Big Data Computing

Master's Degree in Computer Science

2023-2024

Gabriele Tolomei

Department of Computer Science

Sapienza Università di Roma

tolomei@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

Recap from Last Lecture

- Data often come with redundant and noisy (high-dimensional) representations

Recap from Last Lecture

- Data often come with redundant and noisy (high-dimensional) representations
- **Goal:** Extract the maximum possible information from the data while reducing the noise and ignoring redundancies

Recap from Last Lecture

- Data often come with redundant and noisy (high-dimensional) representations
- **Goal:** Extract the maximum possible information from the data while reducing the noise and ignoring redundancies
- PCA achieves this goal by transforming correlated features in the data into **linearly independent** (i.e., orthogonal) components

Recap from Last Lecture

- Data often come with redundant and noisy (high-dimensional) representations
- **Goal:** Extract the maximum possible information from the data while reducing the noise and ignoring redundancies
- PCA achieves this goal by transforming correlated features in the data into **linearly independent** (i.e., orthogonal) components
- As a result, data dimensionality can be reduced to these components

Eigenvectors of the Covariance Matrix

linear transformation

$$\boxed{K\mathbf{e} = \lambda\mathbf{e}}$$

eigenvector

eigenvalue

Eigenvectors of the Covariance Matrix

linear transformation

$$\boxed{K\mathbf{e} = \lambda\mathbf{e}}$$

eigenvector

eigenvalue

When you multiply a matrix by an eigenvector e the resulting vector does not change its direction, but it is only scaled by a factor λ (eigenvalue)

Eigenvectors of the Covariance Matrix

linear transformation

$$\boxed{K\mathbf{e} = \lambda\mathbf{e}}$$

eigenvector

eigenvalue

When you multiply a matrix by an eigenvector e the resulting vector does not change its direction, but it is only scaled by a factor λ (eigenvalue)

In other words, eigenvectors encapsulate all the relevant information to describe a linear transformation (in our case, represented by the covariance matrix K)

Eigenvectors of the Covariance Matrix

linear transformation

$$\boxed{K\mathbf{e} = \lambda\mathbf{e}}$$

eigenvector

eigenvalue

When you multiply a matrix by an eigenvector e the resulting vector does not change its direction, but it is only scaled by a factor λ (eigenvalue)

Principal Components
eigenvectors of the covariance matrix with the largest eigenvalues

How Do We Compute Eigenvectors?

Remember that we want to solve for e the following:

$$K\mathbf{e} = \lambda\mathbf{e}$$

How Do We Compute Eigenvectors?

Remember that we want to solve for e the following:

$$K\mathbf{e} = \lambda\mathbf{e}$$

We can rewrite the system of equations above as:

$$K\mathbf{e} - \lambda\mathbf{e} = 0 \Rightarrow (K - \lambda I)\mathbf{e} = 0$$

I is the identity matrix

How Do We Compute Eigenvectors?

We resort to solving the following homogeneous system:

$$(K - \lambda I)\mathbf{e} = 0$$

How Do We Compute Eigenvectors?

We resort to solving the following homogeneous system:

$$(K - \lambda I)\mathbf{e} = 0$$

Any homogeneous system always has a trivial solution, i.e., the zero vector e = 0

How Do We Compute Eigenvectors?

We resort to solving the following homogeneous system:

$$(K - \lambda I)\mathbf{e} = 0$$

Any homogeneous system always has a trivial solution, i.e., the zero vector $e = 0$

The only way for the homogeneous system above to have a non-trivial solution is for its matrix $(K - \lambda I)$ to be non-invertible, otherwise:

How Do We Compute Eigenvectors?

We resort to solving the following homogeneous system:

$$(K - \lambda I)\mathbf{e} = 0$$

Any homogeneous system always has a trivial solution, i.e., the zero vector e = 0

The only way for the homogeneous system above to have a non-trivial solution is for its matrix (K – λI) to be non-invertible, otherwise:

$$(K - \lambda I)(K - \lambda I)^{-1}\mathbf{e} = 0(K - \lambda I)^{-1}$$

How Do We Compute Eigenvectors?

We resort to solving the following homogeneous system:

$$(K - \lambda I)\mathbf{e} = 0$$

Any homogeneous system always has a trivial solution, i.e., the zero vector $e = 0$

The only way for the homogeneous system above to have a non-trivial solution is for its matrix $(K - \lambda I)$ to be non-invertible, otherwise:

$$\cancel{(K - \lambda I)}\cancel{(K - \lambda I)^{-1}}\boxed{\mathbf{e} = 0}\cancel{(K - \lambda I)^{-1}}$$

How Do We Compute Eigenvectors?

A square matrix is invertible iff its determinant is not 0

How Do We Compute Eigenvectors?

A square matrix is invertible iff its determinant is not 0

$\Downarrow$

If the determinant of the matrix $(K - \lambda I)$ is equal to 0, it is non-invertible

How Do We Compute Eigenvectors?

A square matrix is invertible iff its determinant is not 0

⇩

If the determinant of the matrix $(K - \lambda I)$ is equal to 0, it is non-invertible

⇩

The corresponding homogeneous system will have a non-trivial solution

How Do We Compute Eigenvectors?

1. Find the eigenvalues by solving for λ: det(K – λI) = 0

$$\det\left(\underbrace{\begin{bmatrix} 2-\lambda & 4/5 \\ 4/5 & 3/5-\lambda \end{bmatrix}}_{K-\lambda I}\right) = 0$$

How Do We Compute Eigenvectors?

1. Find the eigenvalues by solving for λ: det(K – λI) = 0

$$\det\Bigg(\underbrace{\begin{bmatrix} 2-\lambda & 4/5 \\ 4/5 & 3/5-\lambda \end{bmatrix}}_{K-\lambda I}\Bigg) = 0$$

$$(2-\lambda)(3/5-\lambda)-(4/5)(4/5) = \lambda^2-13/5\lambda+14/25$$

$\boxed{\lambda^2 - 13/5\lambda + 14/25 = 0}$ characteristic equation of K

How Do We Compute Eigenvectors?

1. Find the eigenvalues by solving for λ: det(K − λI) = 0

$$\det\left(\underbrace{\begin{bmatrix} 2-\lambda & 4/5 \\ 4/5 & 3/5-\lambda \end{bmatrix}}_{K-\lambda I}\right) = 0$$

$$(2-\lambda)(3/5-\lambda)-(4/5)(4/5) = \lambda^2-13/5\lambda+14/25$$

$$\boxed{\lambda^2 - 13/5\lambda + 14/25 = 0}$$ characteristic equation of K

$$\boxed{\lambda_1 = \frac{13+\sqrt{113}}{10} \approx 2.36;}\ \boxed{\lambda_2 = \frac{13-\sqrt{113}}{10} \approx 0.24}$$

How Do We Compute Eigenvectors?

2. Plug each eigenvalue in to find the corresponding eigenvector

$$\underbrace{\begin{bmatrix} 2 & 4/5 \\ 4/5 & 3/5 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} e_{1,1} \\ e_{1,2} \end{bmatrix}}_{\mathbf{e_1}} = \lambda_1 \underbrace{\begin{bmatrix} e_{1,1} \\ e_{1,2} \end{bmatrix}}_{\mathbf{e_1}}$$

$$\underbrace{\begin{bmatrix} 2 & 4/5 \\ 4/5 & 3/5 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} e_{2,1} \\ e_{2,2} \end{bmatrix}}_{\mathbf{e_2}} = \lambda_2 \underbrace{\begin{bmatrix} e_{2,1} \\ e_{2,2} \end{bmatrix}}_{\mathbf{e_2}}$$

How Do We Compute Eigenvectors?

Let's see what happens for λ_1

$$\begin{cases} 2e_{1,1} + 4/5e_{1,2} = \frac{13+\sqrt{113}}{10}e_{1,1} \\ 4/5e_{1,1} + 3/5e_{1,2} = \frac{13+\sqrt{113}}{10}e_{1,2} \end{cases}$$

How Do We Compute Eigenvectors?

Let's see what happens for λ_1

$$\begin{cases} 2e_{1,1} + 4/5e_{1,2} = \frac{13+\sqrt{113}}{10}e_{1,1} \\ 4/5e_{1,1} + 3/5e_{1,2} = \frac{13+\sqrt{113}}{10}e_{1,2} \end{cases}$$

Just replacing λ_1 ~ 2.36

$$\begin{cases} 2e_{1,1} + 0.8e_{1,2} = 2.36e_{1,1} \\ 0.8e_{1,1} + 0.6e_{1,2} = 2.36e_{1,2} \end{cases}$$

How Do We Compute Eigenvectors?

