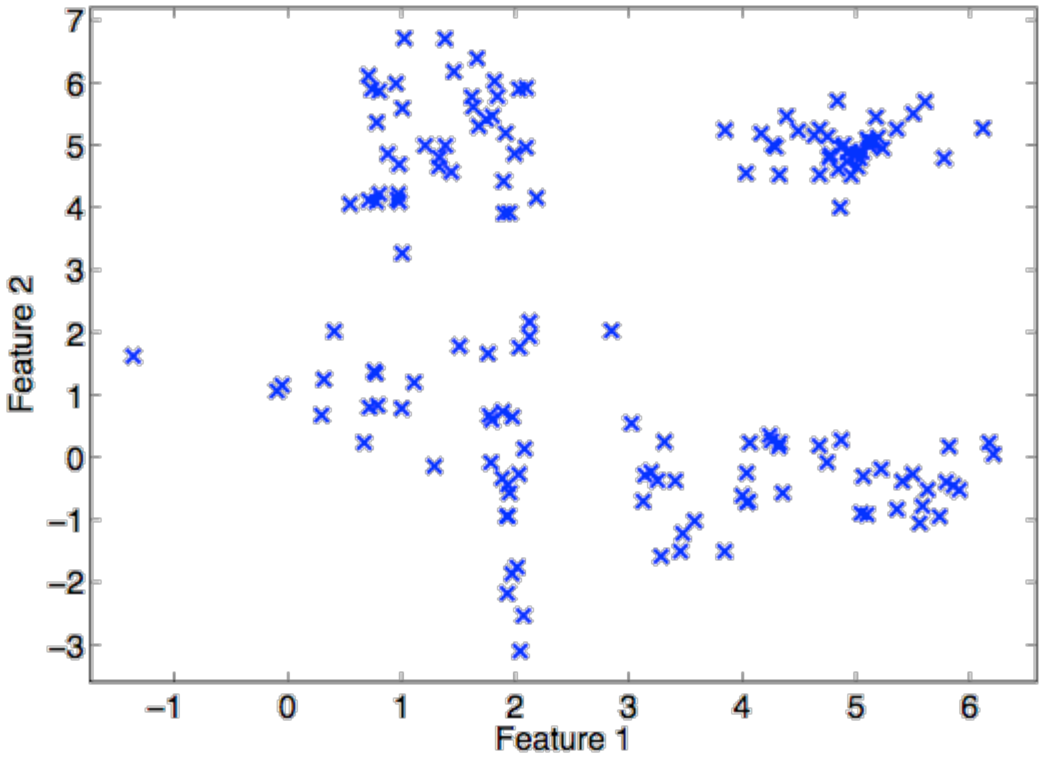
# Unsupervised learning

Gosia Migut

Slides credit: Tom Viering

TU Delft

# Unlabelled data: what now?




- Unsupervised learning: no labels/targets present

# Unsupervised learning

- Clustering
  - Discover structures in unlabelled data

- Dimensionality reduction
  - does not use information about the labels

# Dimensionality reduction

- Many data sets are *high-dimensional*: each instance is described by many features.

- Why do we want to reduce data dimensionality?
- What does it mean to reduce dimensionality?
- How to reduce dimensionality with Principal Component Analysis ?

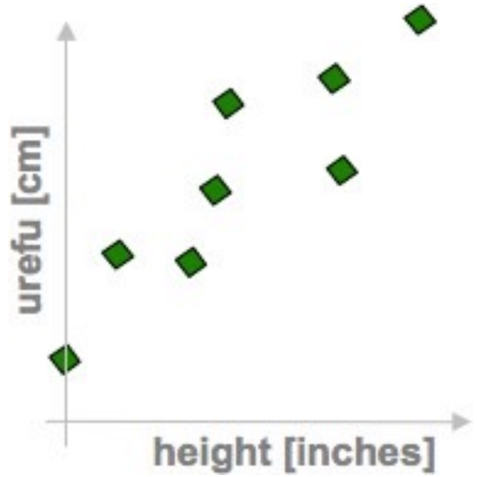
# Dimensionality reduction

- Why do we want to reduce data dimensionality?
  - Make storage or processing of data easier
  - (Visual) discovery of hidden structure in the data
  - Remove redundant and noisy features
  - Intrinsic dimensionality might be smaller

# Dimensionality reduction
*Visual discovery of data structure*

# Dimensionality reduction
## *Redundant features*

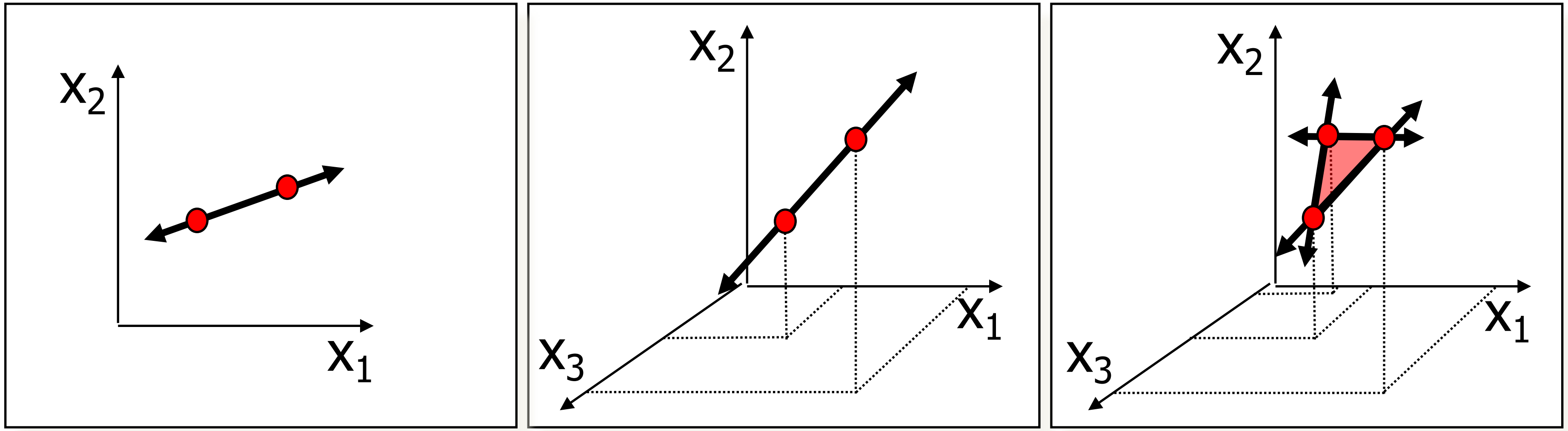
- Get a population, predict some property
  - instances represented as {urefu, height} pairs
  - what is the dimensionality of this data?




"height" = "urefu" in Swahili

# Dimensionality reduction

## *Intrinsic dimensionality*

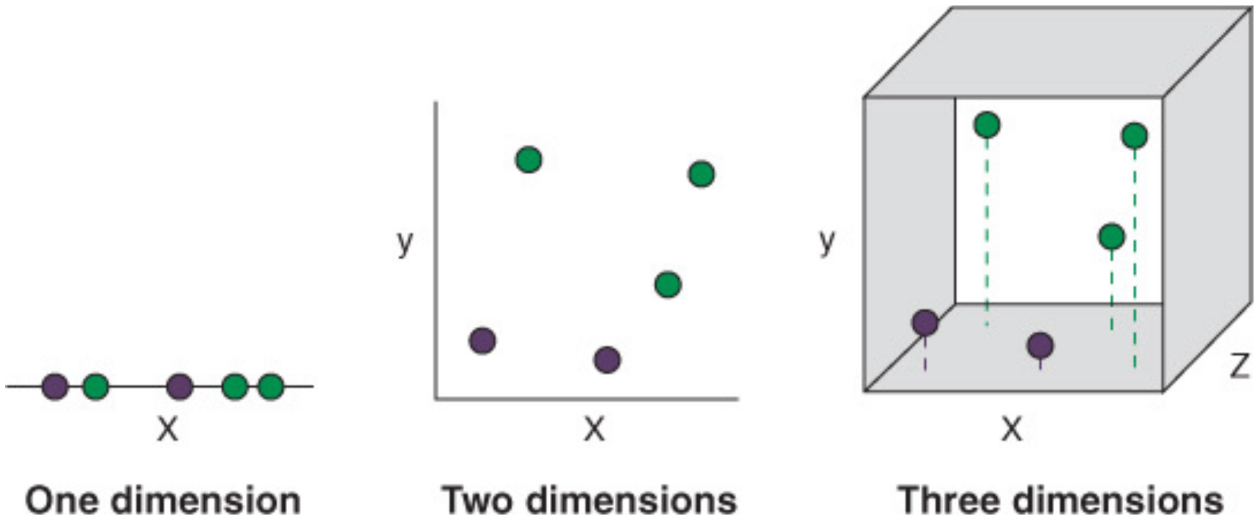



| 2 objects, 2 dimensions<br>➔ 1 dimension | 2 objects, 3 dimensions<br>➔ 1 dimension | 3 objects, 3 dimensions<br>➔ 2 dimension |
|---|---|---|

# Dimensionality reduction

## *Curse of dimensionality*

- As dimensionality grows fewer observations per region
- Statistics need repetition



# Dimensionality reduction

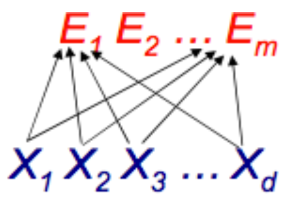
- Many data sets are *high-dimensional*: each instance is described by many features.

- Why do we want to reduce data dimensionality?
- **What does it mean to reduce dimensionality?**
- How Principal Component Analysis reduces dimensionality?

# Reducing dimensionality: methods

- Feature selection
  - pick a subset of the original dimensions $X_1 X_2 X_3 \ldots X_{d-1} X_d$
  - Use domain knowledge
  - Use statistics-based selection methods

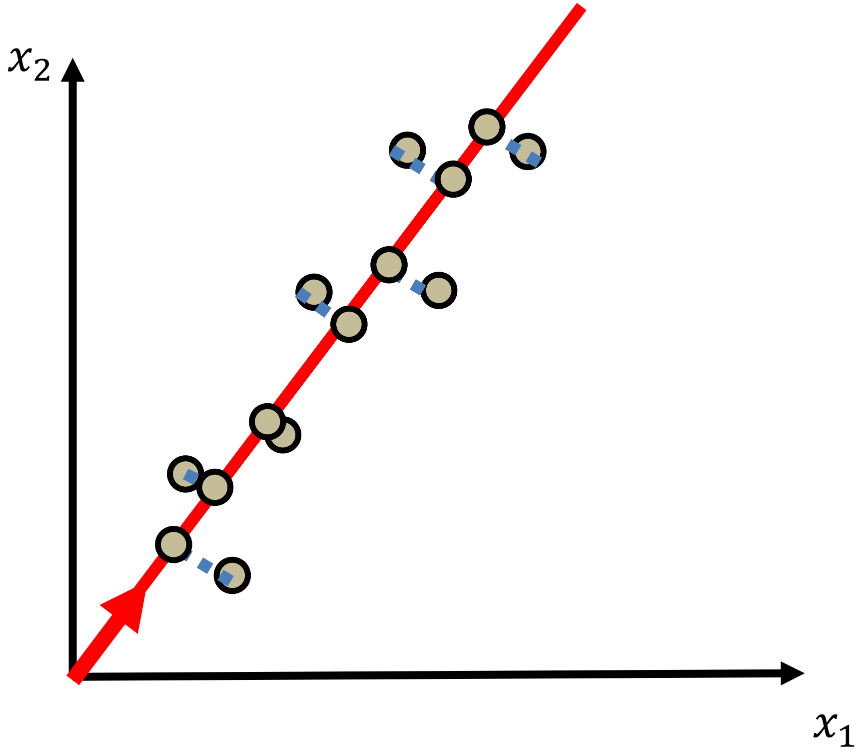
- **Feature extraction**
  - construct a new set of dimensions $E_i = f(X_1 \ldots X_d)$

