4. Dimensionality Reduction

Sometimes, before fitting the model, whether it is a supervised or unsupervised learning problem, it is necessary to reduce the data's dimension (number of features). There are several reasons why we are interested in reducing dimensionality as a previous preprocessing step:

- In most learning algorithms, the complexity depends on the number of features, d, and the number of samples.
- When one feature is decided to be unnecessary, we save the cost of analyzing it.
- When data can be explained with fewer features, we get a better idea about the process that underlies the data, and this allows knowledge extraction.
- When data can be represented in a few dimensions without loss of information, it can be plotted and analyzed visually for structure and outliers.

There are two main methods for reducing dimensionality:

- **Feature selection**, we are interested in finding k features ($k < d$) that give us the most information, and we discard the others.

- **Feature extraction**, we are interested in finding a new set of k features resulting from the combination of the original ones.

4.1 Subset Selection

Unsupervised learning: Dimensionality reduction
Variable type: all

In subset selection, we are interested in finding the best subset of features. There are some simple approaches:

- Remove features with variance 0.

- Forward selection, we start with no variables and add them one by one, at each step adding the one that decreases the error the most until any further addition does not decrease the error (or decreases it only slightly).

- Backward selection, we start with all variables and remove them one by one, at each step removing the one that decreases the error the most or increases it only slightly.

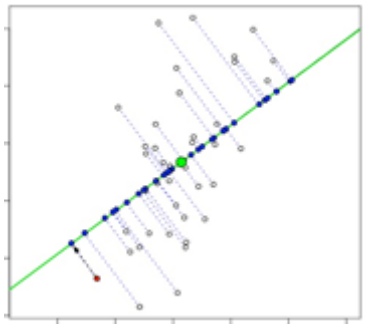
- Feature selection using optimization algorithms.

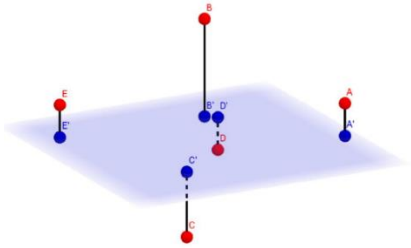
4.2 Principal Component Analysis (PCA)

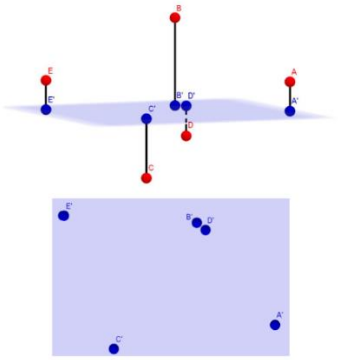
Unsupervised learning: Dimensionality reduction
Variable type: continuous

We are interested in finding a mapping from the inputs in the original *d*-dimensional space to a new k-dimensional space (*k* < *d)*, with minimum loss of information. PCA uses a linear projection.

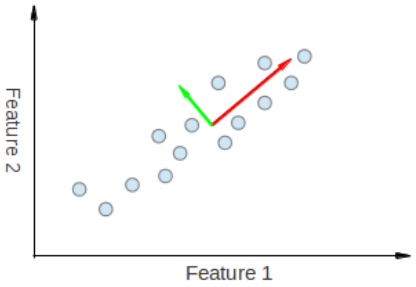
Examples in 2d and 3d.







Principal components analysis (PCA) projects the data **maximizing the variance in the new space**.

