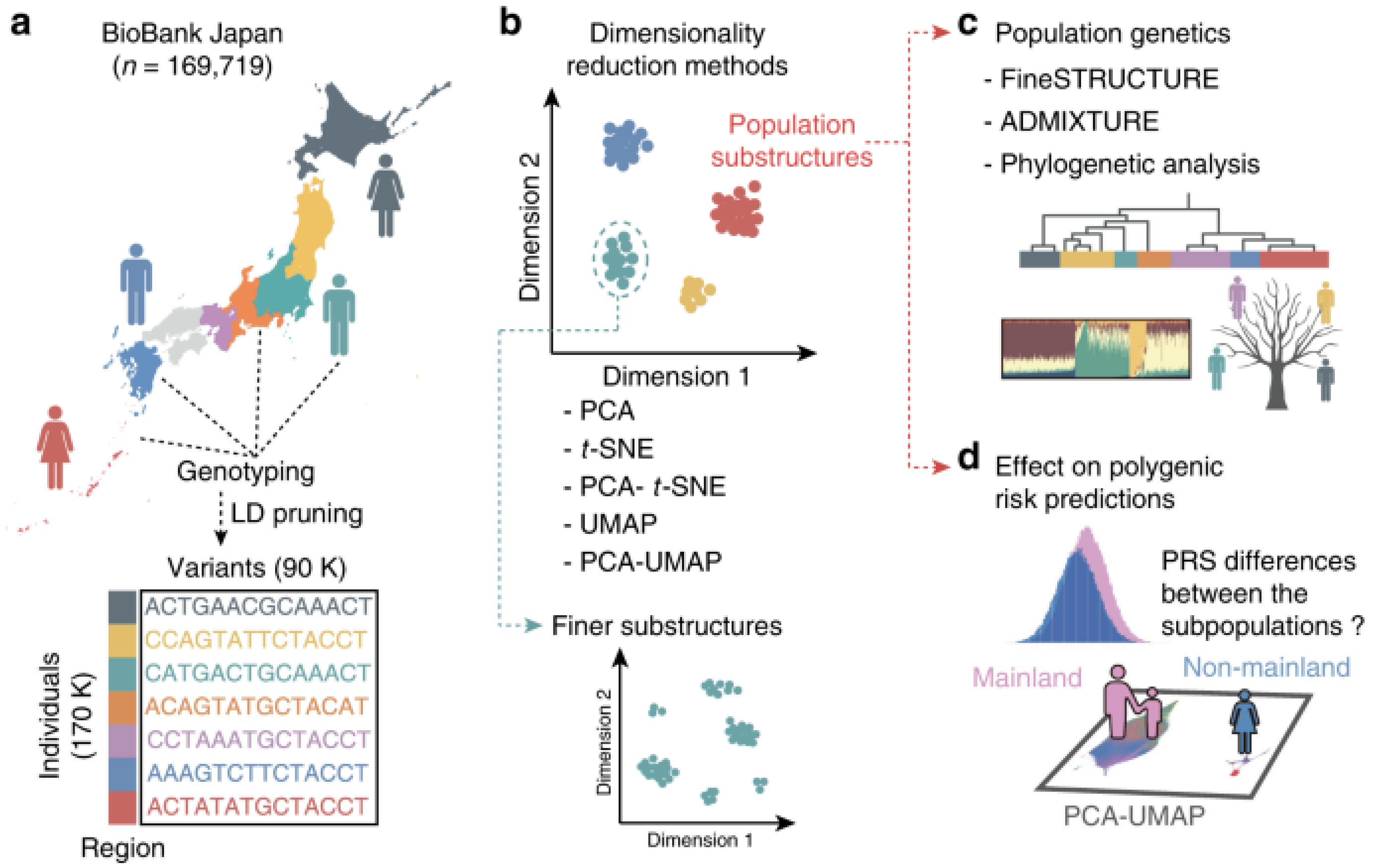
Linear methods for dimensionality reduction (Theory)

Machine Learning



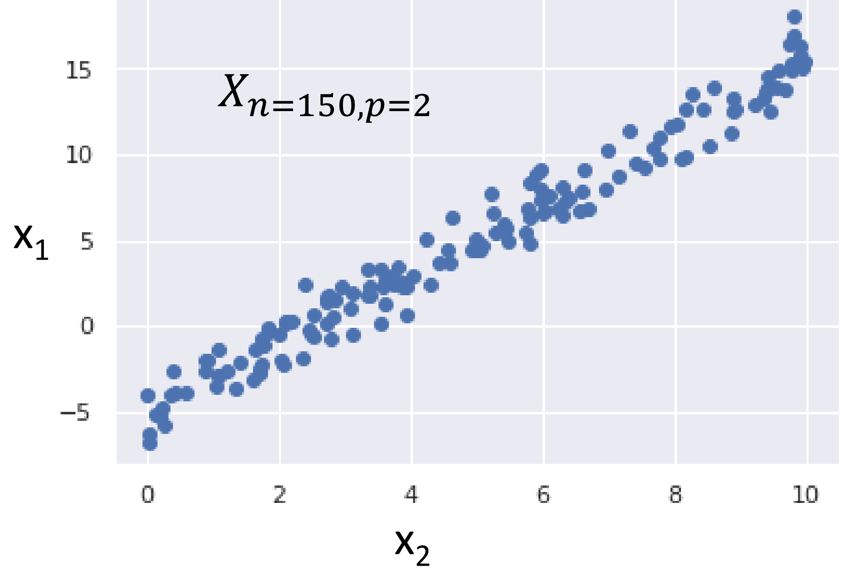
Toni Monleón-Getino, amonleong@ub.edu
04/25/2023

Linear methods for dimensionality reduction

- What is dimensionality reduction?
- Linear methods and Non-linear methods.
- Linear dimensionality reduction.
- Principal Component Analysis (PCA)
- Factor Analysis (FA),
- Linear Discriminant Analysis (LDA)
- Truncated Singular Value Decomposition (SVD)

What is dimension reduction?

- 1. Visualizing data is one of the most important step in the data analysis (if not the most!).
- 2) The right visualization method may reveal problems with the experimental data that can render the results from a standard analysis completely useless.
- 3) Plot that reveal relationships between columns or between rows are more complicated due to the high dimensionality of data ($X_1, \ldots, X_p$).
- 4) We would describe powerful techniques for exploratory data analysis based on dimensions reduction. The general idea is to reduce the dataset (X_{nxp}) to have fewer dimensions, yet approximately preserve important properties, such us the distance between samples.



Not easy to visualize multivariate data

For exemple: to compare each of the 150 samples to each other in this exemple (x_1 vs x_2) we can use an scatterplot, but is impossible since points are high dimensional ($x_1, x_2, \ldots, x_p$)

Data análisis for the Life Sciences with R: Irizarri & Love, 2017. CRC PRESS

When we use dimensionality reduction?

- 1. Not easy to extract useful information from the multivariate data.
- 2) Many bivariate plots are needed.
- 3) Bivariate plots, however, mainly represent correlations between variables (not samples)
- Some statistics for data matrix (X_{nxp}):

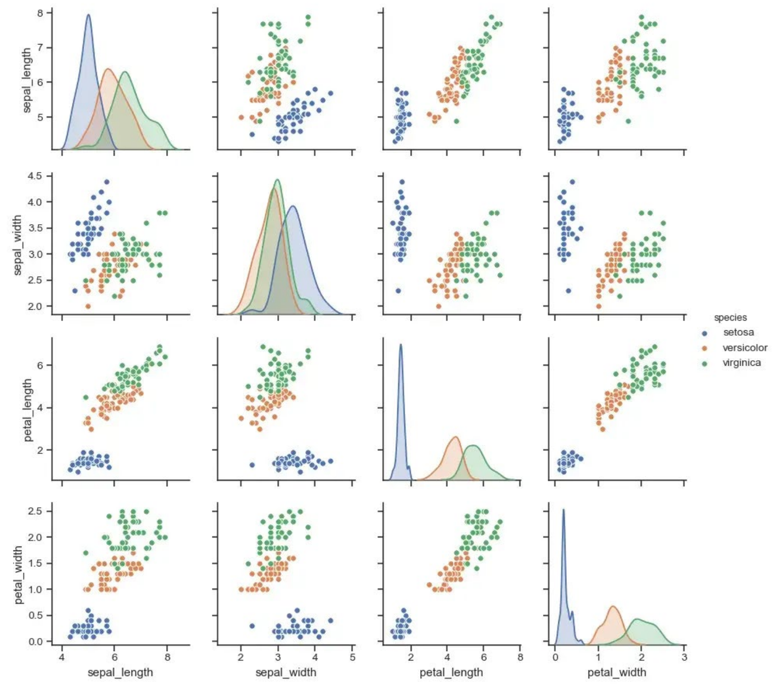
population
$$\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y}$$

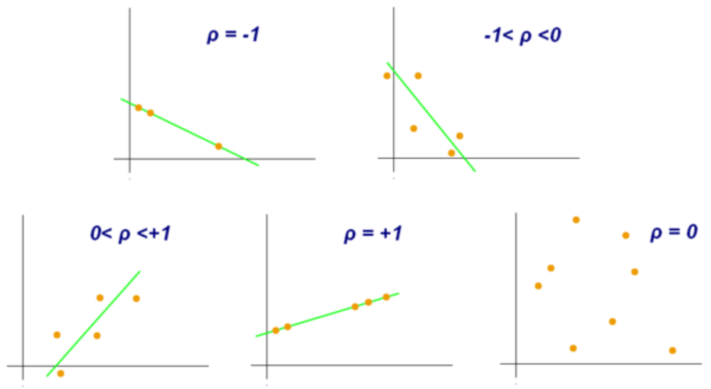
sample
$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Variance-Covariance matrix
$$\Sigma = Cov(X)$$
$$\mathbf{Cov}(\mathbf{X}) = \begin{pmatrix} Var[X_1] & \cdots & Cov(X_1, X_n) \\ \vdots & & \vdots \\ Cov(X_n, X_1) & \cdots & Var[X_n] \end{pmatrix}.$$

Correlation matrix
$$Corr(X) = [diag(\Sigma)]^{-1/2}\, \Sigma\, [diag(\Sigma)]^{-1/2}$$

https://en.wikipedia.org/wiki/Pearson_correlation_coefficient



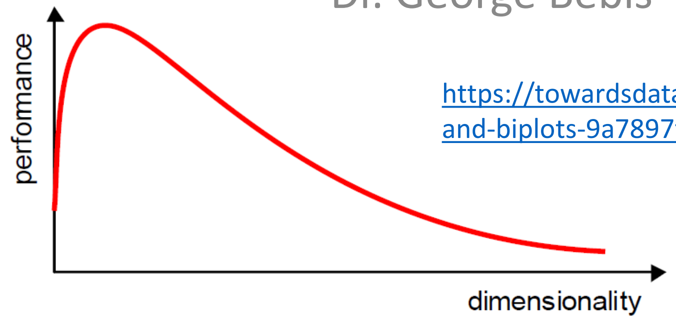


Not easy to visualize multivariate data

Curse of Dimensionality

Pattern Recognition
Dr. George Bebis

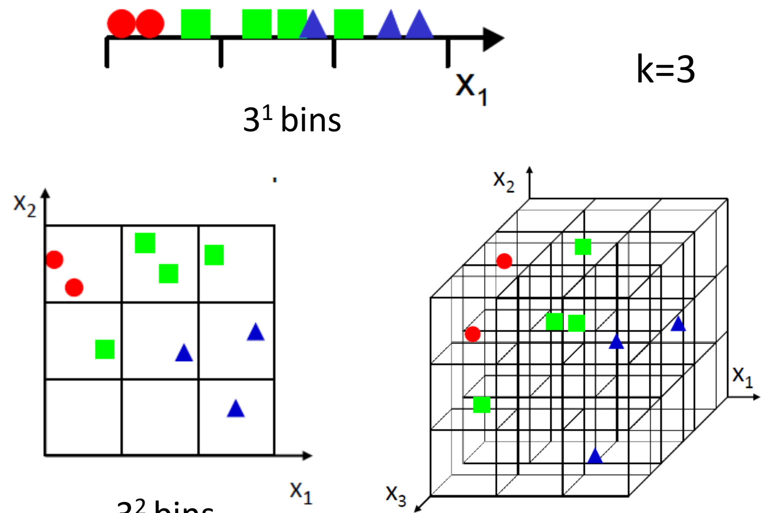
- Increasing the number of p features in X_{nxp} will not always improve classification accuracy.



https://towardsdatascience.com/what-are-pca-loadings-and-biplots-9a7897f2e559

- In practice, the inclusion of more features might actually lead to worse performance (we say multicollinearity or redundancy in statistics).