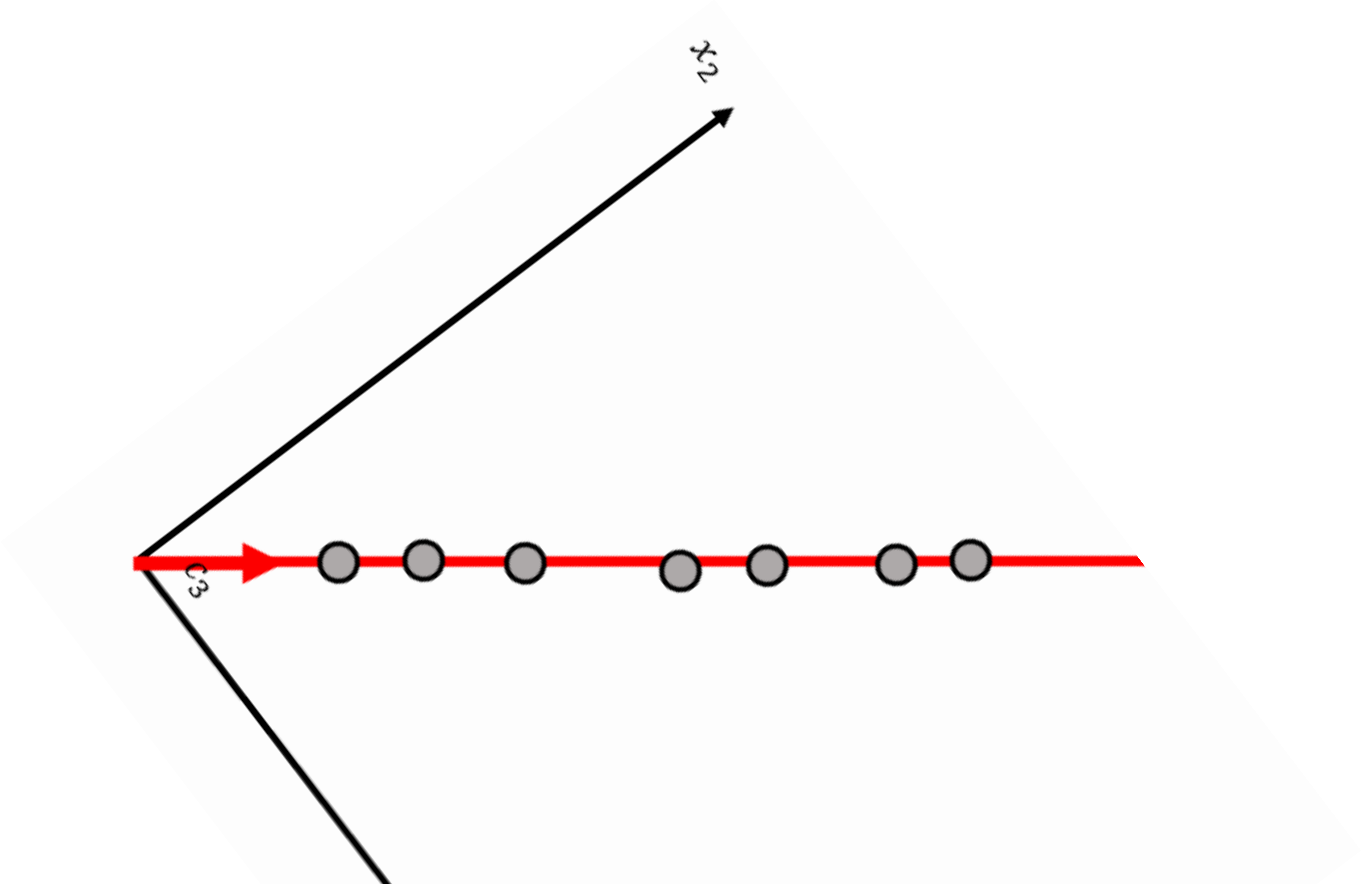



  - (linear) combinations of original

# Reducing dimensionality
## *Feature extraction*

- Many important dimensionality reduction techniques are *linear* techniques
- These project the data onto a *linear subspace* of *lower dimensionality*




TU Delft

# Dimensionality reduction

- Many data sets are *high-dimensional*: each instance is described by many features.

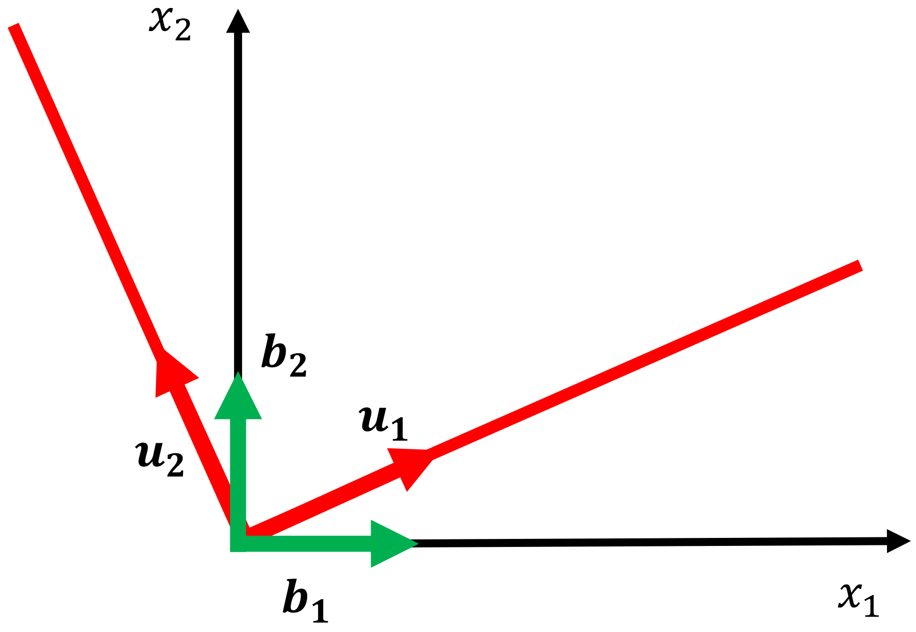
- Why do we want to reduce data dimensionality?
- What does it mean to reduce dimensionality?
- **How Principal Component Analysis reduces dimensionality?**

# Principal component analysis

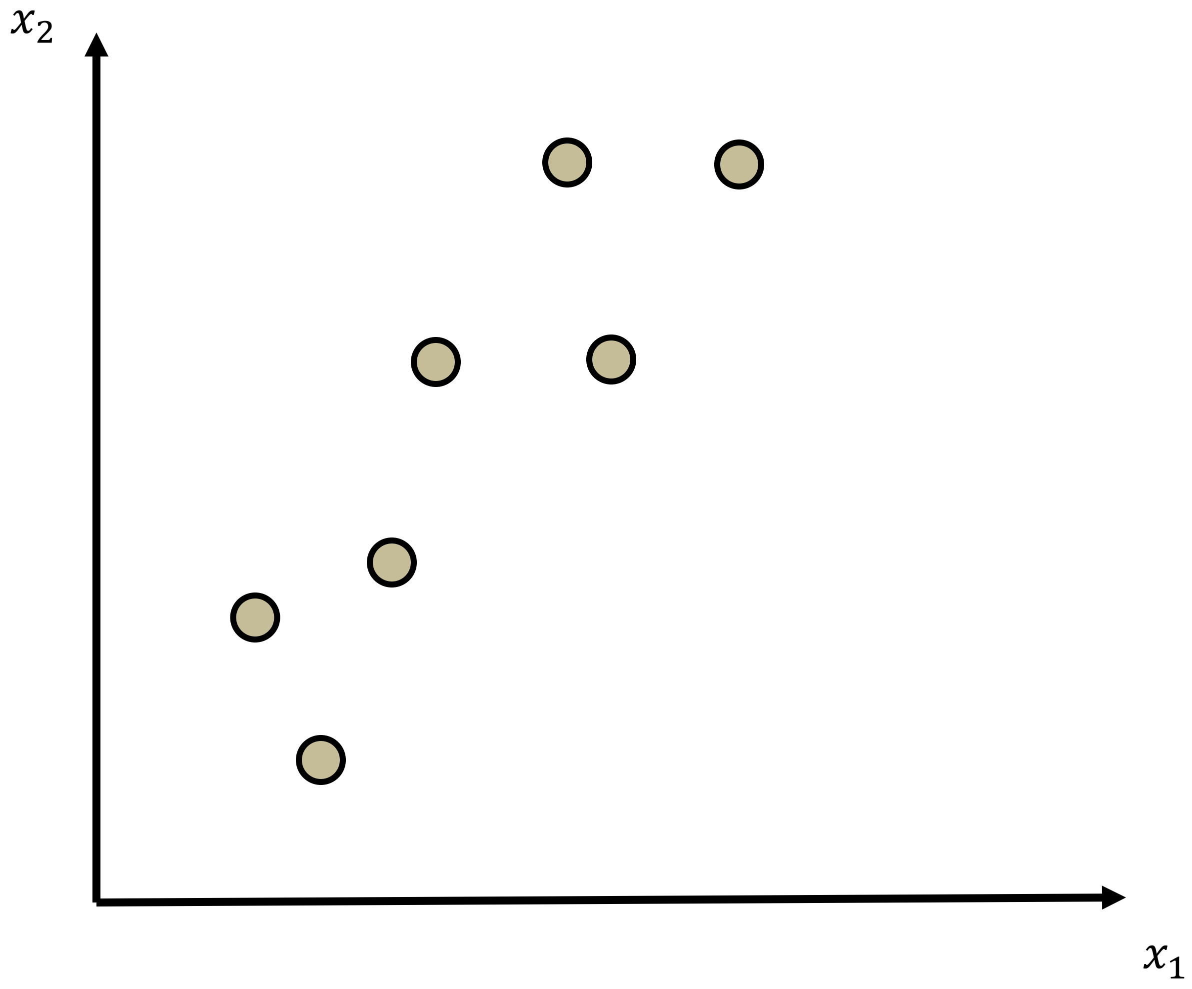
# PCA: What it does

- PCA finds new basis vectors $\boldsymbol{u_1}, \boldsymbol{u_2}$
- Called principal components
- They are orthogonal and have a specific order: the first is the most important
- As many as original dimensionality of data

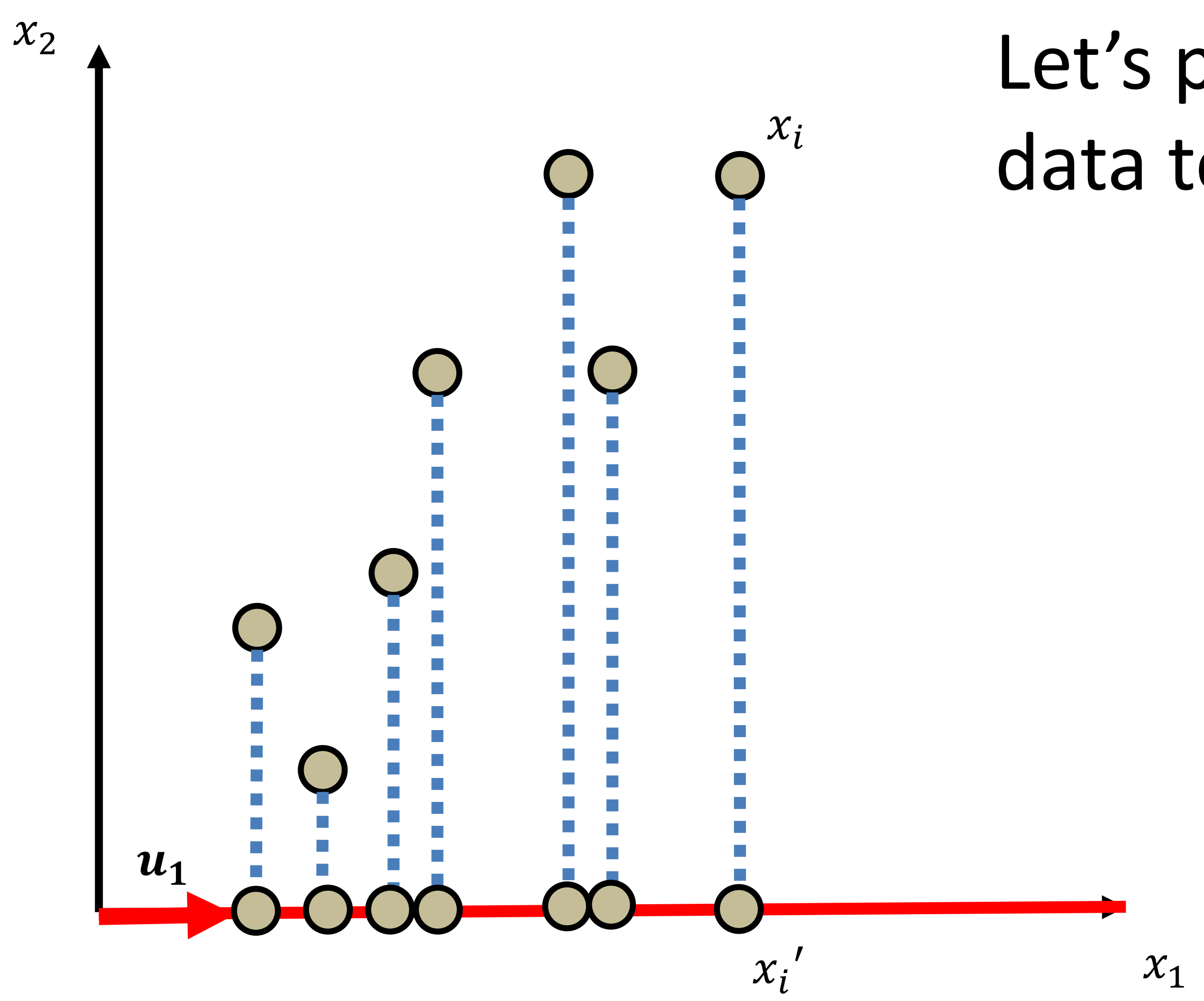
- How to find them?
  - Minimum error formulation
  - Variance maximization formulation
- How to do a dimensionality reduction?
- How many components to use?
- Limitations



