University of Warsaw
Faculty of Economic Sciences

Unsupervised Learning

Winter Semester, 2025/2026

Unsupervised learning: dimension reduction

Literature

Dimension reduction: literature

- Cady, F. (2017). The Data Science Handbook. Wiley.
- Leskovec, J; Rajamaran, A.; Ullman, J. (2014). Mining of Massive Datasets. Cambridge University Press.
- Faul, A. (2019). A Concise Introduction to Machine Learning. Chapman and Hall.
- Chopra, R.; England, A.; Alaudeen, M. (2019). Data Science with Python. Packt Publishing.

General overview

General overview

- Why do we need dimensionality reduction?
  - In general, we do not pose restrictions on the number of dimensions
  - Each numeric feature in a dataset is a dimension
  - Many features may be related
  - Probably we do not need them all in the dataset
  - If we include all of them it is unlikely to improve models
  - It may return arbitrary features
  - Thus, we may look for fewer features than the original set
  - Computational reasons – time and storage
  - Statistical reasons – better generalization
  - Visualization helps to understand data
  - We want to preserve information in original set (as far as possible)

General overview

- We are given some data points in d dimensions
- We would like to convert them to data points in $r < d$ dimensions
- We want to minimize the loss on information (to preserve essential properties)

Principal Component Analysis (PCA)

PCA

- We look for features in a transformed space
- Each dimension in the new space captures the most variation in the original data, when it is projected onto that dimension
- New features (if there are any) shall by highly correlated with some of the original features
- They should not be correlated with any of the other new features
- So we look into using the variance-covariance matrix we recall from correlation and regression
- Most information in the data should be retained if we project the data on some low dimensional manifold
- Advantages are visualization, extracting essential attributes, computational efficiency

PCA

- In simple words
  - We search for those directions in a space that have the highest variance
  - We then project the data onto the subspace of highest variance
  - This structure is encoded in the sample covariance of the data
  - We want to find the eigenvectors and eigenvalues of this covariance

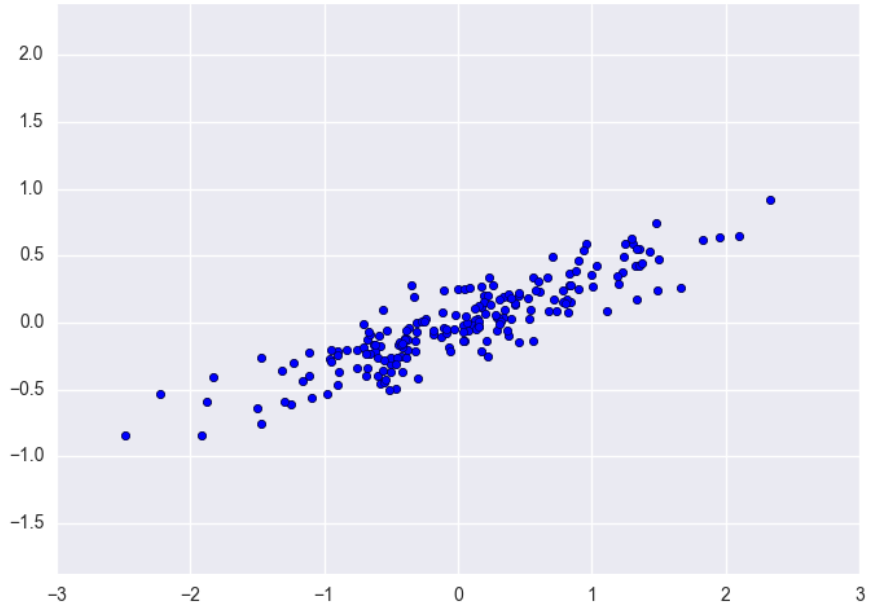
PCA

- We look for r-dim projection, which preserves variance in the best manner
  - We compute the mean vector μ and covariance matrix Σ of the original points
  - We compute eigenvectors and eigenvalues of Σ
  - We select top r eigenvectors
  - We project the points onto a subspace that is spanned by them (y is the new point, whereas x is the old point, the rows of A are the eigenvectors)

