Introduction to Machine Learning
CentraleSupélec Paris — Fall 2017

3. Dimensionality Reduction

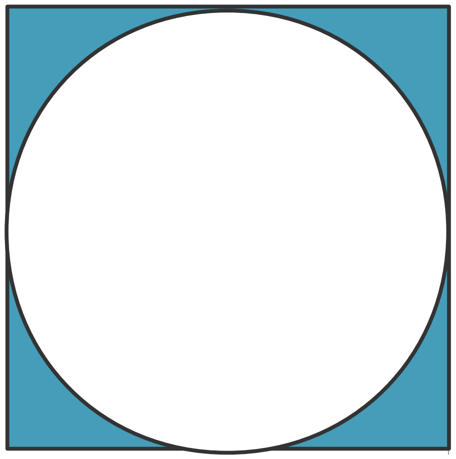
Chloé-Agathe Azencott
Centre for Computational Biology, Mines ParisTech
chloe-agathe.azencott@mines-paristech.fr

Learning objectives

- Give reasons **why** one would wish to **reduce the dimensionality** of a data set.
- Explain the difference between **feature selection** and **feature extraction.**
- Implement some **filter strategies.**
- Implement some **wrapper strategies.**
- Derive the **computation of principal components** from a "max variance" definition
- Implement **PCA.**

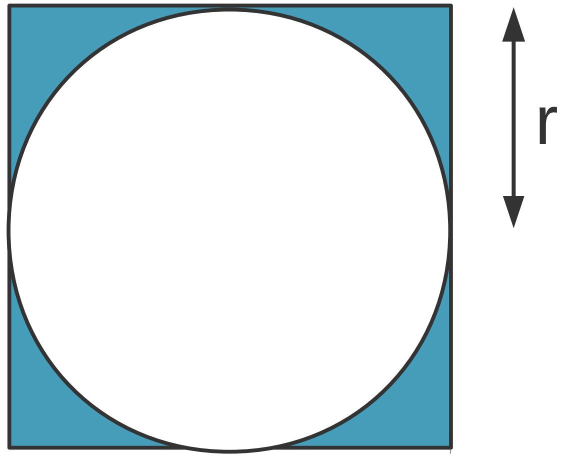
Curse of dimensionality

- Methods / intuitions that work in low dimension may not apply to high dimensions.
- p=2: Fraction of the points within a square that fall outside of the circle inscribed in it:



Curse of dimensionality

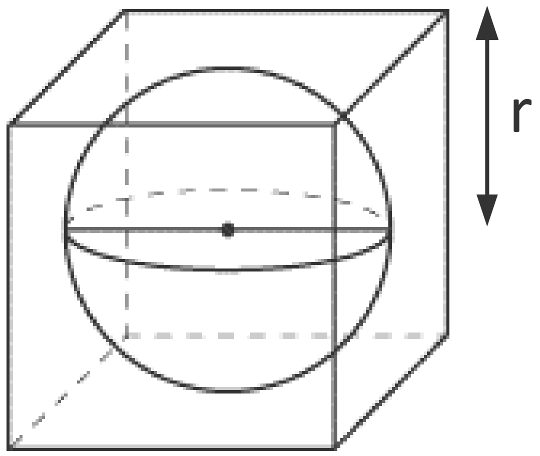
- Methods / intuitions that work in low dimension may not apply to high dimensions.
- p=2: Fraction of the points within a square that fall outside of the circle inscribed in it:



$$1 - \frac{\pi r^2}{4r^2} = 1 - \frac{\pi}{4}$$

Curse of dimensionality

- Methods / intuitions that work in low dimension may not apply to high dimensions.
- p=3: Fraction of the points within a cube that fall outside of the sphere inscribed in it:



$$1 - \frac{4/3\pi r^3}{8r^3} = 1 - \frac{\pi}{6}$$

Curse of dimensionality

- **Volume of a p-sphere:** $\frac{2r^p \pi^{p/2}}{p\Gamma(p/2)}$

The Gamma function Γ generalizes the factorial.
 Γ(n) = (n-1)!

- When p ↗ the proportion of a hypercube outside of its inscribed hypersphere approaches 1.

- What this means:
  - hyperspace is very big
  - all points are far apart

  ⇒ **dimensionality reduction.**

More reasons to reduce dimensionality

- **Computational complexity** (time and space)
- **Interpretability**
- Simpler models are **more robust** (less variance)
- Data **visualization**
- Cost of **data acquisition**
- Eliminate non-relevant attributes that can make it harder for an algorithm to learn.

Approaches to dimensionality reduction

- **Feature selection**

  Choose m < p features, ignore the remaining (p-m)

  - **Filtering** approaches

    Apply a statistical measure to assign a score to each feature (correlation, χ^2-test).

Approaches to dimensionality reduction

- **Feature selection**

  Choose $m < p$ features, ignore the remaining $(p-m)$

  - **Filtering** approaches

    Apply a statistical measure to assign a score to each feature (correlation, χ^2-test).

  - **Wrapper** approaches

    Search problem: Find the best set of features for a given predictive model.

Approaches to dimensionality reduction

- **Feature selection**

  Choose $m < p$ features, ignore the remaining $(p-m)$

  - **Filtering** approaches

    Apply a statistical measure to assign a score to each feature (correlation, χ^2-test).

  - **Wrapper** approaches

    Search problem: Find the best set of features for a given predictive model.

  - **Embedded** approaches

    Simultaneously fit a model and learn which features should be included.

Approaches to dimensionality reduction

- **Feature selection**

  Choose $m < p$ features, ignore the remaining $(p-m)$

  - **Filtering** approaches

    Apply a statistical measure to assign a score to each feature (correlation, χ^2-test).

  - **Wrapper** approaches

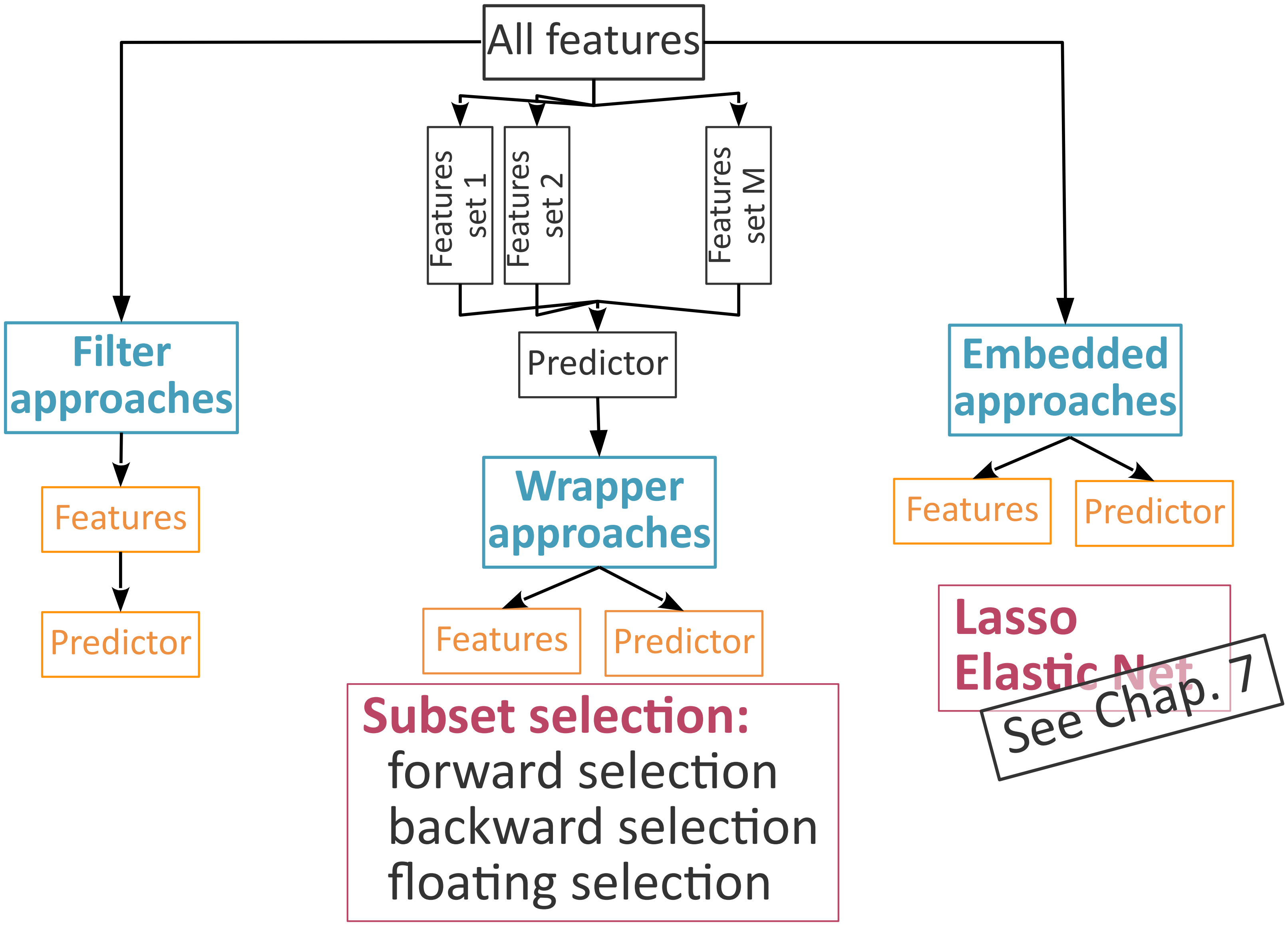
    Search problem: Find the best set of features for a given predictive model.

  - **Embedded** approaches

    Simultaneously fit a model and learn which features should be included.

Are those approaches supervised or unsupervised?

Feature selection: Overview



Subset selection:
forward selection
backward selection
floating selection

Feature selection: Subset selection

Approaches to dimensionality reduction

- **Feature selection**

  Choose m < p features, ignore the remaining (p-m)

  - **Filtering** approaches

    Apply a statistical measure to assign a score to each feature (correlation, χ^2-test).

  - **Wrapper** approaches

    Search problem: Find the best set of features for a given predictive model.

  - **Embedded** approaches

    Simultaneously fit a model and learn which features should be included.

All these feature selection approaches are **supervised.**

Subset selection

- **Goal:** Find the subset of features that leads to the best-performing algorithm.
- **How many subsets of p features are there?**

Subset selection

- **Goal:** Find the subset of features that leads to the best-performing algorithm.
- **Issue:** 2^p such sets.