Let's see what happens for λ_1

$$\begin{cases} 2e_{1,1} + 4/5e_{1,2} = \frac{13+\sqrt{113}}{10}e_{1,1} \\ 4/5e_{1,1} + 3/5e_{1,2} = \frac{13+\sqrt{113}}{10}e_{1,2} \end{cases}$$

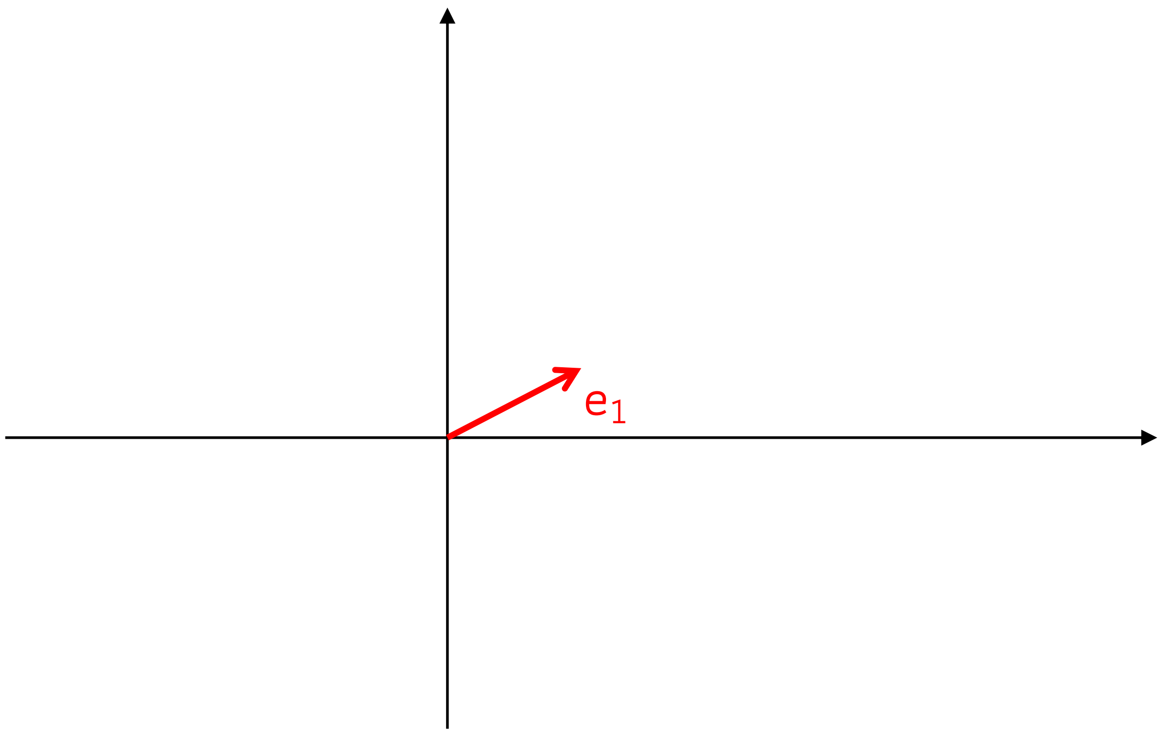
Just replacing λ_1 ~ 2.36

$$\begin{cases} 2e_{1,1} + 0.8e_{1,2} = 2.36e_{1,1} \\ 0.8e_{1,1} + 0.6e_{1,2} = 2.36e_{1,2} \end{cases} \Rightarrow \boxed{e_{1,1} \approx 2.2e_{1,2}}$$

The system has infintely many solutions

How Do We Compute Eigenvectors?

Any vector which satisfies the relationship above works!

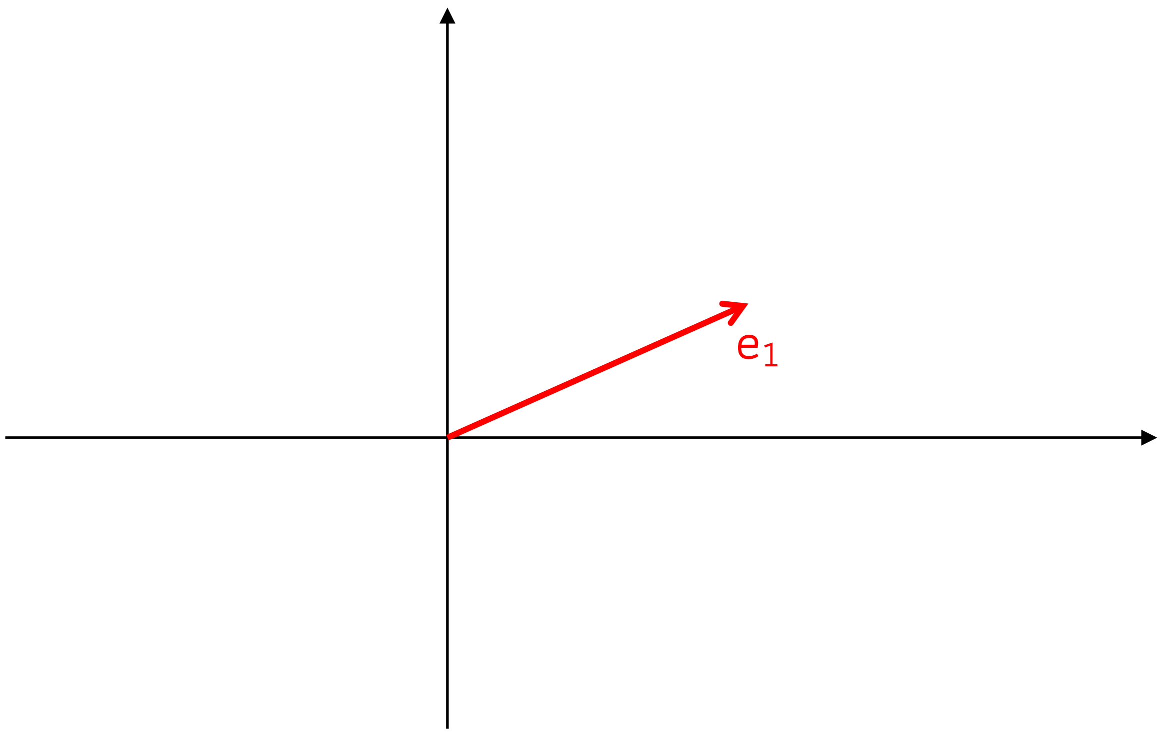


$$\begin{bmatrix} 1.1 \\ 0.5 \end{bmatrix}$$

$$e_{1,1} \approx 2.2 e_{1,2}$$

How Do We Compute Eigenvectors?

Any vector which satisfies the relationship above works!

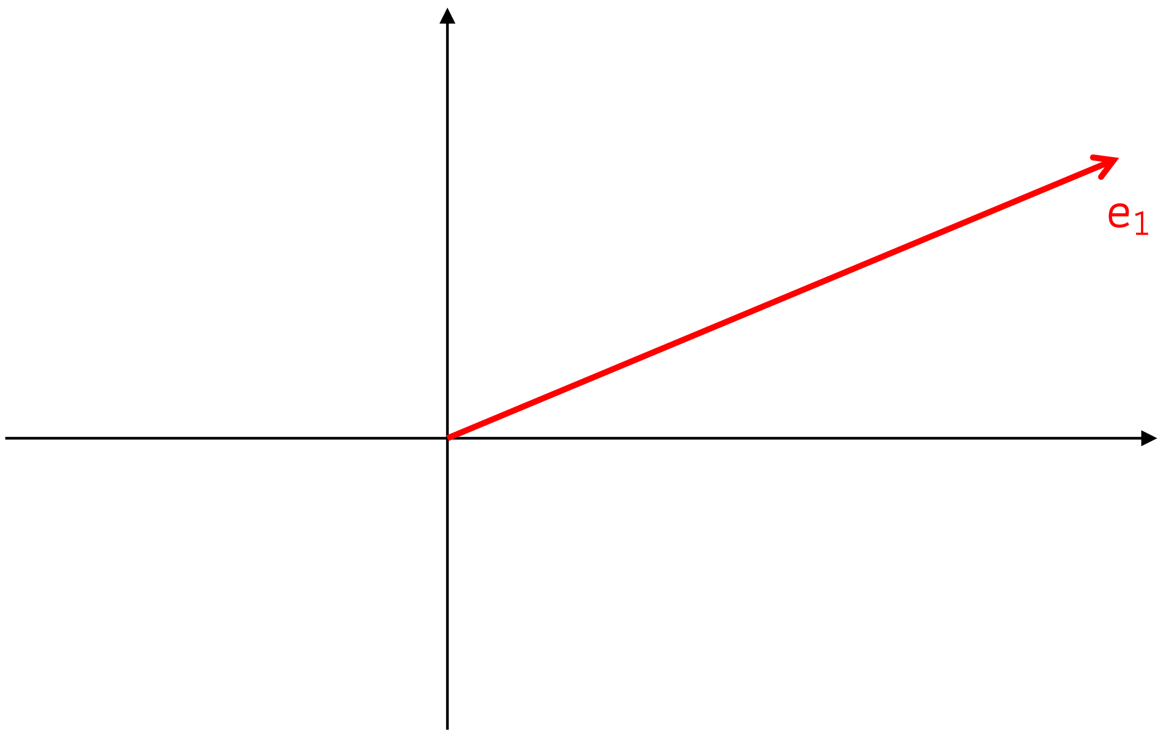


$$\begin{bmatrix} 2.2 \\ 1 \end{bmatrix}$$

$$e_{1,1} \approx 2.2 e_{1,2}$$

How Do We Compute Eigenvectors?

Any vector which satisfies the relationship above works!