- The number of training examples required increases exponentially with dimensionality **d** (i.e., $k^{\mathbf{d}}$).



3^1 bins

3^2 bins

3^3 bins

k: number of bins per feature

What is dimensionality reduction?

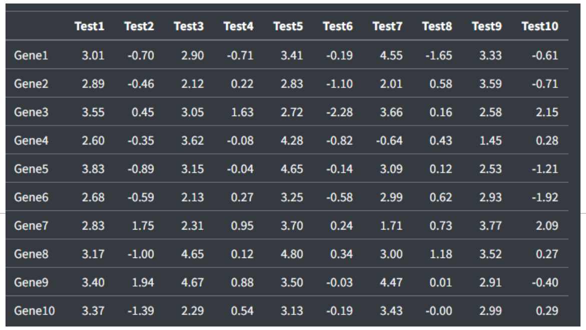
Brief summary

- Dimensionality reduction helps to reduce the number of attributes or features of a dataset X_{nxp}(e.g. thousand of expression genes in a hundred tests/individuals) by selecting important features or combining features to capture variance in a dataset.
- It is often used to improve the performance of machine learning models and to aid visualization.
- Dimensionality reduction (or dimension reduction) is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension.
- Working in high-dimensional spaces can be undesirable for many reasons; raw data are often sparse as a consequence of the curse of dimensionality, and analyzing the data is usually computationally intractable (hard to control or deal with).
- Dimensionality reduction is common in fields that deal with large numbers of observations and/or large numbers of variables, such as signal processing, speech recognition, neuroinformatics, and bioinformatics.
- Dimensionality reduction methods and approaches:
    - Methods are commonly divided into linear and nonlinear approaches.
    - Approaches can also be divided into feature selection and feature extraction.
    - Dimensionality reduction can be used for noise reduction, data visualization, cluster analysis, or as an intermediate step to facilitate other analyses (e.g regression, classification).

See this interactive example: http://www.graphbio1.com/en/ (PCA) for genetic expresion

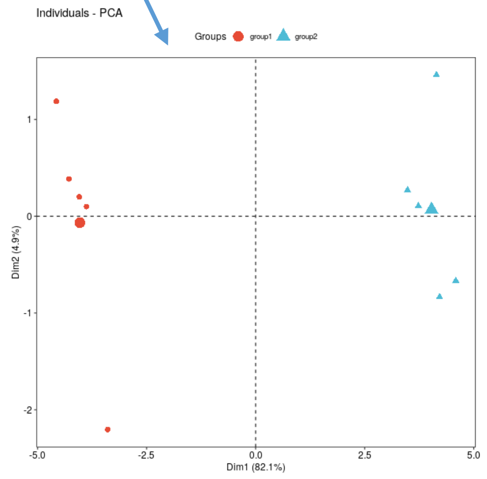
http://www.graphbio1.com/en/

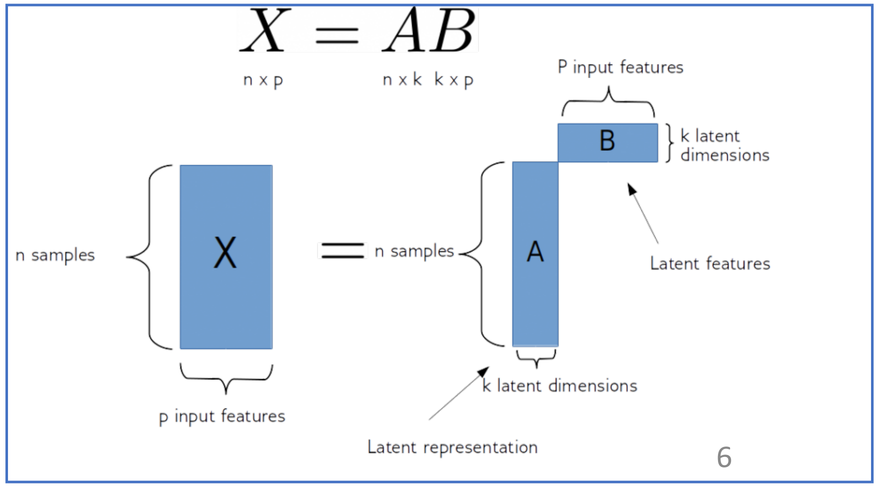
X_{nxp}



	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10
Gene1	3.01	-0.70	2.90	-0.71	3.41	-0.19	4.55	-1.65	3.33	-0.61
Gene2	2.89	-0.46	2.12	0.22	2.83	-1.10	2.01	0.58	3.59	-0.71
Gene3	3.55	0.45	3.05	1.63	2.72	-2.28	3.66	0.16	2.58	2.15
Gene4	2.60	-0.35	3.62	-0.08	4.28	-0.82	-0.64	0.43	1.45	0.28
Gene5	3.83	-0.89	3.15	-0.04	4.65	-0.14	3.09	0.12	2.53	-1.21
Gene6	2.68	-0.59	2.13	0.27	3.25	-0.58	2.99	0.62	2.93	-1.92
Gene7	2.83	1.75	2.31	0.95	3.70	0.24	1.71	0.73	3.77	2.09
Gene8	3.17	-1.00	4.65	0.12	4.80	0.34	3.00	1.18	3.52	0.27
Gene9	3.40	1.94	4.67	0.88	3.50	-0.03	4.47	0.01	2.91	-0.40
Gene10	3.37	-1.39	2.29	0.54	3.13	-0.19	3.43	-0.00	2.99	0.29

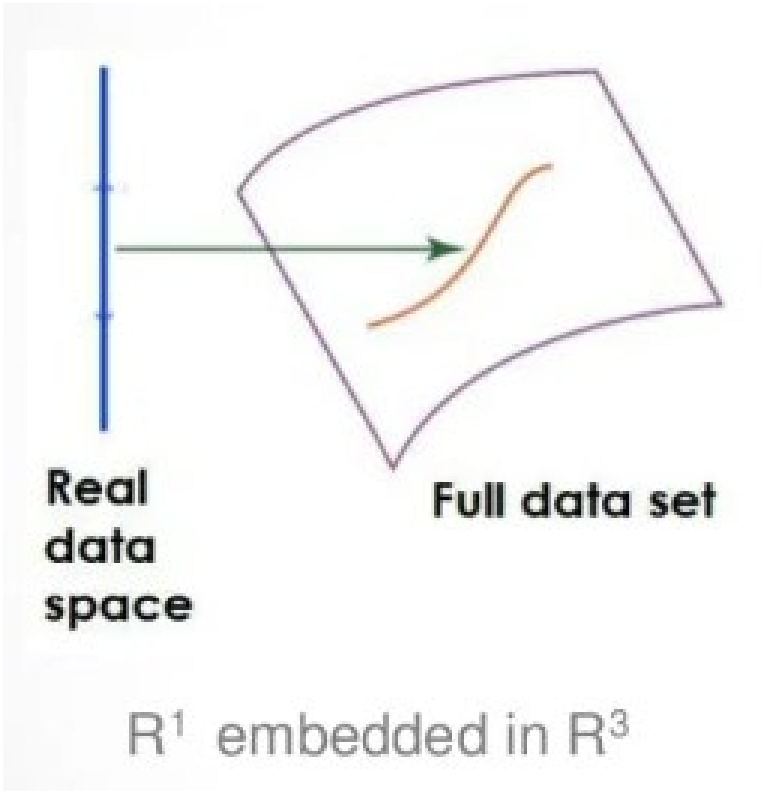
PCA First exemple: GENES VS TEST





Dimension reduction theory: introduction

- More data (X) is collected on an outcome than is useful.
- From full X (full set of data) only a subset is important to predicting and outcome.
- Maybe more variables are correlated (believe that there is some redundancy in those variables).
- This can be conceptualized as a low dimensional shape stuck (embedded) within the larger-dimensional data shape.
- Some common linear methods:
  - 1-Principal Component Analysis (PCA)
  - 2-Factor Analysis (FA)
  - 3-Singular Value Descomposition (SVD)
  - 4-Linear Discriminant Analysis (LDA)
  - 5-Independent Component analysis (ICA)



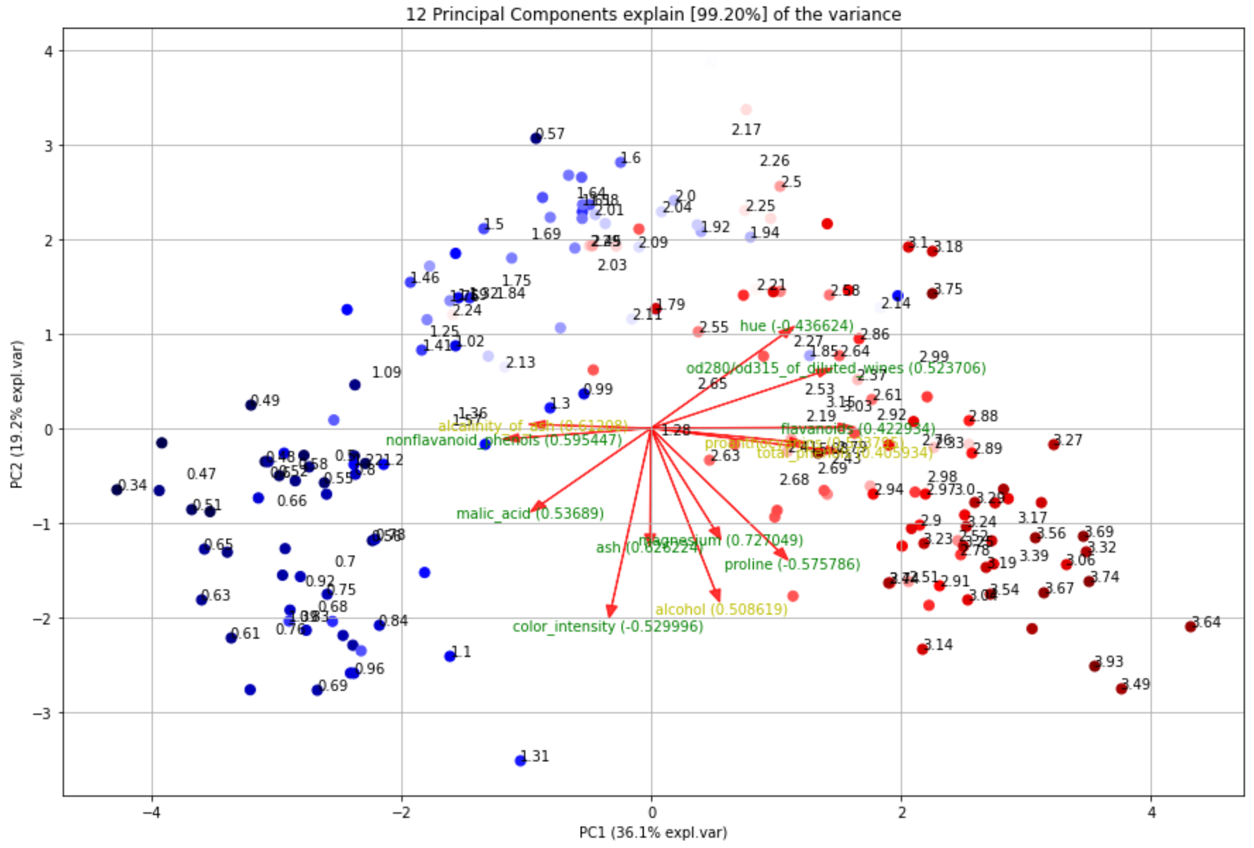
https://www.slideshare.net/ColleenFarrelly/review-of-methods-for-dimension-reduction

PCA example

- We will import the ***wine dataset*** and create a data frame that contains 178 samples, with 13 features and 3 wine classes (see Python script in https://towardsdatascience.com/what-are-pca-loadings-and-biplots-9a7897f2e559) .

- Let's start deeper investigating the flavanoids variable that contributes to PC1. The angle of the flavanoid's arrow is positive, almost horizontal, and has a score of 0.422. This suggests that some of the variances in the x-axis should come from the flavonoids variable. Or in other words, if we would color the samples using the flavanoids values, we should expect to find samples with low values on the left side and high values on the right side.