$$y = A(x - \mu)$$

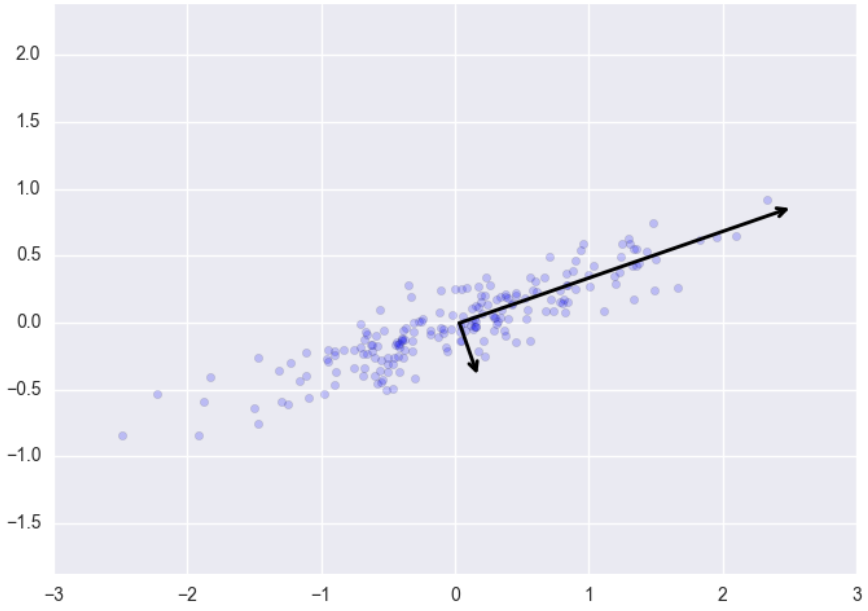
PCA

- Example: 200 points (two feature dimensions); the algorithm aims to learn about the relationship between the x and y values
- It is quantified by a list of the principal axes in the data, and using those axes to describe the dataset
- There is a nearly linear relationship between the x and y variables



PCA

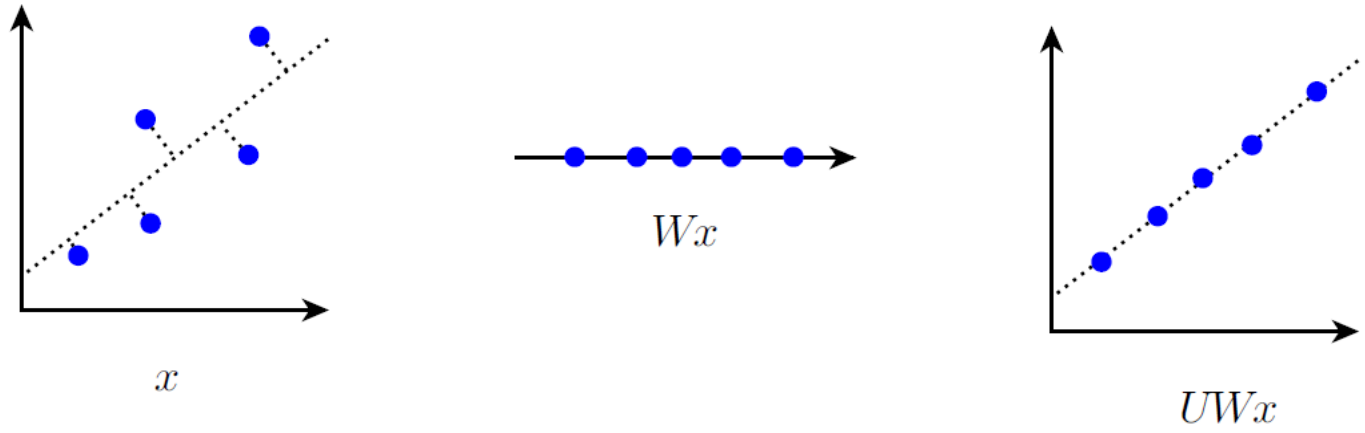
- We find two new features, on which the original data may be projected, rotated and scaled
- Use the "components" to define the direction of the vector, and the "explained variance" to define the squared-length of the vector
- These vectors stand for the principal axes of the data, and the length of the vector is an indication of how "important" that axis is in describing the distribution of the data
- It is a measure of the variance of the data when projected onto that axis
- The projection of each data point onto the principal axes are the "principal components" of the data



