

Modul - Unsupervised and Reinforcement Learning (URL)

Bachelor Programme AAI

04 - Dimensionality Reduction Techniques

Prof. Dr. Marcel Tilly

Faculty of Computer Science, Cloud Computing

Agenda



Code can be found in [DimensionalityReduction.ipynb](#) on GitHub or [hosted on myBinder](#)

- Apply dimension reduction techniques
- Describe the concepts behind principal components and dimensionality reduction
- Apply principal component analysis (PCA) when solving problems using scikit-learn
- Compare manual PCA versus scikit-learn



Dimensionality Reduction

Many modern data domains involve huge numbers of features / dimensions

- Documents: thousands of words, millions of bigrams
- Images: thousands to millions of pixels
- Genomics: thousands of genes, millions of DNA polymorphisms
- What else?



What Is *Dimensionality*?

- To put it simply, *dimensionality* is the number of **dimensions**, **features**, or **variables** associated with a sample of data.
- Often, this can be thought of as a number of columns in a spreadsheet, where each sample is on a new row, and each column describes some attribute of the sample.

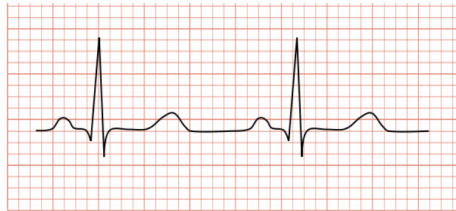
Example:

Pressure (hPa)	Temperatur (C)	Humidity (%)
1050	32.2	12
1026	27.8	80
953	14.8	59

- We have three samples of data, each with three independent features or dimensions.

Example: ECG data

- We have a very large dataset of time series data, such as echo-cardiogram or ECG



- Let's say, that these signals were captured from your company's new model of watch, and we need to look for signs of a heart attack or stroke.
- In looking through the dataset, we can make a few observations:
 - Most of the individual heartbeats are very similar.
 - There is some noise in the data from the recording system or from the patient moving during the recording.
 - Despite the noise, the heartbeat signals are still visible.
 - There is a lot of data – too much to be able to process using the hardware available on the watch.

Dimensionality Reduction

What is the objective?

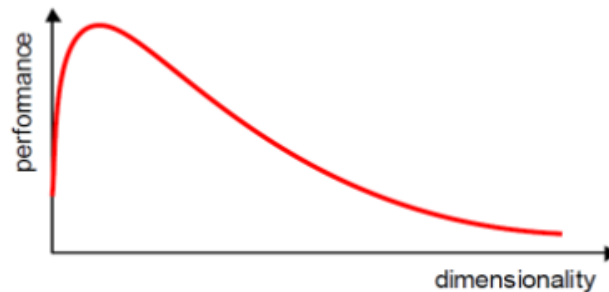
What is the objective?

- Depending upon the problem being solved, or the origin of this dataset, we may want to reduce the number of dimensions per sample without losing the provided information.
- Choose an optimum set of features of lower dimensionality to improve classification accuracy.
- By using dimensionality reduction, we are able to remove much of the noise from the signal, which, in turn, will assist with the performance of the algorithms that are applied to the data as well as reduce the size of the dataset to allow for reduced hardware requirements.

The *Curse of Dimensionality*



- Increasing the number of features will not always improve classification accuracy.
- The inclusion of more features might actually lead to worse performance.
- The number of training examples required increases exponentially with dimensionality p
- Optimization problems will be infeasible as the number of features increases.
- Due to the absolute scale of inherent points in an n -dimensional space, as n maintains to grow, the possibility of recognizing a particular point (or even a nearby point) proceeds to fall.



- The *sparseness* of data is the property of being scanty or scattered.
- It lacks *denseness*, and its high percentage of the variable's cells do not contain actual data - fundamentally full of “empty” or “N/A” values.
- Points in an n-dimensional space frequently become sparse as the number of dimensions grows. The distance between points will extend to grow as the number of dimensions increases.

C1	C2	C3	C4	C5	C6
0	0	0	0	0	1
0	0	0	2	0	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0



Applications of *Dimensionality Reduction*

Pre-processing/feature engineering

- The quality of the information provided during the algorithm development, as well as the correlation between the input data and the desired result, is critical in order for a high-performing solution to be designed.
- We should isolate the most important components of information from the data and provide this to the model so that only the most relevant information is being provided.
- This can also have a secondary benefit in that we have reduced the number of features being provided to the model, so there can be a corresponding reduction in the number of calculations to be completed. This can reduce the overall training time for the system.