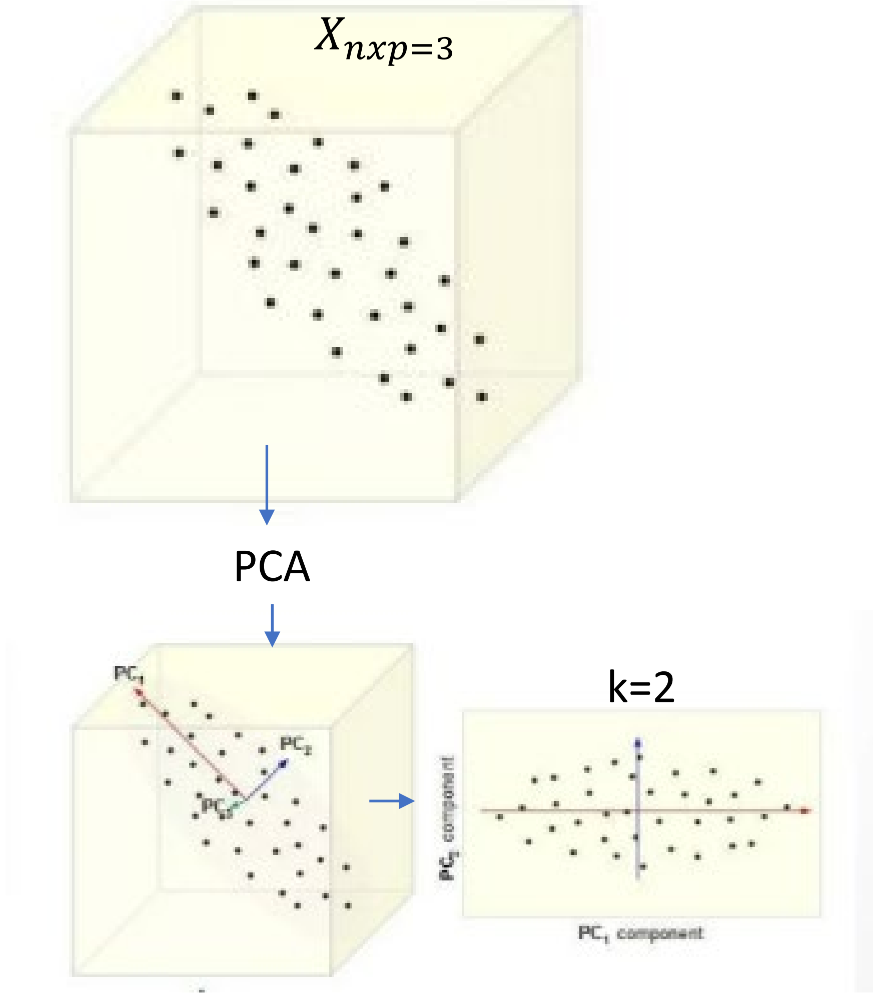
https://towardsdatascience.com/what-are-pca-loadings-and-biplots-9a7897f2e559



Biplot of the wine dataset (samples = 178 and variables = 13). Samples are colored on class information (class=cultivars, type of genetic variety)

Dimension reduction theory: PCA

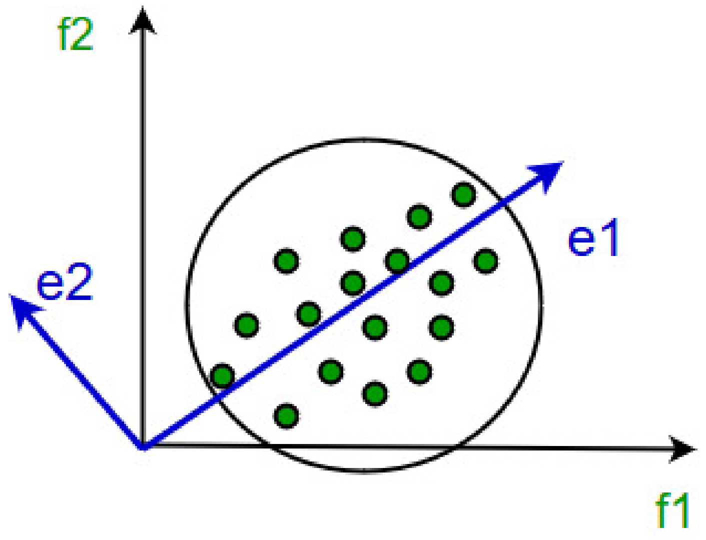
- The traditional method is known as Principal Component Analysis (PCA)
- Assumes subspace of useful data is linear
- Finds a transformation that reduces dimension while accounting for as much variance as possible:
- Because of redundancy between variables in X, you believe that it should be possible to reduce the observed variables into a smaller number of principal components (artificial variables) that will account for most of the variance in the observed variables
- PCA has been proven to work on a real data
- Has been extended to nonlinear assumptions (Kernel PCA)



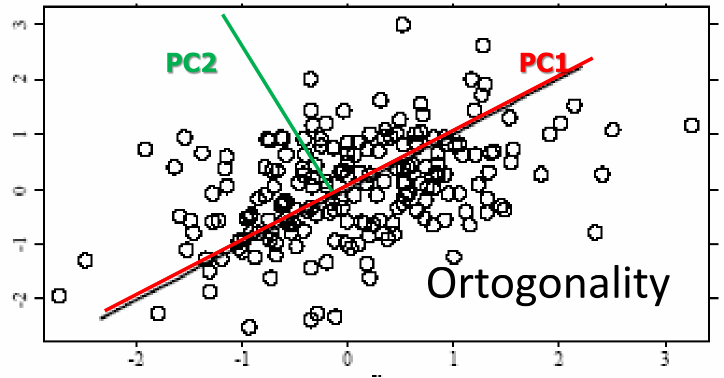
https://www.slideshare.net/ColleenFarrelly/review-of-methods-for-dimension-reduction

Dimension reduction theory: PCA

- PCA method was introduced by Karl Pearson (1857 -1936). It works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum.
- It involves the following steps:
  - Construct the covariance matrix (Σ) of the data.
  - Compute the eigenvectors (A) of this matrix.
  - Eigenvectors corresponding to the largest eigenvalues (λ) are used to reconstruct a large fraction of variance of the original data.
- Hence, we are left with a lesser number of eigenvectors, and there might have been some data loss in the process. But, the most important variances should be retained by the remaining eigenvectors (A).
- Advantages of Dimensionality Reduction with PCA:
  - It helps in data compression, and hence reduced storage space.
  - It reduces computation time.
  - It also helps remove redundant features, if any.
- Disadvantages of Dimensionality Reduction with PCA:
  - It may lead to some amount of data loss.
  - PCA tends to find linear correlations between variables, which is sometimes undesirable.
  - PCA fails in cases where mean and covariance are not enough to define datasets.
  - We may not know how many principal components to keep- in practice, some thumb rules are applied.
  - Fails in case of non linearity



Seeks a projection that preserves as much information in the data as possible.



https://www.geeksforgeeks.org/dimensionality-reduction/

Dimension reduction theory: PCA - Linear Algebra

- Let X be a random vector (X_{nxp}), the principal component can be defined as a linear combination of optimally-weighted observed variables in X.
- Based on how subject scores on a principal component are computed.
- Theory behind PCA is linear Algebra, starting from matrices Cov(X)=Σ , Corr(X).
- The covariance and correlation matrix are symmetrics.

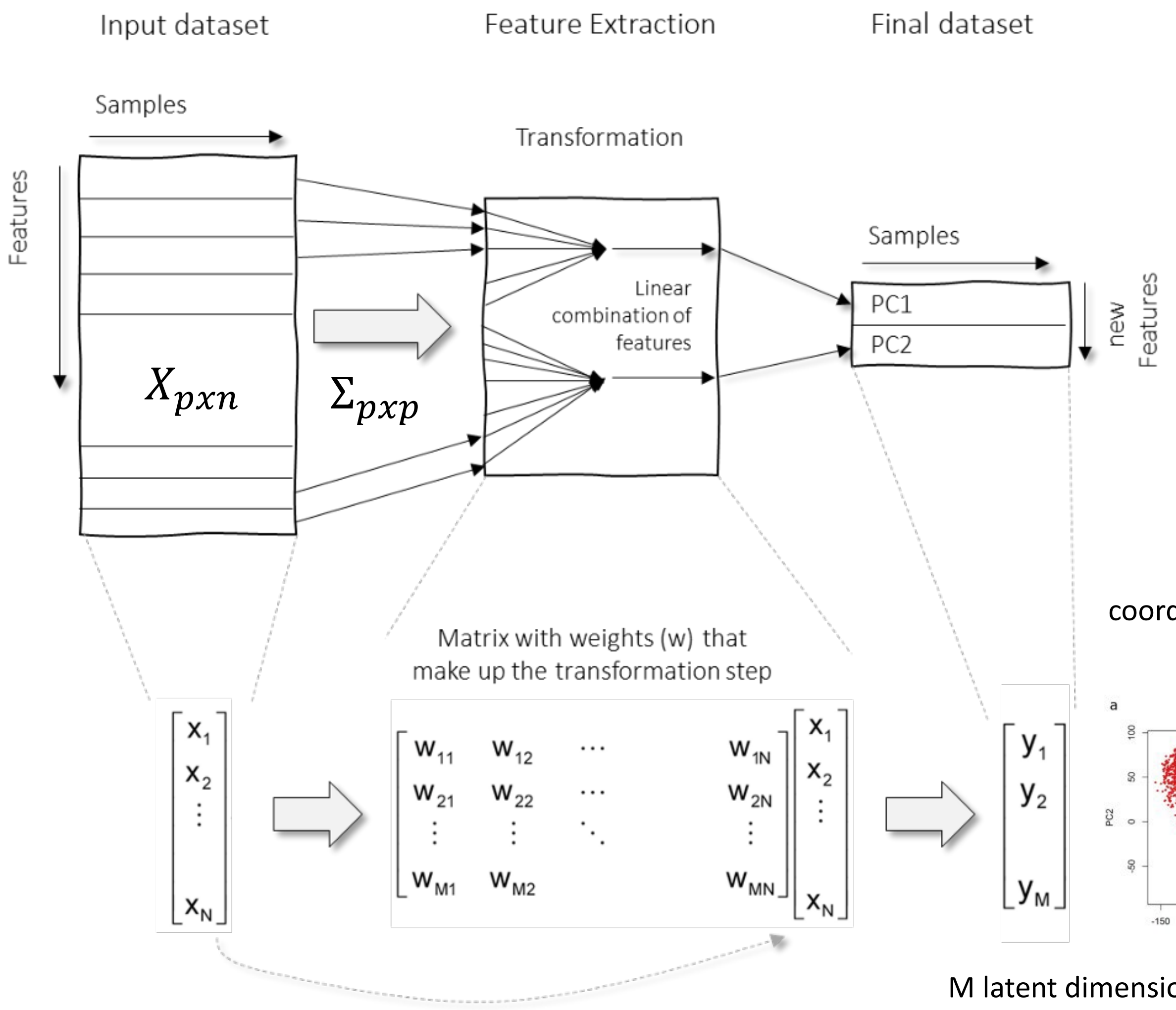
$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

$$\mathbf{Cov}(\mathbf{X}) = \begin{pmatrix} Var[X_1] & \cdots & Cov(X_1, X_n) \\ \vdots & & \vdots \\ Cov(X_n, X_1) & \cdots & Var[X_n] \end{pmatrix}.$$

https://www.slideshare.net/ColleenFarrelly/review-of-methods-for-dimension-reduction

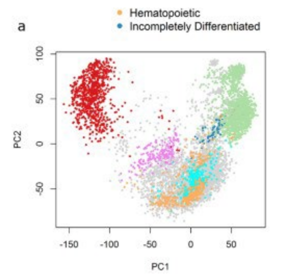
Dimension reduction theory: PCA - Linear Algebra

- Feature extraction approaches can reduce the number of dimensions and at the same time minimize the loss of information.
- To do this, we need a transformation function; y=f(x).
- In the case of PCA, the transformation is limited to a linear function which we can rewrite as a set of weights that make up the transformation step;
- y=Wx, where W are the weights, x are the input features, and y is the final transformed feature space.
- See on the right hand a schematic overview to demonstrate the transformation step together with the mathematical steps.
- See the mathematical annex (at the end of this presentation)
- Later on, the different steps in a schematic way