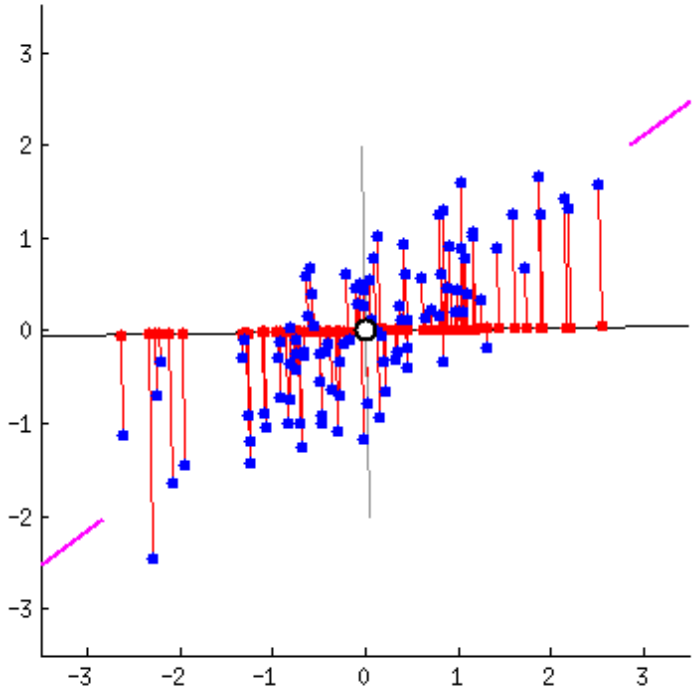
PCA

- We can reduce the dimensionality by 50% and preserve much of the original variance
- The information along the least important principal axis or axes is removed, leaving only the components of the data with the highest variance
- The fraction of variance that is cut out (proportional to the spread of points about the line formed in this figure) is roughly a measure of how much "information" is discarded in this reduction of dimensionality
- The transformation from data axes to principal axes is an affine transformation, which basically means it is composed of a translation, rotation, and uniform scaling

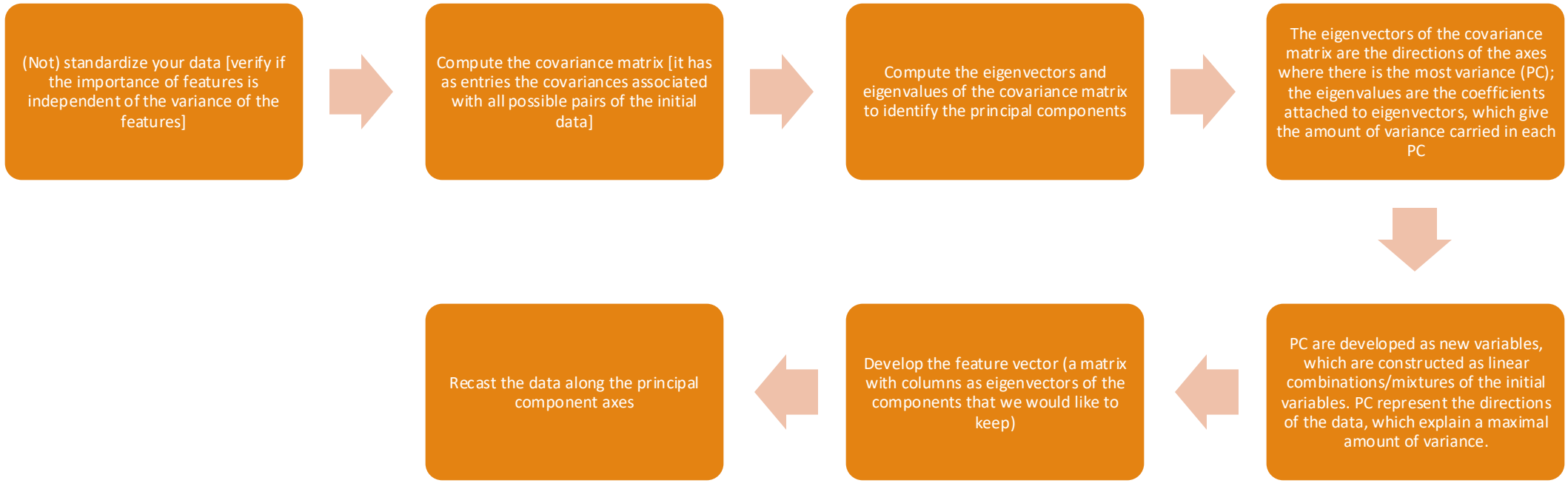
PCA



PCA (Jaadi, 2019)



PCA (Jaadi, 2019)



