# CS-EJ3211 Machine Learning with Python

## Session 6 - Feature Learning

Alexander Pavlyuk

Aalto University
FITech

05.07.23

# Recap - Unsupervised learning

Unsupervised methods:

- Clustering (week 5) $\longleftarrow$
- Feature Learning (week 6)

# Recap - Clustering

- Idea: decomposing a data set into few groups (clusters) of similar data points.
- Hard clustering (K-means) - each data point belongs to exactly one cluster.
- Soft clustering (GMM) - each data point belongs to several clusters with varying degrees of belonging.
- Similarity measures: Euclidean distance, connectivity, etc.

# Recap - Unsupervised learning

Unsupervised methods:

- Clustering (week 5)
- Feature Learning (week 6) $\longleftarrow$

# Feature Learning

The efficiency of ML methods depends on the choice of features:

- More features - the risk of an increase in computational resources used
- Fewer features - the risk of overfitting

The goal of feature learning - find just enough relevant features to achieve high ML methods performance.

# Feature Learning

**Feature Learning** - automate the choice of finding good features.

Feature Learning can be:

- supervised - dictionary learning, ANN
- unsupervised - PCA

Learn a hypothesis map that reads in some representation of a datapoint (e.g features) and transforms it to a set of (new) features:

$$\mathbf{z} = (z_1, ..., z_D) \rightarrow \mathbf{x} = (x_1, ..., x_n) \ ,$$

# Dimensionality reduction

Transform high dimensional (many features) dataset $Z$ to a dataset $X$ with lower dimensionality.

# Dimensionality reduction

Why?

- reduce the use of computational resources
- prevent overfitting
- data visualization

How?

- How to map datapoint to a space with lower dimensionality?
- How to quantify the information loss after dataset transformation?

# Dimensionality reduction - Compression

The goal is to find (learn) a compression map:

$h(\cdot) : \mathbb{R}^D \rightarrow \mathbb{R}^n$,

that transforms a long feature vector $\mathbf{z} \in \mathbb{R}^D$ to a short feature vector $h(\mathbf{z}) = \mathbf{x} \in \mathbb{R}^n$ $(D \gg n)$.

The new feature vector $\mathbf{x} = h(\mathbf{z})$ is a compressed representation (code) of the original feature vector $\mathbf{z}$.

# Dimensionality reduction - Reconstruction

We can reconstruct original vector using a reconstruction map:

$r(\cdot) : \mathbb{R}^n \to \mathbb{R}^D$

Reconstructed original feature vector: $\hat{\mathbf{z}} = r(\mathbf{x})$

Information loss (reconstruction error): $\hat{\mathbf{z}} - \mathbf{z}$

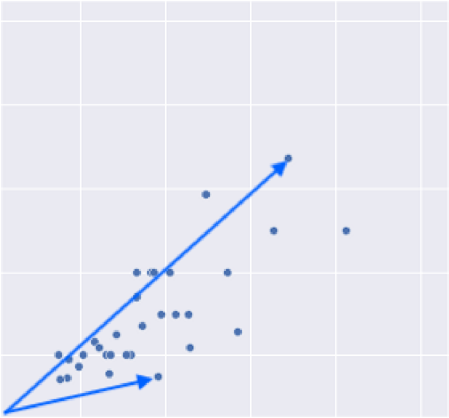
Need to learn a compression map such that: $\hat{\mathbf{z}} \approx \mathbf{z}$, i.e. reduce dimensionality with minimal information loss.

# Principal Component Analyses

**PCA** is a dimensionality reduction method where compression map $h(\cdot)$ is a **linear** map.

**PCA** is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components.

# Linear Transformation

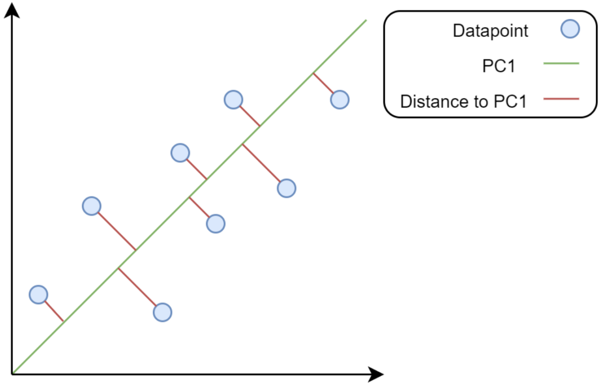


3Blue1Brown

# Principal Component Analyses

PCA, gif

# Principal Component Analyses




- Reconstruction error - the mean of squared distances (red lines)
- This component corresponds to the direction of the largest variance of the data

# Principal Component Analyses

Linear transformation: $h(\cdot) : \mathbb{R}^2 \to \mathbb{R}^1$:
$\mathbf{z} = (z_1, z_2) \to \mathbf{x} = (x_1)$
$\mathbf{x} = W\mathbf{z}$

Reconstruction: $\hat{\mathbf{z}} = W^T\mathbf{x}$

How to quantify the information loss after dataset transformation?