Input dataset
Feature Extraction
Final dataset
Samples
Features
Transformation
X_{pxn}
Σ_{pxp}
Linear combination of features
Samples
PC1
PC2
new Features
Matrix with weights (w) that make up the transformation step
x_1 x_2 ⋮ x_N
w_{11} w_{12} ⋯ w_{1N}
w_{21} w_{22} ⋯ w_{2N}
w_{M1} w_{M2} w_{MN}
y_1 y_2 y_M

coordinates



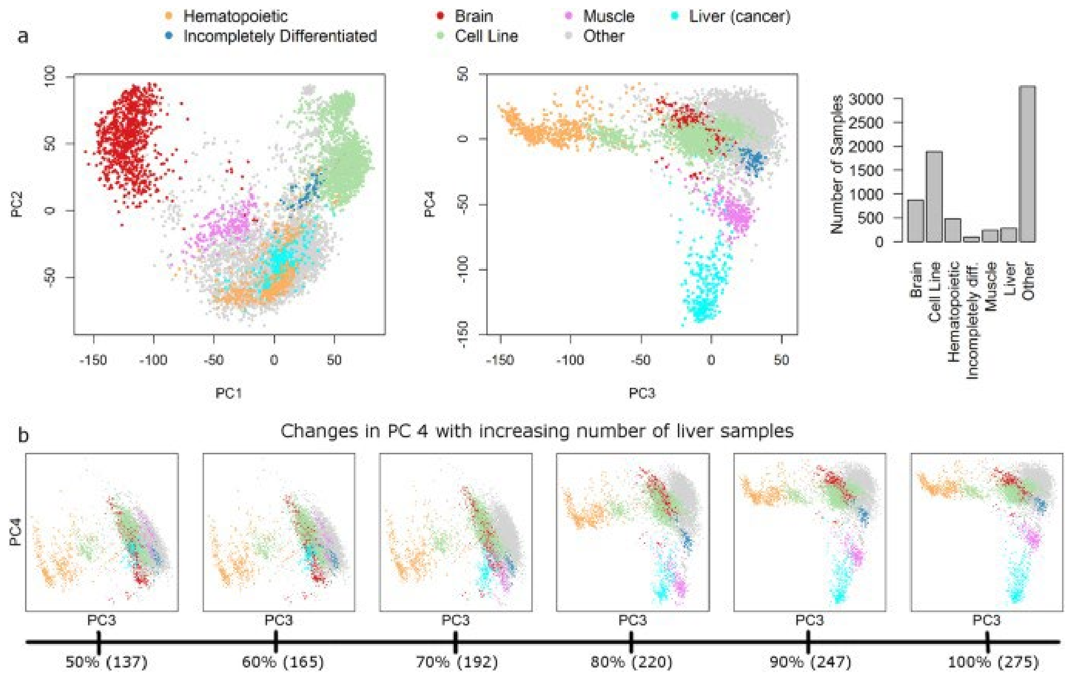
M latent dimensions (usually 2 or 3, PCA1, PCA2, PCA3)

https://towardsdatascience.com/what-are-pca-loadings-and-biplots-9a7897f2e559

PCA-Steps 4 steps

- We are to summarize the PCA process in 4 steps

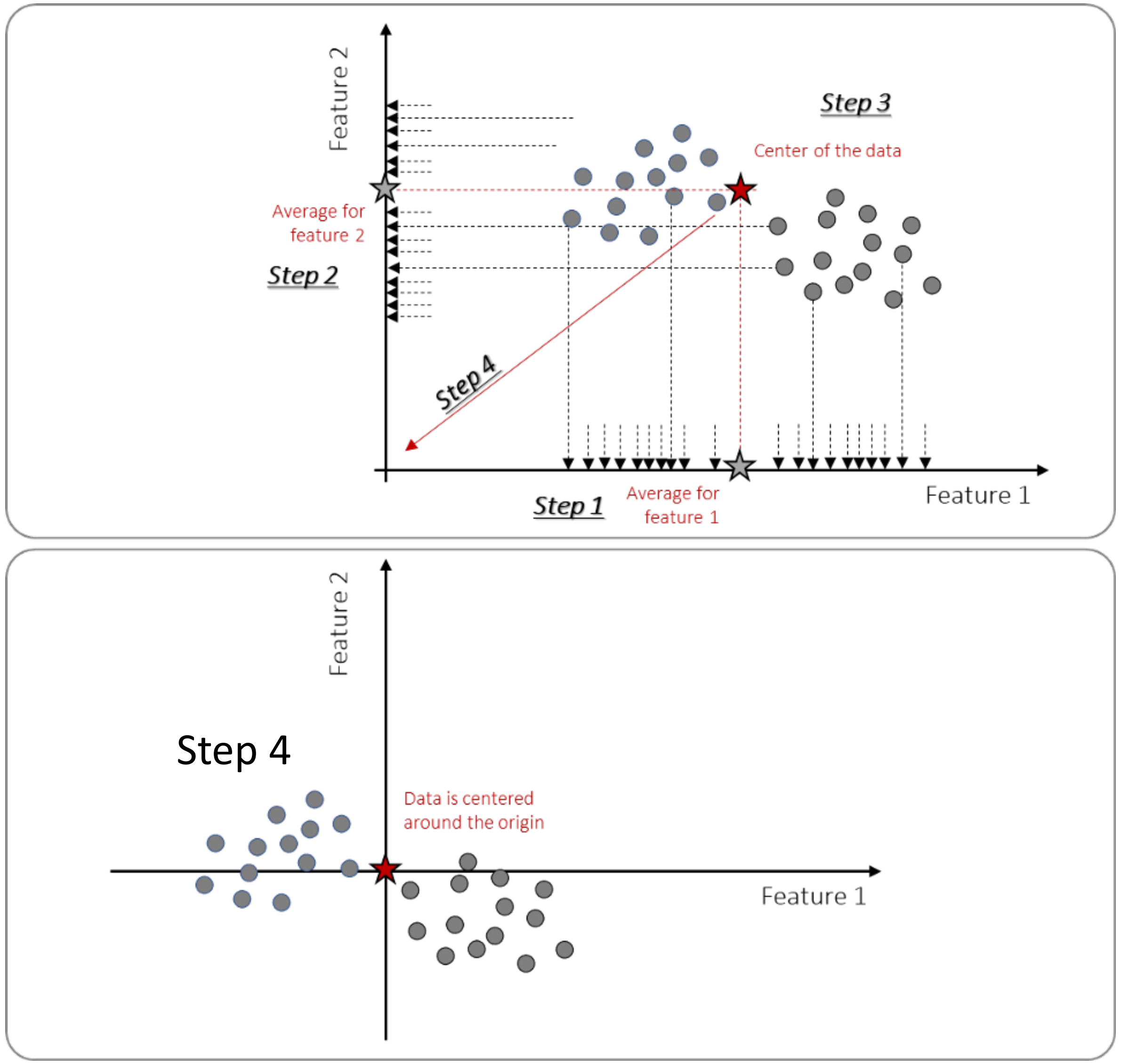
- **STEP 0: STANDARIZATION**
  - Before we do parts 1 to 4, it is crucial to get the data in the right shape by standardization and this should therefore be the very first part.
  - Because we search for the direction with the largest variance, a PCA is very sensitive to variables that have different value ranges or to the presence of outliers.
  - *If there are large differences between the ranges of initial variables, the variables with larger ranges will dominate over those with small ranges.*
  - To prevent this, we need to standardize the range of the initial variables so that each variable contributes equally to the analysis.
  - We can do this by subtracting the mean and dividing it by the standard deviation for each value of each variable.
  - Standardization involves rescaling the features such that they have the properties of a standard normal distribution with a mean of zero and a standard deviation of one.
  - This is also named a z-score standardization for which Scikit-learn has the StandardScaler().
  - Once the standardization is done, all the variables should be on the same scale.



PCA-Steps

Part 1. Center data around the origin.

- The first part is computing the average of the data (illustrated in Figure) which can be done in four smaller steps:
  - First by computing the average per feature **(1 and 2)**, and then the center (**3**).
  - We can now shift the data so that it is centered around the origin(**4**).
  - Note that this transformation step does not change the relative distance between the points but only centers the data around the origin.



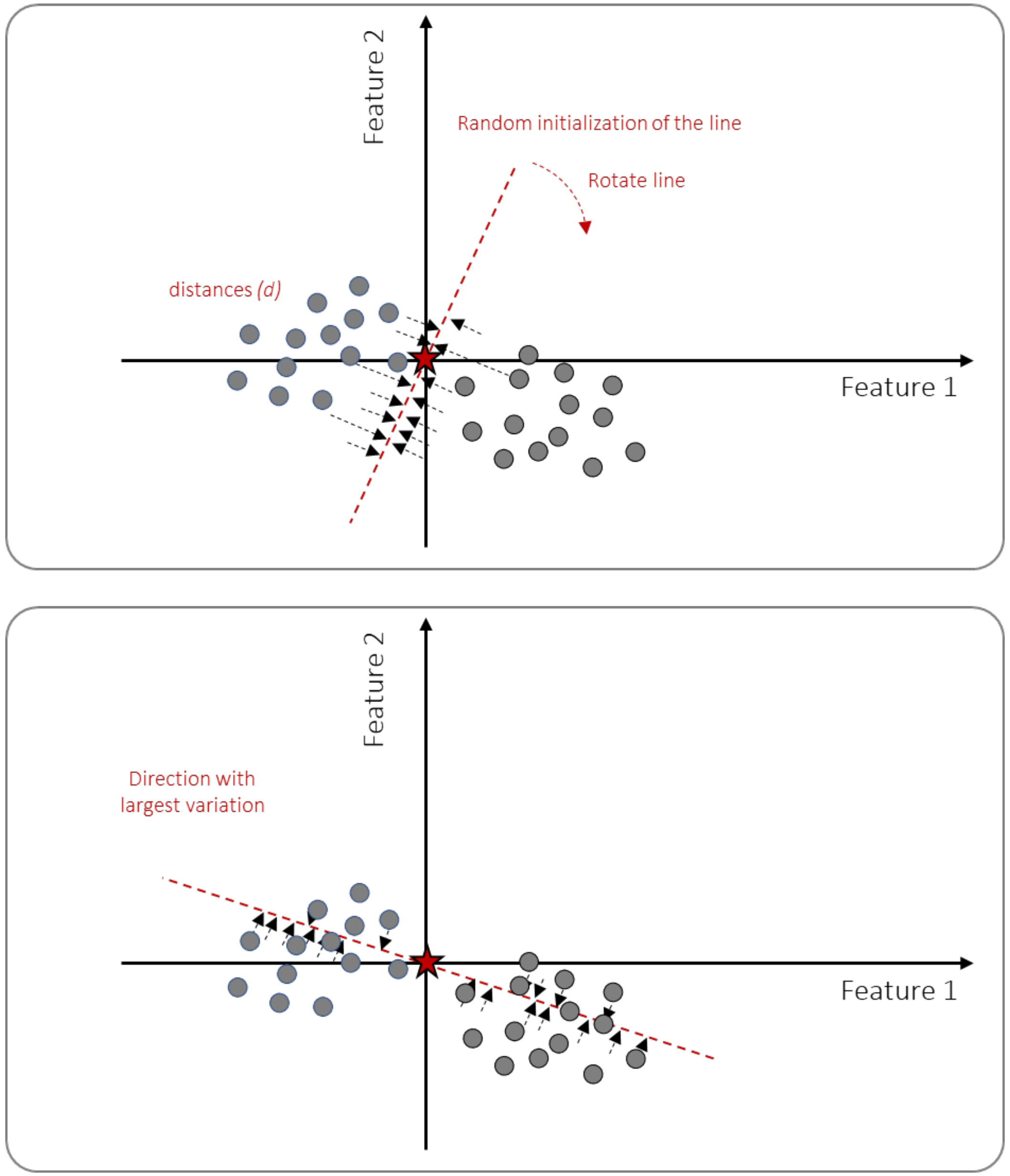
Center data around zero

PCA-Steps

Part 2. Fit the line through origin and data points.

- The next part is to fit a line through the origin and the data points (or samples). This can be done by:
    - 1. drawing a random line through the origin,
    - 2. projecting the samples on the line orthogonally, and then
    - 3. rotating until the best fit is found by minimizing the distances. *However, it is more practical to maximize the distances from the projected data points to the origin* which will lead to the same results.