PCA

PCA

- PCA complexity is cubic in number of original features
  - It is not feasible for high-dimensional datasets
- Alternatively, approximate the sort of projection found by PCA
  - for example, one can use Random Projections
- More scalable, but what about quality of components?
- Can be shown to preserve distance relations from the original data

PCA

- Effective in a wide variety of contexts
- Good starting point in order to visualize the relationships between observations & the variance in the data
- Understand the intrinsic dimensionality of the data
- Offers a straightforward and effcient path to gain insight into high-dimensional data
- Weaknesses:
  - Highly affected by outliers in the data
  - Does not perform well with non-linear relationships in data (manifold learning, multidimensional scaling are more appropriate)

Multidimensional scaling (MDS)

MDS

- MDS is a nonlinear statistical technique originating in psychometrics
- The data used for multidimensional scaling (MDS) are dissimilarities between pairs of objects
- MDS transforms a dataset into a dataset with lower number of dimensions, keeping the distances between the points
  - The main objective of MDS is to represent the dissimilarities as distances between points in a low dimensional space such that the distances correspond as closely as possible to the dissimilarities
- Keeping the distances it allows us to reasonably preserve patterns and clusters
- The coordinates of the new points in the lower dimension no longer have „business” value and are dimensionless
  - The value is carried by the shape of the scatterplot and by the relative distances between points

MDS

- Proximities indicate the overall similarity (or dissimilarity) of the elements in the data
- MDS will look for a spatial configuration of the elements so that the distances between the elements match their proximities as closely as possible
- Often the data are arranged in a square matrix — the proximity matrix
- There are two major groups of methods for deriving proximities:
  - Direct: subjects might either assign a numerical similarity or dissimilarity value to each pair of objects or provide a ranking of the pairs with respect to their similarity or dissimilarity. Both approaches are direct methods of collecting proximity data
  - Indirect: does not require that a subject assigns a numerical value to the elements of the proximity matrix directly. Rather, the proximity matrix itself is derived from other measures. An example of this is data from confusion matrices or from correlation matrices

MDS

- We look for a projection, which preserves inter-point distances
- Assume the appropriate distance measure (does not have to be Euclidean)
- x_i - point in N dimensions
- y_i - corresponding point in $\mathrm{n} < N$ dimensions
- Let us work with Euclidean model $d_{ij} = \sqrt{\sum (x_{il} - x_{jl})^2}$
- It is invariant to translations, rotations or scalings
- Define the distance in a new space $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
- Define the objective (STRESS) function that we want to minimize then, for instance, $STRESS = \sqrt{\frac{\sum_{i<j}(D_{ij} - d_{ij})^2}{\sum_{i<j}(D_{ij})^2}}$ where D_i is the distance in the original dataset and d_i comes from the reduced one

Isomap

Isomap

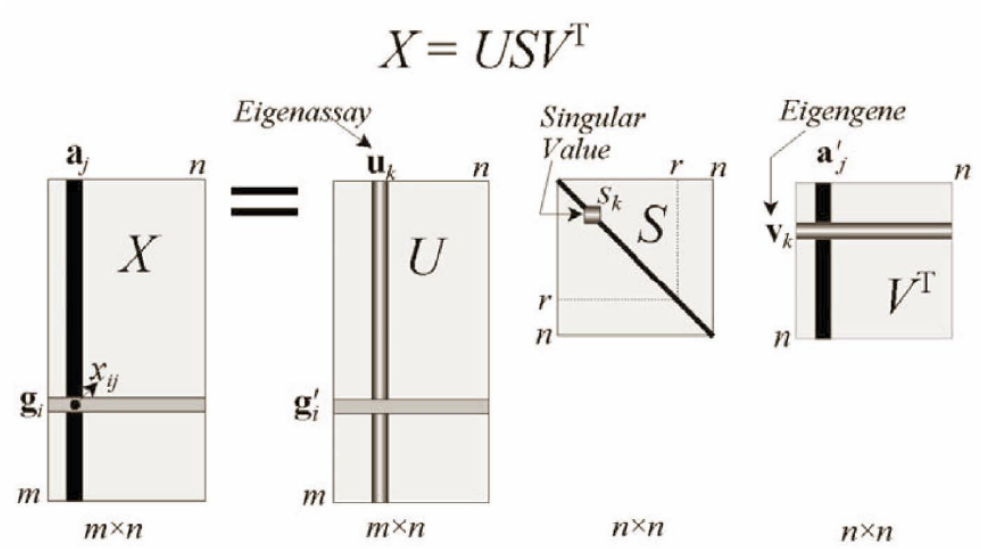
- We look for a projection onto a nonlinear manifold
  - We construct the neighborhood graph G for all x_i, x_j if distance $(x_i, x_j) < \epsilon$ than add edge (x_i, x_j) to G
  - Then we compute the shortest distances along graph $\delta_G(x_i, x_j)$
  - The selection of the algorithm is up to us
  - We apply the multidimensional scaling to $\delta_G(x_i, x_j)$