Noise reduction

- Dimensionality reduction can also be used as an effective noise reduction/filtering technique.
- It is expected that the noise within a signal or dataset does not comprise a large component of the variation within the data.
- Thus, we can remove some of the noise from the signal by removing the smaller components of variation and then restoring the data back to the original dataspace.

Generating plausible artificial datasets

- We can divide the dataset into the components of information (or variation), we can investigate the effects of each components or generate new dataset samples by adjusting the ratios between features.
- We can scale these components, which, in effect, increases or decreases the importance of that specific component. This is also referred to as statistical shape modelling, as one common method is to use it to create plausible variants of shapes.
- It is also used detect facial landmarks in images in the process of active shape modelling.

Financial modelling/risk analysis



- Dimensionality reduction provides a useful toolkit for the finance industry, as being able to consolidate a large number of individual market metrics or signals into a smaller number of components allows for faster, and more efficient computations.
- Similarly, the components can be used to highlight those higher-risk products/companies.

Dimension Reduction lead to Simpler models

Because

- **Simpler to use** by *lower computational complexity*
- **Easier to train** because *needs less examples*
- **Less sensitive to noise** because less noise
- **Easier to explain** because better *interpretable*
- **Generalizes better** through *lower variance*

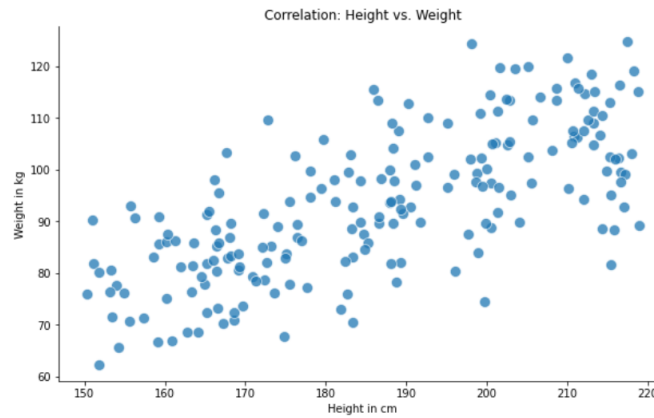


Approaches to *Dimensionality Reduction*

Correlation



Let's imagine, we measure the variables *height* and *weight*:



- **Positive correlation:** tells us that both variables move in the same direction e.g. both values are increasing simultaneously.
- **Negative correlation:** describes inversely correlated variables. Meaning, if one variable is increasing the other is decreasing or vice-versa.
- A correlation coefficient of zero shows that there is no relationship at all.

Approaches to Dimensionality Reduction

Feature selection

Select subset of existing features (without modification)

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \rightarrow \mathbf{y} = \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \cdot \\ \cdot \\ x_{i_K} \end{bmatrix}$$

$K \ll N$

- Linearly project n-dimensional data onto a k-dimensional space
- $k < n$, often $k \ll n$
- Example: project space of 104 words into 3 dimensions

Approaches to Dimensionality Reduction

Feature Extraction

In feature extraction, a set of new features are found. That is found through some mapping function f from the existing features.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \xrightarrow{f(\mathbf{x})} \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_K \end{bmatrix}$$

$K \ll N$

- Linearly project n -dimensional data onto a k -dimensional space
- $k < n$, often $k \ll n$
- Example: project space of 104 words into 3 dimensions

Linear dimensionality reduction

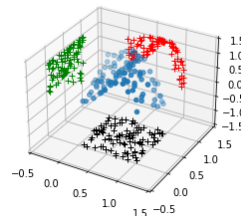
Best k-dimensional subspace for projection depends on task

- **Classification:** maximize separation among classes
 - Example: Linear Discriminant Analysis (LDA)
 - **Regression:** maximize correlation between projected data and response variable
 - Example: Partial Least Squares (PLS)
 - **Unsupervised:** retain as much data variance as possible
 - Example: Principal Component Analysis (PCA)
-
- **Principal Component Analysis (PCA):** It seeks a projection that preserves as much information as possible in the data.
 - **Linear Discriminant Analysis (LDA):** - It seeks a projection that best discriminates the data.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is an exploratory approach to reduce the data set's dimensionality to 2D or 3D, used in exploratory data analysis for making predictive models. Principal Component Analysis is a linear transformation of data set that defines a new coordinate rule such that:

- The highest variance by any projection of the data set appears to laze on the first axis.
- The second biggest variance on the second axis, and so on.