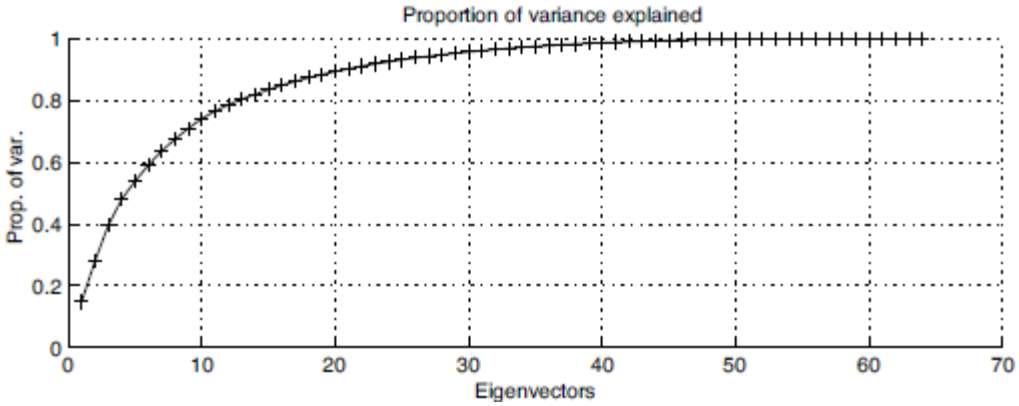


The projection vectors are calculated as the eigenvectors of the covariance matrix of the data:
- The first vector is the eigenvector with the largest eigenvalue.
- The second vector is the eigenvector with the second largest eigenvalue.
- Etcetera

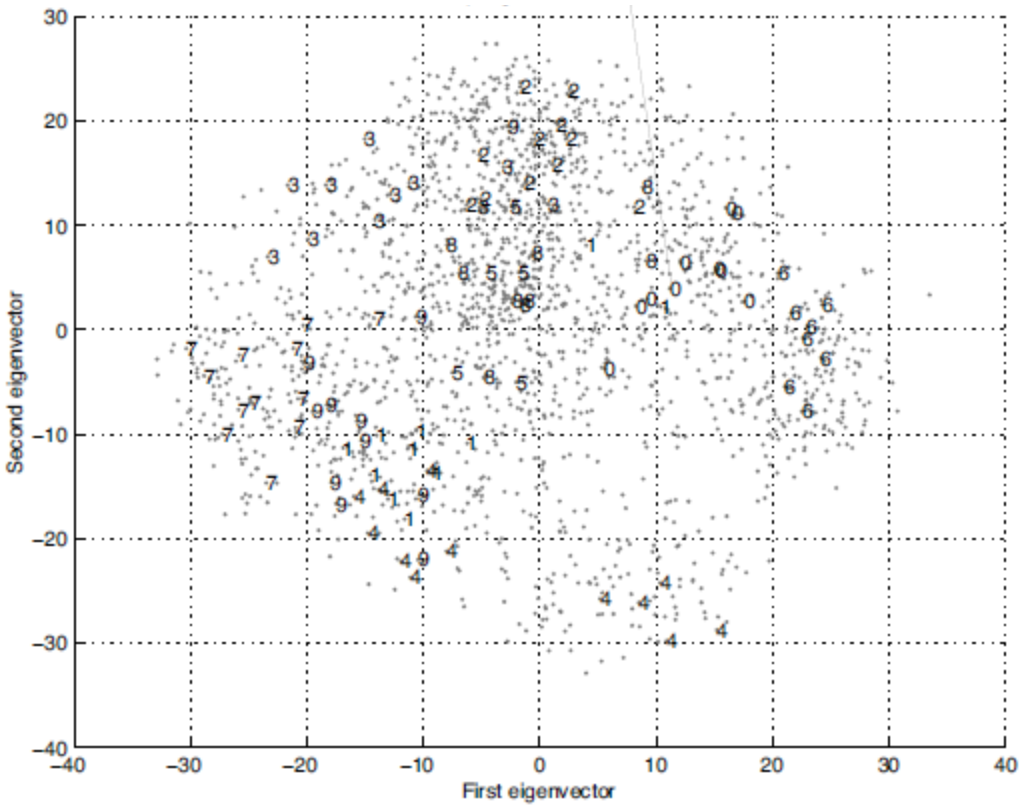
There are d eigenvectors. The new data is obtained projecting the data into the first k eigenvectors.

$$D' = D * V$$

An advantage of PCA is that we can measure the proportion of variance explained by the addition of each feature.