- The fit is computed using the sum of squared distances (SS) as it will eliminate the orientation of the data points surrounding the line. At this point (see Figure), we fitted a line in the direction with the maximum variance


Feature 2
Random initialization of the line
Rotate line
distances (d)
Feature 1
Feature 2
Direction with largest variation
Feature 1

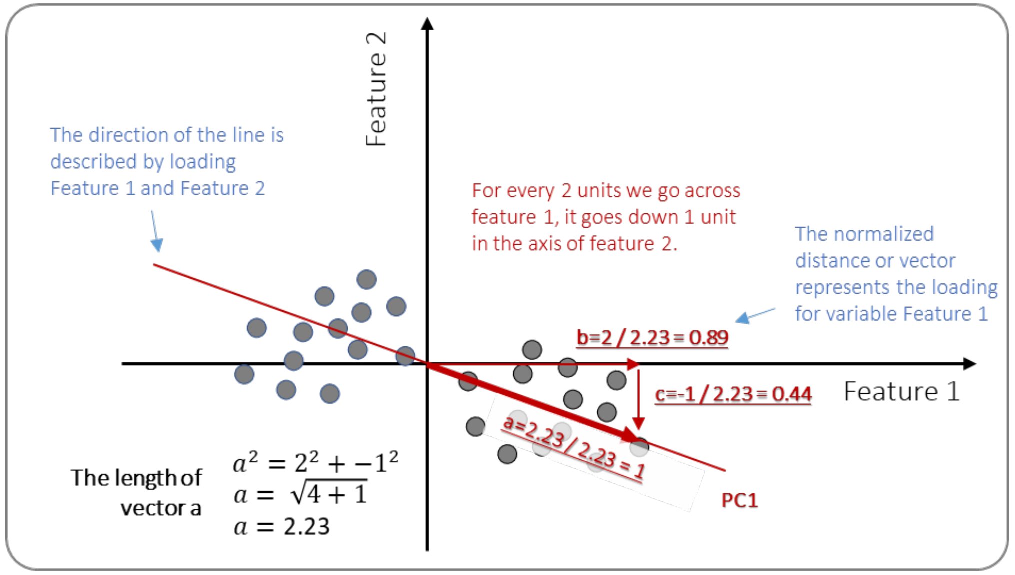
Finding the best fit. Start with a random line (top) and rotate until it fits the data best by minimizing the distances from the data points to the line (bottom). (image by the author)

PCA-Steps

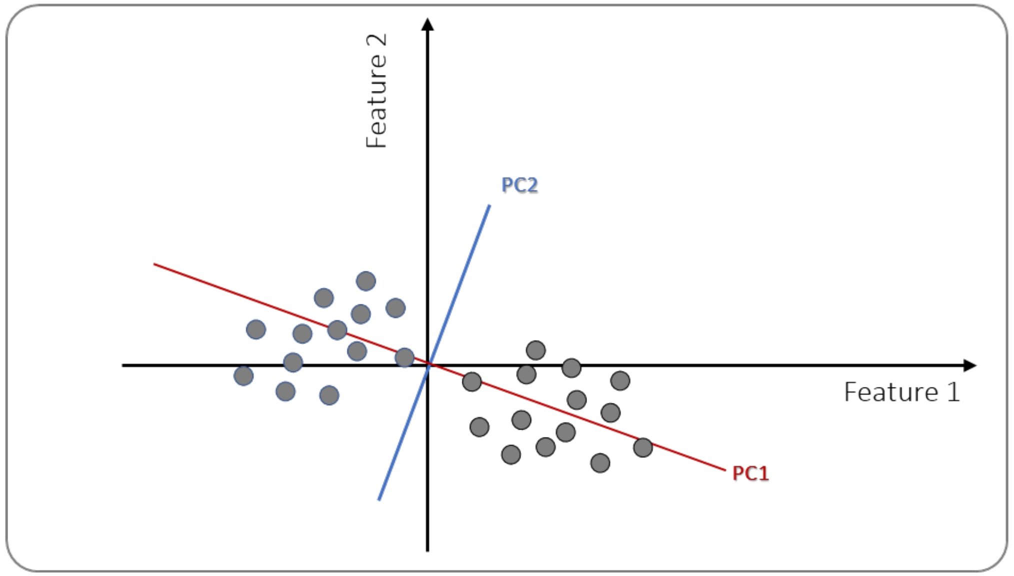
Part 3. Computing the Principal Components and the loadings.

- We determined the best-fitted line in the direction with maximum variation which is now the **1st Principal Component** or **PC1**.
- The next step is to compute the slope of PC1 that describes the contribution of each feature for PC1.
- In this example, we can *visually* observe that data points are spread out more across feature 1 than feature 2 (Figure).
- The slope of the red line is representative of our visual observation; for every 2 units we go across feature 1 (to the right), it goes down 1 unit in the axis of feature 2. Or in other words, to make PC1 (the red line), we need 2 parts of feature 1 and -1 part of feature 2. We can describe these "*parts*" as *vectors b* and *c* which we can then use to compute *vector a.* Vector *a* will get the value of 2.23 (see figure). This is what we call the **eigenvector for this particular PC (in algebra notation A)**

- However, we need to standardize toward the so-called "*unit vector*" which we get by dividing all vectors by *a*=2.23. Thus vector *b*=2/2.23=0.85, vector *c*=1/2.23=0.44 and vector a=1 (*aka the unit vector*). Thus, in other words, the range of these vectors is between -1 and 1. If for example *vector b* would have been very large, such as a value towards 1, it would mean that feature 1 contributes almost entirely to PC1.



Feature 2
The direction of the line is described by loading Feature 1 and Feature 2
For every 2 units we go across feature 1, it goes down 1 unit in the axis of feature 2.
The normalized distance or vector represents the loading for variable Feature 1
b=2 / 2.23 = 0.89
c=-1 / 2.23 = 0.44
Feature 1
a=2.23 / 2.23 = 1
The length of vector a
$a^2 = 2^2 + -1^2$
$a = \sqrt{4+1}$
$a = 2.23$
PC1

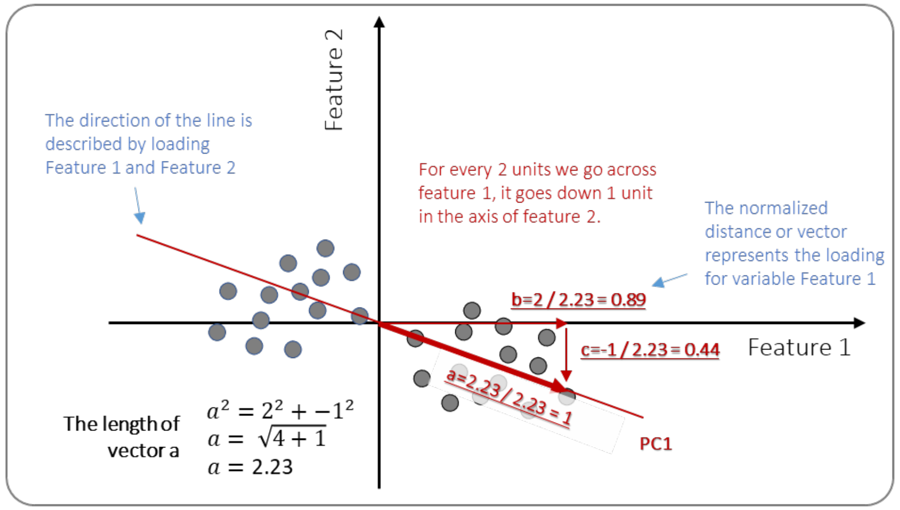


Computing PC1 and PC2 and determining the loadings

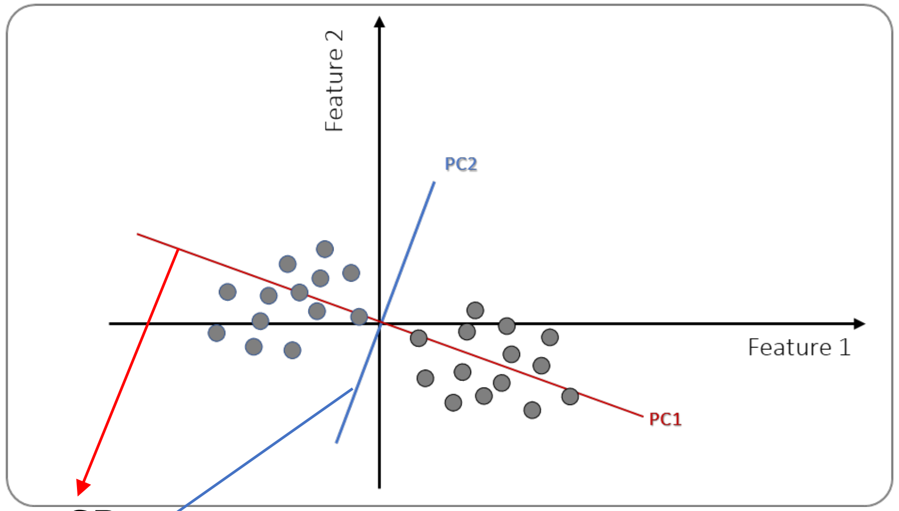
PCA-Steps

Loadings

- *It is important to realize that the principal components are less interpretable and don't have any real meaning since they are constructed as linear combinations of the initial variables.*
- But we can analyze the loadings which describe the importance of the independent variables.
- The loadings are from a numerical point of view, equal to the coefficients of the variables, and provide information about which variables give the largest contribution to the components.
- Loadings range from -1 to 1.
- A high absolute value (towards 1 or -1) describes that the variable strongly influences the component. Values close to 0 indicate that the variable has a weak influence on the component.
- The sign of a loading (+ or -) indicates whether a variable and a principal component are positively or negatively correlated.



Feature 2
The direction of the line is described by loading Feature 1 and Feature 2
For every 2 units we go across feature 1, it goes down 1 unit in the axis of feature 2.
The normalized distance or vector represents the loading for variable Feature 1
b=2 / 2.23 = 0.89
c=-1 / 2.23 = 0.44
Feature 1
a=2.23 / 2.23 = 1
PC1
The length of vector a
$a^2 = 2^2 + -1^2$
$a = \sqrt{4+1}$
$a = 2.23$



$$CP_1 = a_1 x_{1i} + \cdots + a_p x_{pi}$$
$$CP_2 = a'_1 x_{1i} + \cdots + a'_p x_{pi}$$

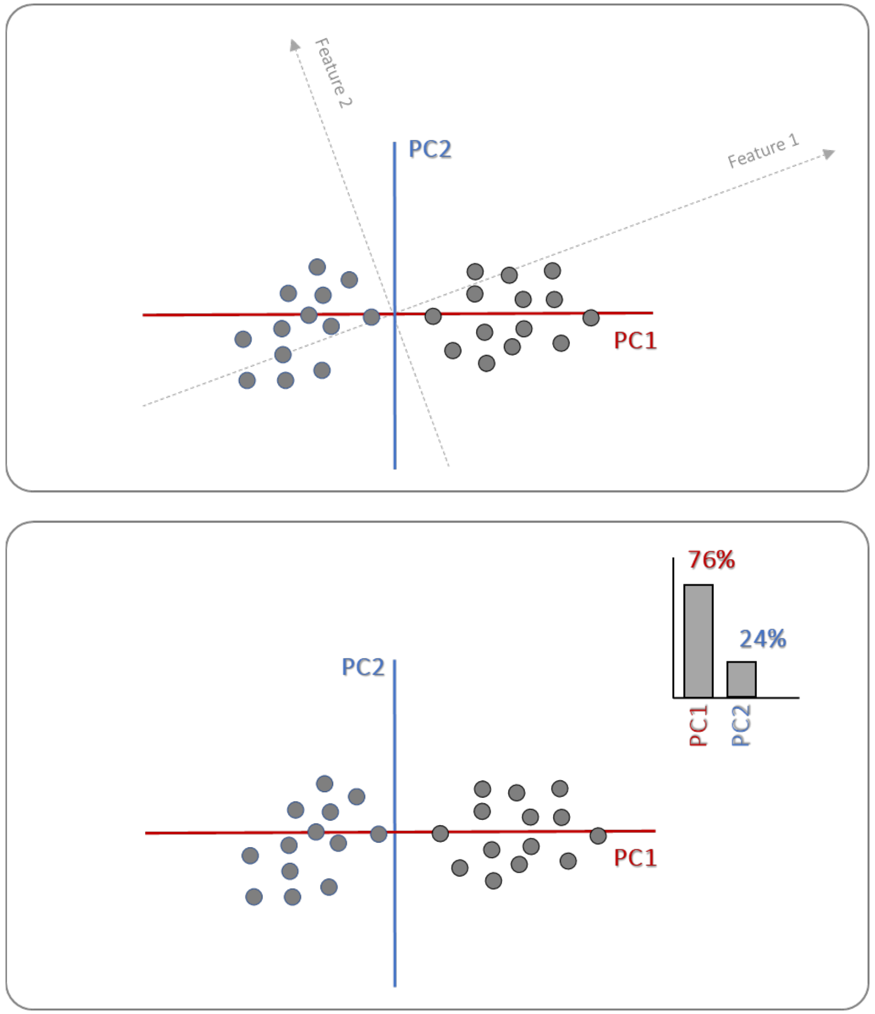
PCA-Steps

Part 4. The transformation and explained variance.

- We computed the PCs and we can now rotate (or transform) the entire dataset in such a manner that the x-axis is the direction where the largest variance is seen (aka PC1).