# PCA: Example

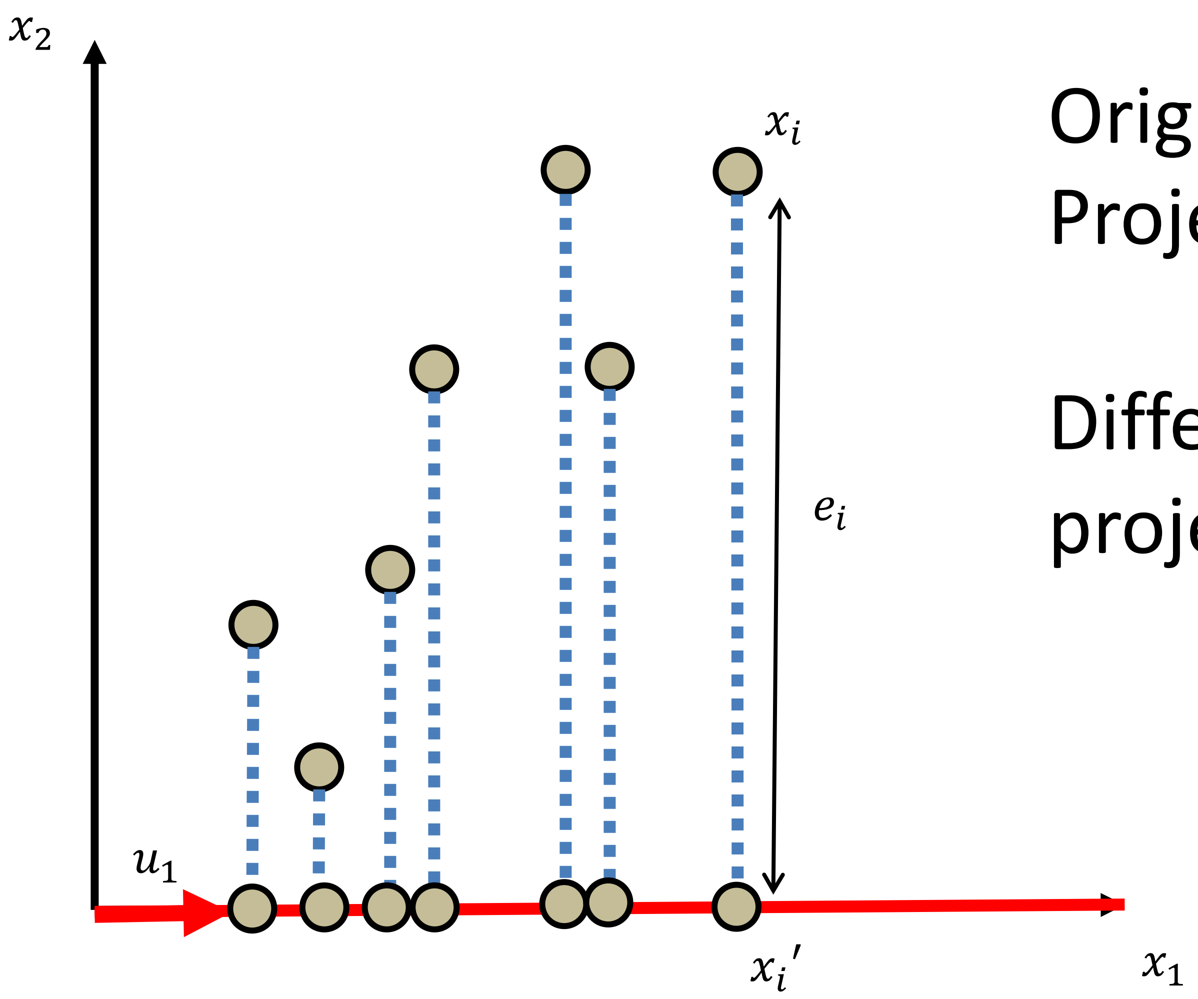


# PCA: Example




Let’s project all data to $\boldsymbol{u_1}$

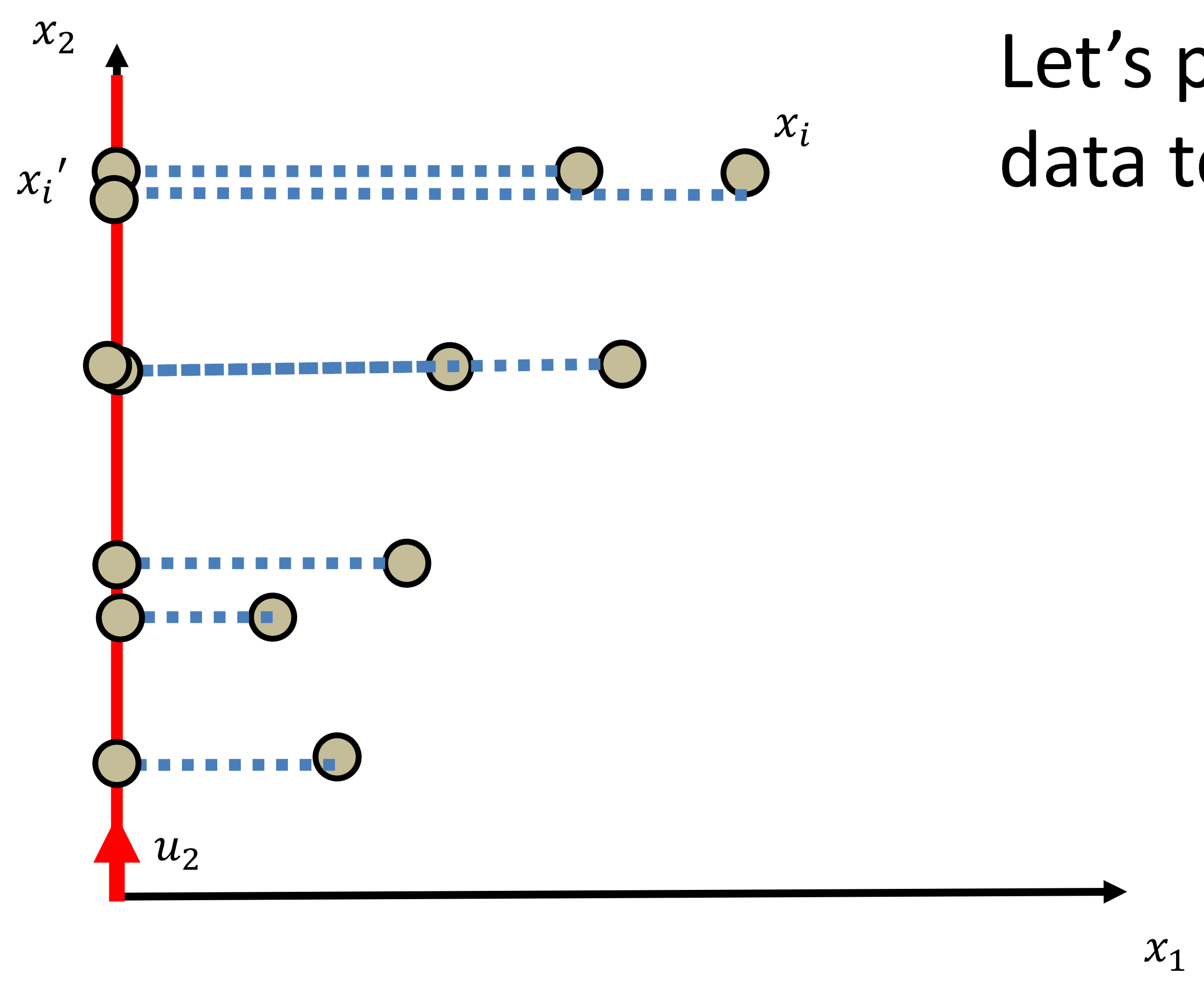
# PCA: Example




Original data $x_i$
Projected data $x_i'$

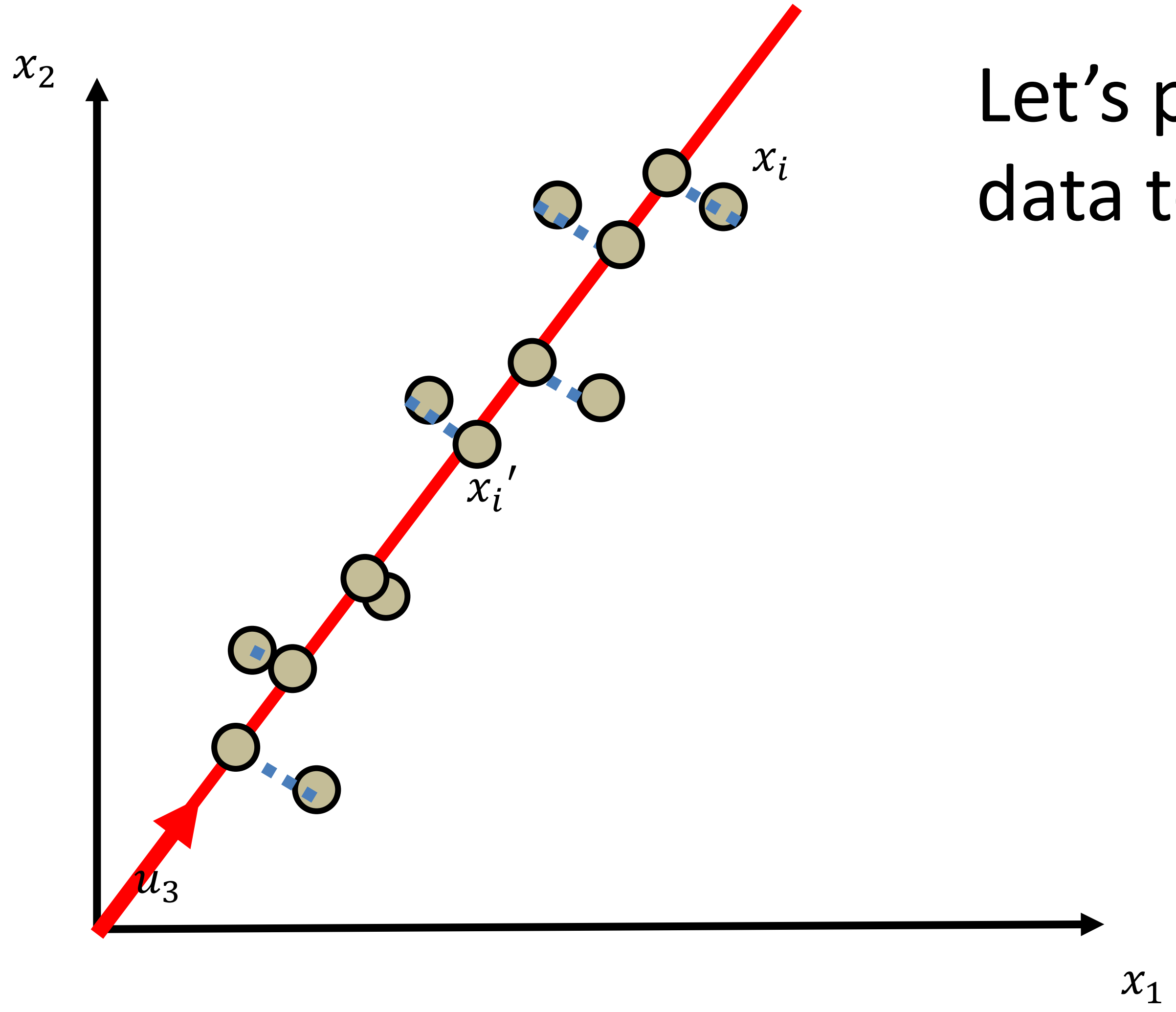
Difference:
projection error $\boldsymbol{e_i}$