Subset selection

- **Goal:** Find the subset of features that leads to the best-performing algorithm.
- **Issue:** 2^p such sets.
- Greedy approach: **forward search**

  Add the "best" feature at each step
  - Initially: $\mathcal{F} = \emptyset$
  - New best feature: $j* = \arg \min_{j \in \{1,\dots,p\}} E(\mathcal{F} \cup \{j\})$
  - stop if $E(\mathcal{F}) < E(\mathcal{F} \cup \{j\})$
  - else: $\mathcal{F} \leftarrow \mathcal{F} \cup \{j\}$

$E(\mathcal{F})$: Error of a predictor trained only using the features in $\mathcal{F}$

Subset selection

- **Goal:** Find the subset of features that leads to the best-performing algorithm.
- **Issue:** 2^p such sets.
- Greedy approach: **forward search**

  Add the "best" feature at each step

  - Initially: $\mathcal{F} = \emptyset$
  - New best feature: $j* = \arg \min_{j \in \{1,\dots,p\}} E(\mathcal{F} \cup \{j\})$
  - stop if $E(\mathcal{F}) < E(\mathcal{F} \cup \{j\})$
  - else: $\mathcal{F} \leftarrow \mathcal{F} \cup \{j\}$

$E(\mathcal{F})$: Error of a predictor trained only using the features in $\mathcal{F}$

?

What is the complexity of this algorithm?

Subset selection

- **Goal:** Find the subset of features that leads to the best-performing algorithm.
- **Issue:** 2^p such sets.
- Greedy approach: **forward search**

  Add the "best" feature at each step

  - Initially: $\mathcal{F} = \emptyset$
  - New best feature: $j* = \arg \min_{j \in \{1,\dots,p\}} E(\mathcal{F} \cup \{j\})$
  - stop if $E(\mathcal{F}) < E(\mathcal{F} \cup \{j\})$
  - else: $\mathcal{F} \leftarrow \mathcal{F} \cup \{j\}$

  Complexity: O(p^2 x C) where C=complexity of training and evaluating the model (might depend on p also).

  Much better than O(2^p)!

$E(\mathcal{F})$: Error of a predictor trained only using the features in $\mathcal{F}$

Subset selection

- Greedy approach: **forward search**

  Add the "best" feature at each step
  - Initially: $\mathcal{F} = \emptyset$
  - New best feature: $j* = \arg\min_{j \in \{1,\dots,p\}} E(\mathcal{F} \cup \{j\})$
  - stop if $E(\mathcal{F}) < E(\mathcal{F} \cup \{j\})$
  - else: $\mathcal{F} \leftarrow \mathcal{F} \cup \{j\}$

  Complexity: $O(p^2)$

$E(\mathcal{F})$: Error of a predictor trained only using the features in $\mathcal{F}$

- **Alternative strategies:**
  - **Backward search:** start from {1, …, p}, eliminate features.
  - **Floating search:** alternatively add q features and remove r features.

Approaches to dimensionality reduction

- **Feature extraction**

  Project the p features on $m < p$ new dimensions

  - Principal Components Analysis (PCA)
  - Factor Analysis (FA)
  - Non-negative Matrix Factorization (NMF)
  - Linear Discriminant Analysis (LDA)
  - Multidimensional scaling (MDS)
  - Isometric feature mapping (Isomap)
  - Locally Linear Embedding (LLE)
  - Autoencoders

Linear

Supervised

Non linear

Most of these approaches are **unsupervised.**

Feature extraction: Principal Component Analysis

Principal Components Analysis (PCA)

- **Goal:** Find a low-dimensional space such that **information loss** is minimized when the data is projected on that space.

Principal Components Analysis (PCA)

- **Goal:** Find a low-dimensional space such that **information loss** is minimized when the data is projected on that space.
- **Unsupervised:** We're only looking at the data, not at any labels.

Principal Components Analysis (PCA)

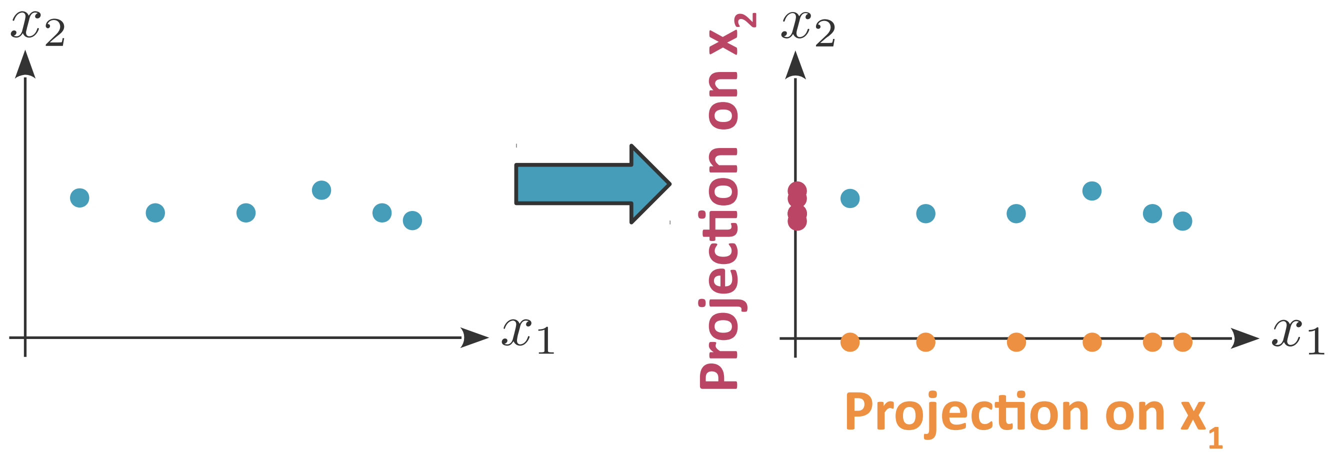
- **Goal:** Find a low-dimensional space such that **information loss** is minimized when the data is projected on that space.
- **Unsupervised:** We're only looking at the data, not at any labels.

In PCA, we want the **variance** to be **maximized.**

Principal Components Analysis (PCA)

- **Goal:** Find a low-dimensional space such that **information loss** is minimized when the data is projected on that space.
- **Unsupervised:** We're only looking at the data, not at any labels.

In PCA, we want the **variance** to be **maximized.**

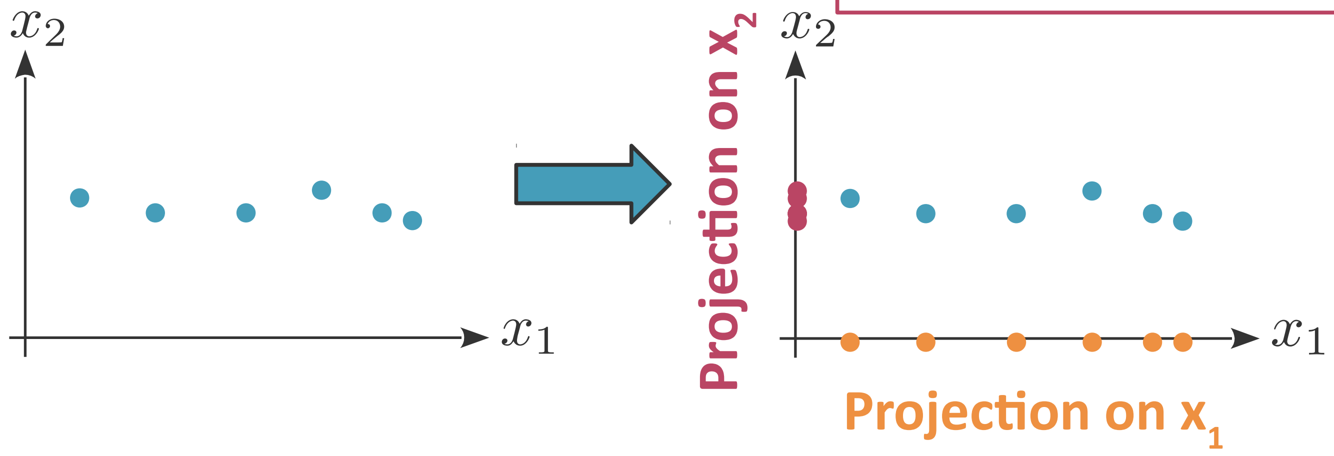


Principal Components Analysis (PCA)

- **Goal:** Find a low-dimensional space such that **information loss** is minimized when the data is projected on that space.
- **Unsupervised:** We're only looking at the data, not at any labels.

In PCA, we want the **variance** to be **maximized.**

Warning! This requires standardizing the features.



Feature standardization

- **Variance** of feature j in data set $\mathcal{D}$: ?

$$\mathcal{D} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^n\} \qquad \boldsymbol{x} \in \mathbb{R}^p$$

Feature standardization

- **Variance** of feature j in data set $\mathcal{D}$:

$$\sigma_j^2 = \frac{1}{n}\sum_{i=1}^{n}(x_j^i - \mu_j)^2 \qquad \mu_j = \frac{1}{n}\sum_{i=1}^{n} x_j^i$$

Feature standardization

- Variance of feature j in data set $\mathcal{D}$:

$$\sigma_j^2 = \frac{1}{n}\sum_{i=1}^{n}(x_j^i - \mu_j)^2 \qquad \mu_j = \frac{1}{n}\sum_{i=1}^{n} x_j^i$$

- Features that take large values will have large variance

Compare [10, 20, 30, 40, 50] with [0.1, 0.2, 0.3, 0.4, 0.5].

Feature standardization

- **Variance** of feature j in data set $\mathcal{D}$:

$$\sigma_j^2 = \frac{1}{n}\sum_{i=1}^{n}(x_j^i - \mu_j)^2 \qquad \mu_j = \frac{1}{n}\sum_{i=1}^{n} x_j^i$$

- Features that take large values will have large variance

  Compare [10, 20, 30, 40, 50] with [0.1, 0.2, 0.3, 0.4, 0.5].

- **Standardization:**
  - **mean centering:** give each feature a mean of 0
  - **variance scaling:** give each feature a variance of 1

$$x_j^i \leftarrow \frac{x_j^i - \mu_j}{\sigma_j}$$

Principal Components Analysis (PCA)

- **Goal:** Find a low-dimensional space such that **variance is maximized** when the data is projected on that space.
- **Assumption:** the data is **centered** i.e. it has mean 0.