- Note that the transformation step will cause the values of the original feature will be lost. Instead, each PC will contain a proportion of the total variation but with the ***explained variance*** we can describe how much variance each PC contains.

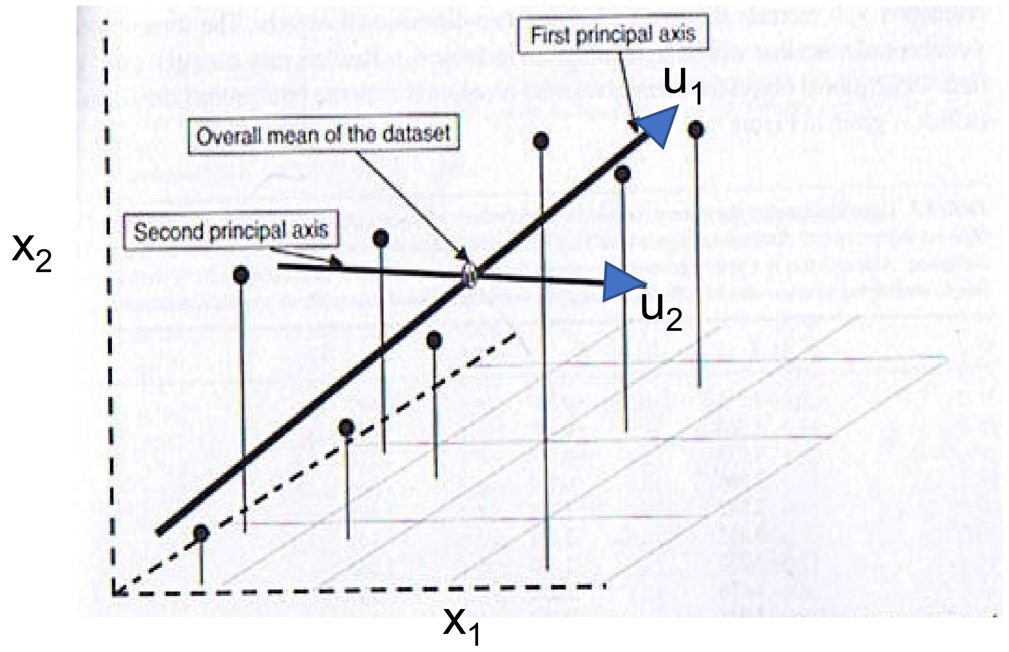
- To compute the *explained variance* we can divide the sum of squared distances (SS) for each PC by the number of data points minus one.



Transformation of the entire dataset and determining computing the explained variance

Interpretation of PCA

- PCA chooses the eigenvectors of the covariance matrix corresponding to the largest eigenvalues ($\lambda_1, \ldots, \lambda_p$).
- The eigenvalues correspond to the variance of the data along the eigenvector directions.
- Therefore, PCA projects the data along the directions where the data varies most.
- PCA preserves as much information in the data by preserving as much variance in the data.



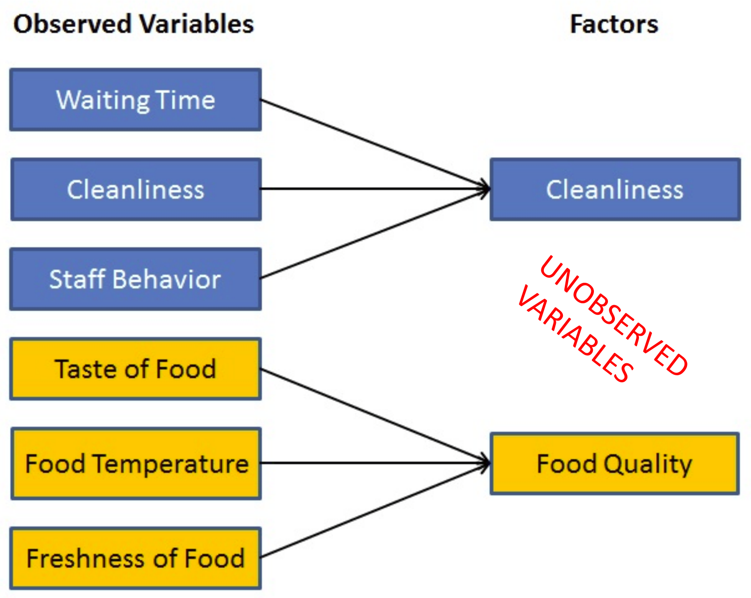
CP1 (u_1): direction of max variance
CP2 (u_2): orthogonal to u_1

You could determine each subject's score (i) on principal component 1 and 2 by using the following formula (for p variables):

$$CP_1 = a_1 x_{1i} + \cdots + a_p x_{pi}$$
$$CP_2 = a'_1 x_{1i} + \cdots + a'_p x_{pi}$$

Factor Analysis (FA)

- Factor analysis is a linear statistical model.
- It is used to explain the variance among the observed variable and condense a set of the observed variable into the unobserved variable called FACTORS.
- Observed variables are modeled as a linear combination of factors and error terms (see in **Source**).
- Factor or latent variable is associated with multiple observed variables, who have common patterns of responses.
- Each factor explains a particular amount of variance in the observed variables. It helps in data interpretations by reducing the number of variables.
- Factor analysis is a method for investigating whether a number of variables of interest X_1, X_2,……., X_p, are linearly related to a smaller number of unobservable factors F_1, F_2,…….., F_k.
- **Types of Factor analysis:**
  - Exploratory Factor Analysis: It is the most popular factor analysis approach among social and management researchers. Its basic assumption is that any observed variable is directly associated with any factor.
  - Confirmatory Factor Analysis (CFA): Its basic assumption is that each factor is associated with a particular set of observed variables. CFA confirms what is expected on the basic.
- **Differences respect PCA:**
  - PCA components explain the maximum amount of variance while factor analysis explains the covariance in data.
  - PCA components are fully orthogonal to each other whereas factor analysis does not require factors to be orthogonal.
  - PCA component is a linear combination of the observed variable while in FA, the observed variables are linear combinations of the unobserved variable or factor.
  - PCA components are uninterpretable. In FA, underlying factors are labelable and interpretable.
  - PCA is a kind of dimensionality reduction method whereas factor analysis is the latent variable method.
  - PCA is a type of factor analysis. PCA is observational whereas FA is a modeling technique.

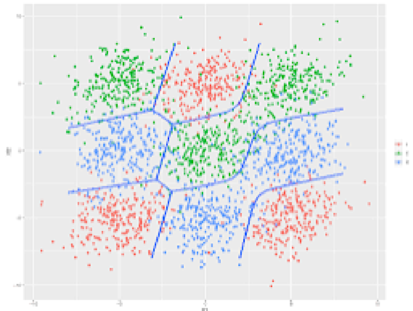


https://medium.com/@hongwy1128/intro-guide-to-factor-analysis-python-84dd0b0fd729

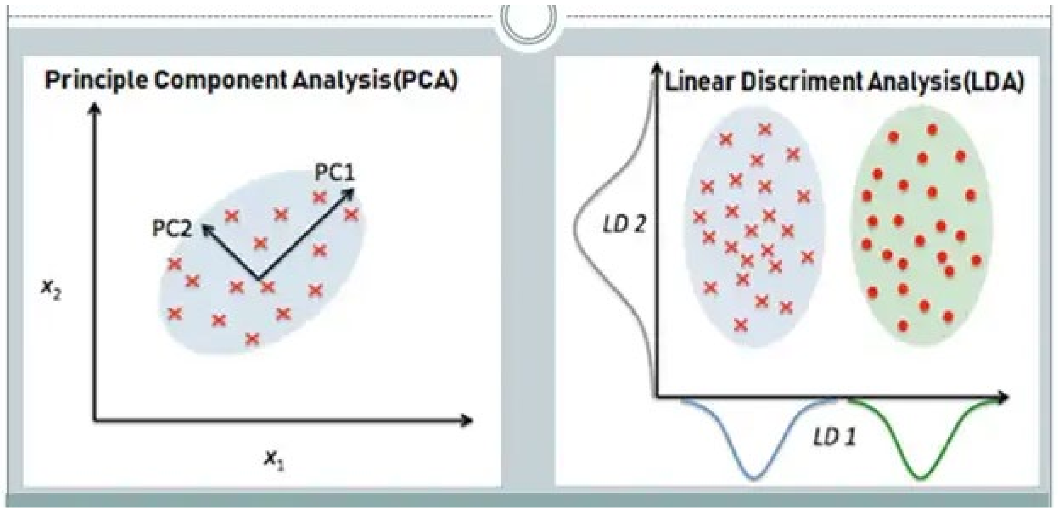
https://www.datacamp.com/tutorial/introduction-factor-analysis

Linear Discriminant Analysis (LDA)

- Linear Discriminant Analysis or Normal Discriminant Analysis or Discriminant Function Analysis is **a dimensionality reduction technique that is commonly used for supervised classification problems**.
- It is used for modelling differences in groups i.e. separating two or more classes (supervised classification method).
- It is used for modelling differences in groups i.e. separating two or more classes.
- It is used to project the features in higher dimension space into a lower dimension space.
- For example, we have two classes, and we need to separate them efficiently:
  - Classes can have multiple features.
  - Using only a single feature to classify them may result in some overlapping as shown in the below figure.
  - So, we will keep on increasing the number of features for proper classification.
- Theory in:
  - https://web.media.mit.edu/~javierhr/files/slidesPCA.pdf and https://cs.fit.edu/~dmitra/ArtInt/ProjectPapers/TheorySubs/PCA-LDA-Lobo.pptx



https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/

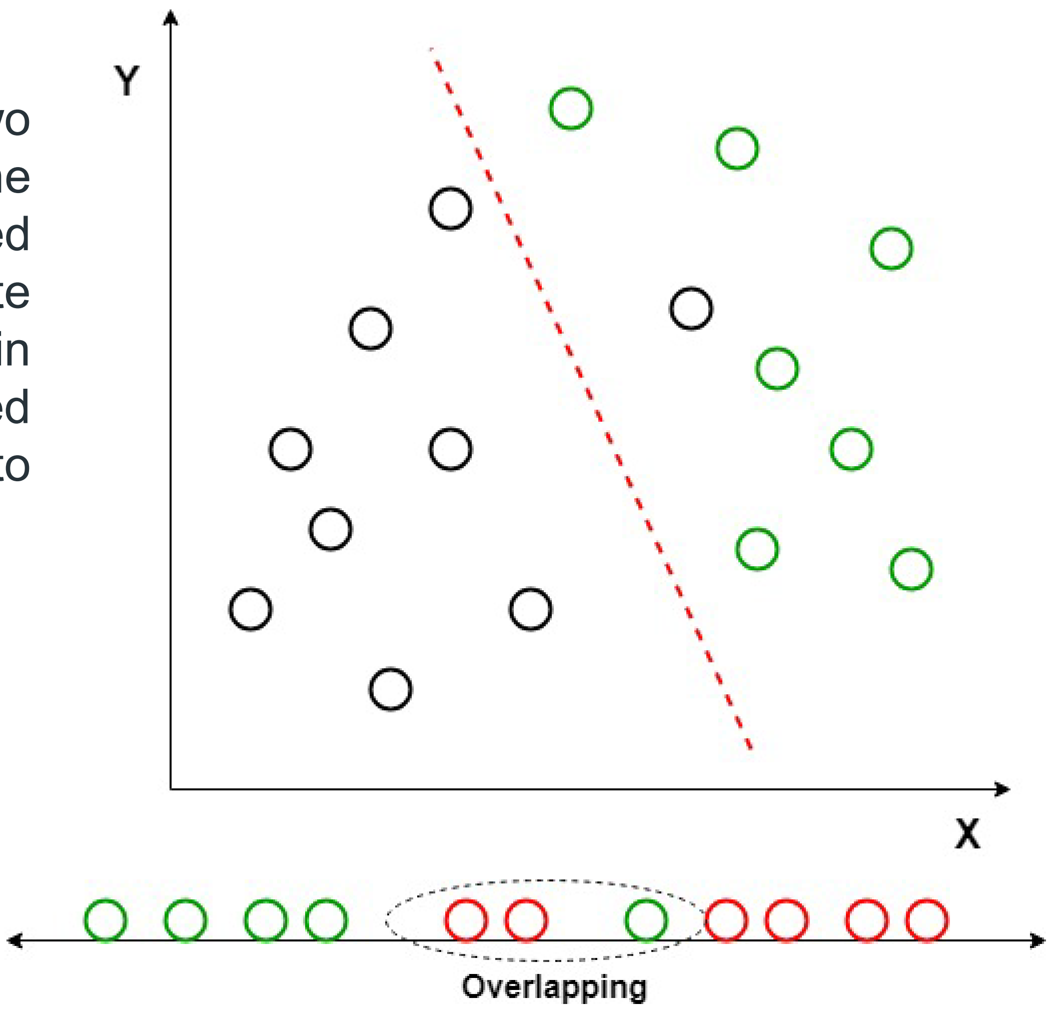

Principle Component Analysis(PCA)
PC1
PC2
x_2
x_1
Linear Discriment Analysis(LDA)
LD 2
LD 1

LDA-Steps

- Suppose we have two sets of data points belonging to two different classes that we want to classify. As shown in the given 2D graph (Figure), when the data points are plotted on the 2D plane, there's no straight line that can separate the two classes of the data points completely. Hence, in this case, LDA (Linear Discriminant Analysis) is used which reduces the 2D graph into a 1D graph in order to maximize the separability between the two classes.