# PCA: Example




Let's project all data to $\boldsymbol{u_2}$

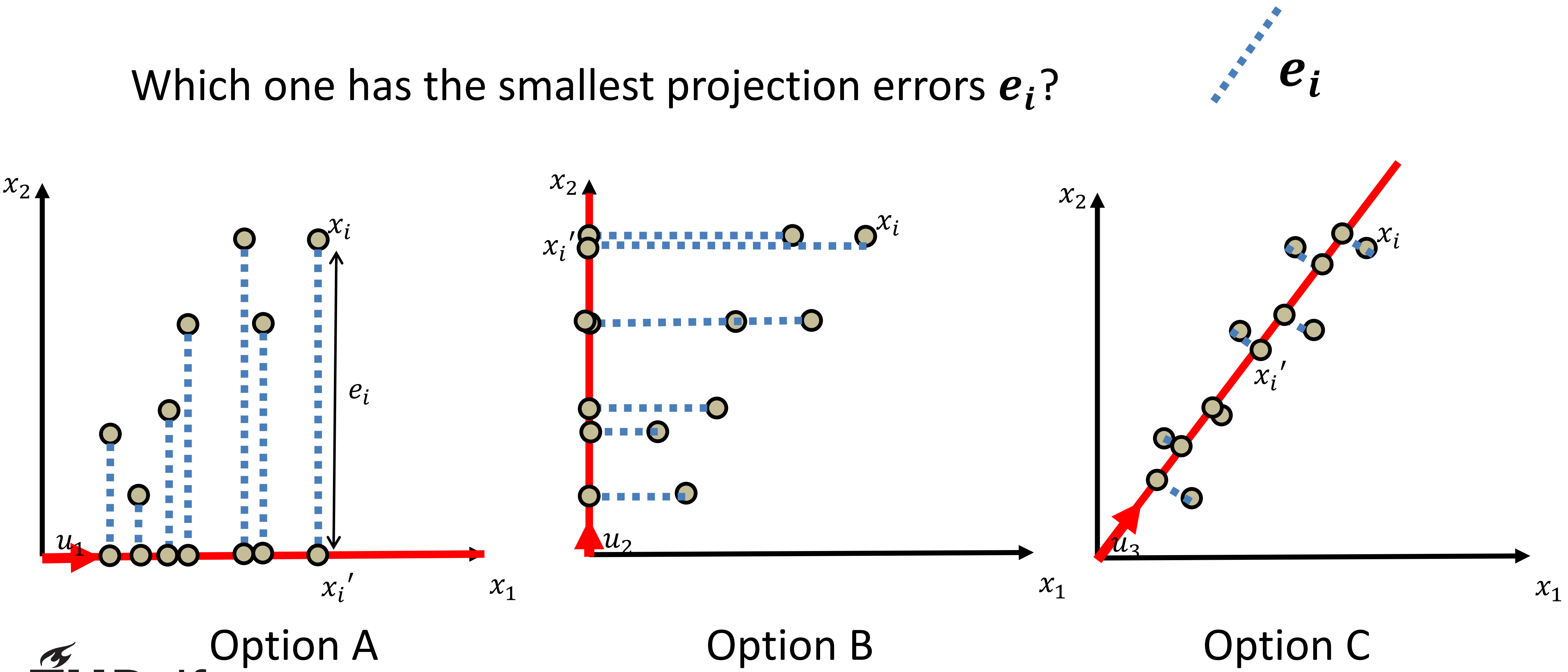
# PCA: Example




Let's project all data to $\boldsymbol{u_3}$

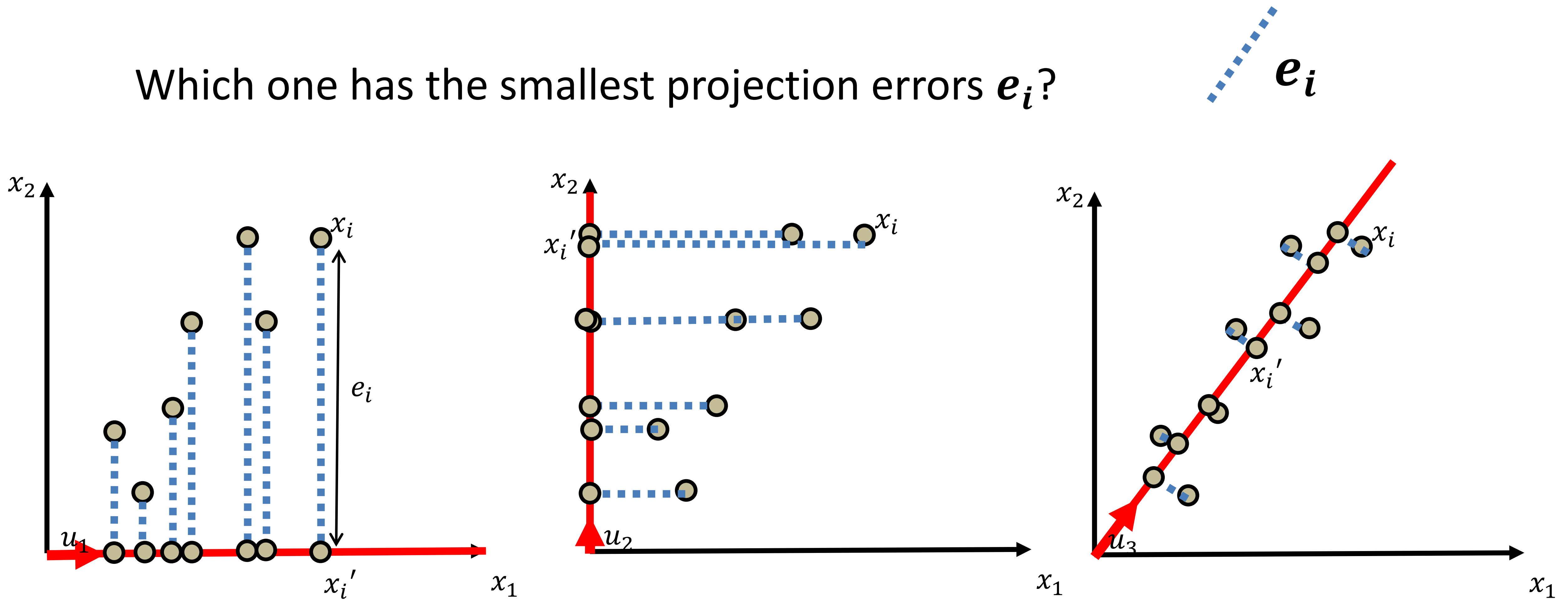
# PCA: Example

Which one has the smallest projection errors $\boldsymbol{e_i}$?



# PCA: Example

Which one has the smallest projection errors $\boldsymbol{e_i}$?



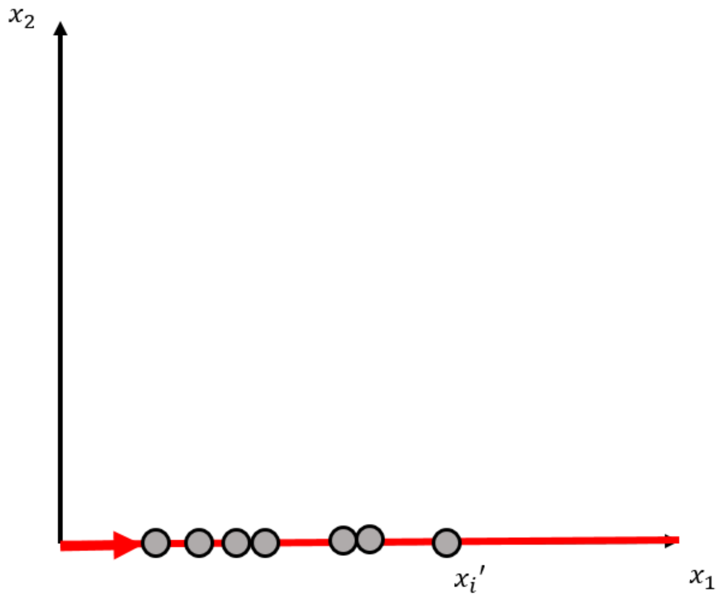

Option A

Option B

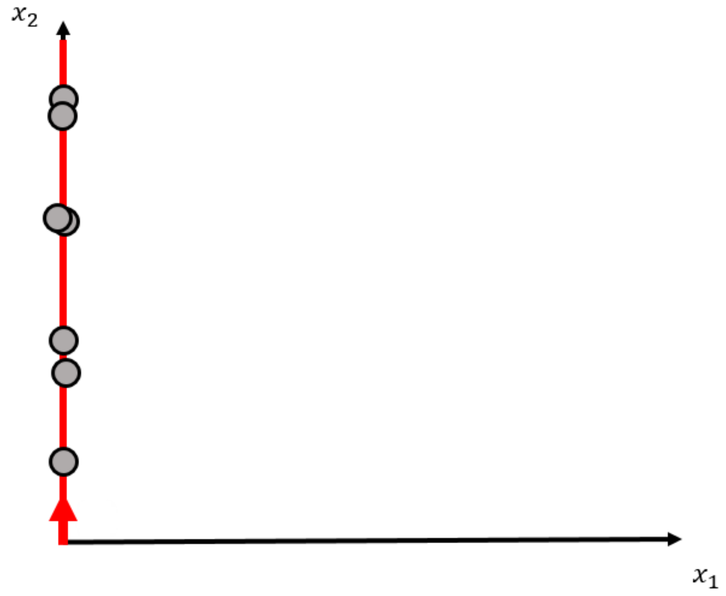
**Option C**

# PCA: Example

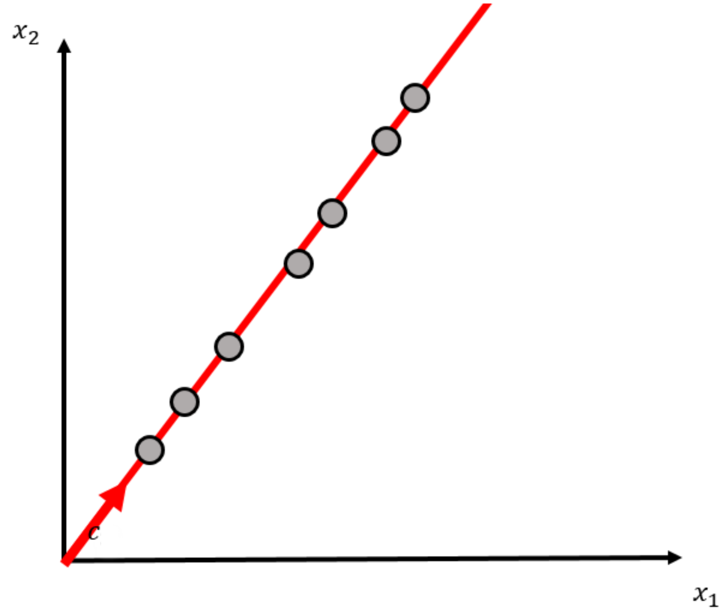
Which one has the largest variance after the projection?