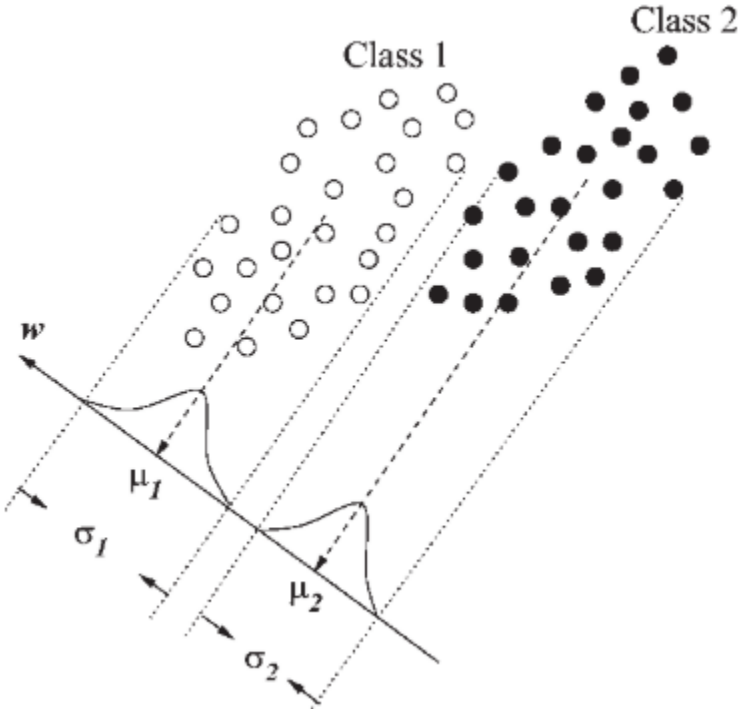
If the first two principal components explain a large percentage of the variance, we can do visual analysis: we can plot the data in a two dimensional space and search visually for structure, groups, outliers, normality, and so forth.



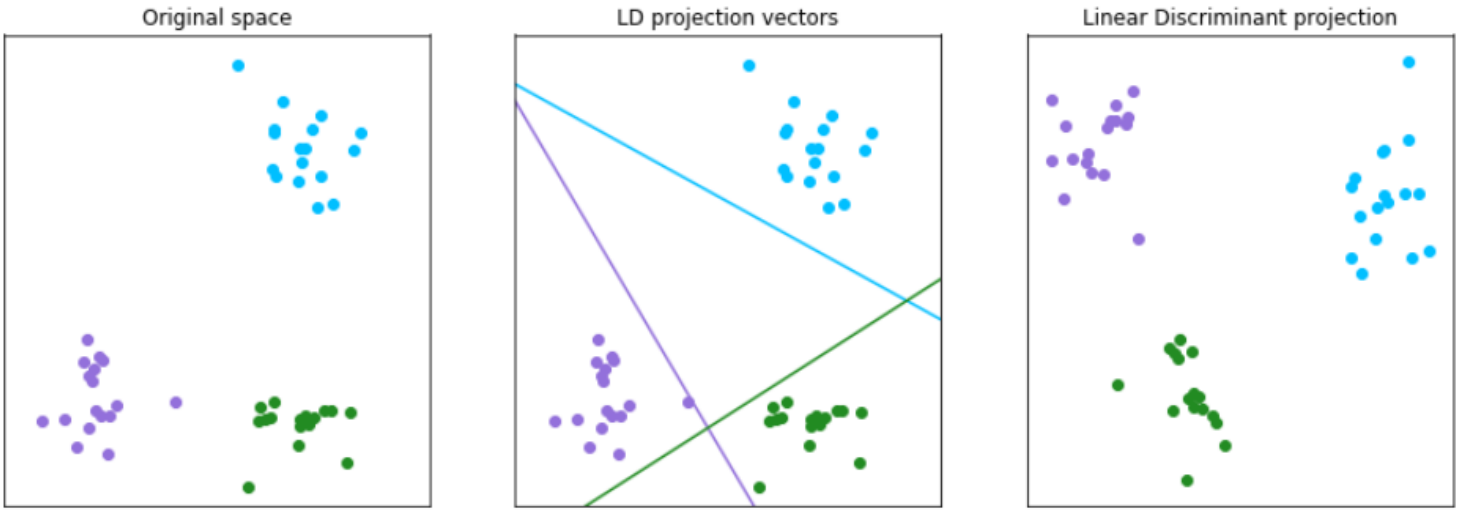
4.3 Linear Discriminant Analysis

Supervised learning: Dimensionality reduction for classification
Variable type: continuous