Information loss is measured with **reconstruction error**:

$$(1/m)\sum_{i=1}^{m} \left\|\mathbf{z}^{(i)} - \hat{\mathbf{z}}^{(i)}\right\|_2^2$$

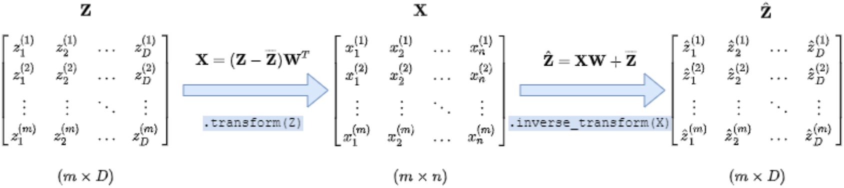
Reconstruction error, gif.
PC1, gif.
Source.

# Principal Component Analyses

We can interpret PC components as follows:
PC1 = $x$ amount of feature 1 + $y$ amount of feature 2

Check out PCA by StatQuest for more explanations.

# Principal Component Analyses



# Student Task 6.1. - Compute PCA

1. Create a PCA object from the sklearn.Decomposition.PCA
2. Find principal components by fitting PCA to the raw dataset
3. Store principal components in W_pca
4. Transform the raw dataset and store the result in Z_hat
5. Compute reconstruction error according to the formula. Remember to multiply the mean squared difference by the number of raw features!

# Student Task 6.2. - Reconstruction Error vs. Number of PCs

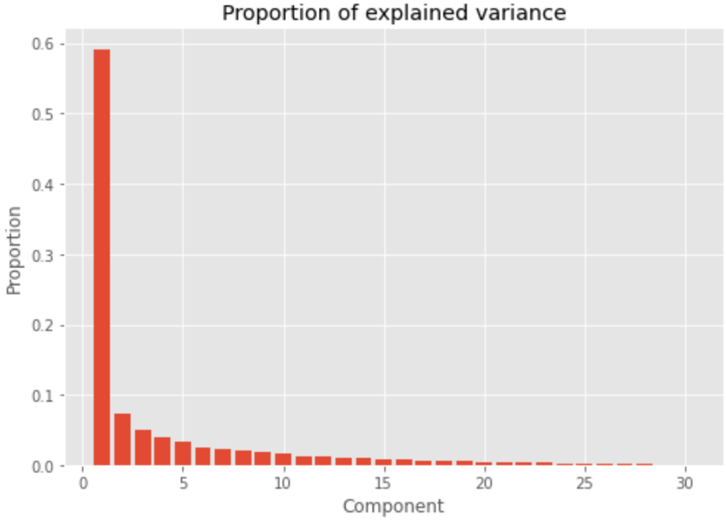
1. Create a loop with $m$ iterations
2. Create PCA object with the number of components increased by 1 from the previous iteration (first iteration - 1 component, second iteration - 2 components, . . . )
3. Fit PCA object, transform the raw dataset, and calculate reconstruction error.

Note: here you should not multiply the reconstruction error by the length of the raw feature vector, since it is redundant in the comparison.

# Student Task 6.3. - Proportion of variance explained

1. Create a PCA object with the defined maximum number of components
2. Fit the PCA object to the raw dataset
3. Access proportion of total variance explained by calling sklearn method pca.explained_variance_ratio_ and store the result in var_ratio variable
4. Calculate the cumulative proportion of total variance explained by calling the numpy method .cumsum(...) on var_ratio
5. Find the minimum number of components for which the cumulative proportion of total variance explained exceeds the defined threshold. Store the result in n_threshold variable.

# Student Task 6.3. - Proportion of variance explained



# Student Task 6.4. - PCA with Linear Regression

1. Create a linear regression model, fit it to the raw training data, and calculate training and validation errors (week 2 material)
2. Create a PCA object with 2 components, fit it to the raw training data
3. Transform training and validation data by applying .transform(. . . ) method
4. Create a linear regression model, fit it to the **transformed** training data, and calculate training and validation errors

# Summary

- PCA tries to find a linear subspace that has maximal variance
- Goal of PCA: nearby points remain nearby, distant points remain distant
- PCA always converges to the same optima
- PCA can be computed easily

# Other methods

- Independent component analysis (ICA) - Wikipedia
- Multidimensional scaling - Wikipedia
- Sammon mapping - Wikipedia
- Isometric mapping of data manifolds (ISOMAP) - Wikipedia
- Locally linear embedding (LLE) - Article
- Other nonlinear methods - Wikipedia

# Additional material

- CS-E4840 "Information Visualization D" course, lecture 9
- Making sense of principal component analysis - StackExchange
- PCA by StatQuest - YouTube
- A blog post about PCA with good visualization - Medium
- 3Blue1Brown lecture about linear transformation - YouTube