Option A




Option B

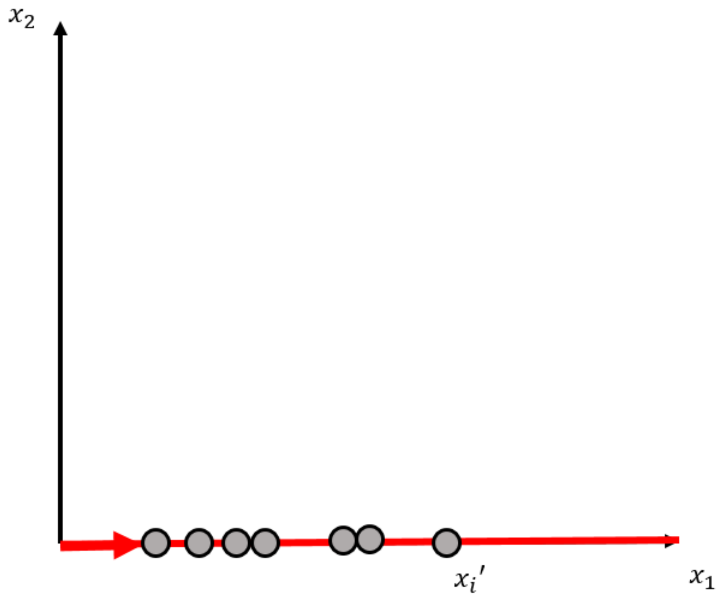



Option C

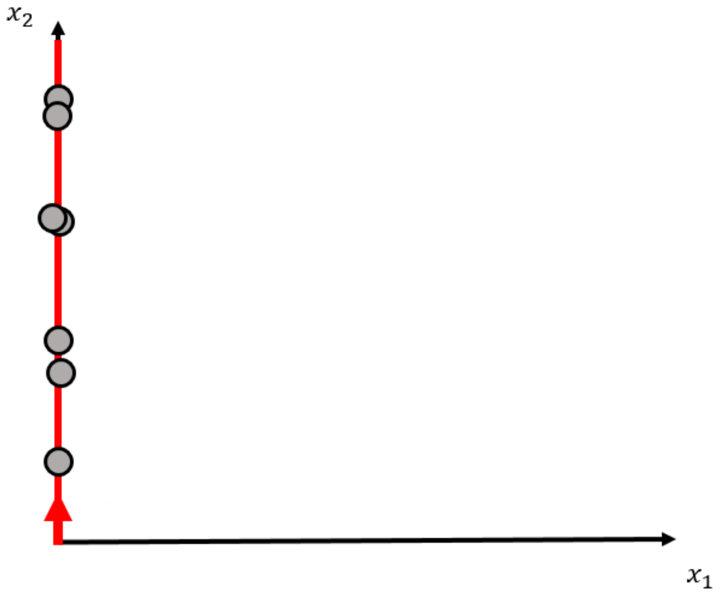
TU Delft

# PCA: Example

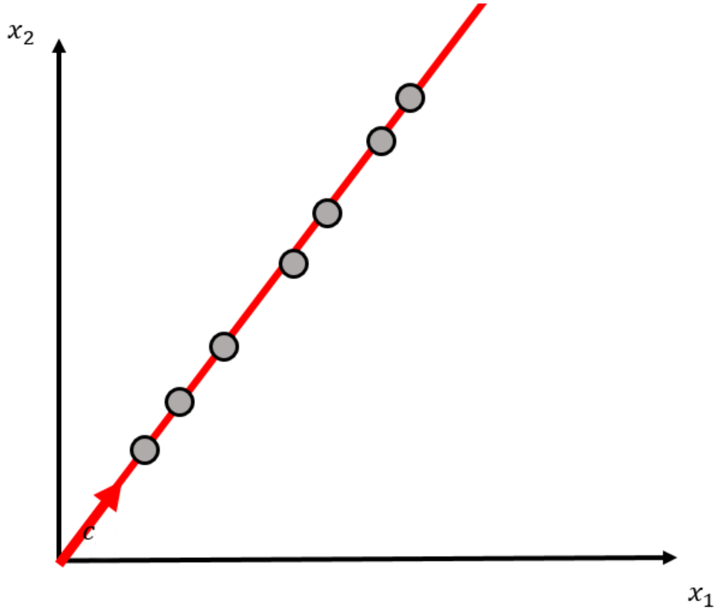
Which one has the largest variance after the projection?




Option A




Option B



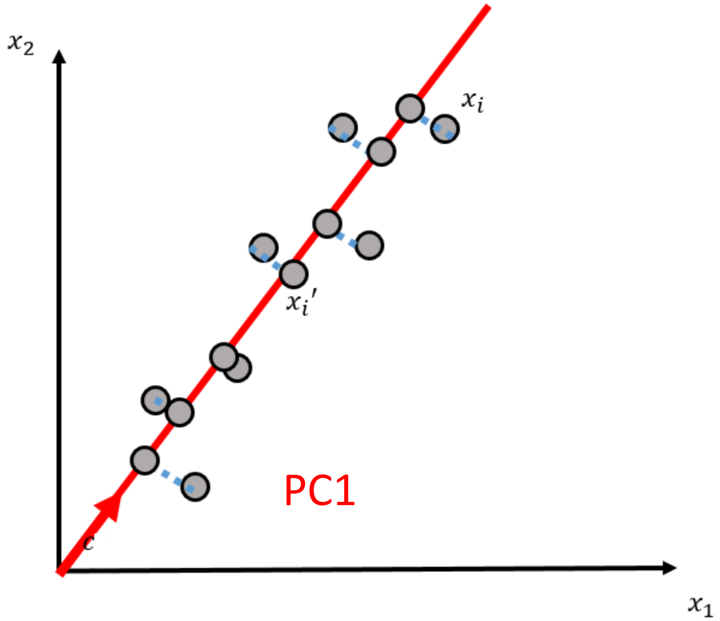

✅ **Option C**

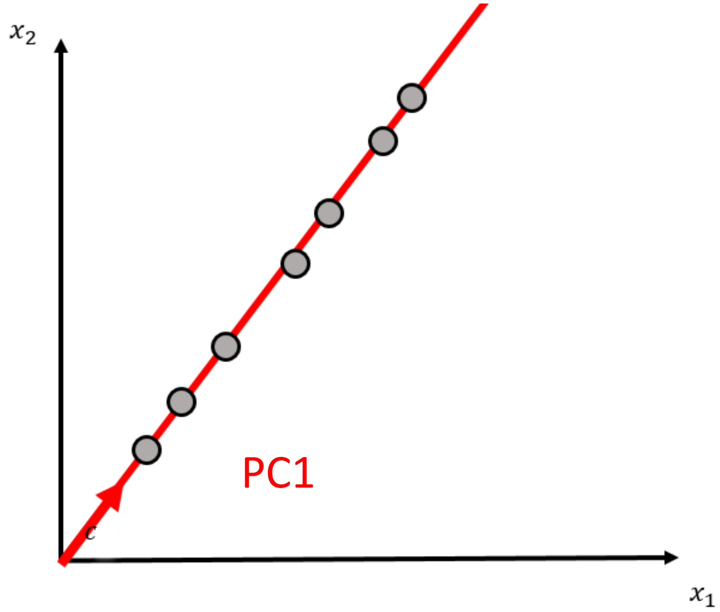
# PCA: Example

The first principal component is chosen such that

- It has the smallest projection error
- It has the largest variance after projection

Equivalent

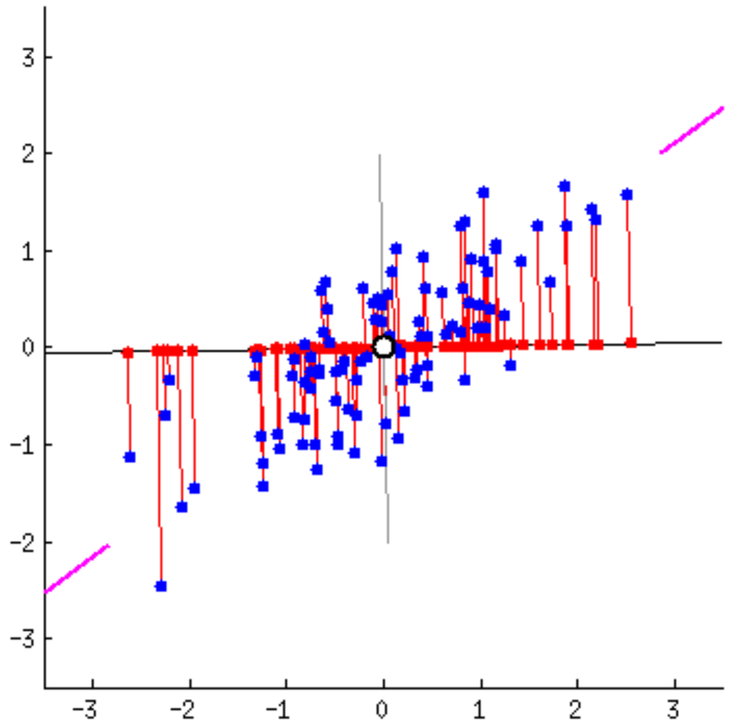



TU Delft

# Principal components analysis

- Principal Components Analysis maps the data onto a *linear subspace*, such that
  - the *variance* of the projected data is *maximized*
  - *The projection error is minimized*



# PCA: minimum error formulation

Let $\boldsymbol{x}$ be original data, $\tilde{x}$ is the projection on $M \leq D$ PC's.

Find $M$ orthogonal PC's such that the projection error

$$\frac{1}{N}\sum_{n=1}^{N}\|x_n - \tilde{x}_n\|^2$$

is smallest.

# PCA: maximum variance formulation

Let $\boldsymbol{x}$ be original data, $\boldsymbol{u_1}$ defines the direction of projection space. Find *M* orthogonal PC's such that the variance after projection is maximized:

$$\max_{\|u_1\|^2=1} var(u_1^T x)$$

TU Delft

# Principal components analysis

- Our objective is to maximize variance:

$$\max_{\|u_1\|^2=1} var(u_1^T x)$$

- The variance of the projected data is given by (S is covariance matrix): (see slide 30 for derivation)

$$var(u_1^T x) = [u_1^T X X^T u_1] = u_1^T S u_1$$

- Enforce constraint using *Lagrange multiplier* $\lambda$ (condition is $u_1^T u_1 = 1$):

$$\max_{\|u_1\|^2=1} var(u_1^T X) = \max_{u_1,\lambda} \; u_1^T S u_1 - \lambda(1 - u_1^T u_1)$$

- Find stationary point by setting derivative with respect to $u_1$ to zero:

$$\boldsymbol{S u_1 = \lambda u_1}$$

- ***This proves that eigenvales of covariance matrix give dimensions with max variance.***

TU Delft

# Principal components analysis

- The variance of the projected data is given by:

$$var(u_1^T x) = \frac{1}{N}\sum_{n=1}^{N}(u_1^T x_n - u_1^T \bar{x})^2 = \frac{1}{N}\sum_{n=1}^{N}(u_1^T x_n - u_1^T \bar{x})(u_1^T x_n - u_1^T \bar{x})^T$$

$$= \frac{1}{N}\sum_{n=1}^{N}(u_1^T x_n - u_1^T \bar{x})(x_n^T u_1 - \bar{x}^T u_1) = \frac{1}{N}\sum_{n=1}^{N} u_1^T (x_n - \bar{x})(x_n^T - \bar{x}^T) u_1 = \frac{1}{N}\sum_{n=1}^{N} u_1^T (x_n - \bar{x})(x_n - \bar{x})^T u_1$$

$$= u_1^T x x^T u_1 = u_1^T S u_1$$

- Where $(u_1^T \bar{x})$ is the mean of projected data where $\bar{x}$ is the sample mean
- Where S is covariance matrix
- $xx^T = S$ if data is zero-mean (mean of each dimension equals zero)

TUDelft

# Eigenvalues & eigenvectors: Definition

- $M$ square matrix, $\lambda$ constant, **e** a non-zero column vector
- $\lambda$ is an eigenvalue of $M$ and **e** is the corresponding eigenvector of $M$ if

$$Me = \lambda e$$

- Avoiding ambiguity regarding length: eigenvector to be *unit vector*
- $\lambda$ and **e** form eigenpairs
- Watch: 3blue1brown: **Eigenvectors and eigenvalues | Essence of linear algebra, chapter 14**

# Principal components analysis

- *Principal components* are given by the eigenvectors of the covariance matrix:

$$\boldsymbol{S}\boldsymbol{u}_{\mathbf{1}} = \boldsymbol{\lambda}\boldsymbol{u}_{\mathbf{1}}$$

- Covariance matrix is positive and semi-definite. This implies that the smallest possible eigenvalue is 0.
- First principal component is given by the eigenvector with the corresponding highest eigenvalue, etc.

# How to find eigenpairs?

- Pivotal condensation
- Power iteration

# How to find eigenpairs? Pivotal condensation

- Restate definition eigenpair $Su = \lambda u$ as:

$$(S - \lambda I)u = 0$$

- For this to hold the determinant of $(S - \lambda I)$ must be 0

- Determinant of $(S - \lambda I)$ is an *n*-th degree polynomial from which we can get the *n* values for λ that are eigenvalues of *S*

# Eigenpairs: Pivotal condensation (example)

- Set $S$ to $\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$
- Then $S - \lambda I$ $\begin{bmatrix} 3-\lambda & 2 \\ 2 & 6-\lambda \end{bmatrix}$
- Determinant is $(3-\lambda)(6-\lambda) - 4$
- Setting to zero, solving equation $\lambda^2 - 9\lambda + 14 = 0$
- Gives solutions $\lambda = 7 \text{ and } \lambda = 2$ being principal eigenvalues
- Let **u** be vector of unknowns $\begin{bmatrix} x \\ y \end{bmatrix}$
- Solve $\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 7 \begin{bmatrix} x \\ y \end{bmatrix}$

TU Delft

# Eigenpairs: Pivotal condensation (example)

- Two equations: $\begin{bmatrix} 3x + 2y & = & 7x \\ 2x + 6y & = & 7y \end{bmatrix}$
- Both saying the same thing $y = 2x$
- Possible eigenvector: $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$
- Make unit vector (divide by length): $\begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}$

- Second eigenvalue: repeat with $\lambda = 2$
- Equation becomes: $x = -2y$
- Second eigenvector: $\begin{bmatrix} 2/\sqrt{5} \\ -1/\sqrt{5} \end{bmatrix}$

# How to find eigenpairs?

- Pivotal condensation
- Power iteration

# How to find eigenpairs? Power iteration

- Start with any unit vector $\mathbf{x_0}$ (of appropriate length)
- Compute until it converges (*):

$$x_{k+1} := \frac{S\ x_k}{\|S\ x_k\|}$$

$\|A\|$ frobenius norm; the square root of the sum of the absolute squares of elements of N

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

- Limiting vector is the *principal eigenvector* (eigenvector with largest eigenvalue)
- When converged, compute eigenvalue $\lambda_1 = x^T S x$
- To find second eigenpair create new matrix $S^* = S - \lambda_1 x x^T$
- Use power iteration on $S^*$ to compute its principal eigenvector, etc.