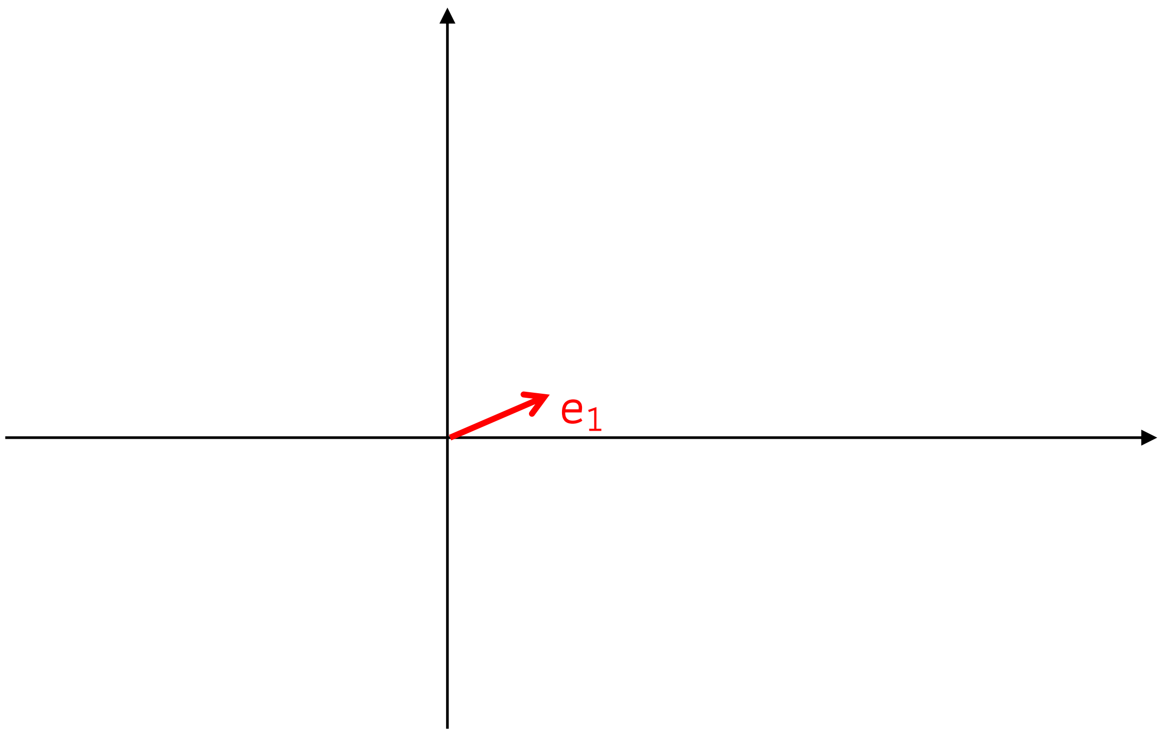


$$\begin{bmatrix} 4.4 \\ 2 \end{bmatrix}$$

$$e_{1,1} \approx 2.2 e_{1,2}$$

How Do We Compute Eigenvectors?

By convention, we restrict to $\|e_1\| = 1$

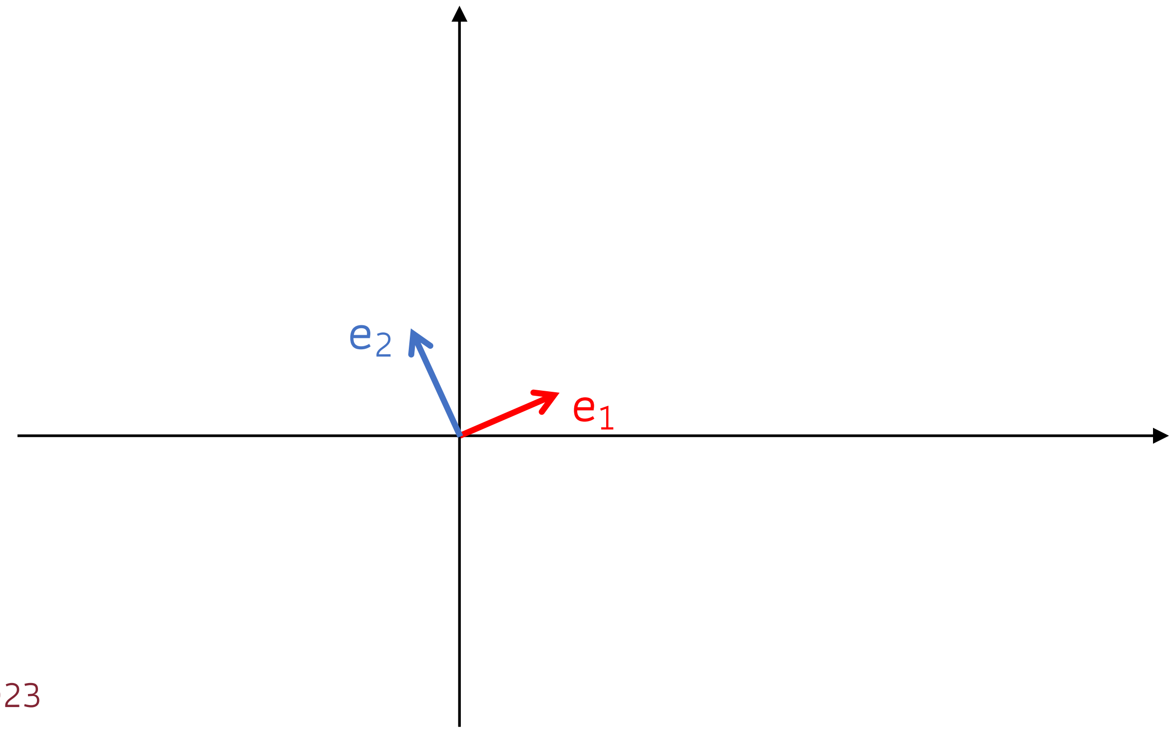


$$\begin{bmatrix} 0.91 \\ 0.41 \end{bmatrix}$$

$$e_{1,1} \approx 2.2 e_{1,2}$$

How Do We Compute Eigenvectors?

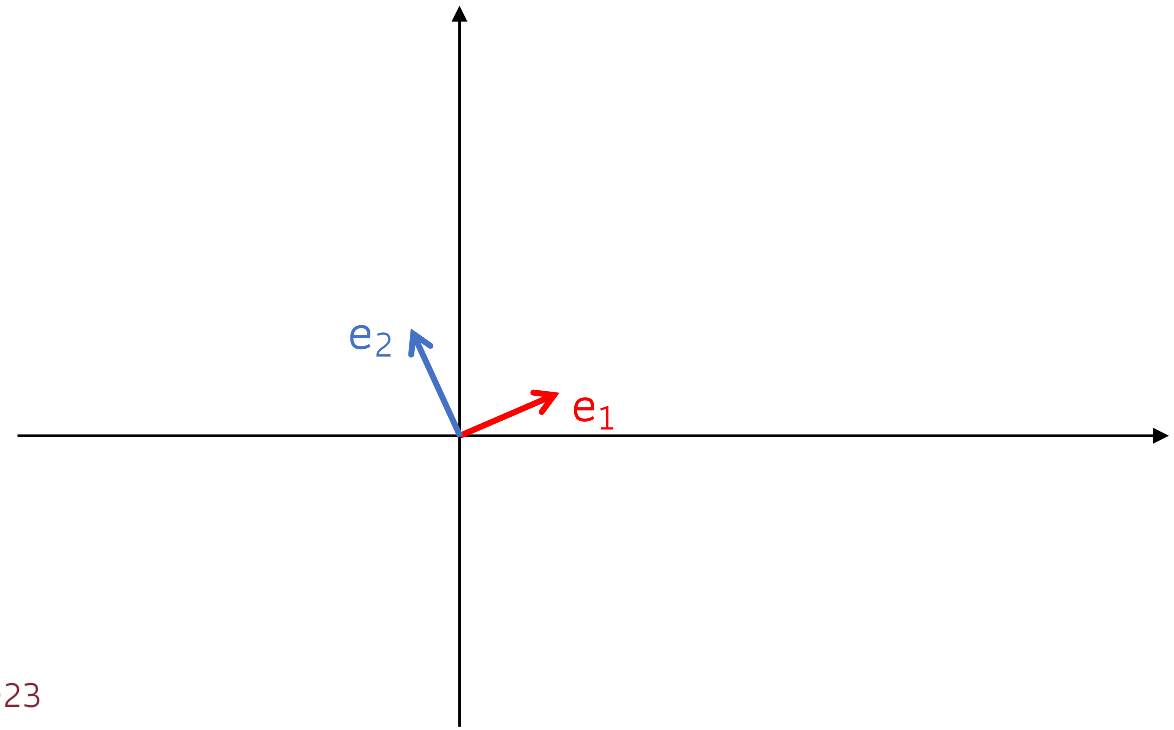
The second eigenvector e_2 can be found by plugging in the smaller eigenvalue λ_2



How Do We Compute Eigenvectors?

