Unsupervised Learning

Sophie de Buyl

2020 – 21

VRIJE
UNIVERSITEIT
BRUSSEL

Unsupervised learning

2021-03-06

Unsupervised Learning

Sophie de Buyl

2020 – 21

Reference: chapter 10 of the book ISLR for PCA and clustering techniques.

Unsupervised Learning

1. Objectives

2. Low dimensional representation of high dimensional data sets
    1. Principal component analysis
    2. (t)SNE

3. Clustering methods
    1. k-means clustering
    2. Dendrogram

Â

Objectives

We aim at interpreting 'big' datasets encoded in matrices $\boldsymbol{X}$ of the form:

$$\boldsymbol{X} = \begin{pmatrix} x^1{}_1 & \dots & x^1{}_p \\ \vdots & \ddots & \vdots \\ x^n{}_1 & \dots & x^n{}_p \end{pmatrix},$$

each line = one observation/experiment
columns = p variables

We want to answer questions such as: Is there some structure in the data? What are the most relevant variables (or combinaisons of variables) to describe the data set?

In practice : you will learn how to reduce dimensionalityÂ of a dataset (via PCA and tSNE) and to perform clustering (via k-means clustering and dendrograms).

Specific questions do depend on the problem at hand.

Objectives

We aim at interpreting 'big' datasets encoded in matrices $\boldsymbol{X}$ of the form:

$$\boldsymbol{X} = \begin{pmatrix} x^1{}_1 & \dots & x^1{}_p \\ \vdots & \ddots & \vdots \\ x^n{}_1 & \dots & x^n{}_p \end{pmatrix},$$

each line = one observation/experiment
columns = $\boldsymbol{p}$ variables

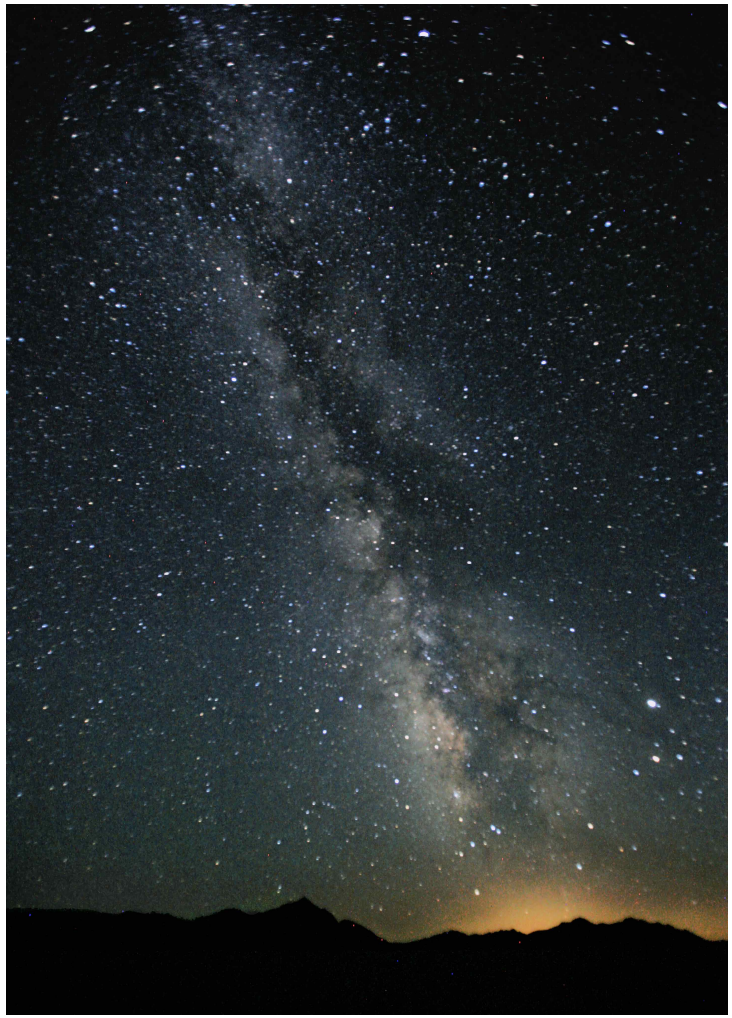
We want to answer questions such as: Is there some structure in the data? What are the most relevant variables (or combinaisons of variables) to describe the data set?

In practice : you will learn how to reduce dimensionalityÂ of a dataset (via PCA and tSNE) and to perform clustering (via k-means clustering and dendrograms).

Specific questions do depend on the problem at hand.