(*) power iteration converges if all eigenvalues are non-negative, which is the case for covariance matrix

# Find eigenpairs with power iteration
# Example

- Let $S = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$
- Start with $\mathbf{x}_0$ being vector with 1s
- Multiply $S\,\mathbf{x}_0$: $$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$$
- Frobenius norm equals $$\sqrt{5^2 + 8^2} = \sqrt{89} = 9.434$$
- Obtain $\mathbf{x}_1$: $$\mathbf{x}_1 = \begin{bmatrix} 0.530 \\ 0.848 \end{bmatrix}$$

# Find eigenpairs with power iteration Example

- Next iteration: $$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} 0.530 \\ 0.848 \end{bmatrix} = \begin{bmatrix} 3.286 \\ 6.148 \end{bmatrix}$$
- Frobenius norm equals 6.971 so $x_2$ becomes

$$\mathbf{x}_2 = \begin{bmatrix} 0.471 \\ 0.882 \end{bmatrix}$$

- Repeat, converges to $\boldsymbol{x} = \begin{bmatrix} \mathbf{0.447} \\ \mathbf{0.894} \end{bmatrix}$
- Principal eigenvalue

$$\lambda = \mathbf{x}^T S \mathbf{x} = \begin{bmatrix} 0.447 & 0.894 \end{bmatrix} \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} 0.447 \\ 0.894 \end{bmatrix} = 6.993$$

# Find eigenpairs with power iteration Example

- To find second eigenpair create new matrix

$$S^* = S - \lambda_1 x x^T$$

- Use power iteration on $S^*$ to compute its principal eigenvector, etc.

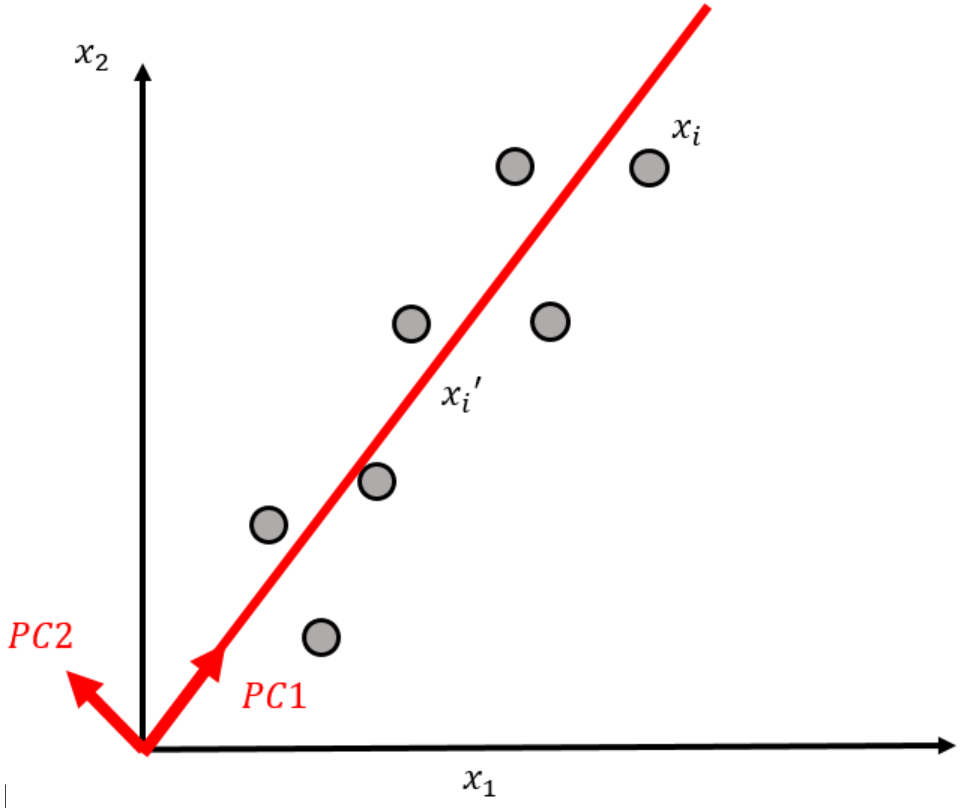
Why power iteration works? Check: https://www.math.usm.edu/lambers/mat610/sum10/lecture14.pdf

# Reducing dimensionality

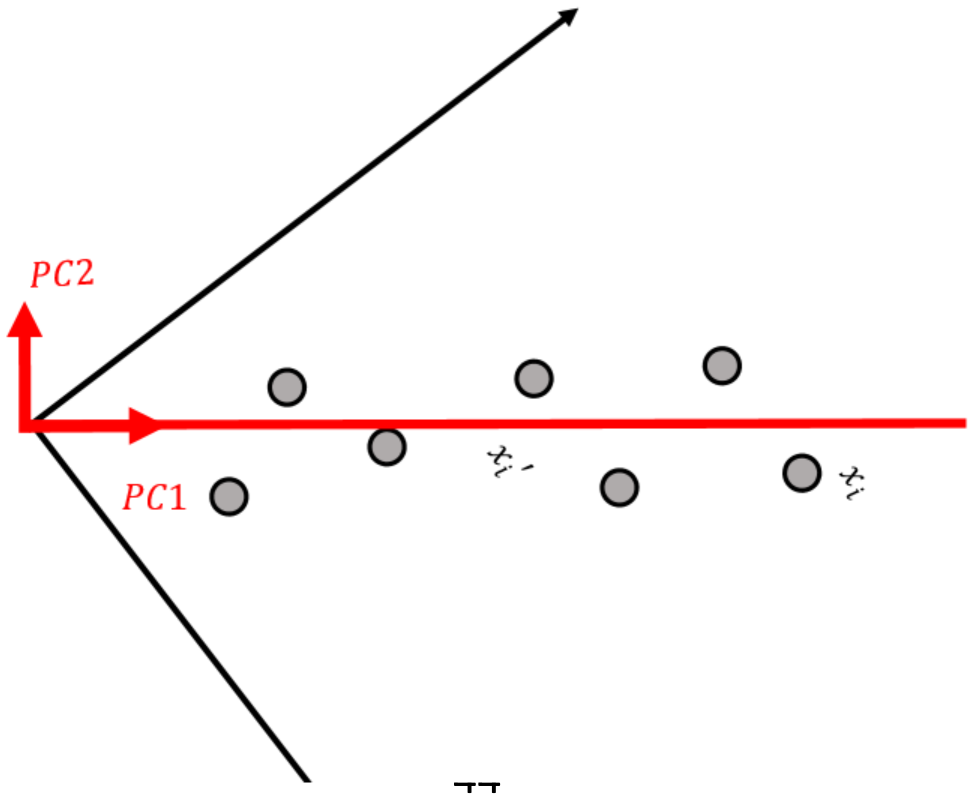
- X matrix whose rows represent points in Euclidean space
- Compute covariance matrix $S$ and its eigenpairs
- $E$ matrix whose columns are the eigenvectors, ordered as largest eigenvalues first
- $X^T E$ : points of X transformed into new coordinate space
  - First axis (largest eigenvalue) most significant
  - Second axis (second eigenpair), next most siginificant
- Let $E_M$ be first $M$ columns of $E$
- Then $X^T E_M$ is *M-dimensional* representation of X