  If not: subtract the mean: $X \leftarrow X - \mu$

- **What formula gives us the projection of x on the direction of w (assuming w to be a unit vector)?**

Principal Components Analysis (PCA)

- **Goal:** Find a low-dimensional space such that **variance is maximized** when the data is projected on that space.
- Projection of X on the direction of **w**: $z = Xw$
- **Compute Var(z) as a function of w and** X. ?

$$z = Xw$$

dimensions: (n, 1) (n, p) x (p, 1)
n samples
p features

unit vector: $||w|| = 1$

Principal Components Analysis (PCA)

- **Goal:** Find a low-dimensional space such that **variance is maximized** when the data is projected on that space.
- Projection of X on the direction of **w**: $z = Xw$
- Var(**z**) can be computed as:

$$
\begin{aligned}
\mathrm{Var}(z) &= \mathrm{Var}(Xw) = \mathrm{Var}(w^\top X^\top) \\
&= \mathbb{E}[((w^\top X^\top) - \mathbb{E}[w^\top X^\top])^2] \\
&= \mathbb{E}[(w^\top X^\top - w^\top \boxed{\mathbb{E}[X]})^2] \quad \leftarrow \text{0 (data centered)} \\
&= \mathbb{E}[w^\top X^\top X w] \\
&= w^\top \boxed{\mathbb{E}[X^\top X]} w \quad \leftarrow \text{Estimated from data: } \Sigma = X^\top X
\end{aligned}
$$

dimensions: (1, p) x (p, n) x (n, p) x (p, 1)

First principal component (PC1)

- **Goal:** find $\boldsymbol{w}_1 = \arg\max_{\boldsymbol{w} \in \mathbb{R}^p} \mathrm{Var}(X\boldsymbol{w})$
  subject to $||\boldsymbol{w}_1|| = 1.$
  What kind of optimization problem do we have?

First principal component (PC1)

- **Goal:** find $\boldsymbol{w}_1 = \arg\max_{\boldsymbol{w} \in \mathbb{R}^p} \mathrm{Var}(X\boldsymbol{w})$ subject to $||\boldsymbol{w}_1|| = 1.$
- Optimization problem: **Maximize f(w) s.t. g(w)=Cte.**

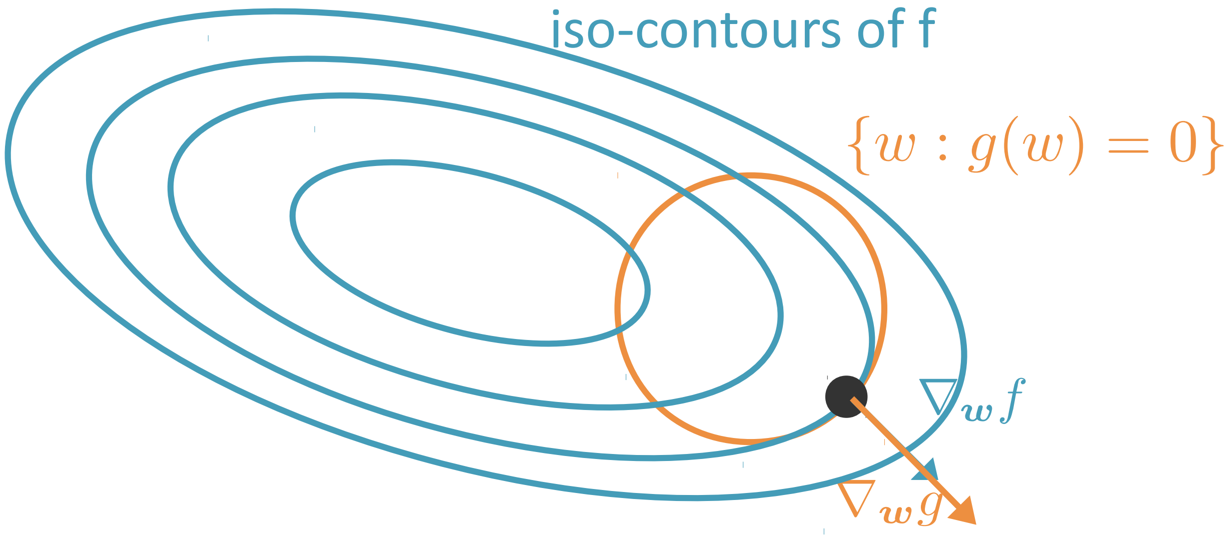
$$f(\boldsymbol{w}) = \boldsymbol{w}^\top \Sigma \boldsymbol{w}$$

$$g(\boldsymbol{w}) = ||\boldsymbol{w}|| - 1$$

First principal component (PC1)

- **Goal:** find $\boldsymbol{w}_1 = \arg\max_{\boldsymbol{w} \in \mathbb{R}^p} \mathrm{Var}(X\boldsymbol{w})$
  subject to $||\boldsymbol{w}_1|| = 1.$
- Optimization problem: **Maximize f(w) s.t. g(w)=0.**

$f(\boldsymbol{w}) = \boldsymbol{w}^\top \Sigma \boldsymbol{w}$ $\qquad$ $g(\boldsymbol{w}) = ||\boldsymbol{w}|| - 1$



$\nabla f(\boldsymbol{w}^*) = \alpha \nabla g(\boldsymbol{w}^*)$ $\qquad$ Lagrangian: $L(\alpha, \boldsymbol{w}) = f(\boldsymbol{w}) - \alpha g(\boldsymbol{w})$

First principal component (PC1)

- Maximize f(w) s.t. g(w)=0

$$L(\alpha, \boldsymbol{w}) = f(\boldsymbol{w}) - \alpha g(\boldsymbol{w})$$

- If $(\alpha^*, \boldsymbol{w}^*)$ maximize the Lagrangian, then
  - $\nabla_\alpha L(\alpha^*, \boldsymbol{w}^*) = 0$ (stationarity point) hence $g(\boldsymbol{w}^*) = 0$
  - For any admissible $\boldsymbol{w}$, $f(\boldsymbol{w}) - \alpha^* g(\boldsymbol{w}) \leq f(\boldsymbol{w}^*) - \alpha g(\boldsymbol{w}^*)$ (by definition of the maximum of L), hence $f(\boldsymbol{w}) \leq f(\boldsymbol{w}^*)$ (because of admissibility).
- That is to say, any maximizer of the Lagrangian gives us a maximizer of f under the constraint that g=0.

First principal component (PC1)

- **Goal:** find $\boldsymbol{w}_1 = \arg\max_{\boldsymbol{w}\in\mathbb{R}^p} \mathrm{Var}(X\boldsymbol{w})$
  subject to $||\boldsymbol{w}_1|| = 1.$
- Optimization problem: **Maximize f(w) s.t. g(w)=0.**

$\nabla f(\boldsymbol{w}^*) = \alpha \nabla g(\boldsymbol{w}^*)$ Lagrangian: $L(\alpha, \boldsymbol{w}) = f(\boldsymbol{w}) - \alpha g(\boldsymbol{w})$

$$L(\alpha, \boldsymbol{w}) = \underbrace{\boxed{\mathrm{Var}(X\boldsymbol{w})}}_{\boldsymbol{w}^\top \Sigma \boldsymbol{w}} - \alpha(\boldsymbol{w}^\top \boldsymbol{w} - 1)$$

Hence:

$$\boldsymbol{w}_1 = \arg\max_{\boldsymbol{w}\in\mathbb{R}^p} \left(\boldsymbol{w}^\top \Sigma \boldsymbol{w} - \alpha(\boldsymbol{w}^\top \boldsymbol{w} - 1)\right)$$

?

First principal component (PC1)

$$\boldsymbol{w}_1 = \arg\max_{\boldsymbol{w}\in\mathbb{R}^p} \left(\boldsymbol{w}^\top \Sigma \boldsymbol{w} - \alpha(\boldsymbol{w}^\top \boldsymbol{w} - 1)\right)$$

- The maximum is an extremum point, hence necessarily the gradient is zero.

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}_1) = 0$$

$$2\Sigma\boldsymbol{w}_1 - 2\alpha\boldsymbol{w}_1 = 0$$

- Hence $\Sigma\boldsymbol{w}_1 = \alpha\boldsymbol{w}_1$

  What does it tell us about α, w_1 w.r.t the matrix Σ ?

First principal component (PC1)

$$\boldsymbol{w}_1 = \arg\max_{\boldsymbol{w}\in\mathbb{R}^p} \left(\boldsymbol{w}^\top \Sigma \boldsymbol{w} - \alpha(\boldsymbol{w}^\top \boldsymbol{w} - 1)\right)$$

- The maximum is an extremum point, hence necessarily the gradient is zero.

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}_1) = 0$$

$$2\Sigma\boldsymbol{w}_1 - 2\alpha\boldsymbol{w}_1 = 0$$

- Hence $\Sigma\boldsymbol{w}_1 = \alpha\boldsymbol{w}_1$

  $(\alpha, \boldsymbol{w}_1)$ are an **(eigenvalue, eigenvector)** of Σ

First principal component (PC1)

$$\boldsymbol{w}_1 = \arg\max_{\boldsymbol{w}\in\mathbb{R}^p} \left(\boldsymbol{w}^\top \Sigma \boldsymbol{w} - \alpha(\boldsymbol{w}^\top \boldsymbol{w} - 1)\right)$$

- $(\alpha, \boldsymbol{w}_1)$ are an **(eigenvalue, eigenvector)** of Σ
- Now we want to find the maximum of all possible extremum points

$$\boldsymbol{w}_1 = \arg\max_{(\alpha, \boldsymbol{w})\ \text{(eigenvalue, eigenvector) of } \Sigma} \left(\boldsymbol{w}^\top \alpha \boldsymbol{w} - \alpha(\boldsymbol{w}^\top \boldsymbol{w} - 1)\right)$$
$$\boldsymbol{w}_1 = \arg\max_{(\alpha, \boldsymbol{w})\ \text{(eigenvalue, eigenvector) of } \Sigma} \alpha$$

- Hence w_1 is the eigenvector of Σ corresponding to its largest eigenvalue.

Second principal component (PC2)

- **Goal:** find $\mathbf{w}_2$
  - of **unit length**
  - **orthogonal** to $\mathbf{w}_1$
  - such that the projection of x on $\mathbf{w}_2$ has **maximal variance.**

Write out each of these requirements. ?

Second principal component (PC2)

- Goal: find $\mathbf{w_2}$
  - of unit length $||\boldsymbol{w}_2|| = 1$
  - orthogonal to $\mathbf{w_1}$ $\boldsymbol{w}_2^\top \boldsymbol{w}_1 = 0$
  - such that the projection of x on $\mathbf{w_2}$ has maximal variance. $\boldsymbol{w}_2 = \arg\max_{\boldsymbol{w} \in \mathbb{R}p} \mathrm{Var}(X\boldsymbol{w})$

- Hence the following optimization problem (Lagrangian):

$$\boldsymbol{w}_2 = \arg\max_{\boldsymbol{w} \in \mathbb{R}^p} \left(\boldsymbol{w}^\top \Sigma \boldsymbol{w} - \alpha(\boldsymbol{w}^\top \boldsymbol{w} - 1) - \beta(\boldsymbol{w}^\top \boldsymbol{w}_1) \right).$$

?