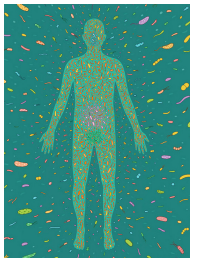
Many other techniques exist for dimensional reduction or clustering. No methods dominate over the other ones, the best method in unsupervised learning highly depends on the dataset.

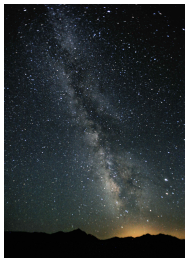
Example of data set and questions: gut microbiome



images from American Museum of Natural History and wikipedia

Example of data set and questions: gut microbiome





images from American Museum of Natural History and wikipedia

We have about 10 times more microbes on and in our bodies that they are stars in the Milky Way. Our gut microbial community is actually the most dense ecosystem known, which makes it a pretty complex system. Since about 10 years ago, it become possible to known the composition of the human associated microbial communities due to the advances in sequencing technology. Big questions in the field include: What is a healthy gut microbial composition? How to control this composition (in particular to restore a healthy composition in case of disease perturbing the system)?

The dataset we will use to illustrate unsupervised learning techniques consists of the bacterial compositions of the feces of 33 individuals (only relative abundances can be measured). The dataset is from 2011 [Arumugam, Nature 2011]. Today the feces of many more people have been sampled, as well as other body sites. In particular, the Human Microbiome Project counts about ten thousands samples.

Example of data set and questions: gut microbiome

We will consider a dataset which consists of the abundance of 248 microbial species of 33 individuals, see a snapshot here under:

0.0000268	0.0000798	0.0000636	0.0000565	0.000115896	0.0000432	0.0000316	0.00019609	0.00004	0.0000716
0.000615703	0.000541181	0.001046353	0.000665158	0.000689109	0.000437158	0.000512171	0.000408606	0.000397813	0.000482034
0.000510041	0.000068	0.0000724	0.0000687	0.0000645	0.0000416	0.0000714	0.0000388	0.000130294	0.0000472
0.000178808	0.00000586	0.00000189	0.0000147	0.0000544	0.0000849	0.0000117	0.000068	0.0000198	0.00000488
0.000116489	0.000296717	0.000183091	0.000108476	0.0000976	0.0000494	0.000222225	0.000196857	0.000327481	0.000348413
0.001166081	0.000722	0.000718418	0.000507162	0.00055814	0.000568596	0.00067678	0.000475078	0.000552965	0.000549491
0.000258469	0.000334943	0.000395204	0.000251151	0.000166682	0.000299899	0.000201003	0.000251086	0.000343657	0.000321792
0.000475983	0.000336855	0.000446418	0.000371842	0.000326289	0.00057566	0.00037902	0.000729107	0.000322915	0.000404254
0.000587723	0.000615169	0.000573969	0.000582565	0.000425491	0.000473737	0.000519671	0.000501205	0.000323562	0.000506146
0.000056	0	0.0000152	0.0000378	0.0000124	0.0000191	0.0000282	0.0000109	0.0000384	0.00000822
0.000806178	0.000735476	0.000642349	0.00037878	0.00068775	0.000684564	0.000770342	0.000795664	0.00078233	0.000714904
0.00027869	0.0004671	0.00042568	0.000231702	0.000455785	0.000526811	0.000426485	0.000269044	0.000270495	0.0002747
0.000333809	0.000461454	0.000401274	0.000570715	0.000913697	0.000579964	0.000611882	0.000211419	0.000343369	0.000397113

How to represent the dataset in an instructive manner?

Specific questions: which are the most abundance species? are they shared by all individuals? What are the species that are the most variable in between individuals?

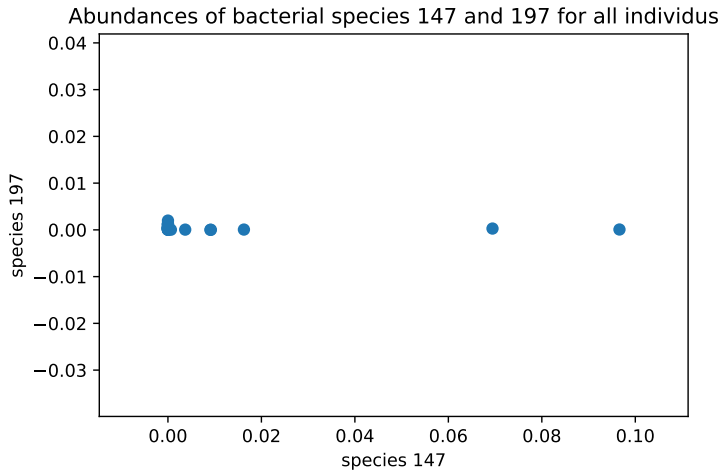
Low dimensional representation of high dimensional data sets