- Here, Linear Discriminant Analysis uses both the axes (X and Y) to create a new axis and projects data onto a new axis in a way to maximize the separation of the two categories and hence, reducing the 2D graph into a 1D graph.

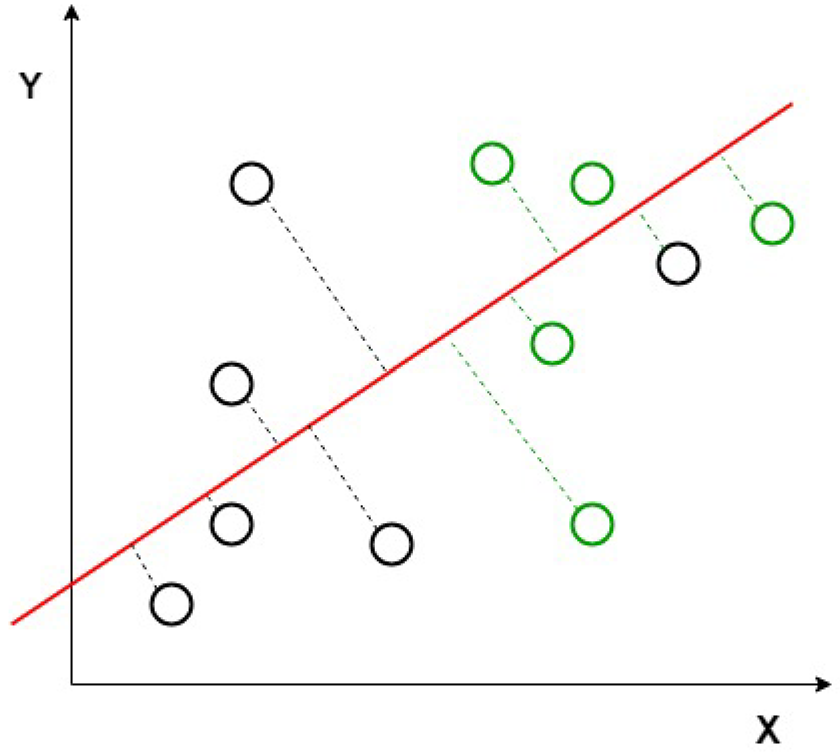
- Two criteria are used by LDA to create a new axis:
  - Maximize the distance between means of the two classes.
  - Minimize the variation within each class.

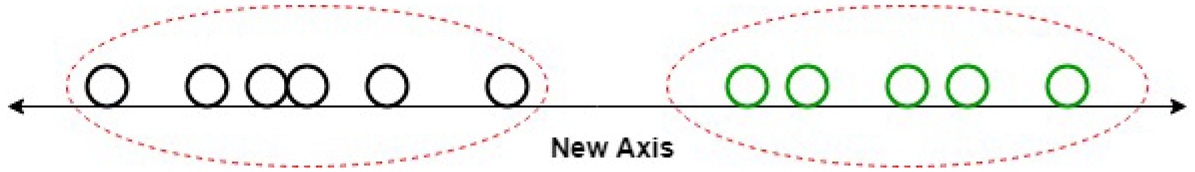


https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/

LDA-Steps

- In the right-hand graph, it can be seen that a new axis (in red) is generated and plotted in the 2D graph such that it maximizes the distance between the means of the two classes and minimizes the variation within each class.
- In simple terms, this newly generated axis increases the separation between the data points of the two classes.
- After generating this new axis using the above-mentioned criteria, all the data points of the classes are plotted on this new axis and are shown in the figure given below.





- But Linear Discriminant Analysis fails when the mean of the distributions are shared, as it becomes impossible for LDA to find a new axis that makes both the classes linearly separable. In such cases, we use non-linear discriminant analysis.

https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/

SVD: a numerical method

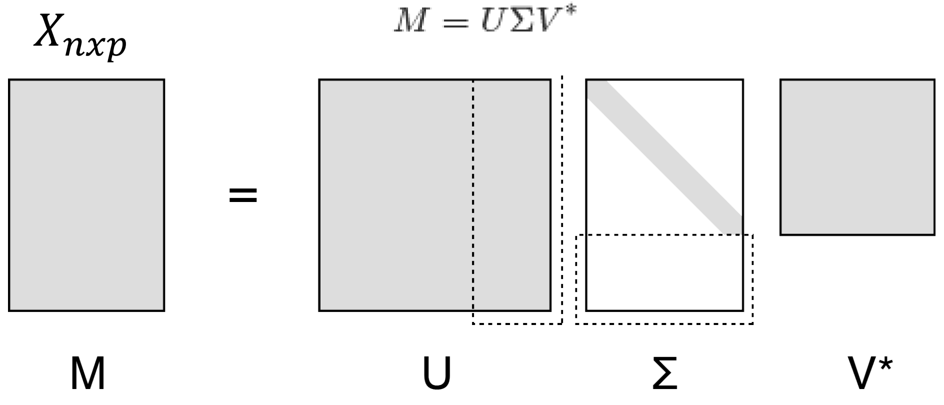
- SVD is a numerical method computationally faster: **SVD is another way to do PCA that tends to be more numerically stable. The singular value decomposition or SVD is a powerful tool in linear algebra. Understanding what the decomposition represents geometrically is useful for having an intuition for other matrix properties and also helps us better understand algorithms that build on the SVD (see in https://gregorygundersen.com/blog/2018/12/10/svd/).**
- SVD is a numerical method and PCA is an analysis approach (like least squares). You can do PCA using SVD, or you can do PCA doing the eigen-decomposition of X^TX (or XX^T), or you can do PCA using many other methods, just like you can solve least squares with a dozen different algorithms like Newton's method or gradient descent or SVD etc.
- When it comes to dimensionality reduction, the Singular Value Decomposition (SVD) is a popular method in linear algebra for matrix factorization in machine learning. Such a method shrinks the space dimension from N-dimension to K-dimension (where K<N) and reduces the number of features.
- SVD constructs a matrix with the row of users and columns of items and the elements are given by the users' ratings. Singular value decomposition decomposes a matrix into three other matrices and extracts the factors from the factorization of a high-level (user-item-rating) matrix. (see mathematics in https://towardsdatascience.com/recommender-system-singular-value-decomposition-svd-truncated-svd-97096338f361).
- More Theory in:

http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm

$$\begin{bmatrix} \\ M \\ \\ \end{bmatrix}\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \end{bmatrix}\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix}$$

X_{nxp}

$$\begin{aligned} MV &= U\Sigma \\ MVV^* &= U\Sigma V^* \\ M &= U\Sigma V^* \end{aligned}$$



The canonical diagram of the SVD decomposition of a matrix M. The columns of U are the orthonormal left singular vectors; Σ is a diagonal matrix of singular values; and the rows of V* are the orthonormal right singular vectors.

https://gregorygundersen.com/blog/2018/12/10/svd/

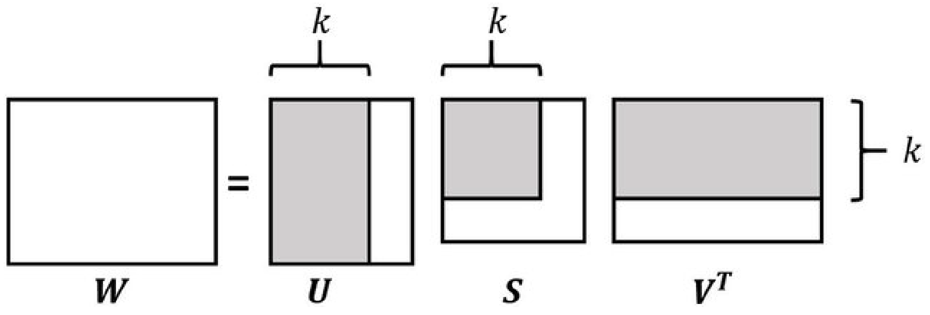
Truncated Singular Value Decomposition (SVD)

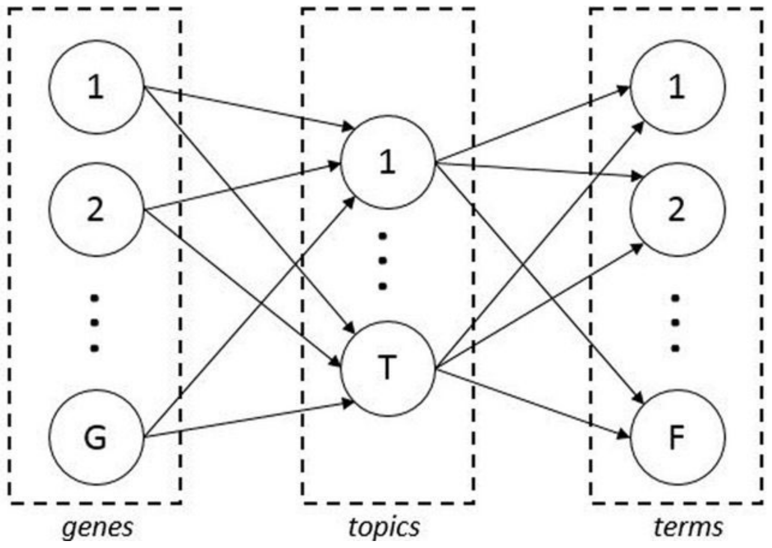
- When it comes to matrix factorization technique, truncated **Singular Value Decomposition** (**SVD**) is a popular method to produce features that factors a matrix M into the three matrices U, Σ, and V.
- Truncated SVD shares similarity with PCA while SVD is produced from the data matrix and the factorization of PCA is generated from the covariance matrix.
- Unlike regular SVDs, truncated SVD produces a factorization where the number of columns can be specified for a number of truncation. For example, given an n x n matrix, truncated SVD generates the matrices with the specified number of columns, whereas SVD outputs n columns of matrices.

- Dimensionality reduction using truncated SVD (aka LSA):

  - This transformer performs linear dimensionality reduction by means of truncated singular value decomposition (SVD). Contrary to PCA, this estimator does not center the data before computing the singular value decomposition. This means it can work with sparse matrices efficiently.

  - **The advantages of truncated SVD over PCA:** Truncated SVD can deal with sparse matrix to generate features' matrices, whereas PCA would operate on the entire matrix for the output of the covariance matrix.