Second principal component (PC2)

$$\boldsymbol{w}_2 = \arg\max_{\boldsymbol{w}\in\mathbb{R}^p} \left(\boldsymbol{w}^\top \Sigma \boldsymbol{w} - \alpha(\boldsymbol{w}^\top \boldsymbol{w} - 1) - \beta(\boldsymbol{w}^\top \boldsymbol{w}_1)\right).$$

- Take the derivative, set it to 0

$$2\Sigma\boldsymbol{w}_2 - 2\alpha\boldsymbol{w}_2 - \beta\boldsymbol{w}_1 = 0$$

$$2\underbrace{\boldsymbol{w}_1^\top \Sigma \boldsymbol{w}_2}_{\text{scalar}} - 2\alpha\underbrace{\boldsymbol{w}_1^\top \boldsymbol{w}_2}_{0} - \boldsymbol{w}_1^\top \beta \boldsymbol{w}_1 = 0$$

$$\begin{aligned} &= \boldsymbol{w}_2^\top \Sigma \boldsymbol{w}_1 \\ &= \lambda_1 \boldsymbol{w}_2^\top \boldsymbol{w}_1 \\ &= 0 \end{aligned}$$

Hence $\beta = 0$

$$\Longrightarrow \boxed{2\Sigma\boldsymbol{w}_2 - 2\alpha\boldsymbol{w}_2 = 0}$$

Second principal component (PC2)

- The second principal component is given by the eigenvector of Σ with the **second largest eigenvalue.**
- And so on and so forth for all other PCs.

Singular value decomposition

- **SVD of X:** $X = UDV^\top$

  p x p
  orthogonal

  n x p

  n x n
  orthogonal

  n x p
  diagonal, non-neg

- Cov(X): $$\begin{aligned} X^\top X &= VD^\top U^\top UDV^\top \\ &= V(D^\top D)V^\top \end{aligned}$$

- **Eigendecomposition of Σ:**

  p x p
  diagonal, eigenvalues

  p x p $\Sigma = Q\Lambda Q^{-1}$

  p x p
  j-th col=j-th eigenvector

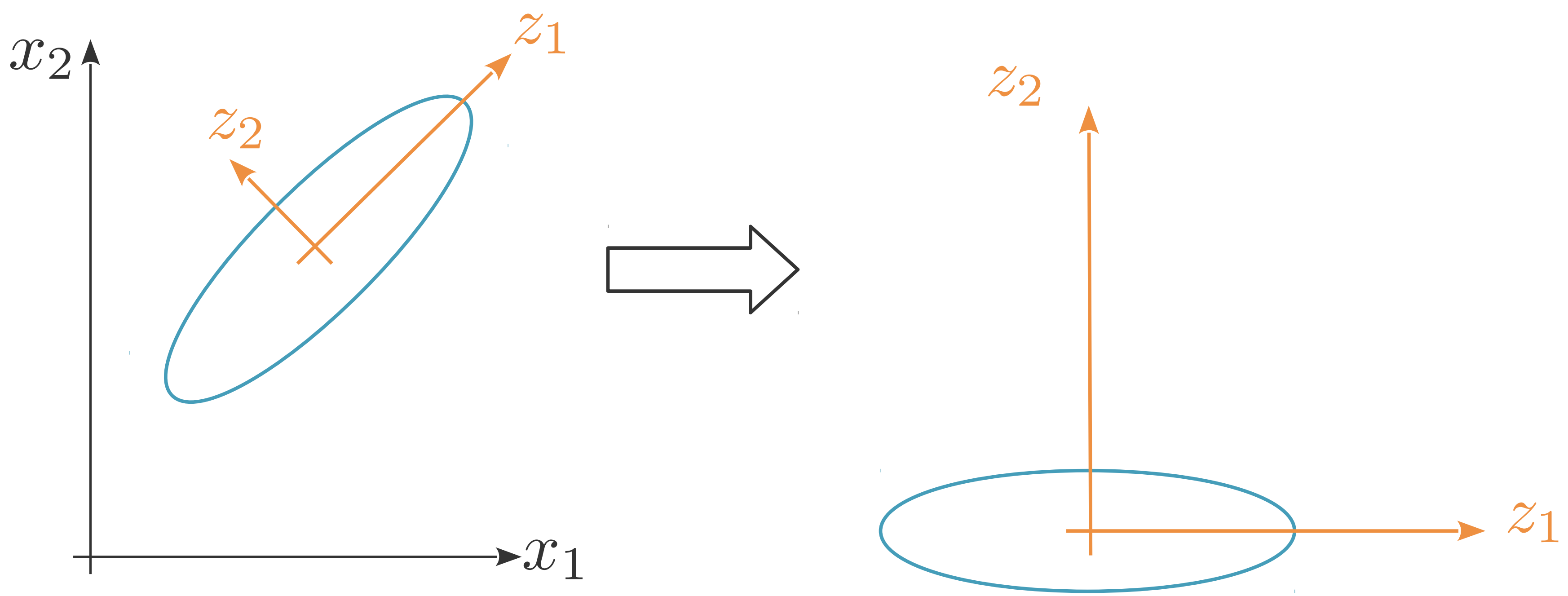
  – The singular values of X are the square root of the eigenvalues of Σ.

  – The left-singular vectors of X (the columns of U) are the eigenvectors of Σ.

What PCA does

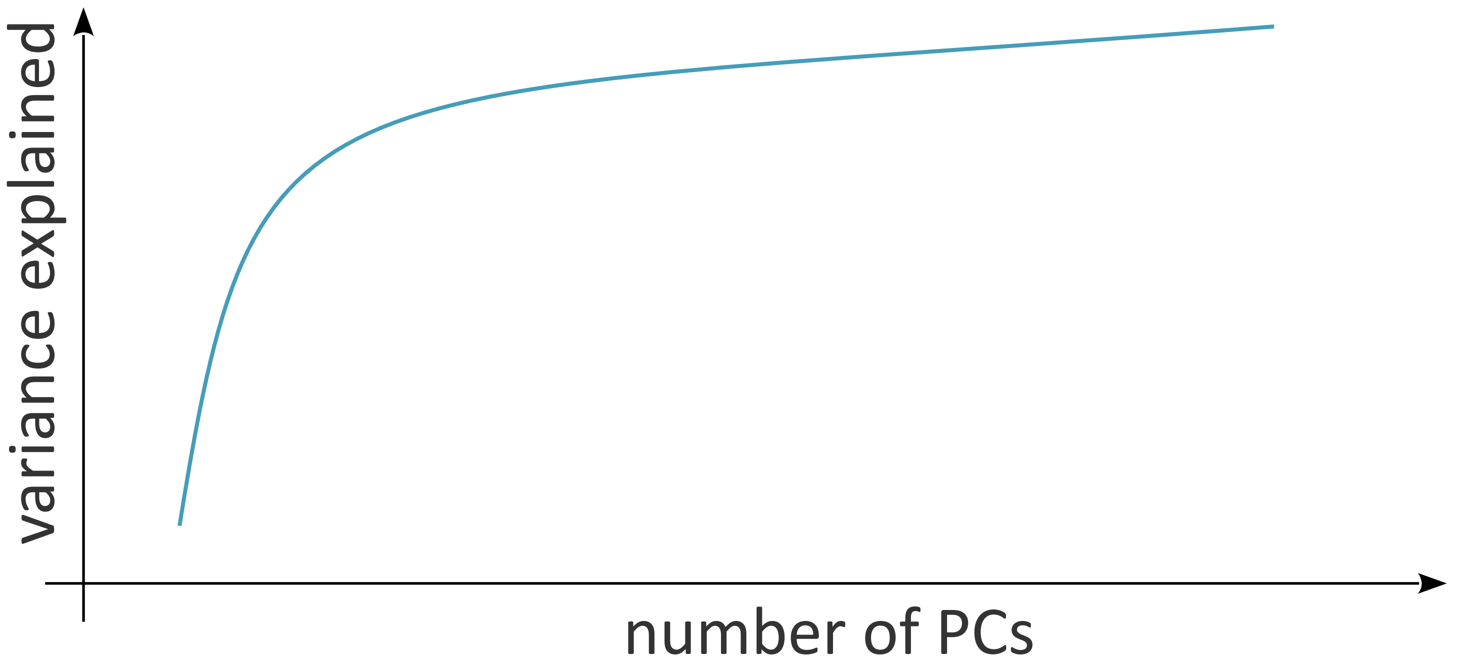
- W: p x m matrix of the **m leading eigenvectors** of Σ.
- **m first PCs:** $Z = (X - \boldsymbol{\mu})W$

dimensions: (n, m) (n, p) x (p, m) $\Sigma = (X - \boldsymbol{\mu})^\top (X - \boldsymbol{\mu})$



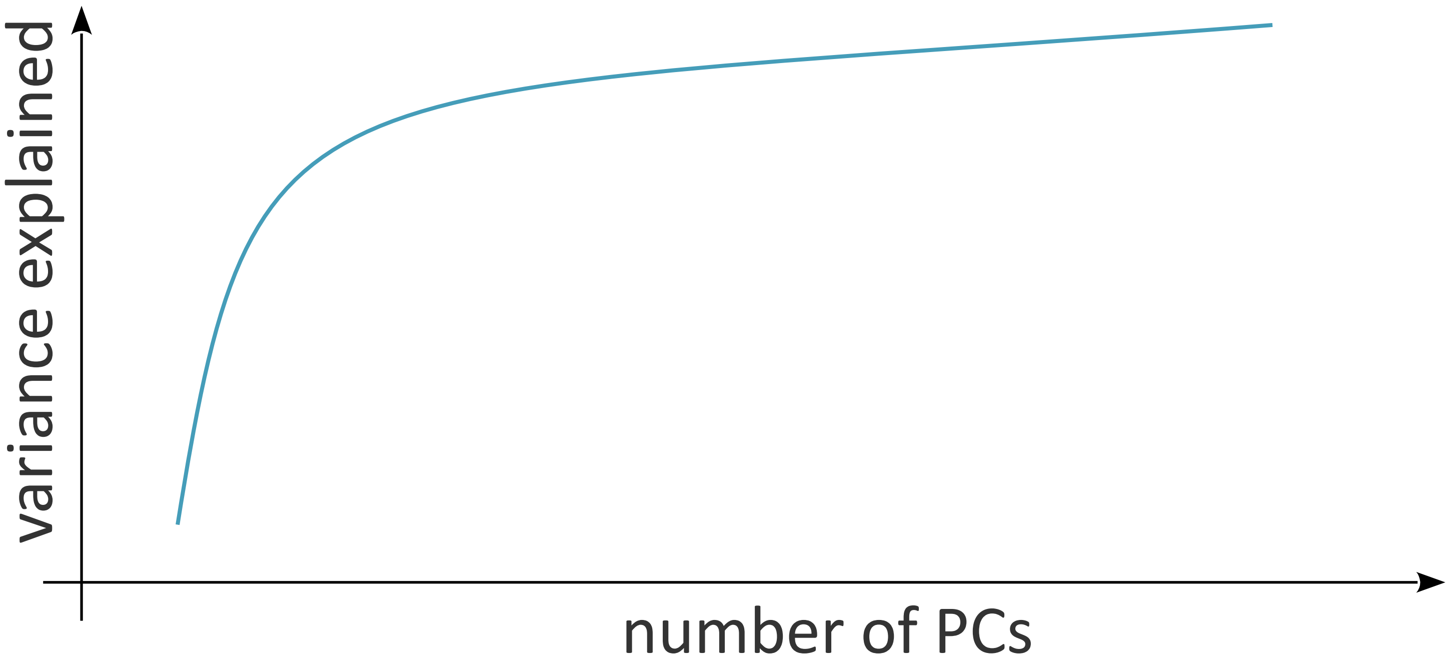
How to choose m

- **Percentage of variance explained:**
  - Total variance in the data = Tr(Σ) = $\lambda_1 + \lambda_2 + ... + \lambda_p$
  - The first m PCs account for $\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_m}{\lambda_1 + \lambda_2 + \cdots + \lambda_p}$ of the total variance.
- **Scree graph:**