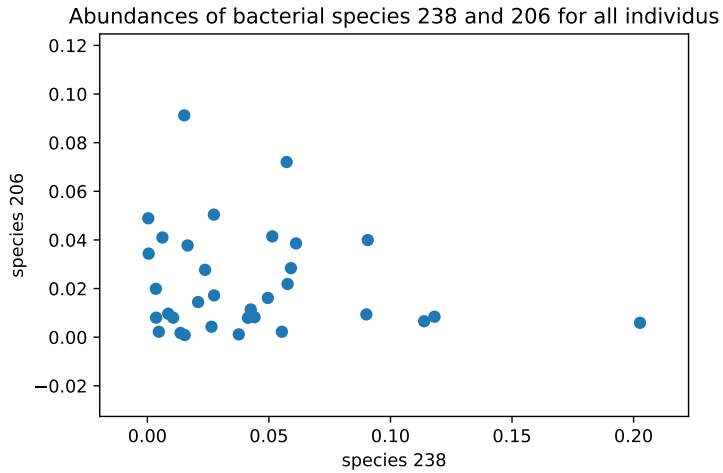
1. Objectives

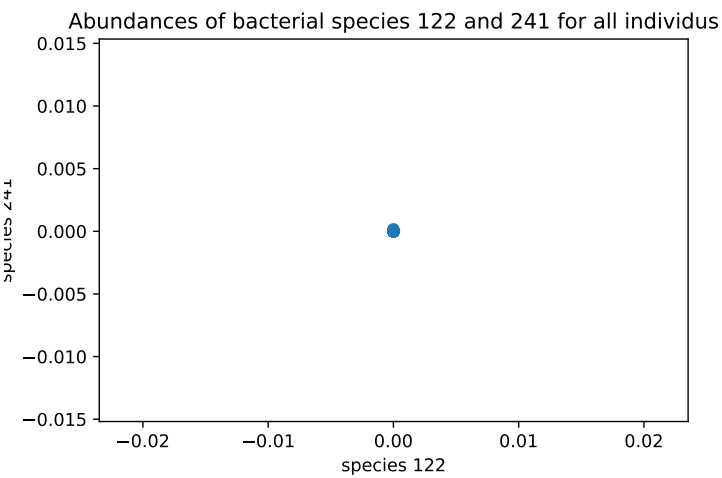
2. Low dimensional representation of high dimensional data sets
    1. Principal component analysis
    2. (t)SNE

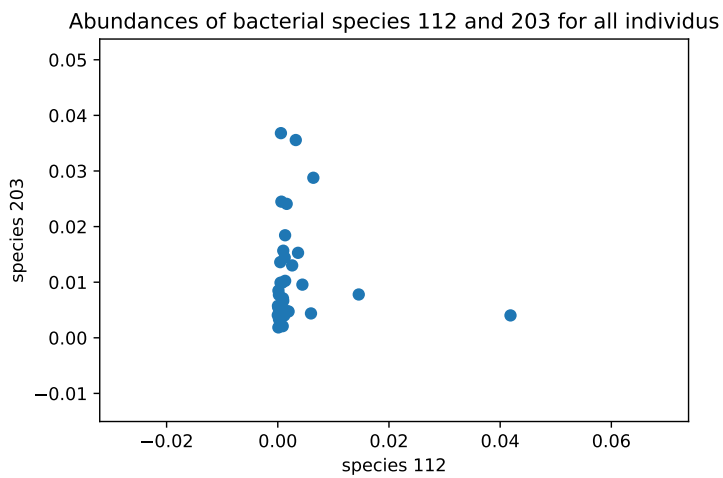
3. Clustering methods
    1. k-means clustering
    2. Dendrogram

How to represent our dataset in 2d? Plot $\frac{248 \times 247}{2}$ figures?

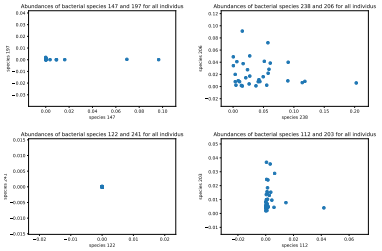








How to represent our dataset in 2d? Plot $\frac{248 \times 247}{2}$ figures?