The second eigenvector e_2 can be found by plugging in the smaller eigenvalue λ_2

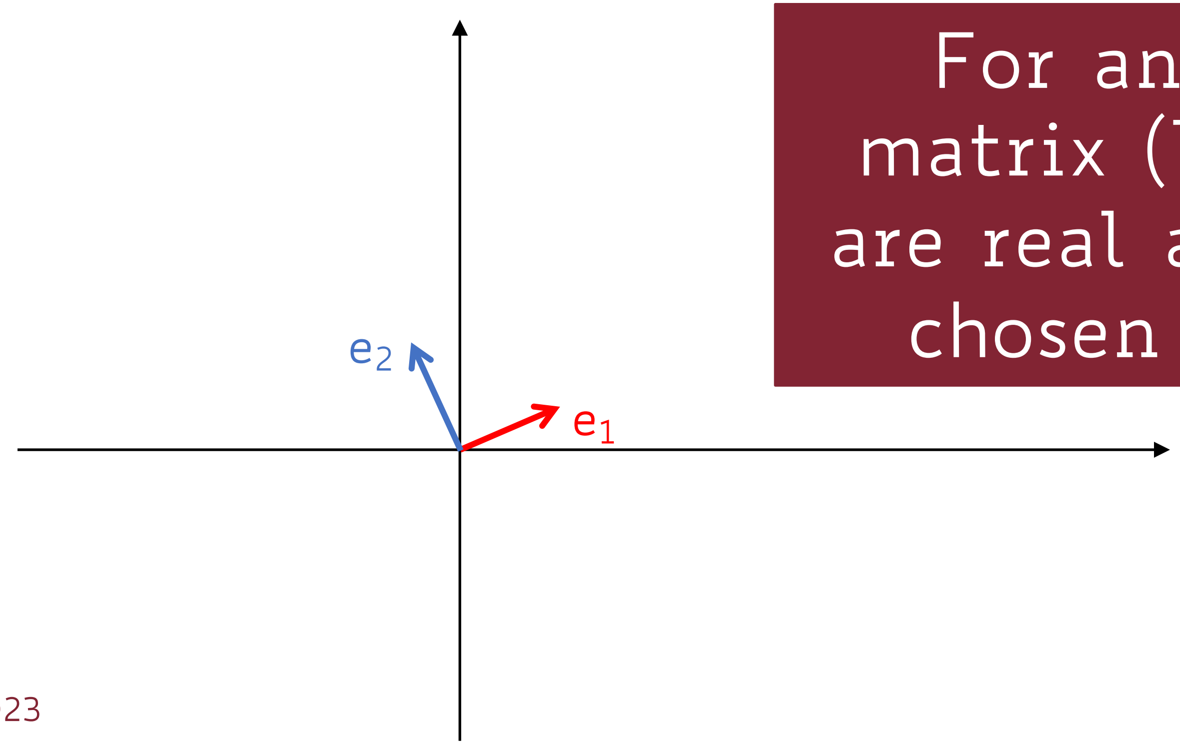
This is just orthogonal to the previously found e_1



How Do We Compute Eigenvectors?

The second eigenvector e_2 can be found by plugging in the smaller eigenvalue λ_2

This is just orthogonal to the previously found e_1

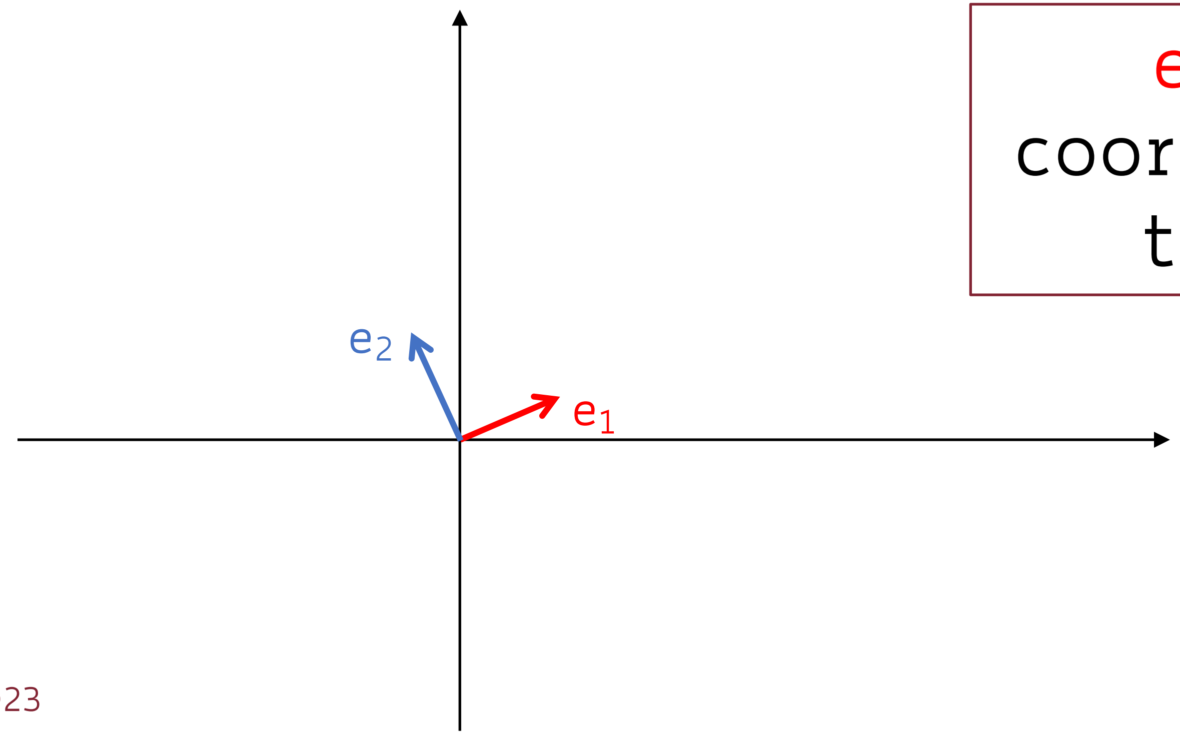


For any dxd real symmetric matrix (like K), the eigenvalues are real and eigenvectors can be chosen real and orthonormal

How Do We Compute Eigenvectors?

The second eigenvector e_2 can be found by plugging in the smaller eigenvalue λ_2

This is just orthogonal to the previously found e_1



e_1 and e_2 are the new coordinate system replacing the original x_1 and x_2

$$\mathbf{e_1} = \begin{bmatrix} 0.91 \\ 0.41 \end{bmatrix} \quad \mathbf{e_2} = \begin{bmatrix} -0.41 \\ 0.91 \end{bmatrix}$$

Principal Components

$$\mathbf{e}_1 = \begin{bmatrix} 0.91 \\ 0.41 \end{bmatrix} \quad \mathbf{e}_2 = \begin{bmatrix} -0.41 \\ 0.91 \end{bmatrix}$$

e_1 is the 1st principal component as it is the eigenvector corresponding to the largest eigenvalue

e_2 is the 2nd principal component as it is the eigenvector corresponding to the smallest eigenvalue

Projecting to New Dimensions: 2-d Case

- e_1 and e_2 identify our new coordinate system (principal components)

Projecting to New Dimensions: 2-d Case

- $\mathbf{e}_1$ and $\mathbf{e}_2$ identify our new coordinate system (principal components)
- Both of them are 2-dimensional vectors

Projecting to New Dimensions: 2-d Case

- $\mathbf{e}_1$ and $\mathbf{e}_2$ identify our new coordinate system (principal components)
- Both of them are 2-dimensional vectors
- Let $\mathbf{x} = (x_1, x_2)$ be a point (i.e., a vector) in the original ($\mathbf{x}_1$, $\mathbf{x}_2$)-space

Projecting to New Dimensions: 2-d Case

- $\mathbf{e}_1$ and $\mathbf{e}_2$ identify our new coordinate system (principal components)
- Both of them are 2-dimensional vectors
- Let $\mathbf{x} = (x_1, x_2)$ be a point (i.e., a vector) in the original $(\mathbf{x}_1, \mathbf{x}_2)$-space

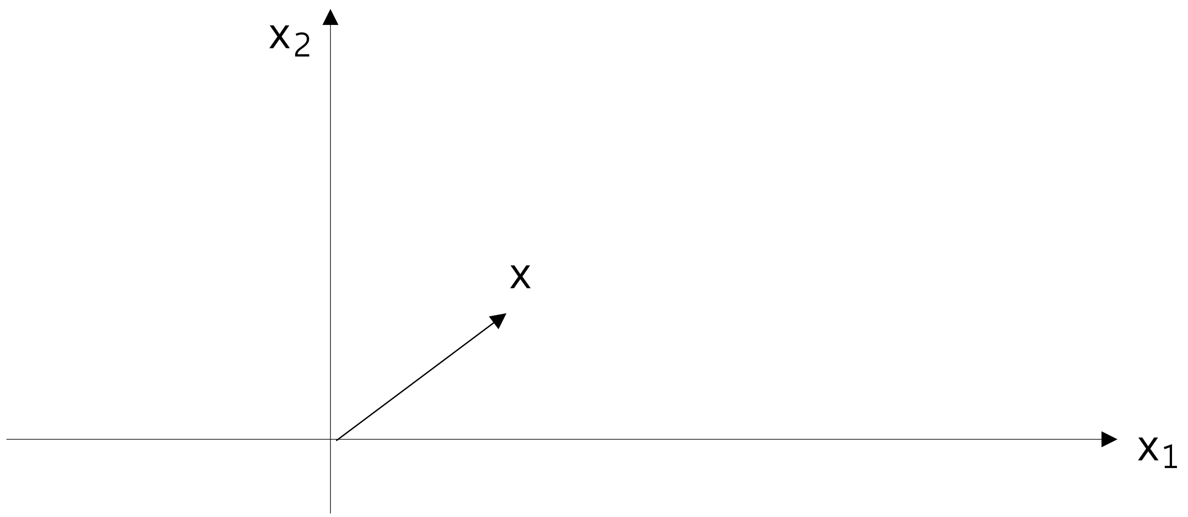
Goal

We want to represent x in the new $(\mathbf{e}_1, \mathbf{e}_2)$-coordinate system