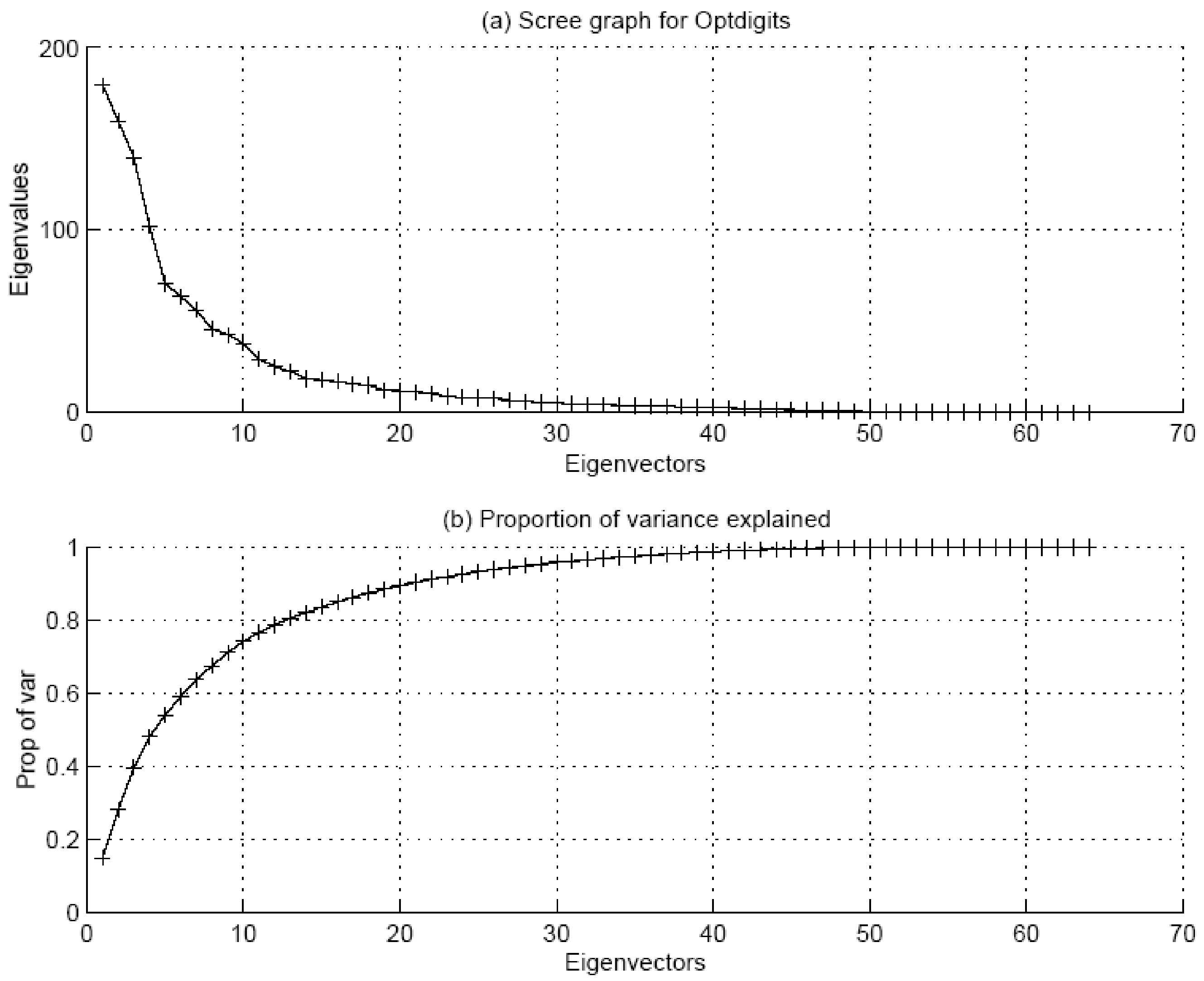


How to choose m

- Pick enough components to explain a fixed percentage of the variance.
- Find the "elbow" (where adding more components doesn't really help much).
- **Scree graph:**



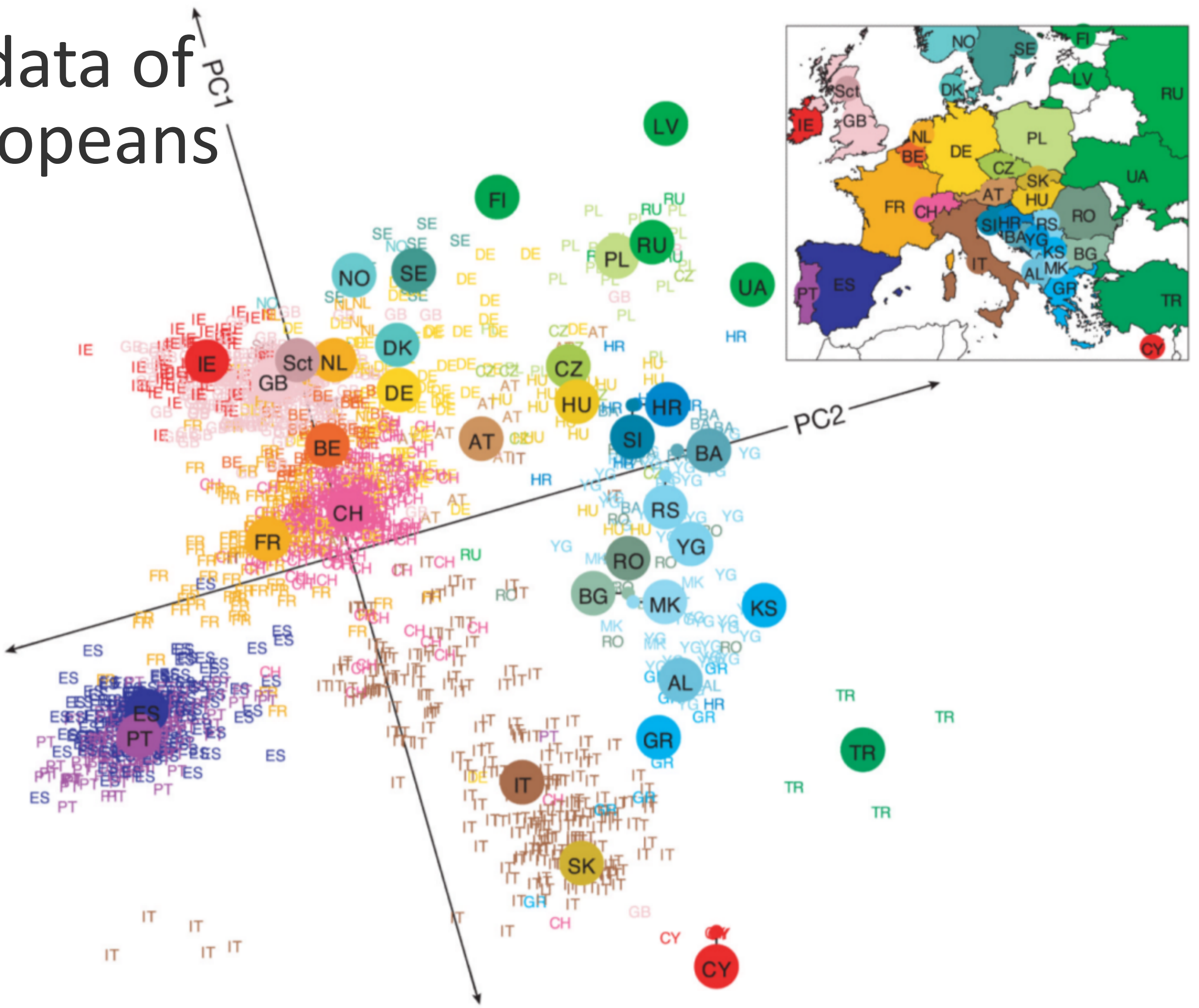
Scree graph (example)



Optdigits dataset of the UCI repository [Ethem Alpaydin]

PCA example: Population genetics

Genetic data of 1387 Europeans



Novembre et al, 2008

Non-linear feature extraction

Multidimensional scaling (MDS)

- Find a mapping that preserves the **dissimilarities** between the data points.

$$\arg\min_{Z \in \mathbb{R}^{n \times m}} \sum_{t=1}^{n} \sum_{u=t+1}^{n} \left(||\boldsymbol{z}^t - \boldsymbol{z}^u|| - d_{tu} \right)^2$$

Multidimensional scaling (MDS)

- Find a mapping that preserves the **dissimilarities** between the data points.

$$\arg\min_{Z\in\mathbb{R}^{n\times m}} \sum_{t=1}^{n}\sum_{u=t+1}^{n} \left(||\boldsymbol{z}^t - \boldsymbol{z}^u|| - d_{tu}\right)^2$$

  - In **Euclidean space:** $d_{tu} = ||\boldsymbol{x}^t - \boldsymbol{x}^u||$

    Equivalent to PCA!

Multidimensional scaling (MDS)

- Find a mapping that preserves the **dissimilarities** between the data points.

$$\arg \min_{Z \in \mathbb{R}^{n \times m}} \sum_{t=1}^{n} \sum_{u=t+1}^{n} \left(||\boldsymbol{z}^t - \boldsymbol{z}^u|| - d_{tu} \right)^2$$

  - In **Euclidean space:** $d_{tu} = ||\boldsymbol{x}^t - \boldsymbol{x}^u||$

    Equivalent to PCA!
  - dissimilarity can come from **another metric**

?