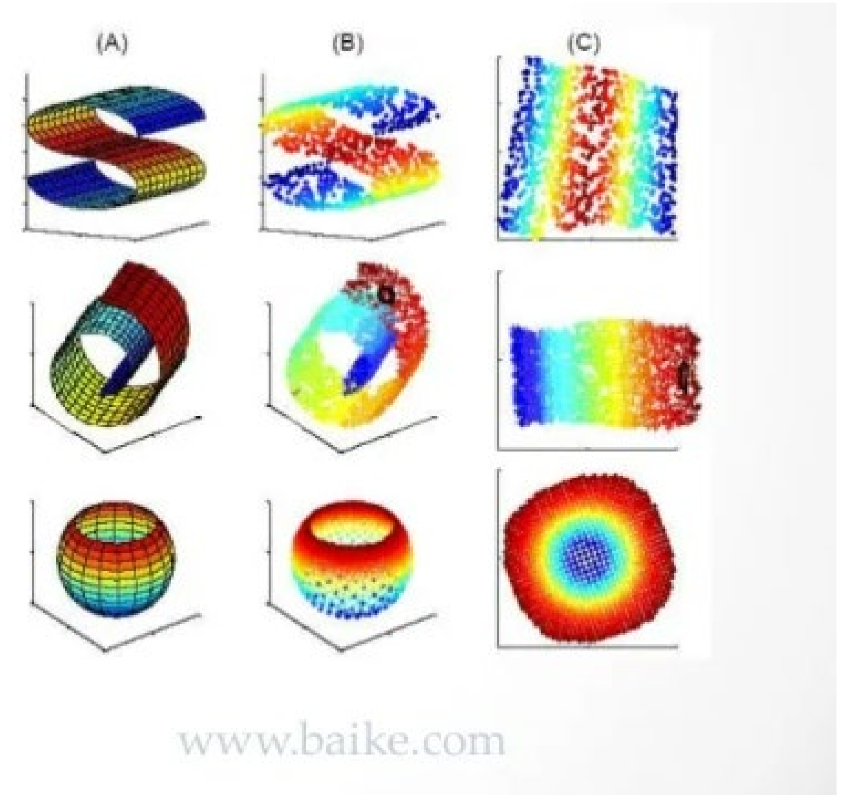


e.g.: Computational algorithms to predict Gene Ontology annotations

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html
https://towardsdatascience.com/recommender-system-singular-value-decomposition-svd-truncated-svd-97096338f361

Dimension reduction theory: Linear assumption

- What if the predictor subspace important to an outcome is not linear?
- What if there are inherent curves in these relationships?
- In these situations, suggest PCA cannot adequately reduce dimensionality without losing important information.
- Nonlinear methods will be studied in future lessons.



https://www.slideshare.net/ColleenFarrelly/review-of-methods-for-dimension-reduction

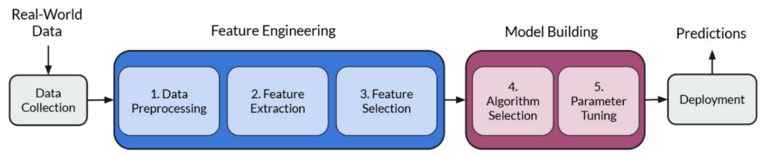
Libraries in python

- There are numerous libraries in python to perform dimension reduction, here we indicate an example of the AutoML type: Autocluster (see in https://github.com/wywongbd/autocluster/blob/master/reports/autocluster_ppt.pdf)
- Autocluster: https://github.com/wywongbd/autocluster .
- Autocluster is an automated machine learning (AutoML) toolkit for performing clustering tasks.
- Autocluster automatically optimizes the configuration of a clustering problem:
  - choice of dimension reduction algorithm
  - choice of clustering model
  - setting of dimension reduction algorithm's hyperparameters
  - setting of clustering model's hyperparameters
- autocluster provides 3 different approaches to optimize the configuration (with increasing complexity):
  - random optimization
  - bayesian optimization
  - bayesian optimization + meta-learning (warmstarting)

List of dimension reduction algorithms in sklearn supported by autocluster's optimizer:

Method	Hyperparameters
TSNE	`n_components` (integer) `perplexity` (float) `early_exaggeration` (float)
PCA	`n_components` (integer) `svd_solver` (categorical) `whiten` (categorical)
Incremental PCA	`n_components` (integer) `whiten` (categorical) `batch_size` (integer)
Kernel PCA	`n_components` (integer) `kernel` (categorical)
Fast ICA	`n_components` (integer) `algorithm` (categorical) `fun` (categorical) `whiten` (categorical)
Truncated SVD	`n_components` (integer) `algorithm` (categorical)
Latent Dirichlet Allocation(LDA)	`n_components` (integer) `learning_method` (categorical)

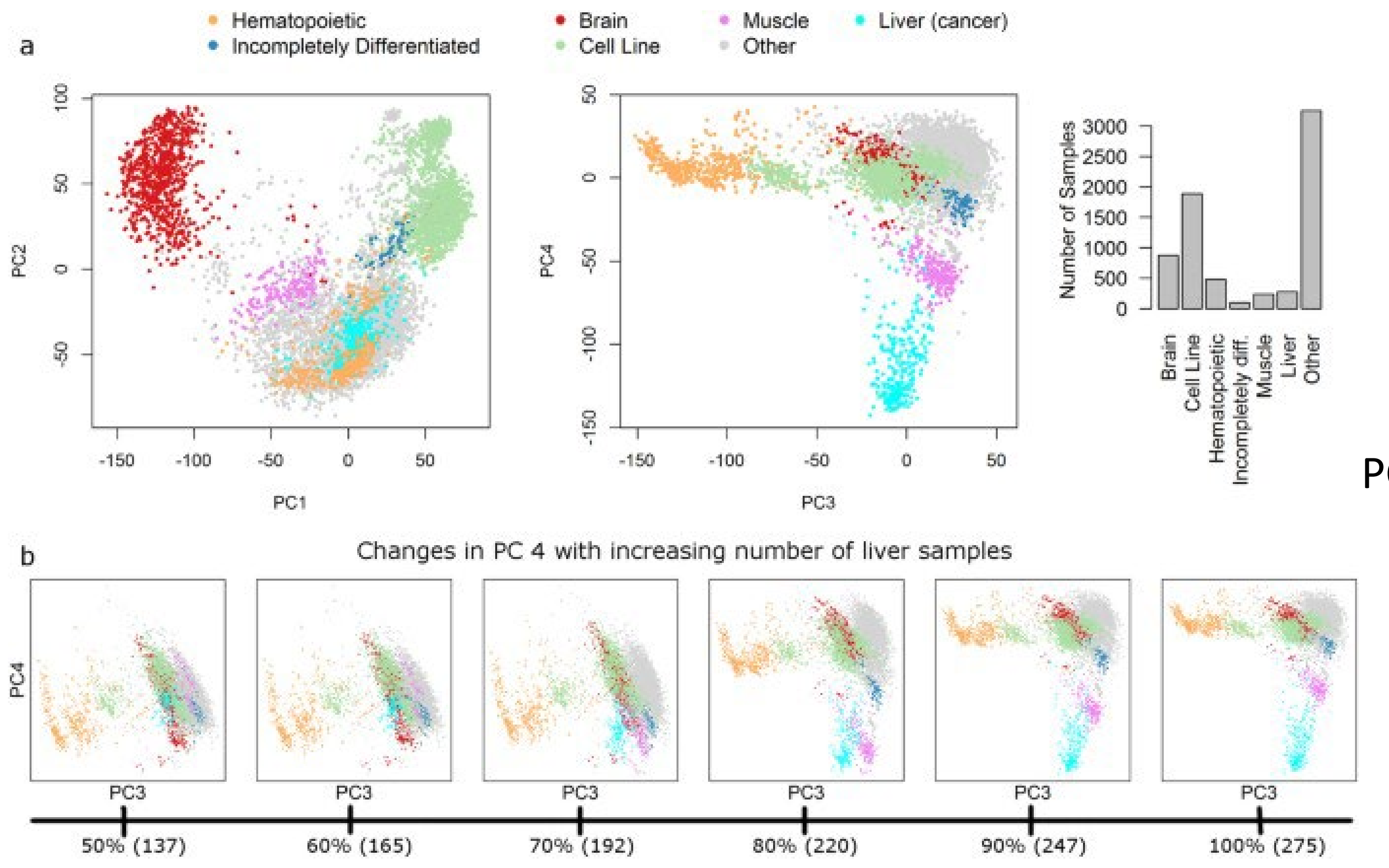
The ML pipeline



Bibliography and links

- https://medium.com/machine-learning-researcher/dimensionality-reduction-pca-and-lda-6be91734f567
- https://www.analyticsvidhya.com/blog/2018/08/dimensionality-reduction-techniques-python/
- Dimensionality Reduction. Section 3.8 (Duda et al.). CS479/679 Pattern Recognition. Dr. George Bebis
- https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction

Mathematical annex for PCA



PCA example

https://www.nature.com/articles/srep25696

Dimension reduction theory: PCA - Linear Algebra

Basen on Pattern Recognition. Dr. George Bebis

- If x∈R^N, then it can be written a linear combination of an orthonormal set of N basis vectors <v$_1$,v$_2$,…,v$_N$> in R^N (e.g., using the standard base):

$$v_i^T v_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{x} = \sum_{i=1}^{N} x_i v_i = x_1 v_1 + x_2 v_2 + \ldots + x_N v_N$$

$$\text{where} \quad x_i = \frac{\mathbf{x}^T v_i}{v_i^T v_i} = \mathbf{x}^T v_i$$

$$\mathbf{x} : \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ . \\ . \\ x_N \end{bmatrix}$$

N= number of variabes (=p)

- PCA seeks to approximate x in a subspace of R^N using a new set of K<<N basis vectors <u$_1$, u$_2$, …,u$_K$> in R^N:

$$\hat{\mathbf{x}} = \sum_{i=1}^{K} y_i u_i = y_1 u_1 + y_2 u_2 + \ldots + y_K u_K$$

$$\text{where} \quad y_i = \frac{\mathbf{x}^T u_i}{u_i^T u_i} = \mathbf{x}^T u_i$$

$$\hat{\mathbf{x}} : \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_K \end{bmatrix}$$

(reconstruction)

such that $\| \mathbf{x} - \hat{\mathbf{x}} \|$ is minimized!

(i.e., minimize information loss)

Dimension reduction theory: PCA - Linear Algebra

- The "optimal" set of basis vectors $<u_1, u_2, \ldots, u_K>$ can be found as follows (we will see why):

(1) Find the eigenvectors u_i of the covariance matrix of the (training) data Σ_x

$$\Sigma_x u_i = \lambda_i u_i$$

(2) Choose the K "largest" eigenvectors u_i (i.e., corresponding to the K "largest" eigenvalues λ_i)

$<u_1, u_2, \ldots, u_K>$ correspond to the "optimal" basis!

We refer to the "largest" eigenvectors u_i as principal components.

PCA – Steps (mathematically)

- Suppose we are given $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M$ (N x 1) vectors

N: # of features

M: # data

Step 1: compute sample mean

$$\bar{\mathbf{x}} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i$$

Step 2: subtract sample mean (i.e., center data at zero)

$$\Phi_i = \mathbf{x}_i - \bar{\mathbf{x}}$$

Step 3: compute the sample covariance matrix Σ_x

$$\Sigma_{\mathbf{x}} = \frac{1}{M}\sum_{i=1}^{M}(\mathbf{x}_\mathrm{i} - \bar{\mathbf{x}})(\mathbf{x}_\mathrm{i} - \bar{\mathbf{x}})^T = \frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i^T = \frac{1}{M}AA^T$$

where $A=[\Phi_1\ \Phi_2\ ...\ \Phi_M]$
i.e., the columns of A are the Φ_i
(N x M matrix)

PCA - Steps

Step 4: compute the eigenvalues/eigenvectors of Σ_x

$$\Sigma_x u_i = \lambda_i u_i$$

where we assume $\lambda_1 > \lambda_2 > \ldots > \lambda_N$

Note : most software packages return the eigenvalues (and corresponding eigenvectors) is decreasing order – if not, you can explicitly put them in this order)

Since Σ_x is symmetric, $<u_1,u_2,\ldots,u_N>$ form an orthogonal basis in R^N and we can represent any $\mathbf{x} \in R^N$ as:

$$\mathbf{x} - \overline{\mathbf{x}} : \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ . \\ . \\ x_N \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ . \\ . \\ y_N \end{bmatrix}$$

$$\mathbf{x} - \overline{\mathbf{x}} = \sum_{i=1}^{N} y_i u_i = y_1 u_1 + y_2 u_2 + \ldots + y_N u_N$$

$$y_i = \frac{(\mathbf{x} - \overline{\mathbf{x}})^T u_i}{u_i^T u_i} = (\mathbf{x} - \overline{\mathbf{x}})^T u_i \qquad if \ \| u_i \| = 1$$

i.e., this is just a “change” of basis!

Note : most software packages normalize u_i to unit length to simplify calculations; if not, you can explicitly normalize them)

PCA - Steps

Step 5: dimensionality reduction step – approximate **x** using only the first K eigenvectors (K<<N) (i.e., corresponding to the K largest eigenvalues where K is a parameter):

$$\mathbf{x} - \bar{\mathbf{x}} = \sum_{i=1}^{N} y_i u_i = y_1 u_1 + y_2 u_2 + \ldots + y_N u_N$$

$$\Big\Downarrow$$

approximate $\mathbf{x}\ by\ \hat{\mathbf{x}}$
using first K eigenvectors only

$$\hat{\mathbf{x}} - \bar{\mathbf{x}} = \sum_{i=1}^{K} y_i u_i = y_1 u_1 + y_2 u_2 + \ldots + y_K u_K$$

(reconstruction)

$$\mathbf{x} - \bar{\mathbf{x}} : \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ y_N \end{bmatrix} \rightarrow \hat{\mathbf{x}} - \bar{\mathbf{x}} : \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_K \end{bmatrix}$$

note that if K=N, then $\hat{\mathbf{x}} = \mathbf{x}$
(i.e., zero reconstruction error)