Projecting to New Dimensions: 2-d Case

1. Center x around the mean of each dimension

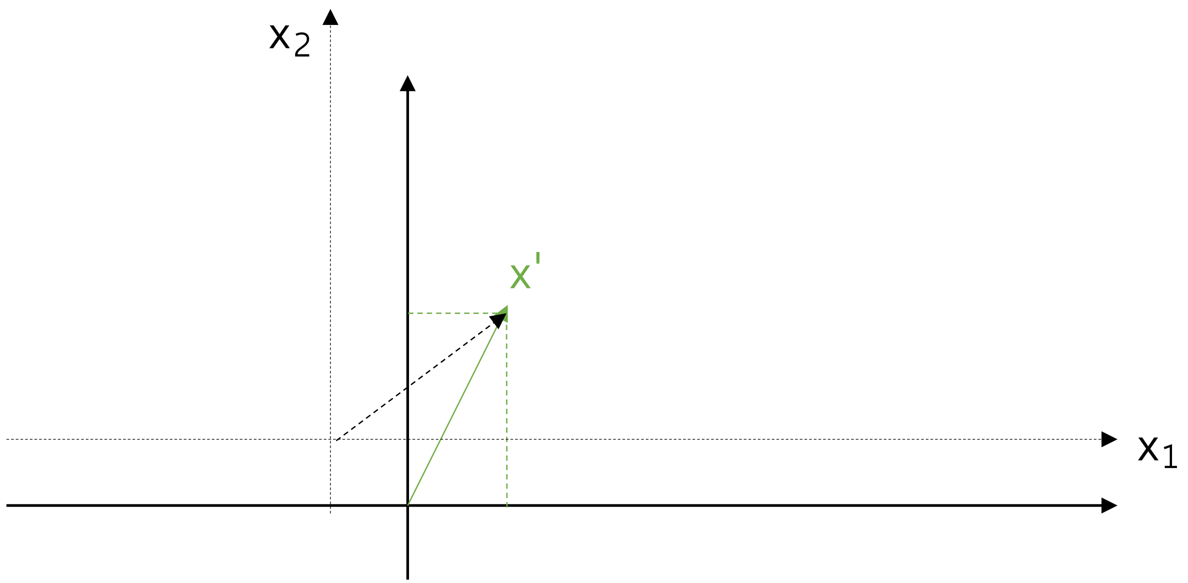
$$\mathbf{x}' = \mathbf{x} - \boldsymbol{\mu} = (x_1 - \mu_1, x_2 - \mu_2)$$



Projecting to New Dimensions: 2-d Case

1. Center x around the mean of each dimension

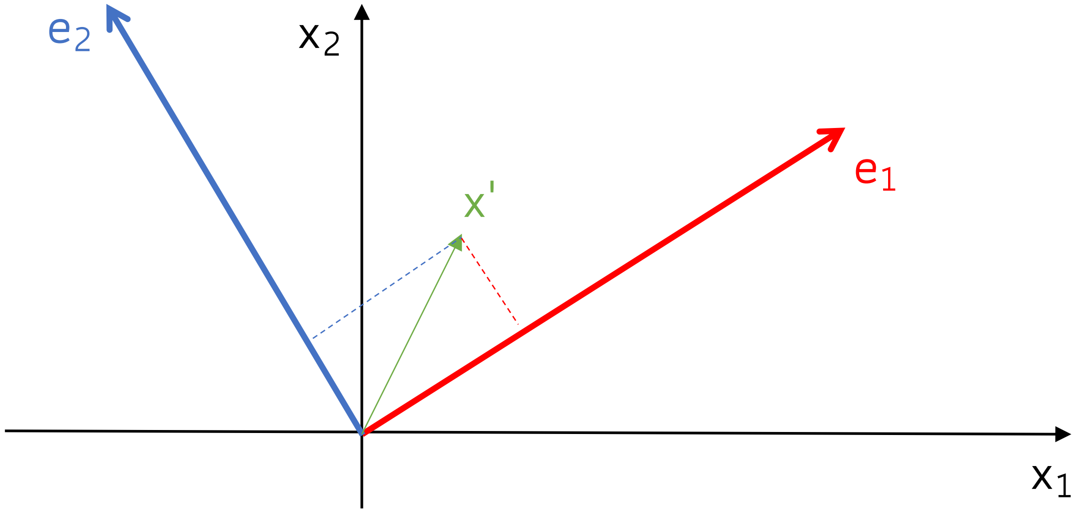
$$\mathbf{x}' = \mathbf{x} - \boldsymbol{\mu} = (x_1 - \mu_1, x_2 - \mu_2)$$



Projecting to New Dimensions: 2-d Case

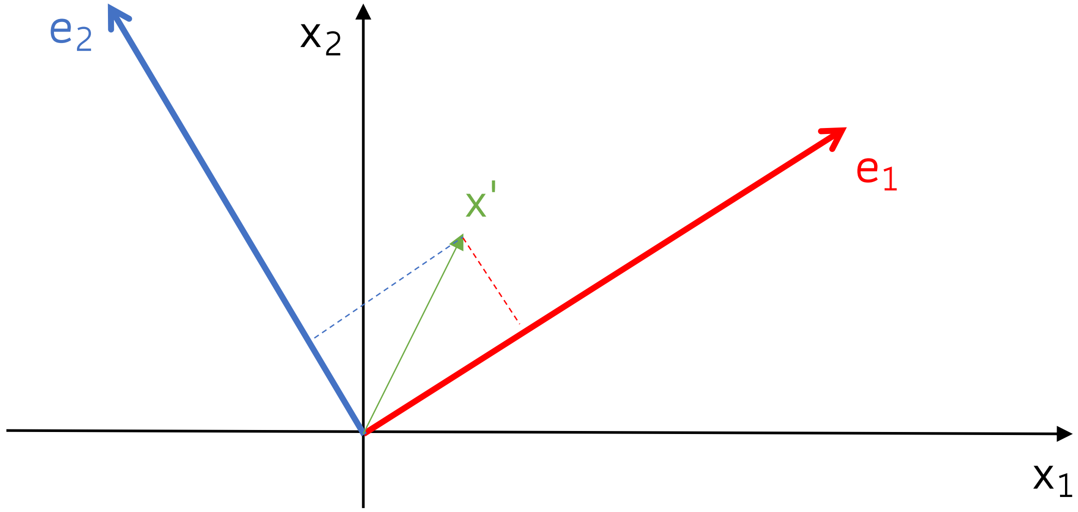
2. Project x' on each dimension e_1 and e_2

$$\mathbf{x}' = \underbrace{(x'_1, x'_2)}_{\text{coordinates of } \mathbf{x}' \text{ in the } (\mathbf{e}_1, \mathbf{e}_2)\text{-space}} = ({\mathbf{x}'}^T \mathbf{e}_1, {\mathbf{x}'}^T \mathbf{e}_2)$$



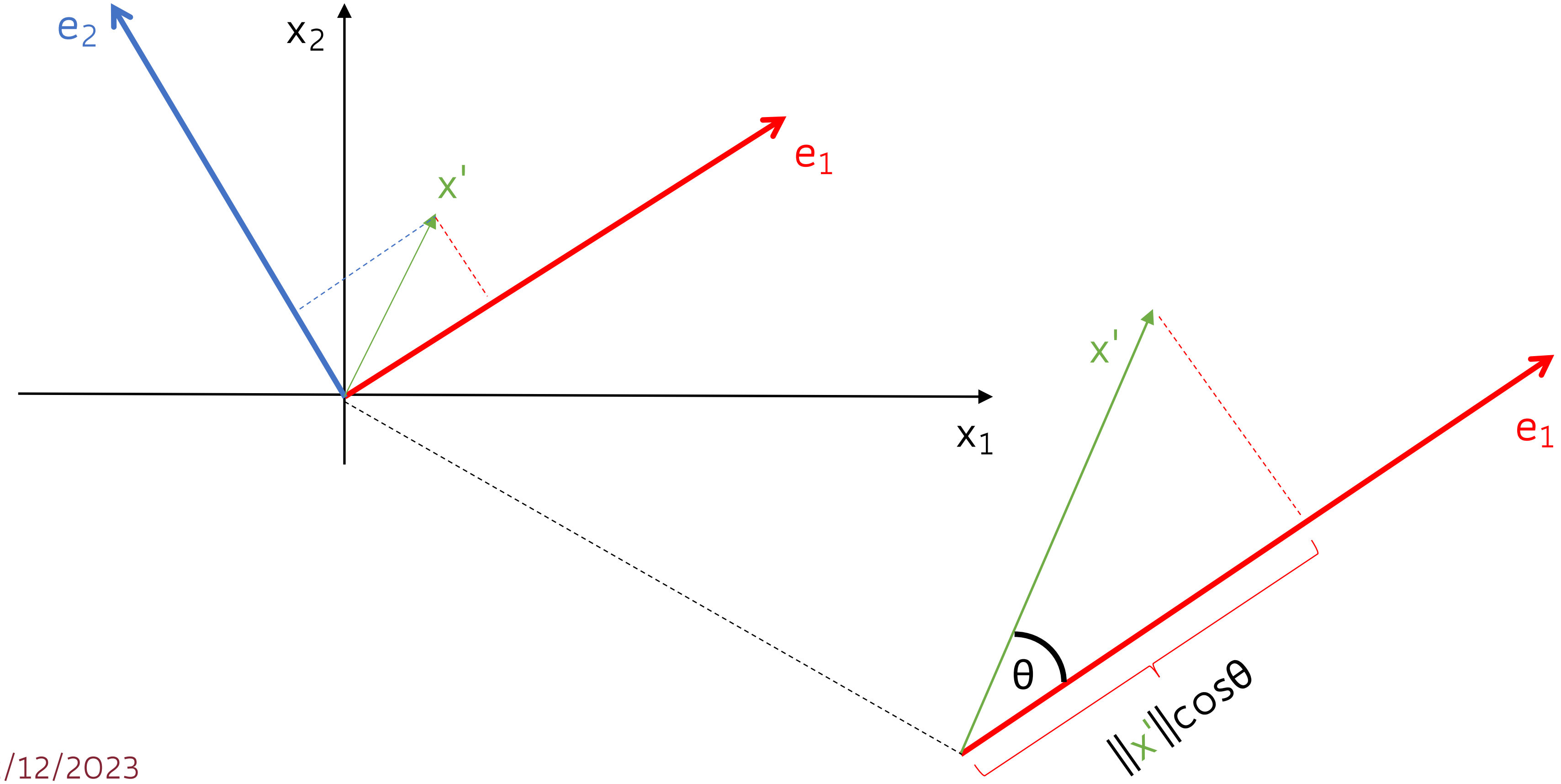
Projecting to New Dimensions: 2-d Case

Why the dot product?