Usage of PCA:

- To reduce the number of dimensions in the dataset.
- To find patterns in the high-dimensional dataset
- To visualize the data of high dimensionality
- To ignore noise
- To improve classification
- To gets a compact description
- To captures as much of the original variance in the data as possible

History of PCA

- PCA was invented in 1901 by Karl Pearson and Harold Hotelling as an analog of the Principal axis theorem.
- Mathematically the main objective of PCA is to:
 - Find an orthonormal basis for the data
 - Sort dimensions in the order of importance
 - Discard the low significance dimensions
 - Focus on uncorrelated and Gaussian components

Lower dimensional projections

- Obtain new feature vector by transforming the original features x_1, \dots, x_n

$$z_1 = w_0^1 + \sum_i w_i^1 x_i$$

...

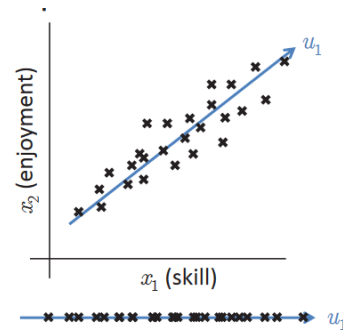
$$z_k = w_0^k + \sum_i w_i^k x_i$$

- New features are linear combinations of old ones
- Reduces dimension when $k < n$

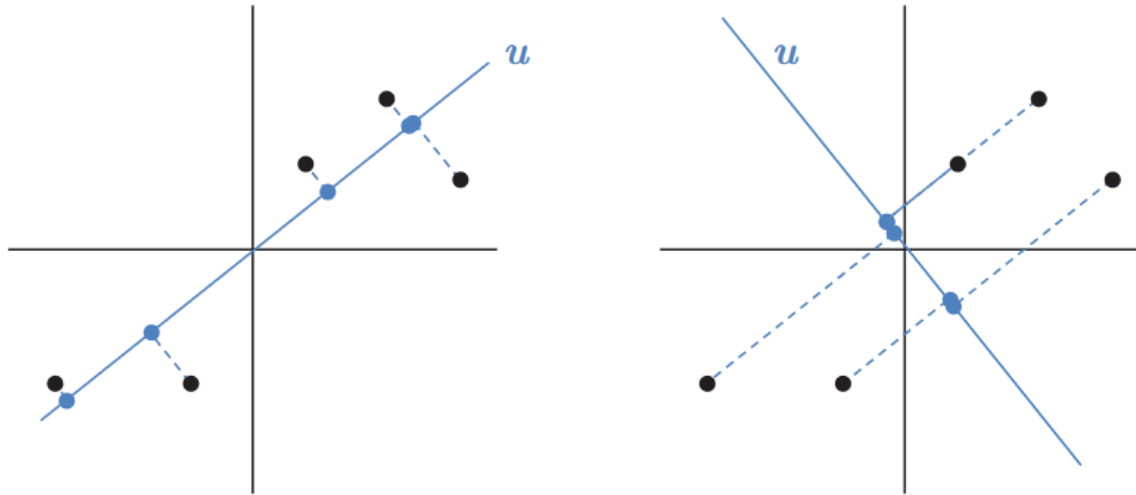
Using a new basis for the data

- Project a point into a (lower dimensional) space:
 - Point: $x = (x_1, \dots, x_n)$
 - Select a basis – set of unit (length 1) basis vectors (u_1, \dots, u_k)
 - with an orthonormal basis: $u_j \cdot u_j = 1$, and $u_j \cdot u_l = 0$ for $j \neq l$
 - select a center \tilde{x} , defines offset of space
 - best coordinates in lower dimensional space defined by dot-products: for (z_1, \dots, z_k) ,

$$z_j^i = (x^i - \tilde{x}) \cdot u_j$$



Which projection is better?



Goal: Select the direction of u !

Revisit: Vector Projections



Basic definitions

$$\mathbf{x}^T \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos \theta$$

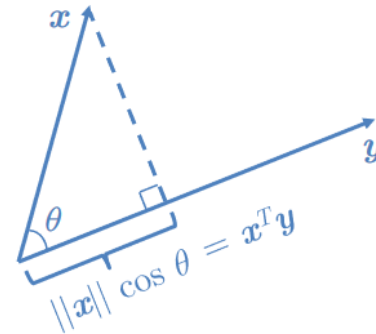
$$\cos \theta = \text{adj} / \text{hyp}$$

Then

$$\text{adj} = \text{hyp} \cos \theta$$

$$= \|\mathbf{x}\| \cos \theta$$

$$= \mathbf{x}^T \mathbf{y} \text{ [assuming } \|\mathbf{y}\| = 1 \text{ (unit vector)]}$$