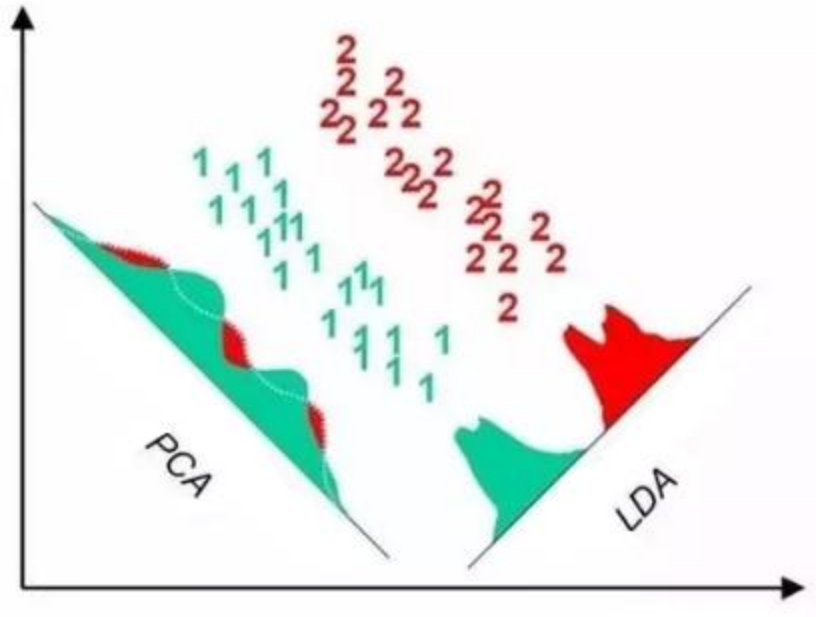
Linear discriminant analysis (LDA) is a **supervised** method for dimensionality reduction **for classification** problems. It was designed to project the data of two classes maximixing the difference between the centroids and minimizing the variance of groups in the new space. It assumes the data of each group follows a multivariate normal distribution and the variances among groups are the same.