Projecting to New Dimensions: 2-d Case

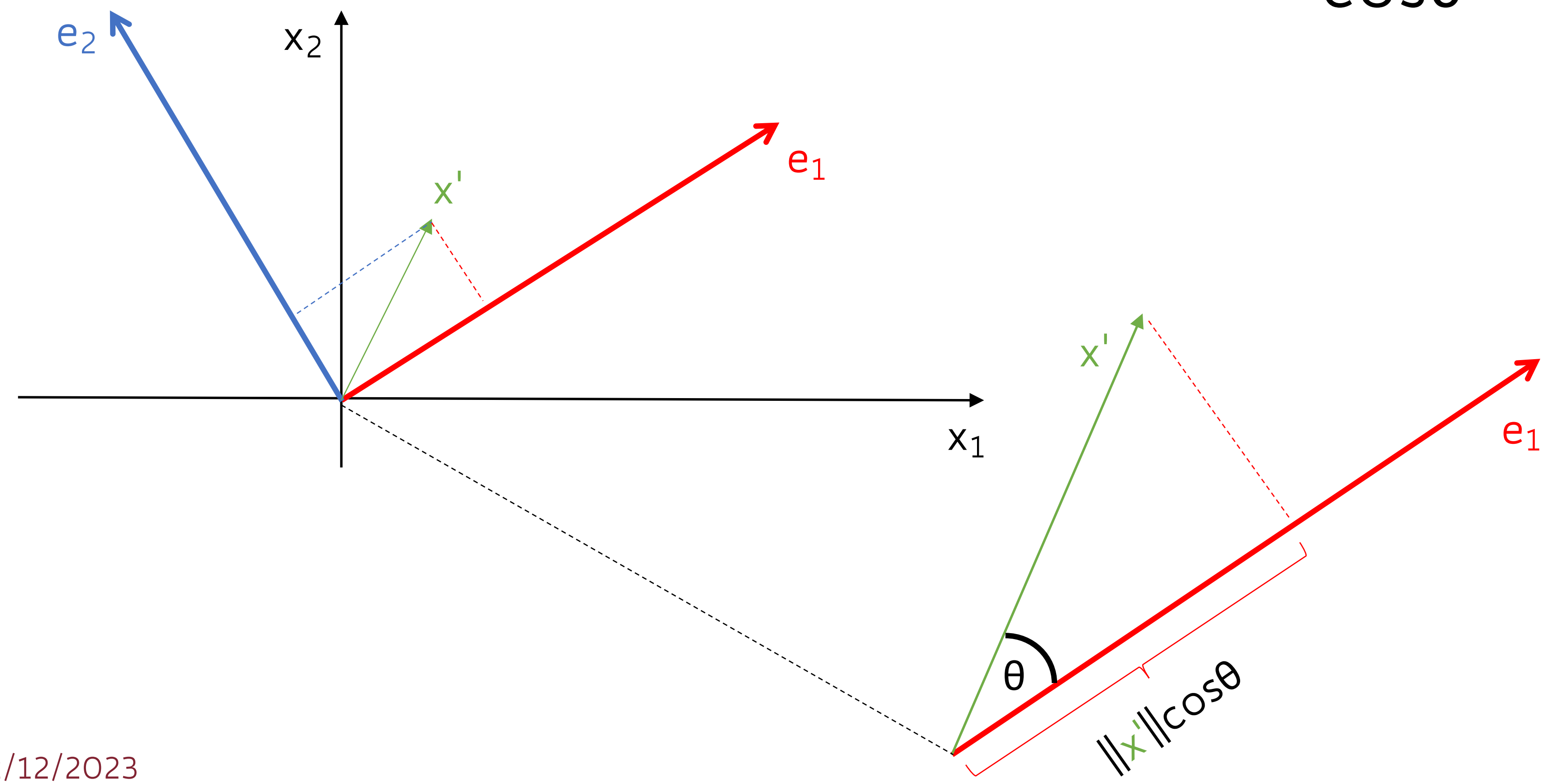
Why the dot product?



Projecting to New Dimensions: 2-d Case

Why the dot product?

$$\cos\theta = (x'e_1)/\|x'\|\|e_1\|$$

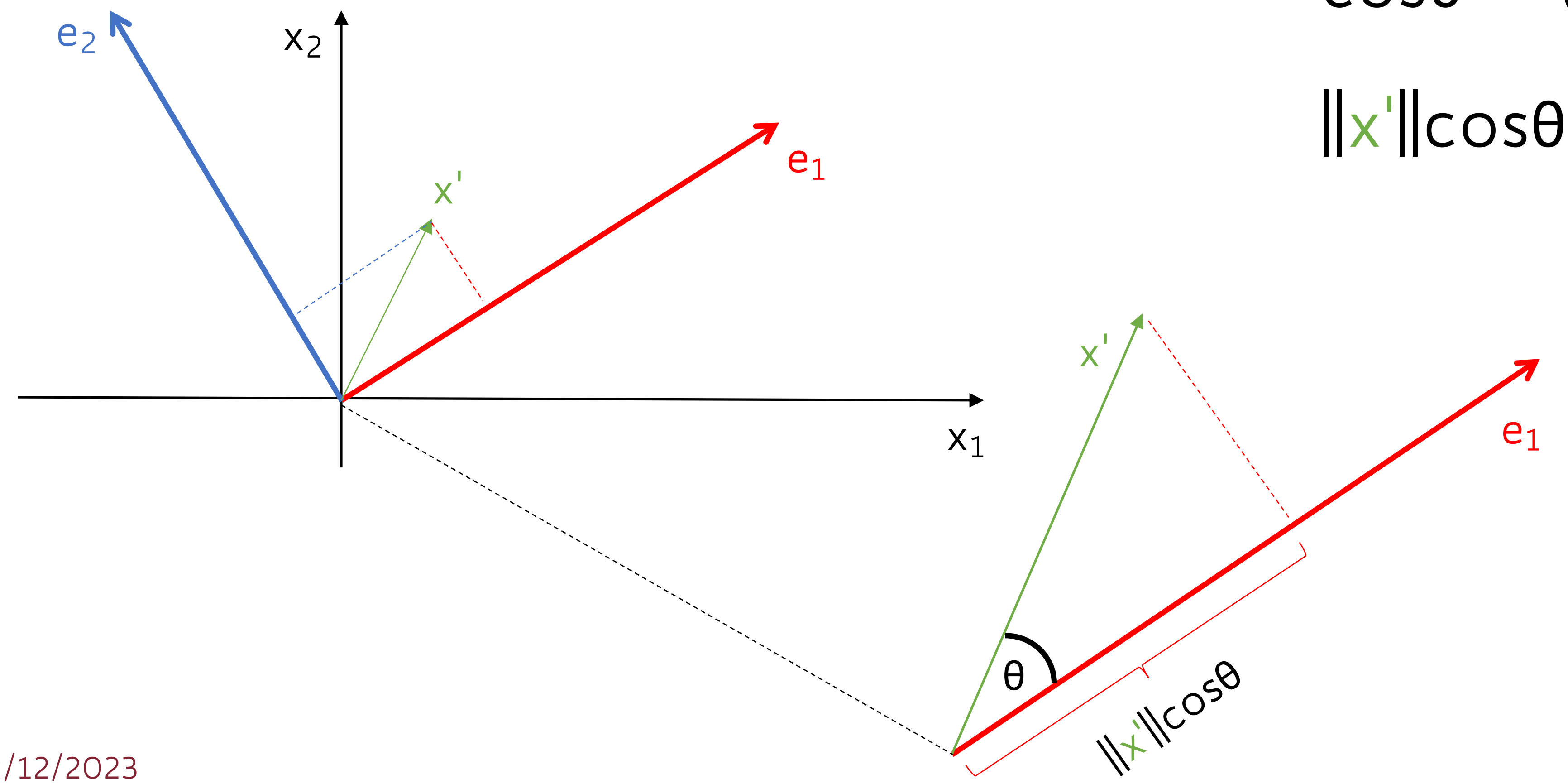


Projecting to New Dimensions: 2-d Case

Why the dot product?