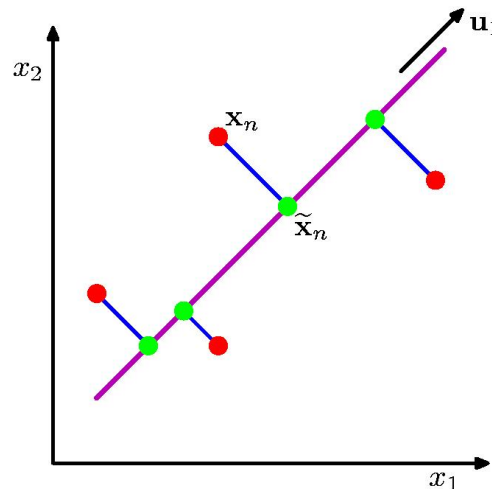
Dot product is the length of the projection!

How does PCA work?



The orthogonal projection of data from high dimensions to lower dimensions such that:

- Maximizes the variance of the projected line (purple)
- Minimizes the MSE between the data points and projections (blue)



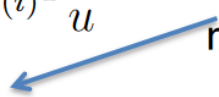
Maximize the variance

Let $x^{(i)}$ be the i^{th} data point minus the mean.

Choose unit-length u to maximize:

$$\begin{aligned}\frac{1}{m} \sum_{i=1}^m (x^{(i)T} u)^2 &= \frac{1}{m} \sum_{i=1}^m u^T x^{(i)} x^{(i)T} u \\ &= u^T \left(\frac{1}{m} \sum_{i=1}^m x^{(i)} x^{(i)T} \right) u.\end{aligned}$$

Covariance matrix Σ

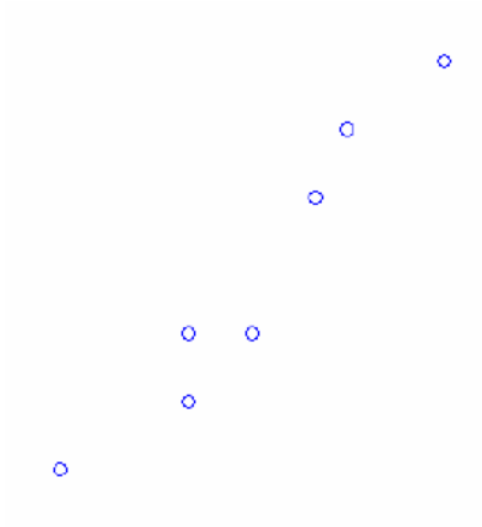


Let $\|u\|=1$ and maximize. Using the method of Lagrange multipliers, can show that the solution is given by the principal eigenvector of the covariance matrix!

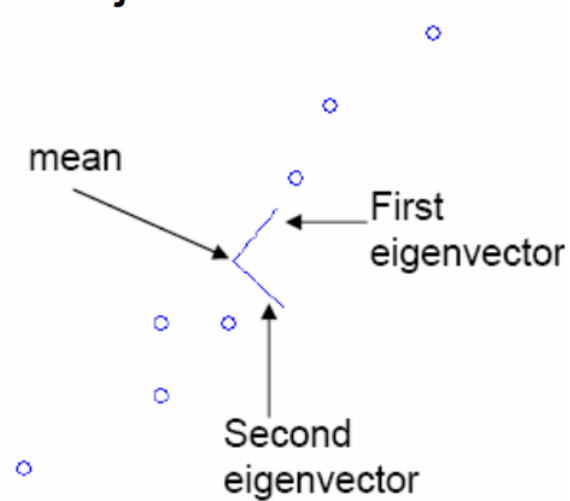
Eigenvector



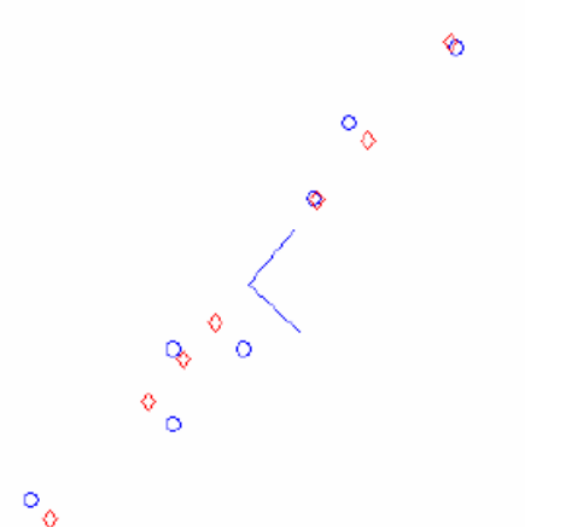
Data:



Projection:



Reconstruction:



Steps involved in PCA

- Start from m by n data matrix X
 - Standardize and **Recenter**: subtract mean from each row of X

$$X_c = X - \tilde{X}$$

- Compute covariance matrix:

$$\Sigma = 1/m \sum X_c^T X_c$$

- Find eigenvectors and values of Σ
- Principal components: k eigenvectors with highest eigenvalues

- Eigenvalues and eigenvector play important roles in PCA.
- The eigenvectors and eigenvalues of a covariance matrix (or correlation) describe the source of the PCA. Eigenvectors (main components) determine the direction of the new attribute space, and eigenvalues determine its magnitude.
- The PCA's main objective is to reduce the data's dimensionality by projecting it into a smaller subspace, where the eigenvectors form the axes.
- The eigenvectors define only the new axes' directions because they all have a size of 1.
 - Consequently, to decide which eigenvector(s), we can discard without losing much information in the subspace construction and checking the corresponding eigenvalues.
 - The eigenvectors with the highest values are the ones that include more information about the distribution of our data.

Covariance Matrix

The classic PCA approach calculates the covariance matrix, where each element represents the covariance between two attributes.

$$\sigma(x, y) = \frac{1}{1 - n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

if we mean center before we get:

$$\sigma(x, y) = \frac{x^T y}{1 - n}$$

```
import pandas as pd
import numpy as np

X = np.array([[0, 3, 4], [1, 2, 4], [3, 4, 5]])
np.cov(X, rowvar=False)
```

Covariance Matrix

Now imagine, a dataset with three features x , y , and z .

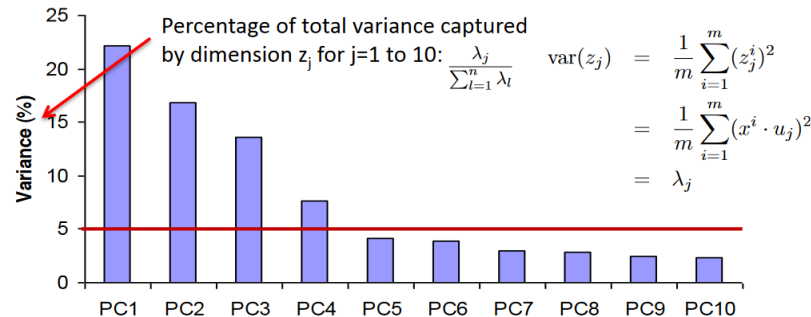
- Computing the covariance matrix will yield us a 3 by 3 matrix.
- This matrix contains the covariance of each feature with all the other features and itself.

$$C(x, y, z) = \begin{bmatrix} var_x & covar_{x,y} & covar_{x,z} \\ covar_{y,x} & var_y & covar_{y,z} \\ covar_{z,x} & covar_{z,y} & var_z \end{bmatrix}$$

The covariance matrix is symmetric and feature-by-feature shaped. The diagonal contains the variance of a single feature, whereas the non-diagonal entries contain the covariance.

Dimensionality reduction with PCA

- In high-dimensional problem, data usually lies near a linear subspace, as noise introduces small variability
- Only keep data projections onto principal components with large eigenvalues
- Can ignore the components of lesser significance.



- You might lose some information, but if the eigenvalues are small, you do not lose much

Choosing the Number of Principal Components

To choose k , note that

$$\text{average squared projected error} = \frac{1}{n} \sum_{i=1}^n \left\| \mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)} \right\|^2$$

$$\text{total variation in the data} = \frac{1}{n} \sum_{i=1}^n \left\| \mathbf{x}^{(i)} \right\|^2$$

Typically choose k to be the smallest value s.t. the error is upper-bounded.

e.g. To bound the error at 1%, choose k s.t.

$$\frac{\frac{1}{n} \sum_{i=1}^n \left\| \mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)} \right\|^2}{\frac{1}{n} \sum_{i=1}^n \left\| \mathbf{x}^{(i)} \right\|^2} \leq 0.01$$

This is equivalent to retaining 99% of the variance.

Algorithm

- 1) Try PCA with $k = 1$
- 2) Compute projected data
- 3) Check if ratio below threshold
- 4) Repeat

taken from Andrew Ng

Summary

- Today we talked about the need of dimensionality reduction
 - various approaches
 - PCA and the math behind
- Applications

Code can be found in [DimensionalityReduction.ipynb](#) on GitHub or [hosted on myBinder](#)

Exercise



- On GitLab: https://inf-git.fh-rosenheim.de/aai-url/04_exercise
- Work on Python, again!
- Work on dimensionality reduction
- Try PCA