# Reducing Dimensionality

## ORIGINAL SPACE: 2D



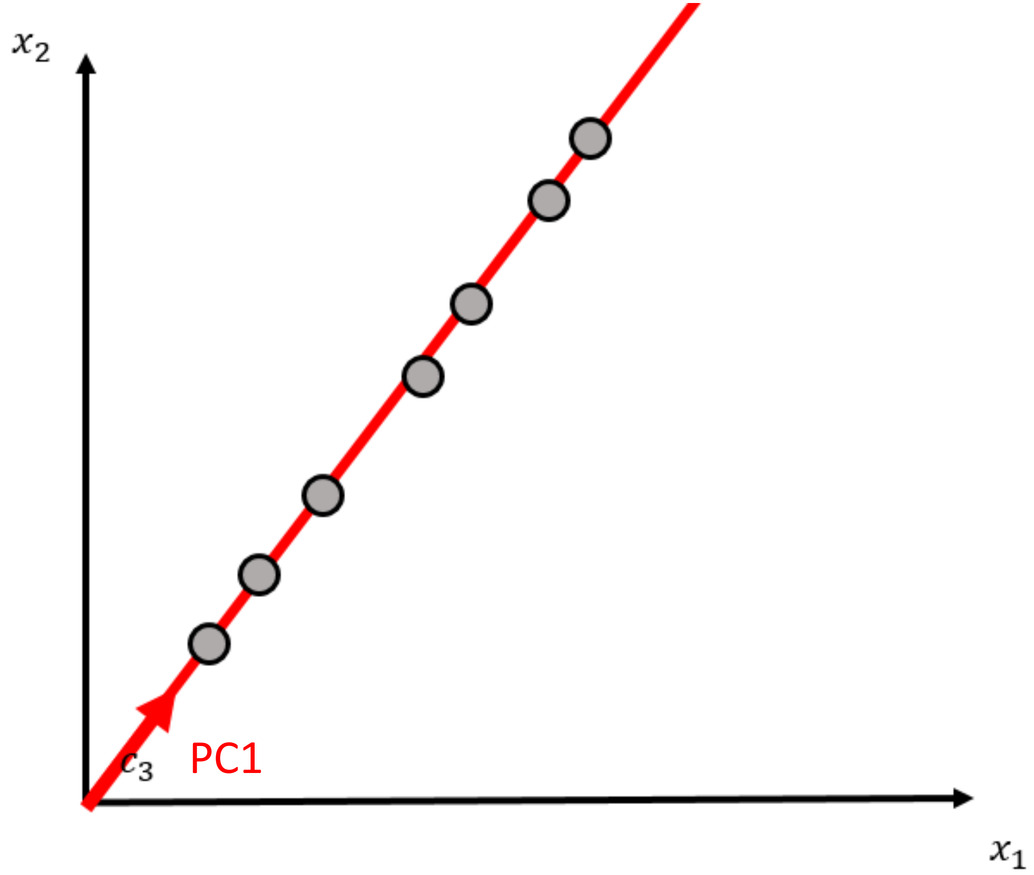

## PCA SPACE: 2D



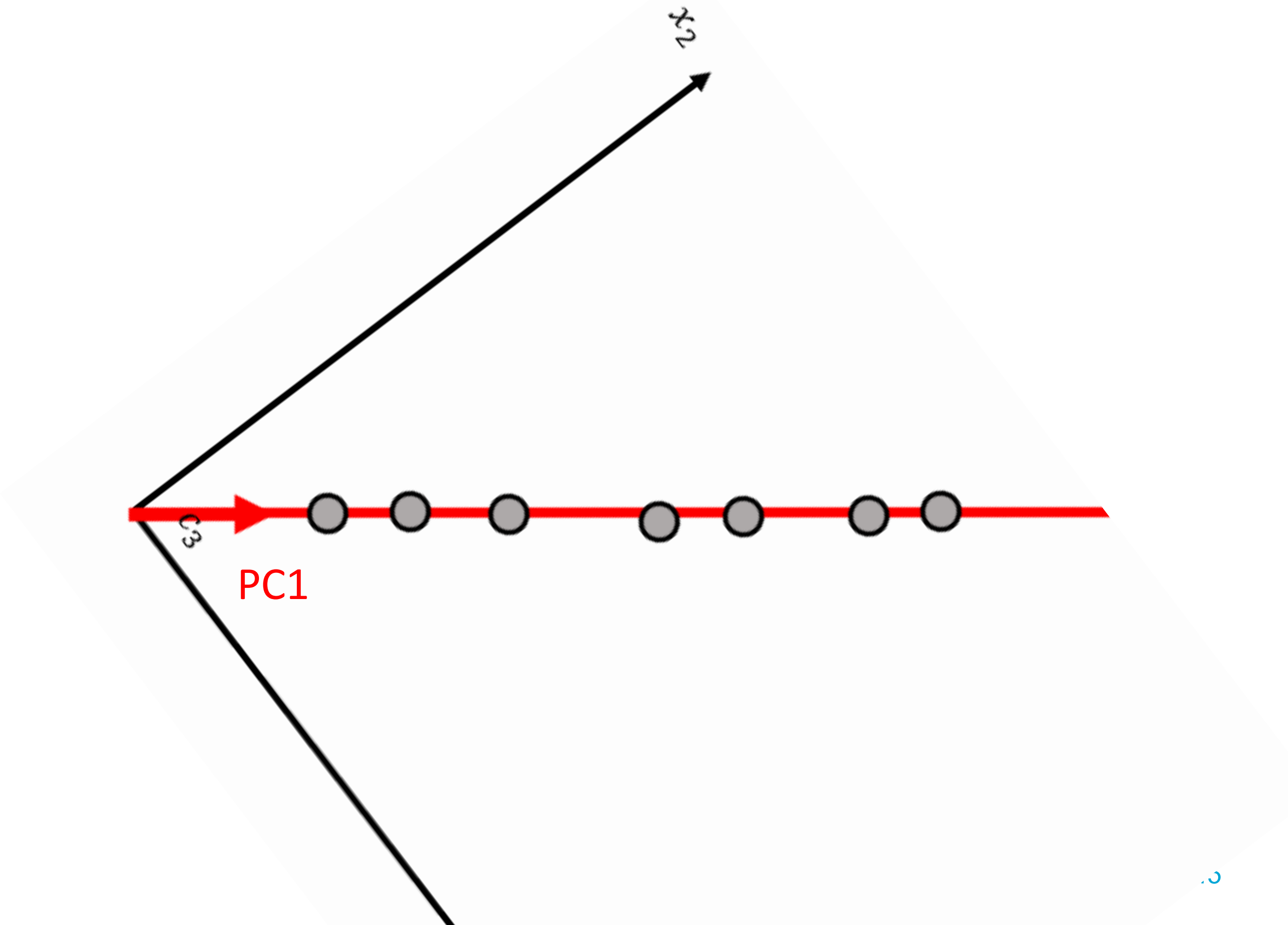

TU Delft

# Reducing Dimensionality

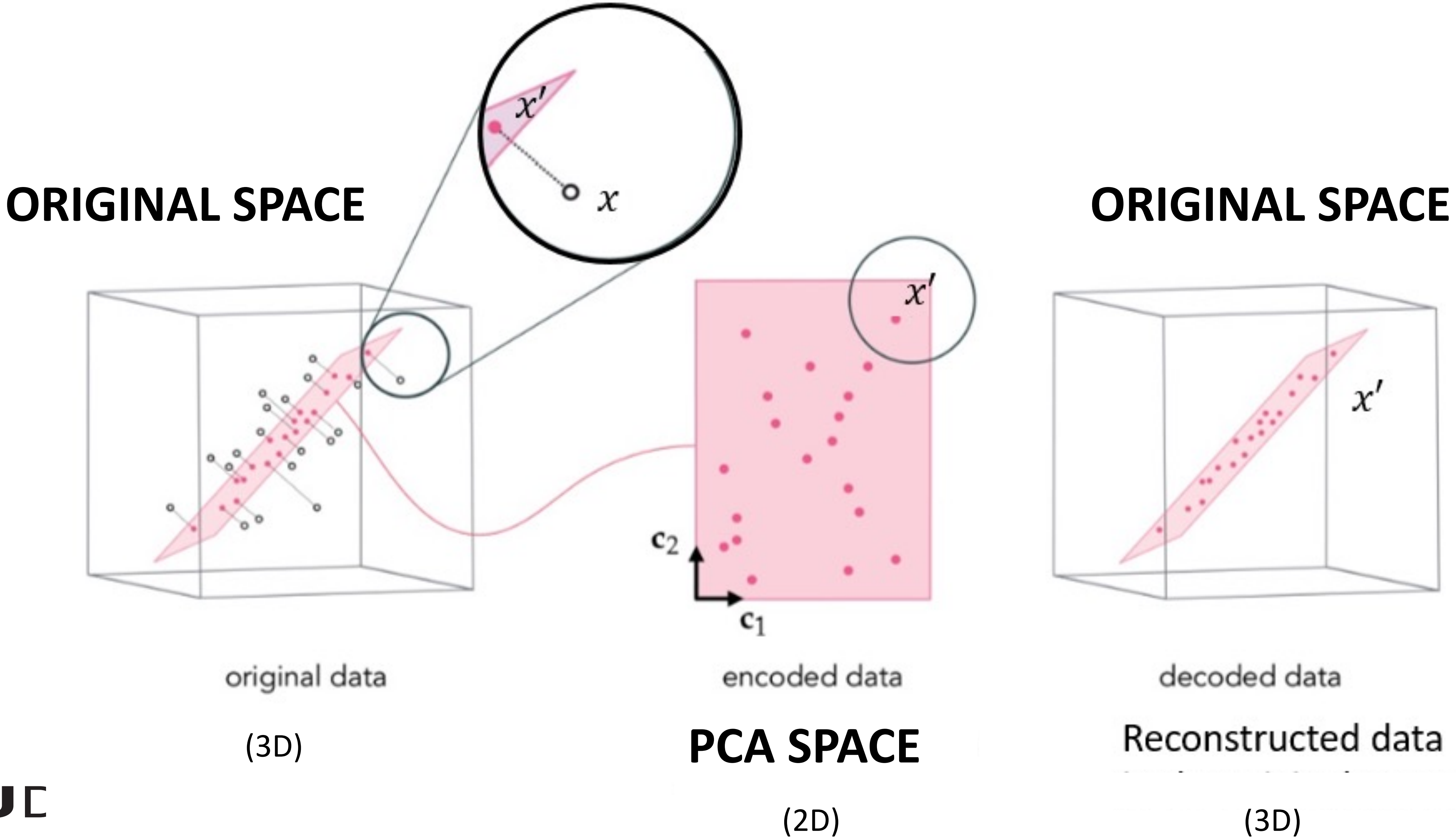
## ORIGINAL SPACE: 2D



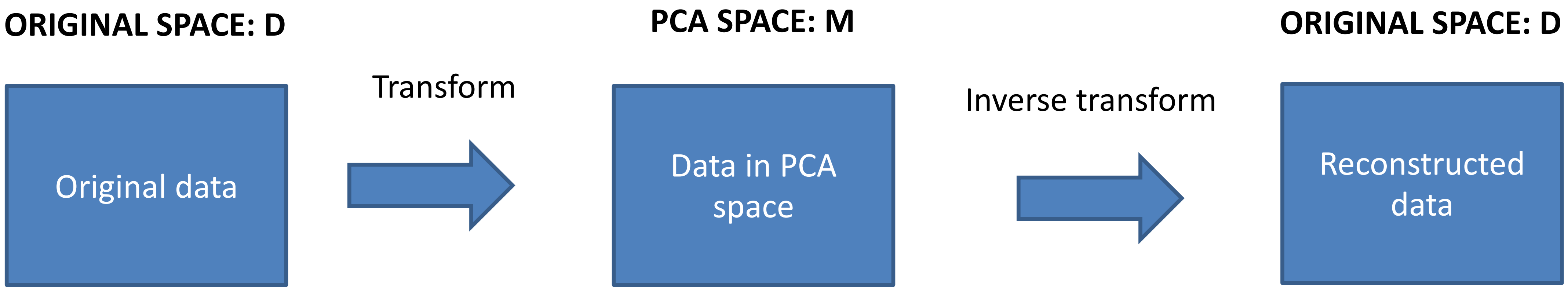

## PCA SPACE: 1D



# Reducing Dimensionality in 3D



# PCA




- Dimensionality reduction if M < D

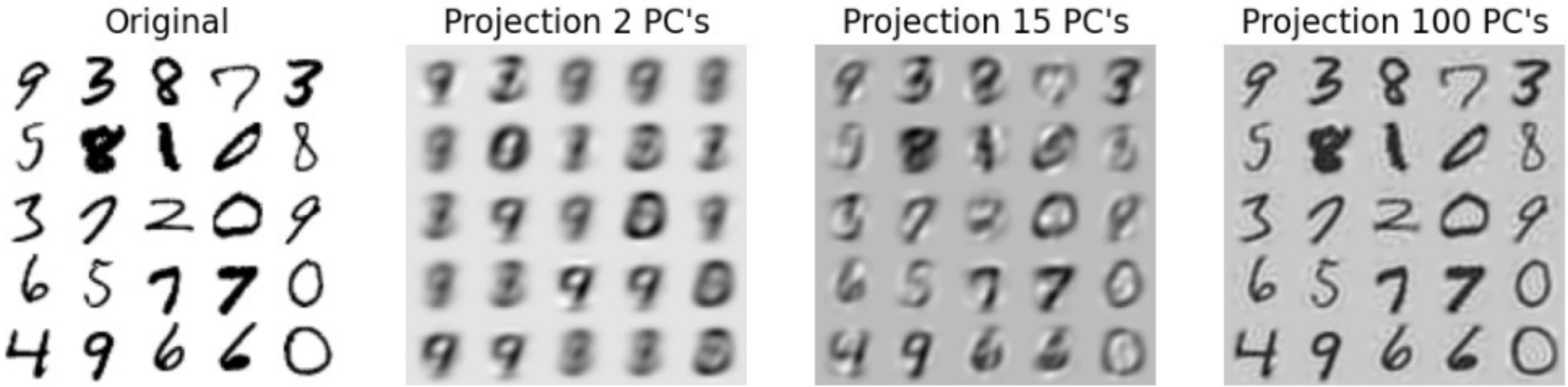
# PCA on MNIST

- Why can we expect to reduce the dimensionality?

- Many pixels are correlated
- Some pixels are always black together / white together
- Redundant information