Multidimensional scaling (MDS)

- Find a mapping that preserves the **dissimilarities** between the data points.

$$\arg\min_{Z\in\mathbb{R}^{n\times m}} \sum_{t=1}^{n}\sum_{u=t+1}^{n} \left(||\boldsymbol{z}^t - \boldsymbol{z}^u|| - d_{tu}\right)^2$$

  - In **Euclidean space:** $d_{tu} = ||\boldsymbol{x}^t - \boldsymbol{x}^u||$

    Equivalent to PCA!

  - dissimilarity can come from **another metric**

    $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$

$$d(\boldsymbol{x},\boldsymbol{v}) = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{v} \text{ (identity of indiscernables)}$$

$$d(\boldsymbol{x},\boldsymbol{v}) = d(\boldsymbol{v},\boldsymbol{x}) \text{ (symmetry)}$$

$$d(\boldsymbol{x},\boldsymbol{v}) \leq d(\boldsymbol{x},\boldsymbol{w}) + d(\boldsymbol{w},\boldsymbol{v}) \text{ (triangular inequality)}$$

Multidimensional scaling (MDS)

- Find a mapping that preserves the **dissimilarities** between the data points.

$$\arg\min_{Z \in \mathbb{R}^{n \times m}} \sum_{t=1}^{n} \sum_{u=t+1}^{n} \left(||\boldsymbol{z}^t - \boldsymbol{z}^u|| - d_{tu} \right)^2$$

  - In **Euclidean space:** $d_{tu} = ||\boldsymbol{x}^t - \boldsymbol{x}^u||$

    Equivalent to PCA!
  - dissimilarity can come from **another metric**
  - … or be **non-metric.**

Multidimensional scaling (MDS)

- Find a mapping that preserves the **dissimilarities** between the data points.

$$\arg \min_{Z \in \mathbb{R}^{n \times m}} \sum_{t=1}^{n} \sum_{u=t+1}^{n} \left(||\boldsymbol{z}^t - \boldsymbol{z}^u|| - d_{tu} \right)^2$$

  - In **Euclidean space:** $d_{tu} = ||\boldsymbol{x}^t - \boldsymbol{x}^u||$

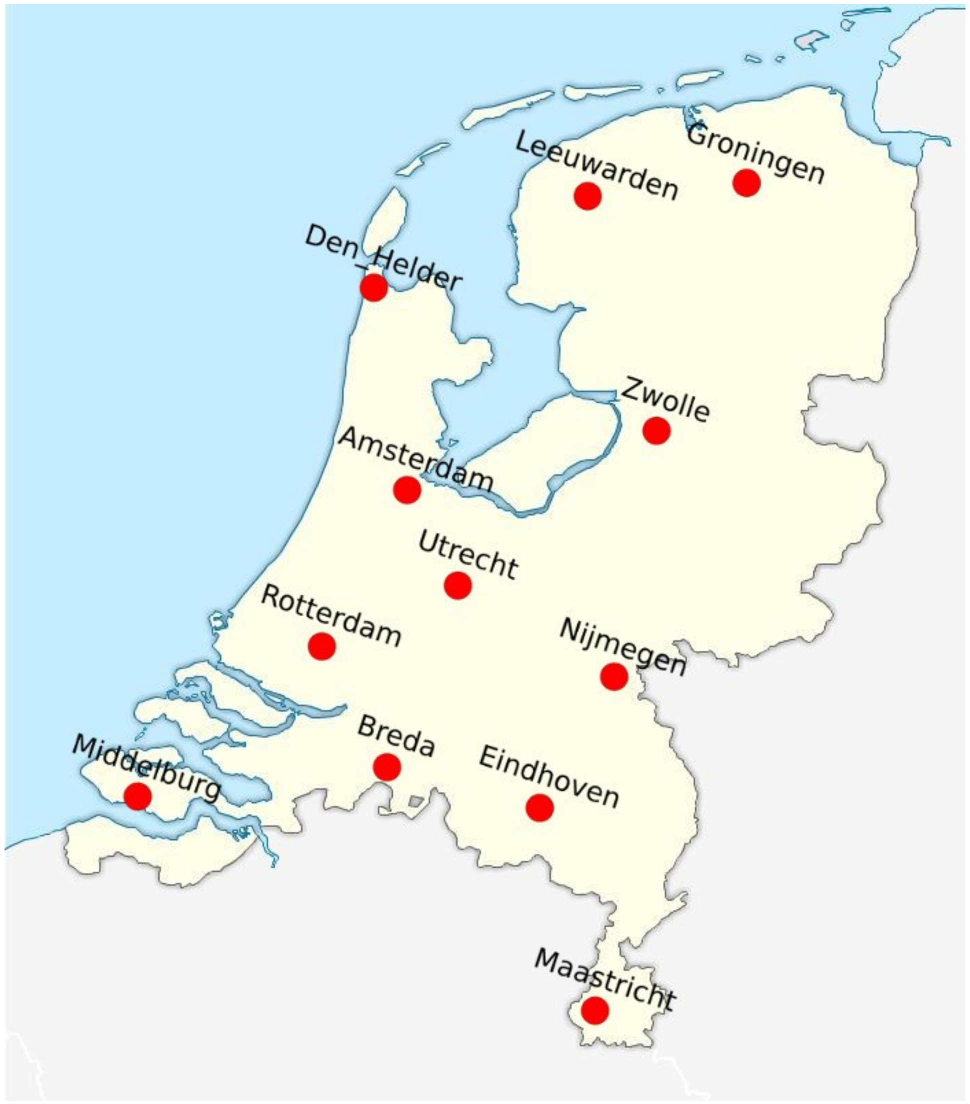
    Equivalent to PCA!
  - dissimilarity can come from **another metric**
  - … or be **non-metric.**

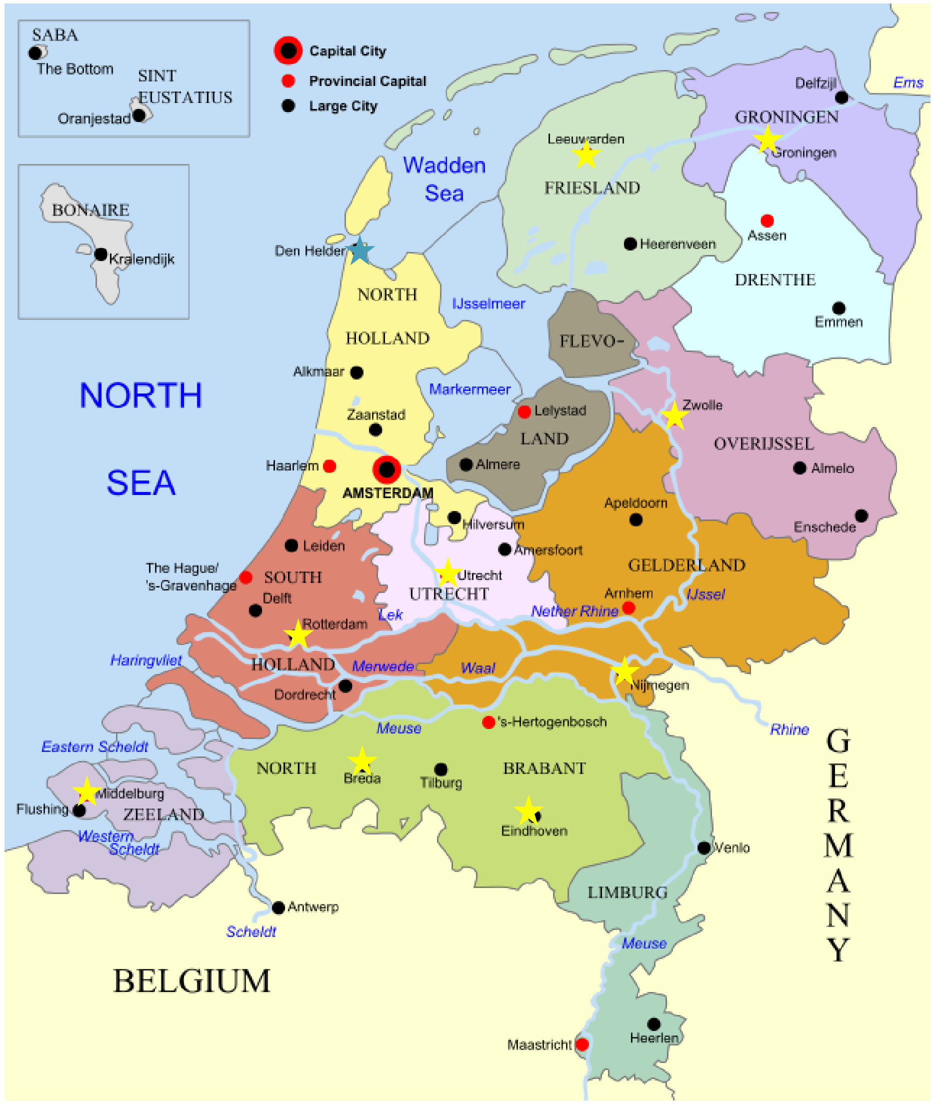
Application: place cities on a map so as to respect their pairwise distances.

Works up to translation / rotation.