Clearly representing all 2d plots is not an instructive manner of representing our dataset. We will see two techniques to choose more wisely two combinations of coordinates to potentially see structure in the dataset. The first one is principal component analysis and its goal is to capture most of the variability of the dataset. The second one is called tSNE and its goal is to preserve as much as possible the local structure of the dataset. More precisely, it aims a preserving small pairwise distances between samples in low dimension.

PCA

1. Objectives

2. Low dimensional representation of high dimensional data sets
    1. Principal component analysis
    2. (t)SNE

3. Clustering methods
    1. k-means clustering
    2. Dendrogram

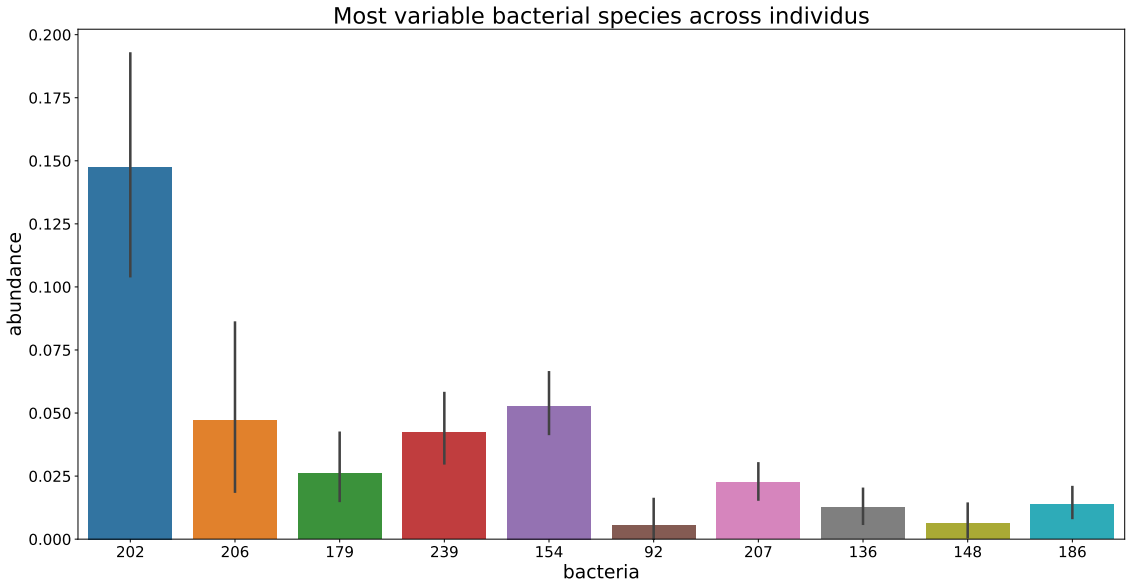
Idea behind PCA

Instead of representing the dataset with the original variables (the bacterial abundances), the idea of Principal Component Analysis (PCA) is to find linear combinations of these original variables which capture most of the variability of the dataset.

How to find these new variables?

Idea behind PCA

We could think of simply selecting the variables with highest variance. Here is a figure representing the mean values and variances (black lines) of the 10 most variable bacterial species:

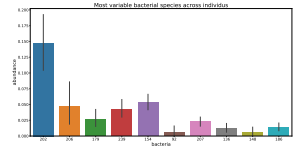


However, these variables are correlated...

We want **uncorrelated** variables and such that the first variable (called the first principal component) accounts for the largest variance, the second variable accounts for the largest variance when the first principal component dimension is not considered, etc..

Idea behind PCA

We could think of simply selecting the variables with highest variance. Here is a figure representing the mean values and variances (black lines) of the 10 most variable bacterial species:



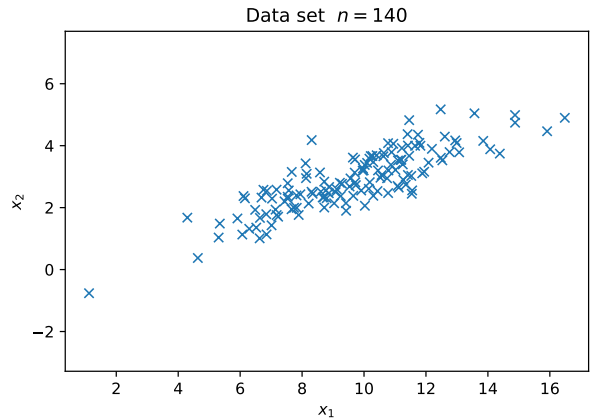
However, these variables are correlated...