$\cos\theta = (x'e_1)/\|x'\|\|e_1\|$

$\|x'\|\cos\theta = x'e_1/\|e_1\|$



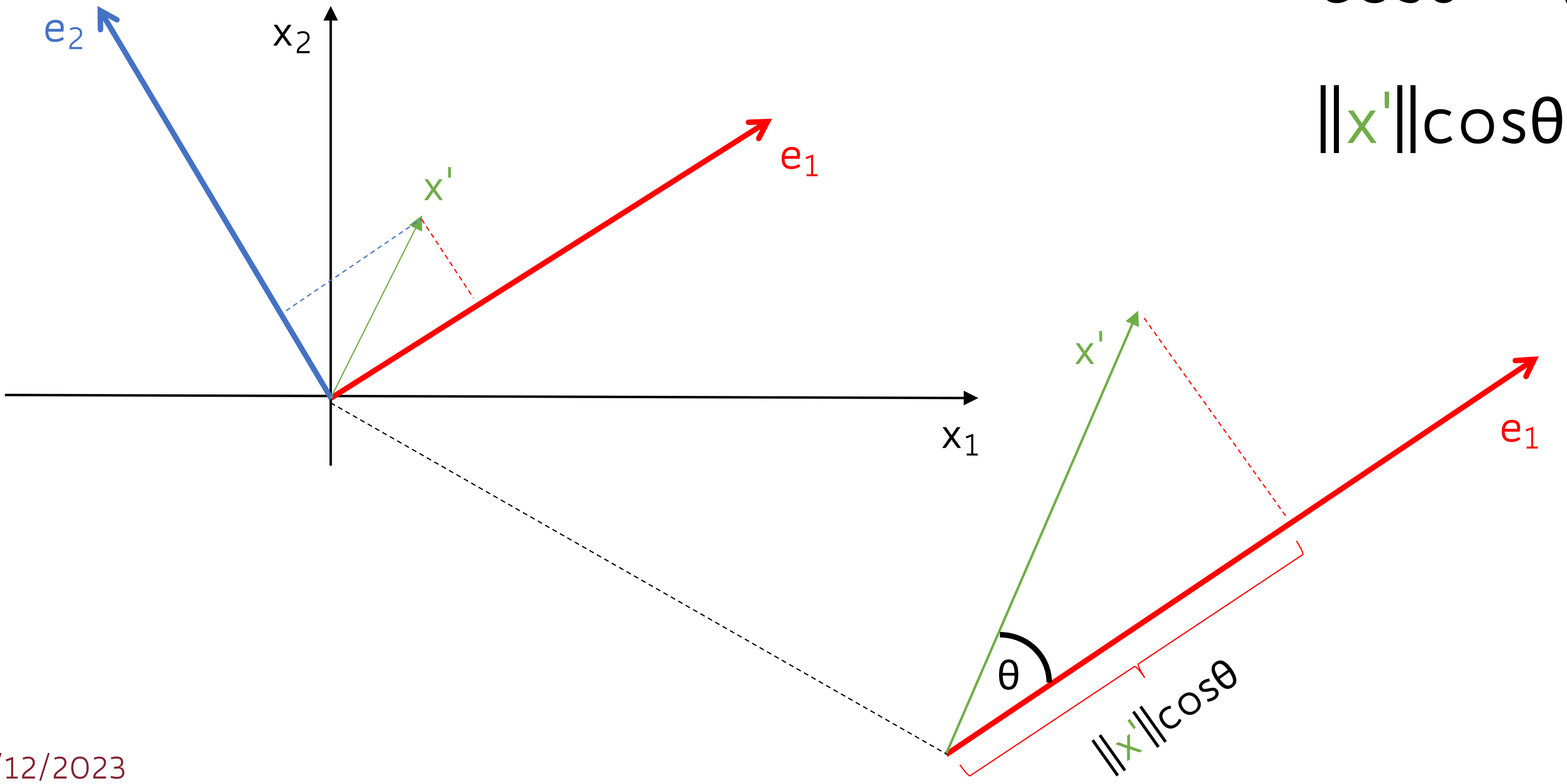
Projecting to New Dimensions: 2-d Case

Why the dot product?

$$\cos\theta = (x'e_1)/\|x'\|\|e_1\|$$

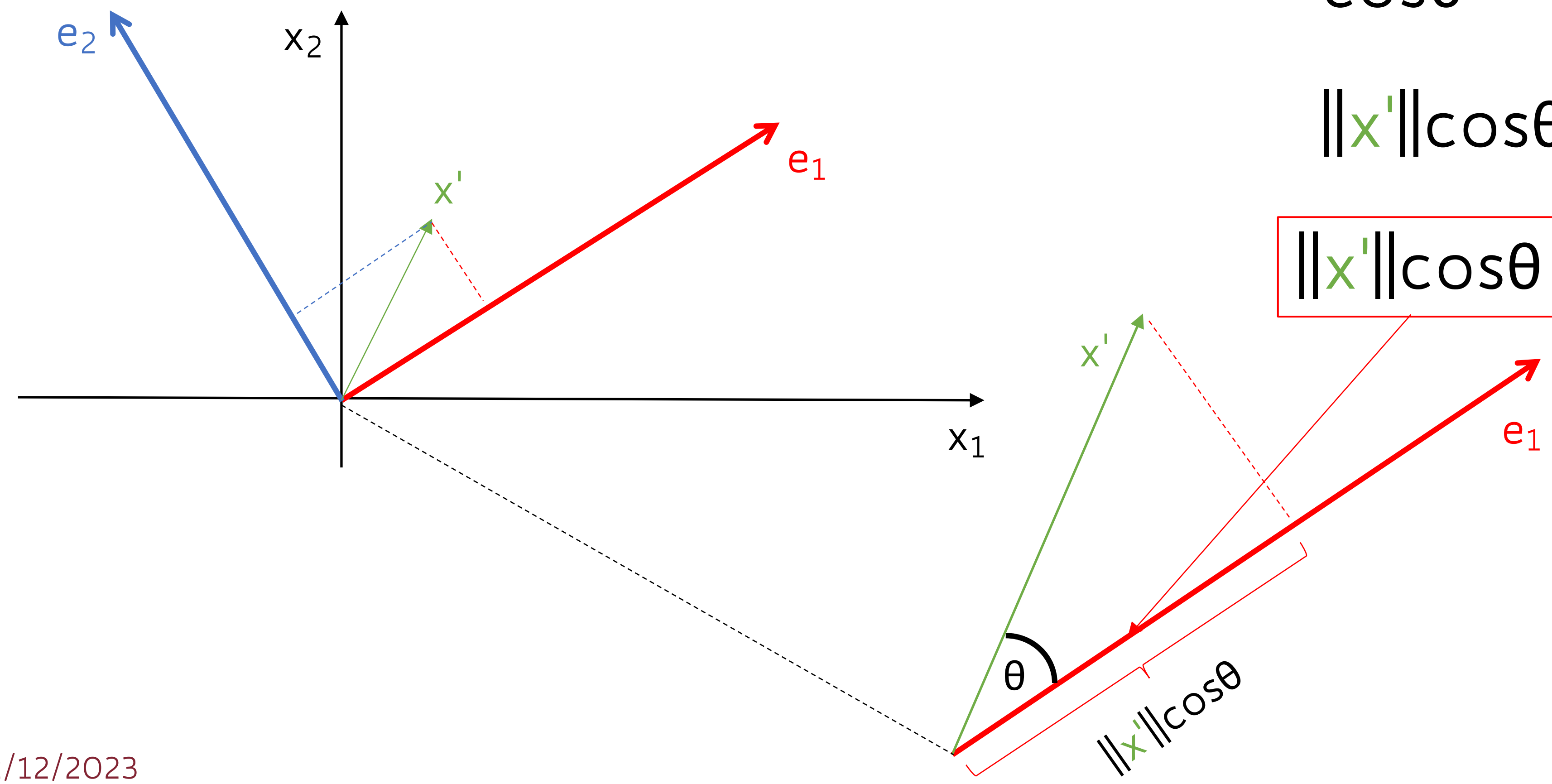
$$\|x'\|\cos\theta = x'e_1/\|e_1\|$$

$\|e_1\| = 1$



Projecting to New Dimensions: 2-d Case

Why the dot product?



