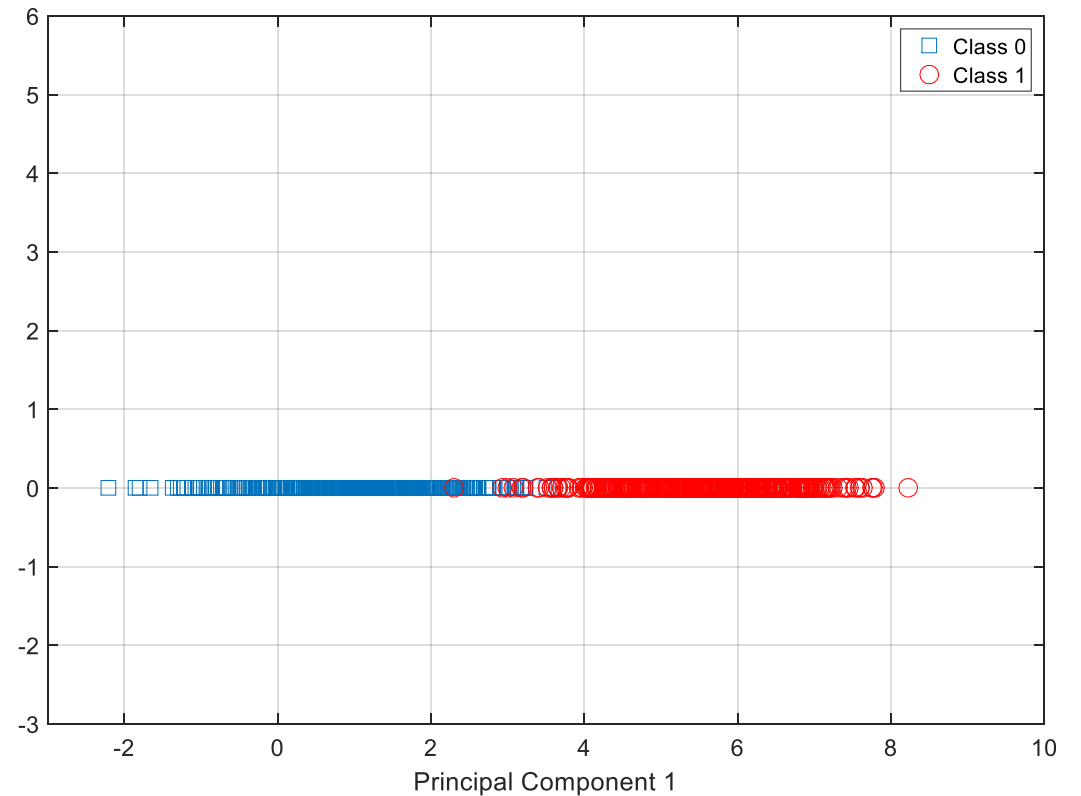
# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

### Geometric Intuition:



*Change of coordinates: Linear combinations of features*



*Ignoring the Second Component/Feature*

# Dimensionality Reduction

## Feature Extraction - Principal Component Analysis:

### Practical Considerations and Limitations:

- Data should be normalized before using PCA for dimensionality reduction.
- Usually, we normalize every feature by subtracting mean of that feature followed by dividing with standard deviation of the feature.
- The covariance matrix of the reduced feature is projection along orthogonal components (directions) and therefore features are uncorrelated to each other. In other words, PCA decorrelates the features.
- **Limitation:**
  - PCA does not consider the separation of data with respect to class label and therefore we do not have a guarantee the mapping of the data along dimensions of maximum variance results in the new features good enough for class discrimination.

**Solution:** Linear Discriminant Analysis (LDA) - Find mapping directions along which the classes are best separated.