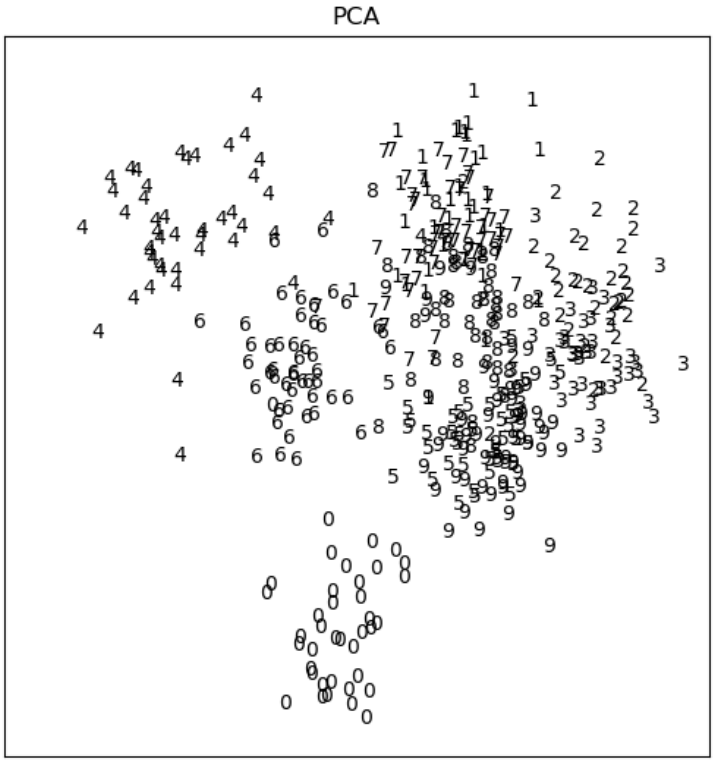


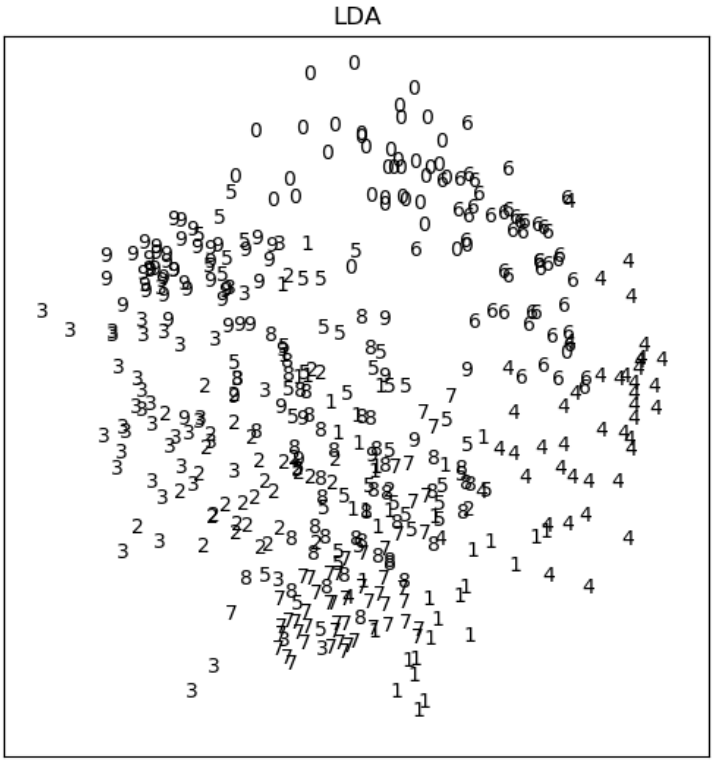
Scikit learn implementation performs LDA one feature at a time calculating the projection vector that better separates the data of the specific class and the rest. Then, it sort the projection vectors based on the quality of the separation. Finally, it projects the data on the projection vectors (one per each class). The dimensionality of the new data is the minimum between the number of features and number of classes.



Difference between Principal Component Analysis and Linear Discriminant Analysis.







4.4 Multidimensional Scaling

Unsupervised learning: Visualization
Variable type: all

Let us say for N points, we are given the distances between pairs of points, d_{ij}, for all $i, j = 1,2,\dots,N$. We do not know the exact coordinates of the points, their dimensionality, or how the distances are calculated. Multidimensional scaling (MDS) is the method for placing these points in a dimensional space, for example two-dimensional, such that the Euclidean distance between them there is as close as possible to d_{ij}. MDS solves an optimization problem which minimices the stress:

$$Stress = \sqrt{\frac{\sum_{i<j}\left(d_{ij} - \widehat{d_{ij}}\right)^2}{\sum_{i<j} d_{ij}^2}}$$

J.B. Kruskal proposed in [Multidimensional scaling by optimizing goodness of fit o a nonmetric hypothesis (1964)] an interpretation of stress:

Stress	20%	10%	5%	2.5%	0%
Goodness of fit	poor	fair	good	excellent	perfect

Note: the python implementation of MDS uses as stress:

$$Stress = \sum_{i<j}\left(d_{ij} - \widehat{d_{ij}}\right)^2$$

MDS can be used for dimensionality reduction by calculating pairwise Euclidean distances in the d-dimensional x space and giving this as input to MDS, which then projects it to a lower-dimensional space so as to preserve these distances.