Singular Value Decomposition (SVD)

SVD

- It includes efficient algorithms (i.a. Matlab MVD)
- It is sufficient also for very large dimensional data
- Some implementations find only just top N eigenvectors

SVD



Data X, one row per data point

US gives coordinates of rows of X in the space of principle components

S is diagonal, $S_k > S_{k+1}$, S_k^2 is kth largest eigenvalue

Rows of V^T are unit length eigenvectors of X^TX.

If cols of X have zero mean, then $X^TX = c\,\Sigma$ and eigenvects are the Principle Components

[from Wall et al., 2003]

SVD

- In order to generate principle components
  - Substract mean $\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x^n$ from each data point to create zero-centered date
  - Create matrix X with one row vector per data point
  - Solve SVD: $X = USV^T$
  - Output principle components: columns of V
    - Eigenvectors in V are sorter from largest to smallest eigenvalues
    - S is diagonal

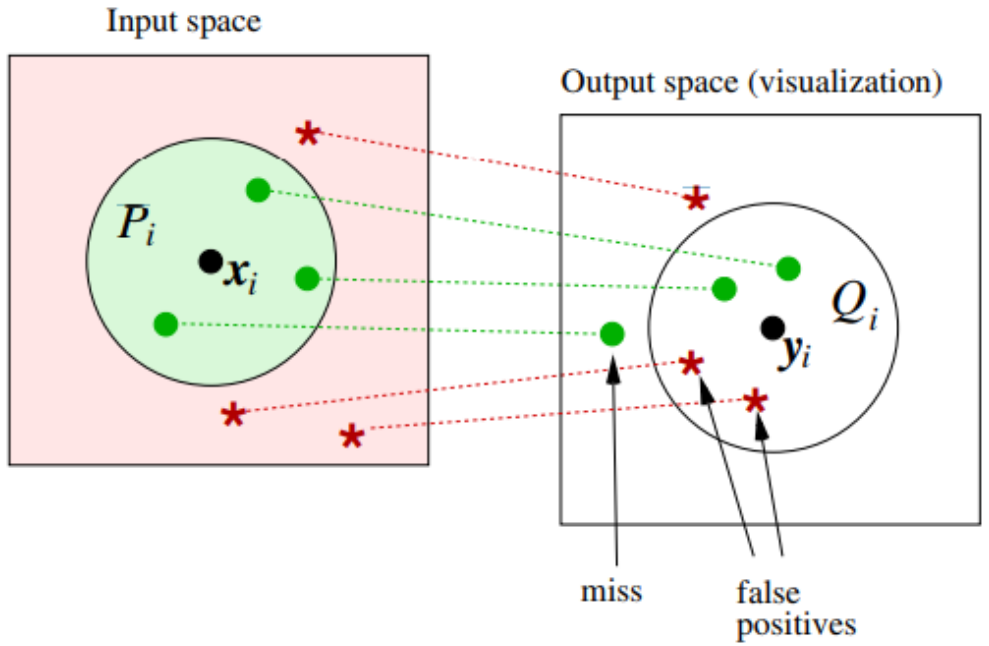
SVD

- In order to project a point into PC coordinates
  - $V^T x$
  - If x_i is i-th row of data matrix X, then
    - i-th row of $US = V^T x_i^T$
    - $(US)^T = V^T X^T$
    - When you aim to project a column vector x to M-dimensional PC subspace, take the first M coordinates of $V^T x$

Dimension reduction and visualization

DR and visualization (Imperial, 2019)

- Kaski & Jaakko 2011
  - a high-dimensional data set cannot, in general, be faithfully represented in a lower-dimensional space
  - it turns out that under a specific but general goal the choice can be expressed as an interesting tradeoff
  - it can miss some similarities (it can place similar points far apart as false negatives) or it can bring dissimilar data points to close together as false positives



Thank you!