Reconstructing a map of the Netherlands

http://www.wouterspekkink.org/?p=299





IsoMap

- Incorporate **local structure** in MDS
- Create a **neighborhood graph**
  - connect each point to its K nearest neighbors (or all neighbors within a distance ε)
  - weight by distance
- Compute **geodesic distances** on the neighborhood graph

  length of the shortest (weighted) path – e.g. Djikstra
- Apply **MDS** to the resulting **dissimilarity matrix.**

IsoMap on “Swiss roll” data

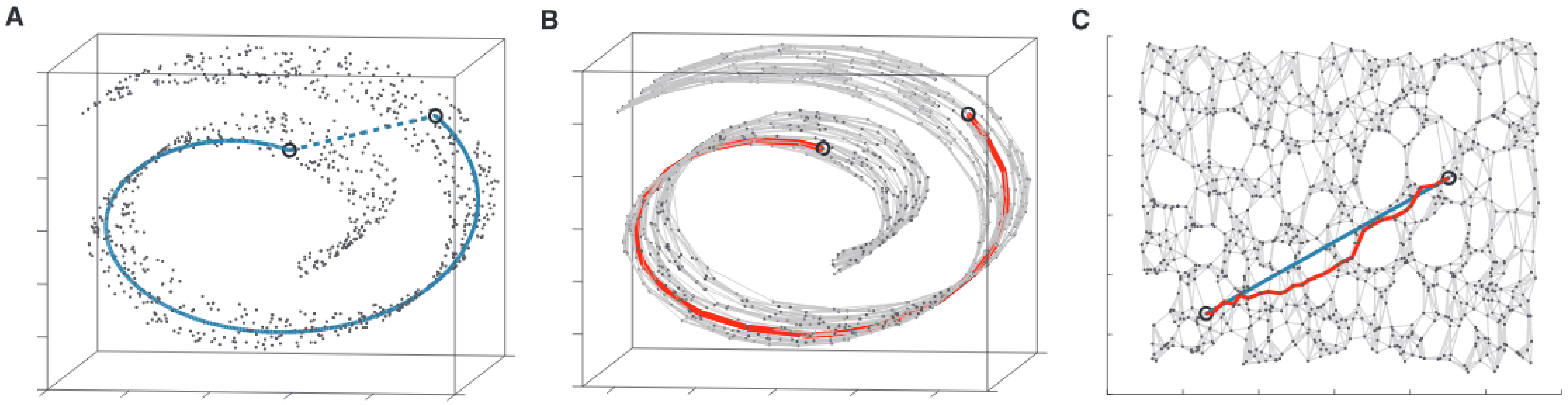


Fig. 3. The “Swiss roll” data set, illustrating how Isomap exploits geodesic paths for nonlinear dimensionality reduction. (**A**) For two arbitrary points (circled) on a nonlinear manifold, their Euclidean distance in the high-dimensional input space (length of dashed line) may not accurately reflect their intrinsic similarity, as measured by geodesic distance along the low-dimensional manifold (length of solid curve). (**B**) The neighborhood graph *G* constructed in step one of Isomap (with *K* = 7 and *N* = 1000 data points) allows an approximation (red segments) to the true geodesic path to be computed efficiently in step two, as the shortest path in *G*. (**C**) The two-dimensional embedding recovered by Isomap in step three, which best preserves the shortest path distances in the neighborhood graph (overlaid). Straight lines in the embedding (blue) now represent simpler and cleaner approximations to the true geodesic paths than do the corresponding graph paths (red).

Tennenbaum et al. (2000) **A Global Geometric Framework for Nonlinear Dimensionality Reduction.** *Science.* `http://web.mit.edu/cocosci/Papers/sci_reprint.pdf`

t-Stochastic Neighbor Embedding (tSNE)

- Very popular at the moment
- Approximate the distribution of pairwise distances in the data by a t-distribution (Student's)

$$\arg\min_{\boxed{Q}} \sum_{i=1}^{n} \boxed{KL}(\boxed{P_i} \,\|\, Q_i)$$

follows a t-distribution

- distribution of the conditional probability that **x**^{i} picks **x**^{j} as a neighbor
- neighbors are picked in proportion to their probability density under a Gaussian centered in **x**^{i}.

$$P_i(\boldsymbol{x}^j) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{||\boldsymbol{x}^j - \boldsymbol{x}^i||^2}{2\sigma^2}\right)$$

Kullback-Leibler divergence (measures how much P diverges from Q).

https://distill.pub/2016/misread-tsne/

Summary

- **Goal:** Use $m < p$ features.
- **Feature selection:**
  - **Filter approaches**
  - **Wrapper approaches:** Subset selection

    Greedy search of the best set of features.
  - **Embedded approaches:** Lasso, Elastic Net
- **Feature extraction:** unsupervised
  - Linear: **Principal Components Analysis**

    Center the data and align it with axes of largest variance.
  - Non-linear: **MDS, IsoMap, tSNE**

    Preserve local distances/structure.

References

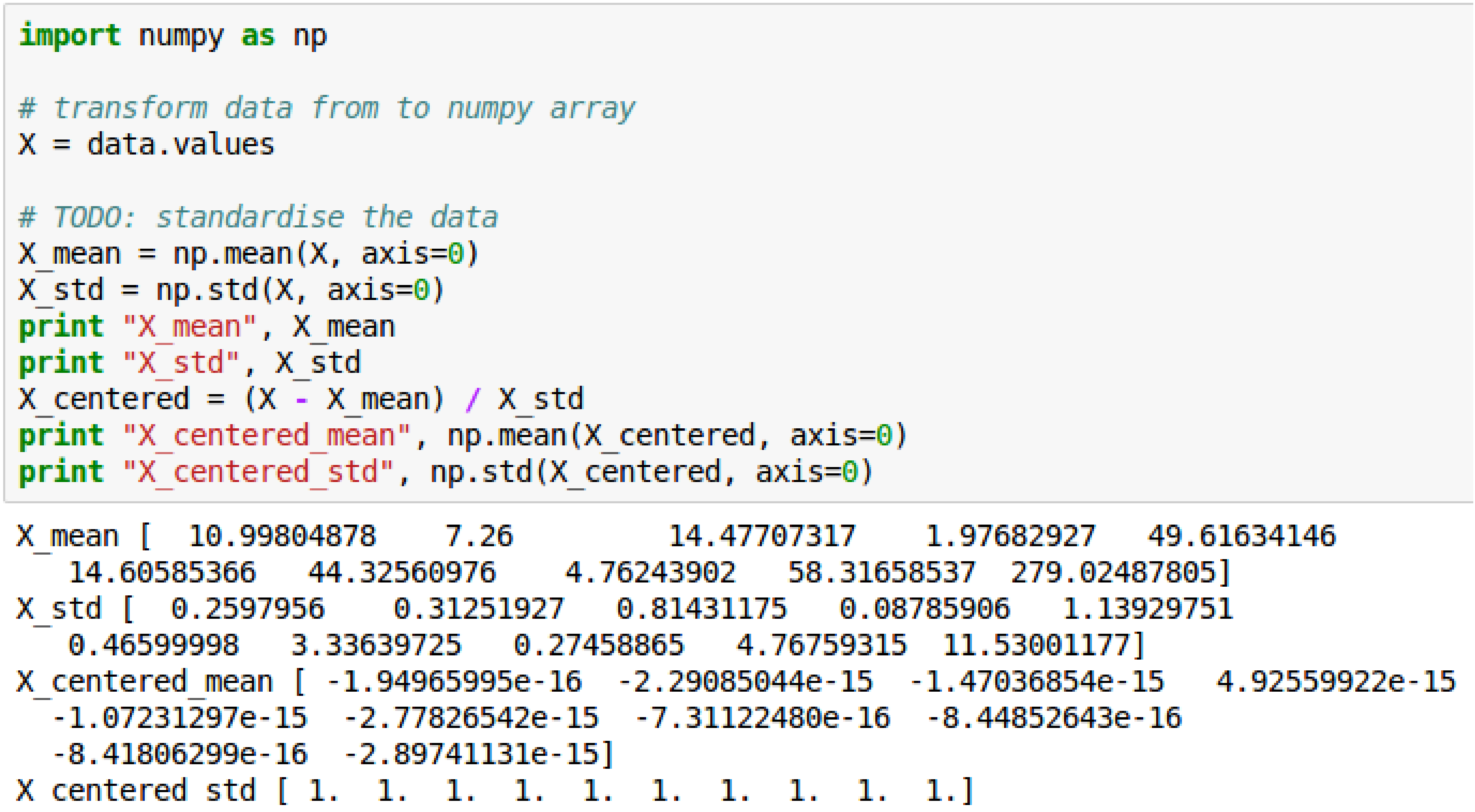
- *An Introduction to Feature Selection.* In: Applied Predictive Modeling. (2013) Kuhn and Johnson
  `https://link.springer.com/chapter/10.1007/978-1-4614-6849-3_19`

  This book chapter should be available to you from within CentraleSupelec. If you cannot access it, please let me know.
  - **Feature Selection:** Chapter 19.1 – 19.3
- *A Course in Machine Learning.*
  `http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf`
  - **PCA:** Chapter 15.2
- *The Elements of Statistical Learning.*
  `http://web.stanford.edu/~hastie/ElemStatLearn/`
  - **PCA:** Chapter 14.5.1
  - **MDS:** Chapter 14.8
- To go further
  - *An Introduction to Variable and Feature Selection* (2003), Guyon and Elisseeff.
    `http://www.jmlr.org/papers/v3/guyon03a.html`

Lab 1: some pointers

- `If pandas.plotting` does not work
  - Update your version of pandas
    - Pip: `pip install pandas --upgrade`
    - Anaconda: `conda update pandas`
  - Or try using `pandas.tools.plotting` instead.

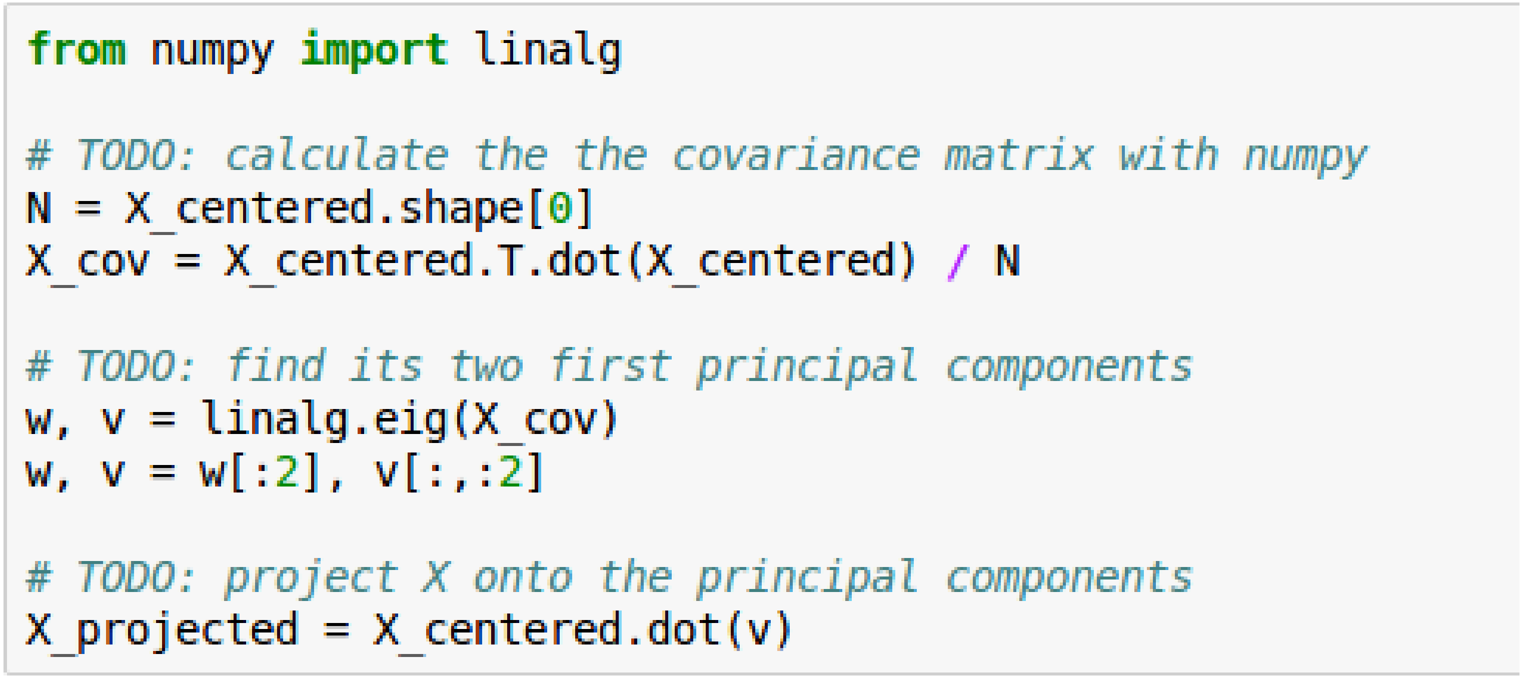
Standardization



```python
import numpy as np

# transform data from to numpy array
X = data.values

# TODO: standardise the data
X_mean = np.mean(X, axis=0)
X_std = np.std(X, axis=0)
print "X_mean", X_mean
print "X_std", X_std
X_centered = (X - X_mean) / X_std
print "X_centered_mean", np.mean(X_centered, axis=0)
print "X_centered_std", np.std(X_centered, axis=0)
```

```
X_mean [  10.99804878    7.26          14.47707317    1.97682927   49.61634146
   14.60585366   44.32560976    4.76243902   58.31658537  279.02487805]
X_std [  0.2597956    0.31251927   0.81431175   0.08785906   1.13929751
   0.46599998   3.33639725   0.27458865   4.76759315  11.53001177]
X_centered_mean [ -1.94965995e-16  -2.29085044e-15  -1.47036854e-15   4.92559922e-15
  -1.07231297e-15  -2.77826542e-15  -7.31122480e-16  -8.44852643e-16
  -8.41806299e-16  -2.89741131e-15]
X centered std [ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.]
```