$\cos\theta = (x'e_1)/\|x'\|\|e_1\|$

$\|x'\|\cos\theta = x'e_1/\|e_1\|$

$\boxed{\|x'\|\cos\theta = x'e_1}$

$\boxed{\|e_1\| = 1}$

Projecting to New Dimensions: 2-d Case

The new coordinates of the original data point x according to the eigenvectors e_1 and e_2 are as follows:

$$\mathbf{x}' = \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix} = \begin{bmatrix} \mathbf{x}'^T \mathbf{e}_1 \\ \mathbf{x}'^T \mathbf{e}_2 \end{bmatrix} = \begin{bmatrix} (x_1 - \mu_1)e_{1,1} + (x_2 - \mu_2)e_{1,2} \\ (x_1 - \mu_1)e_{2,1} + (x_2 - \mu_2)e_{2,2} \end{bmatrix}$$

Projecting to New Dimensions: *d*–dimensions

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Original d-dimensional data point

Projecting to New Dimensions: *d*–dimensions

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Original d-dimensional data point

$$\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k$$
$$k \ll d, \ \mathbf{e}_i \in \mathbb{R}^d$$

k << d principal components

Projecting to New Dimensions: *d*–dimensions

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Original d-dimensional data point

$$\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k$$
$$k \ll d, \ \mathbf{e}_i \in \mathbb{R}^d$$

k << d principal components

1. Mean centering

$$\mathbf{x}' = \mathbf{x} - \boldsymbol{\mu} = \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \\ \vdots \\ x_d - \mu_d \end{bmatrix}$$

Projecting to New Dimensions: *d*–dimensions

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Original d-dimensional data point

$$\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k$$
$$k \ll d, \;\; \mathbf{e}_i \in \mathbb{R}^d$$

k << d principal components

2. Projection to principal components

$$\mathbf{x}' = \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_k \end{bmatrix} = \begin{bmatrix} (\mathbf{x}-\boldsymbol{\mu})^T \mathbf{e}_1 \\ (\mathbf{x}-\boldsymbol{\mu})^T \mathbf{e}_2 \\ \vdots \\ (\mathbf{x}-\boldsymbol{\mu})^T \mathbf{e}_k \end{bmatrix} = \begin{bmatrix} (x_1-\mu_1)e_{1,1} + (x_2-\mu_2)e_{1,2} + \ldots + (x_d-\mu_d)e_{1,d} \\ (x_1-\mu_1)e_{2,1} + (x_2-\mu_2)e_{2,2} + \ldots + (x_d-\mu_d)e_{2,d} \\ \vdots \\ (x_1-\mu_1)e_{k,1} + (x_2-\mu_2)e_{k,2} + \ldots + (x_d-\mu_d)e_{k,d} \end{bmatrix}$$

Why Eigenvector with the Largest Eigenvalue

- We have only "visually proved" that eigenvector **e** turns towards the direction of the greatest variance

Why Eigenvector with the Largest Eigenvalue

- We have only "visually proved" that eigenvector **e** turns towards the direction of the greatest variance
- To actually prove it, we need to show that **e** maximizes the variance among all possible projections

Why Eigenvector with the Largest Eigenvalue

- We have only "visually proved" that eigenvector **e** turns towards the direction of the greatest variance
- To actually prove it, we need to show that **e** maximizes the variance among all possible projections
- Moreover, we pick the eigenvector with the **largest eigenvalue** λ because λ is exactly the variance of the data along that eigenvector

Why Eigenvector with the Largest Eigenvalue

- We have only "visually proved" that eigenvector **e** turns towards the direction of the greatest variance
- To actually prove it, we need to show that **e** maximizes the variance among all possible projections
- Moreover, we pick the eigenvector with the **largest eigenvalue** λ because λ is exactly the variance of the data along that eigenvector

More details available here:
https://github.com/gtolomei/big-data-computing/raw/master/extra/Notes_on_Principal_Component_Analysis.pdf

How Many Dimensions?

- In a d-dimensional space we may have $\mathbf{e}_1, \ldots, \mathbf{e}_d$ length-1 eigenvectors

How Many Dimensions?

- In a d-dimensional space we may have $e_1, \ldots, e_d$ length-1 eigenvectors
- We want to select k dimensions (k << d)

How Many Dimensions?

- In a d-dimensional space we may have $\mathbf{e}_1, \ldots, \mathbf{e}_d$ length-1 eigenvectors
- We want to select k dimensions (k << d)
- Eigenvalue λ_i is the variance along $\mathbf{e}_i$ and the overall variance of the data is the sum of all the eigenvalues

How Many Dimensions?

- In a d-dimensional space we may have $\mathbf{e}_1, \ldots, \mathbf{e}_d$ length-1 eigenvectors
- We want to select k dimensions (k << d)
- Eigenvalue λ_i is the variance along $\mathbf{e}_i$ and the overall variance of the data is the sum of all the eigenvalues

⇩

Pick the subset of k eigenvectors that "explain" the most variance

How Many Dimensions?

1. Sort eigenvectors by eigenvalues such that $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_d$

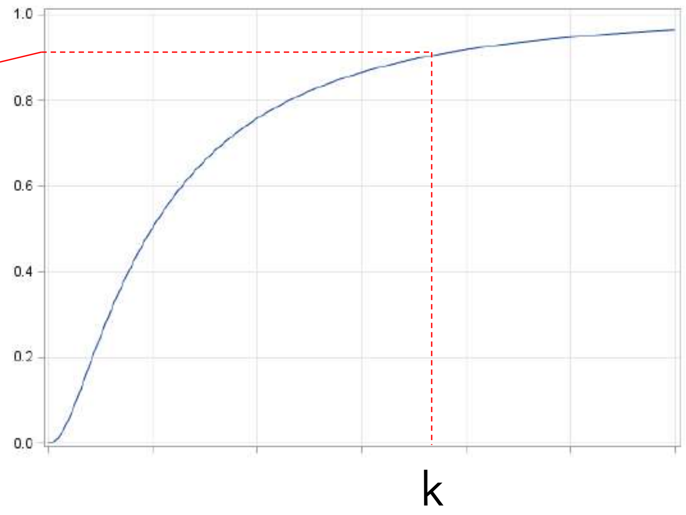
How Many Dimensions?

1. Sort eigenvectors by eigenvalues such that $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_d$

2. Pick the first k eigenvectors that explain x% of the total variance

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

e.g., x = 90÷95%



Practical Issues of PCA

- Covariance and variance are highly sensitive to large values

Practical Issues of PCA

- Covariance and variance are highly sensitive to large values
- If one dimension takes on extremely large values w.r.t. other dimensions the former will be considered as the principal component

Practical Issues of PCA

- Covariance and variance are highly sensitive to large values
- If one dimension takes on extremely large values w.r.t. other dimensions the former will be considered as the principal component

Solution

Normalize each dimension to 0-mean and 1-std-deviation

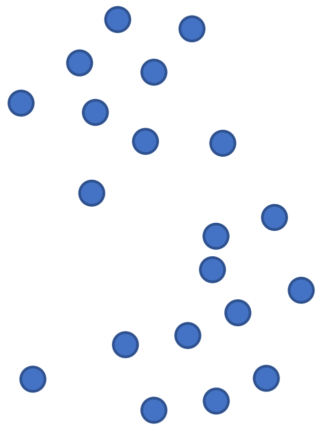
$$z = \frac{x - \mu}{\sigma}$$

Practical Issues of PCA

- PCA assumes the projection subspace is linear, i.e., an hyperplane:
  - 1-d → straight line, 2-d → flat surface, …

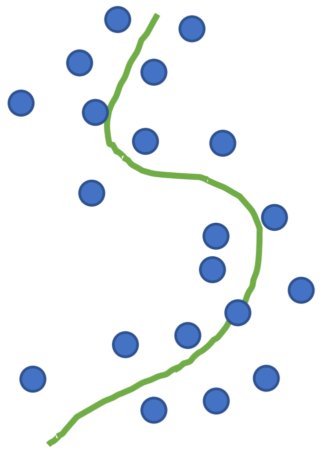
Practical Issues of PCA

- PCA assumes the projection subspace is linear, i.e., an hyperplane:
  - 1-d → straight line, 2-d → flat surface, …
- If data live in a low-dimensional but not linear space (i.e., manifold), PCA may not work nicely



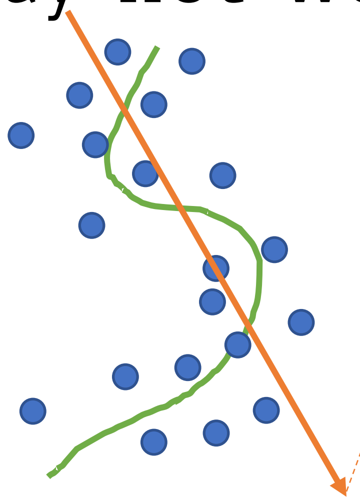
Practical Issues of PCA

- PCA assumes the projection subspace is linear, i.e., an hyperplane:
  - 1-d → straight line, 2-d → flat surface, …
- If data live in a low-dimensional but not linear space (i.e., manifold), PCA may not work nicely



Practical Issues of PCA

- PCA assumes the projection subspace is linear, i.e., an hyperplane:
  - 1-d → straight line, 2-d → flat surface, …
- If data live in a low-dimensional but not linear space (i.e., manifold), PCA may not work nicely



PCA will find a straight line and will not mimic non-linearity

Take-Home Message of Today

- PCA is a dimensionality reduction technique to represents high-dimensional data into low-dimensional linear subspace

Take-Home Message of Today

- PCA is a dimensionality reduction technique to represents high-dimensional data into low-dimensional linear subspace
  - The original space is rotated to make dimensions uncorrelated (i.e., independent)

Take-Home Message of Today

- PCA is a dimensionality reduction technique to represents high-dimensional data into low-dimensional linear subspace
  - The original space is rotated to make dimensions uncorrelated (i.e., independent)
- Reduced size of data means faster processing and smaller storage

Take-Home Message of Today

- PCA is a dimensionality reduction technique to represents high-dimensional data into low-dimensional linear subspace
  - The original space is rotated to make dimensions uncorrelated (i.e., independent)
- Reduced size of data means faster processing and smaller storage
- PCA can be very expensive for many very large-scale applications

Take-Home Message of Today

- PCA is a dimensionality reduction technique to represents high-dimensional data into low-dimensional linear subspace
    - The original space is rotated to make dimensions uncorrelated (i.e., independent)
- Reduced size of data means faster processing and smaller storage
- PCA can be very expensive for many very large-scale applications
- If data do not live on a linear subspace PCA may not work well