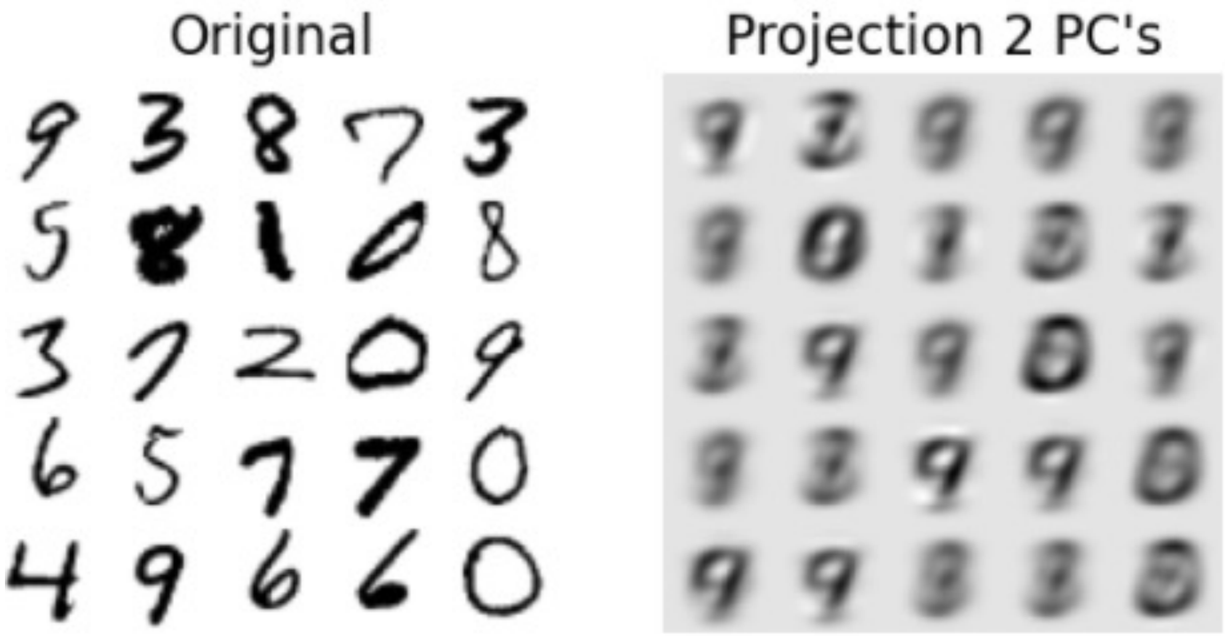
# PCA on MNIST

- PCA reconstructions

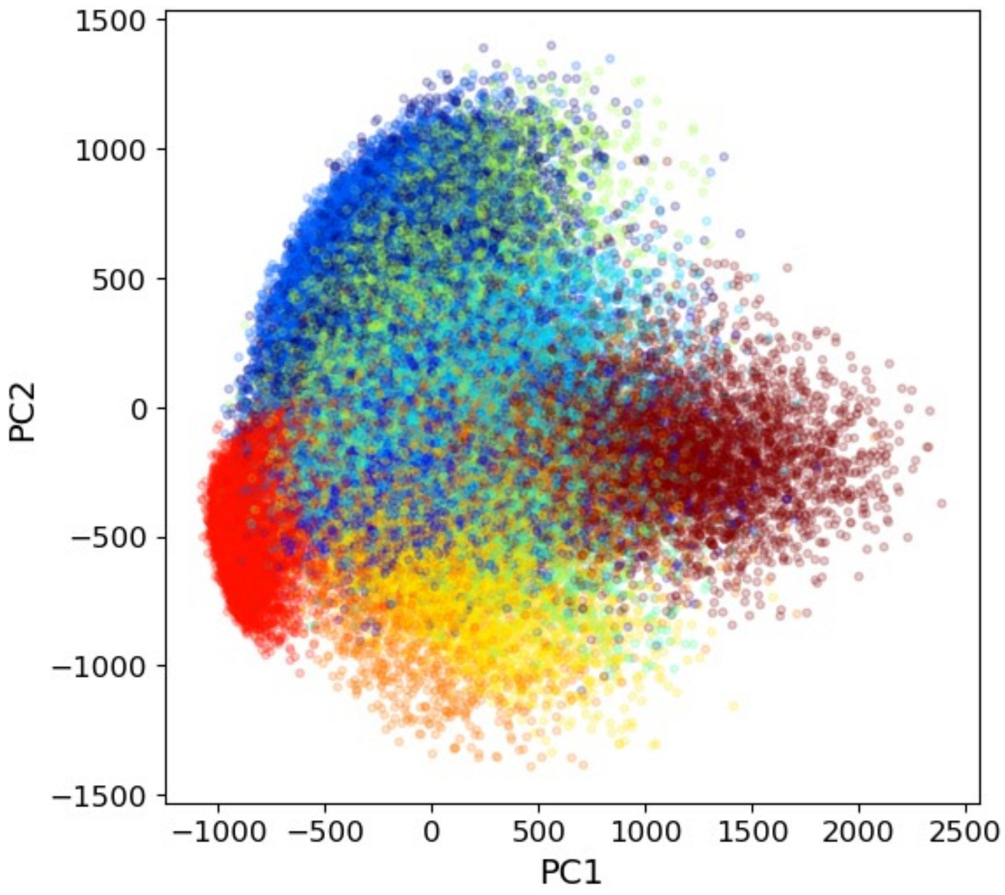


# PCA for visualization

**ORIGINAL IMAGE SPACE (786 D)**




**PCA SPACE 2D**




Colors indicate class

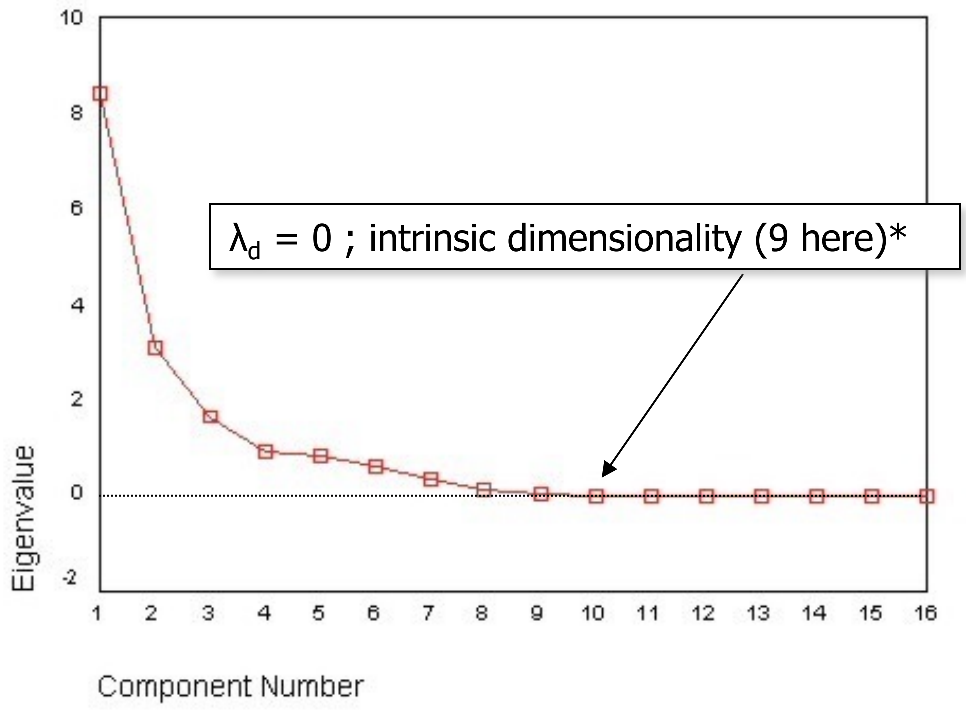
# PCA scree plot

- *Scree plot* of eigenvalues shows amount of variance retained by the eigenvectors (*principal components, PCs*):

- Eigenvalue represents the total amount of variance that can be explained by a given principal component.

- First *M* PCs explains $\frac{\sum_{m=1}^{M} \lambda_m}{\sum_{d=1}^{D} \lambda_d} * 100\%$ of variance

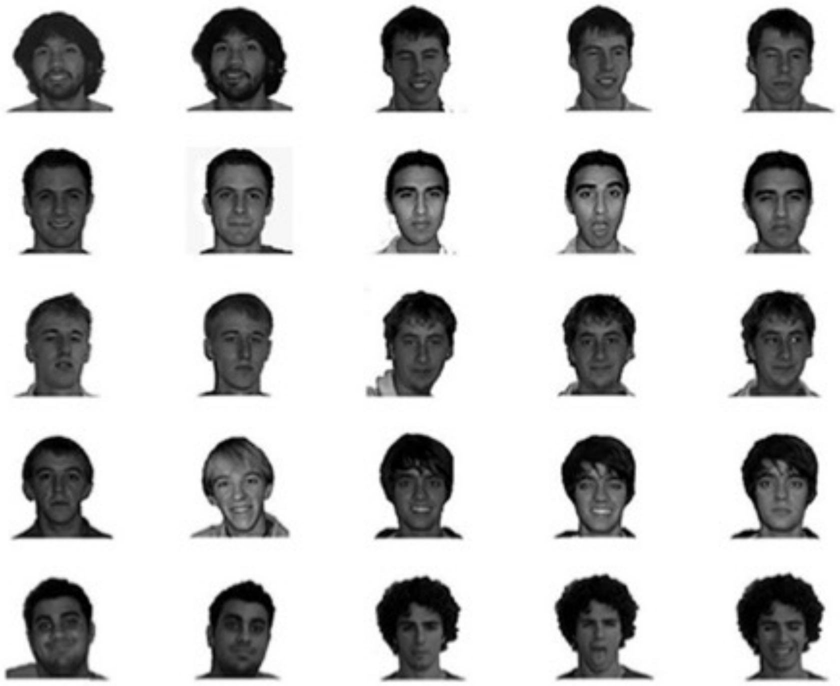
- For example: keep 95% of original variance
- Equivalent: accept a projection error of 5%




*the eigenvalues corresponding to 9 first components are positive and non-zero; some eigenvalues are zero if e.g. there are correlated features.

# Eigenfaces

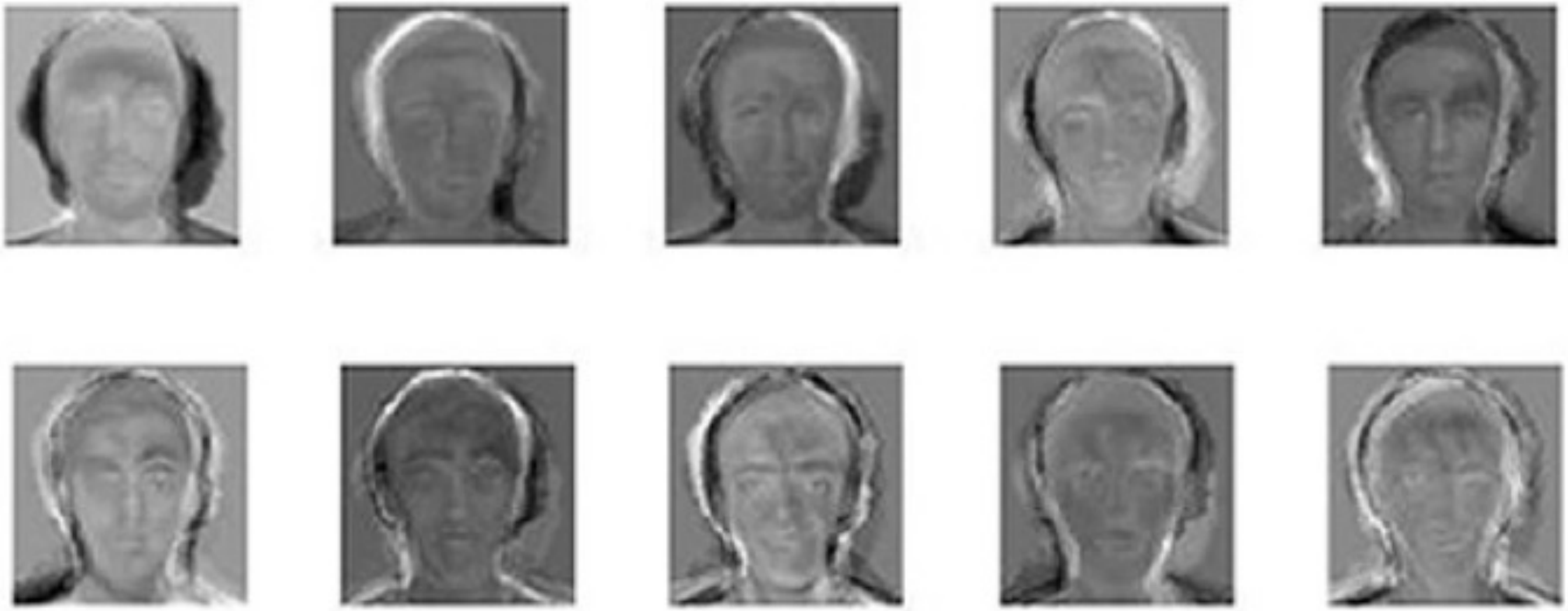
- Suppose we are applying PCA on the following set of face images:



- Image is matrix; *but* represented as a row vector !
- Eigenvectors also row vector, so eigenvector is also an image !

# Eigenfaces

- Example of first eigenvectors of set of face images (faces were aligned):



# Principal components analysis

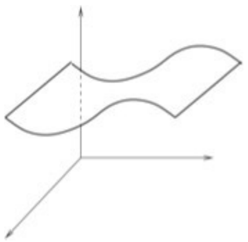
- Example reconstructions of face images and digits (using 30D PCA subspace):

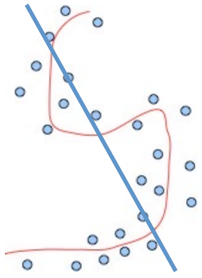
original

reconstructed

# PCA: practical issues

- Covariance extremely sensitive to large values
  - Multiply some dimensions by 1000
    - Dominates covariance
    - Becomes a principal component
  - Normalize each dimension to zero mean and unit variance: $x' = \frac{x - \mu}{\sigma}$
- PCA assumes underlying subspace is linear
  - 1d: straight line, 2d: plane

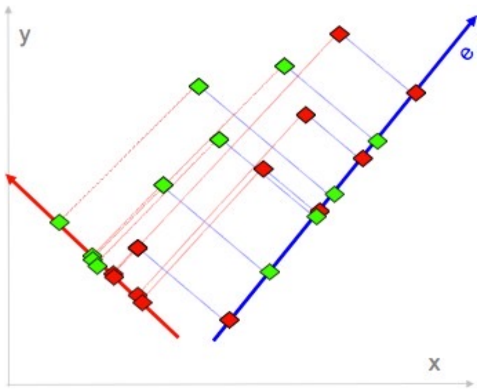




TU Delft

# PCA and classification

- PCA is unsupervised
  - maximizes overall variance of the data along a set of directions
  - does not know anything about class labels
  - can pick direction that makes it hard to separate classes
- Discriminative approach
  - look for a dimension that makes it easy to  separate classes
    - Lower dimensions, simpler models, thus less overfitting
    - Training is faster



# Principal Components Analysis

- Pros
  - allows to visualize high dimensional data
  - dramatic reduction in size of data
    - faster processing (as long as reduction is fast), smaller storage
- Cons
  - too expensive for many applications (Twitter, web)
  - need to understand assumptions behind the methods (linearity)

# PCA resources

- http://setosa.io/ev/principal-component-analysis/
- http://peterbloem.nl/blog/pca