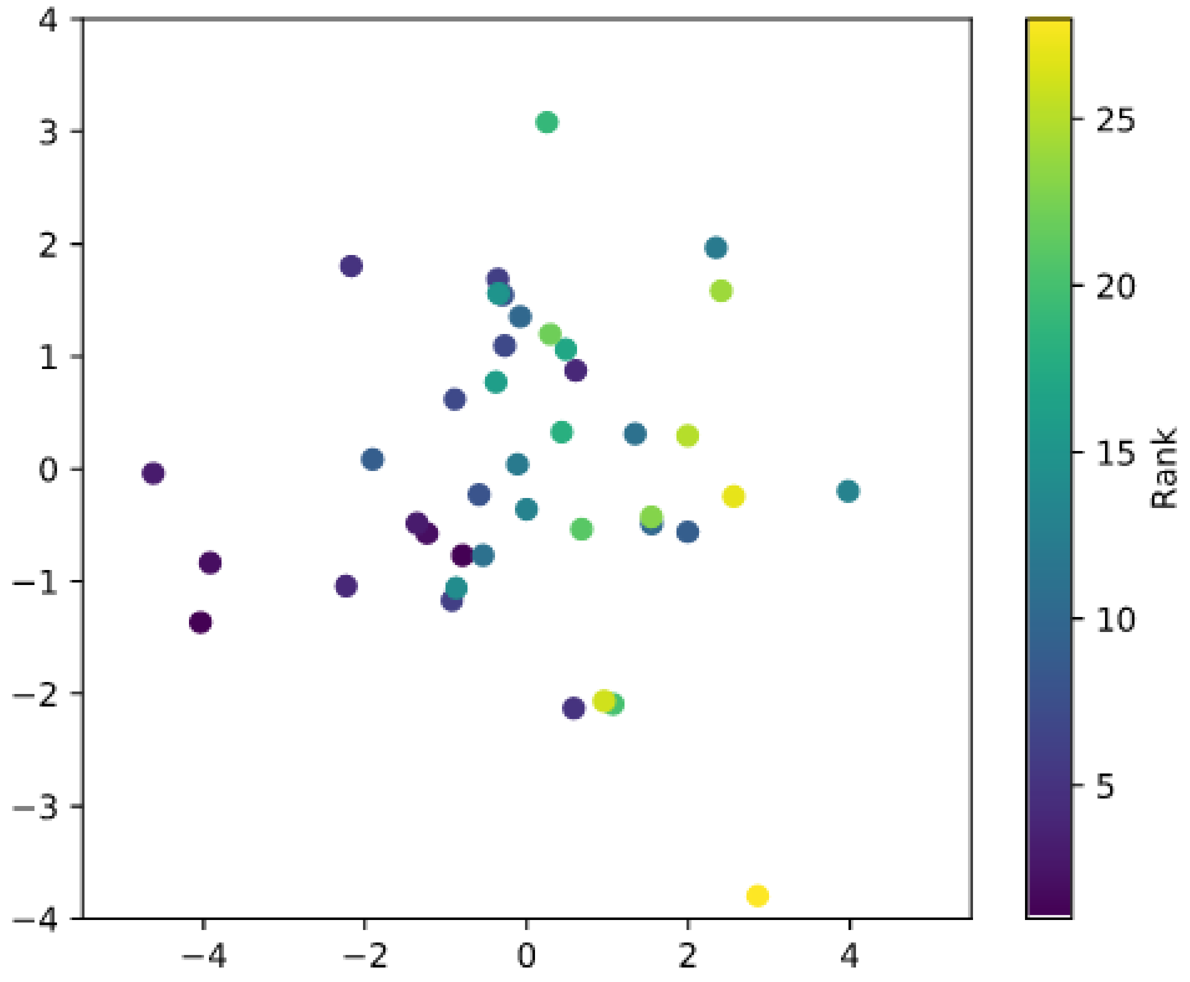
Finding the first 2 components



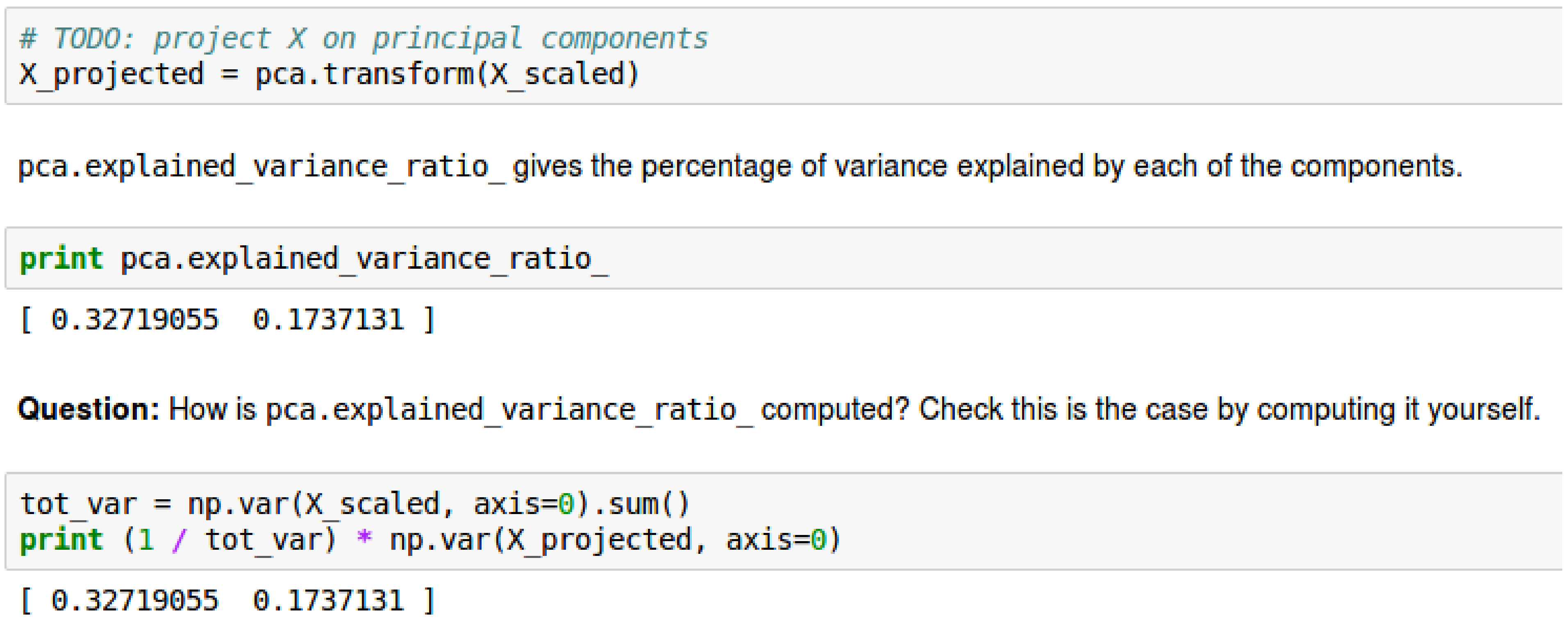
```
from numpy import linalg

# TODO: calculate the the covariance matrix with numpy
N = X_centered.shape[0]
X_cov = X_centered.T.dot(X_centered) / N

# TODO: find its two first principal components
w, v = linalg.eig(X_cov)
w, v = w[:2], v[:,:2]

# TODO: project X onto the principal components
X_projected = X_centered.dot(v)
```

```python
# create figure and axis objects
fig, ax = plt.subplots(figsize=(6, 5))
# create scatterplot on axis N.B. we record the return value to feed to the colorbar
cax = ax.scatter(X_projected[:, 0], X_projected[:, 1], c=my_data['Rank'],
                 cmap=plt.get_cmap('viridis'))
# Set axis limits
ax.set_xlim([-5.5, 5.5])
ax.set_ylim([-4, 4])
# Create color bar
plt.colorbar(cax, label='Rank')
```

```
<matplotlib.colorbar.Colorbar at 0x7f05bb1a2310>
```



Computing the percentage of variance explained (sklearn)



```
# TODO: project X on principal components
X_projected = pca.transform(X_scaled)
```

`pca.explained_variance_ratio_` gives the percentage of variance explained by each of the components.

```
print pca.explained_variance_ratio_
```

```
[ 0.32719055  0.1737131 ]
```

Question: How is `pca.explained_variance_ratio_` computed? Check this is the case by computing it yourself.

```
tot_var = np.var(X_scaled, axis=0).sum()
print (1 / tot_var) * np.var(X_projected, axis=0)
```

```
[ 0.32719055  0.1737131 ]
```