We want **uncorrelated** variables and such that the first variable (called the first principal component) accounts for the largest variance, the second variable accounts for the largest variance when the first principal component dimension is not considered, etc..

Check HMP_explore_data.ipynb to reproduce this barplot. An important remark is that the variance of each bacterial species depends on the units used. One should obviously use the same units here for each bacterial species in order to have a meaningful comparison. For other types of datasets, this can be more subtle. Imagine that your dataset consists of runners and the features measured are the times of different races: a marathon, a 100 meters race,... There if you use the same units (e.g. minutes), you will put effectively a very high weight on the marathon. One way to avoid this is to normalize the dataset and impose a variance of one for all features (here all races).

A ‘toy’ example: PCA from 2d to 1d

Consider the series of points (x_1, x_2) drawn from a bivariate normal distribution:



Suppose you want to represent this dataset in 1d and grasp most of the variability of the sample: Would x_1 or x_2 do the job?

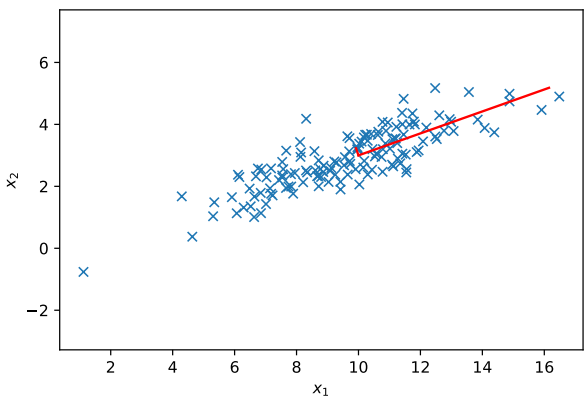
No! The direction containing most of the variability of this ’cloud’ of points is a combination of x_1 and x_2.

A 'toy' example: PCA from 2d to 1d

To quantify the variability of the dataset, we compute the covariance matrix Σ.

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}$$

What can you say about the relative values of the coefficient in this matrix?



We want to change coordinates from x_1 and x_2 to two new variables z_1 and z_2, independent of each other, i.e. $\sigma_{z_1 z_2} = 0$, and such that z_1 goes along the red direction.

A 'toy' example: PCA from 2d to 1d

A first remark is that we are not interested in the mean values of the variables. We first center the samples by considering $\hat{x}_i = x_i - \mu_i$. Then we do a coordinate change

$$z_1 = \hat{x}_1 \varphi^1{}_1 + \hat{x}_2 \varphi^2{}_1 \,, \qquad z_2 = \hat{x}_1 \varphi^1{}_2 + \hat{x}_2 \varphi^2{}_2 \,.$$

Two conditions on φ:

- we want z_1 and z_2, called the **principal components** (PCs), to be uncorrelated:

  $$\Rightarrow \Sigma_z = \varphi^T \Sigma \varphi \qquad \text{should be diagonal.}$$

  where Σ is the covariance matrix in the original coordinates, i.e. $\frac{1}{n} \hat{X}^T \hat{X}$.
- finding the direction z_1 with highest variance is **not** a uniquely defined problem $\Rightarrow$ to impose unicity (up to a sign) we impose the normalisation $\sum_i (\varphi^i{}_1)^2 = 1$. The same holds for the other principal components z_i.

Now we simply need to remember that a symmetric matrix, such as Σ, can be diagonalized by an orthonormal matrix! Moreover the matrix corresponding to the coordinate change performing the diagonalisation is orthogonal and build out of the eigenvectors of Σ (placed column-wise).

A 'toy' example: PCA from 2d to 1d

A first remark is that we are not interested in the mean values of the variables. We first center the samples by considering $\hat{x}_i = x_i - \mu_i$. Then we do a coordinate change

$$z_1 = \hat{x}_1 {\varphi^1}_1 + \hat{x}_2 {\varphi^2}_1 , \qquad z_2 = \hat{x}_1 {\varphi^1}_2 + \hat{x}_2 {\varphi^2}_2 .$$

Two conditions on φ:

- we want z_1 and z_2, called the **principal components** (PCs), to be uncorrelated:

  $$\Rightarrow \Sigma_z = \varphi^T \Sigma \varphi \qquad \text{should be diagonal.}$$

  where Σ is the covariance matrix in the original coordinates, i.e. $\frac{1}{n} \hat{X}^T \hat{X}$.
- finding the direction z_1 with highest variance is **not** a uniquely defined problem $\Rightarrow$ to impose unicity (up to a sign) we impose the normalisation $\sum_i (\varphi^i{}_1)^2 = 1$. The same holds for the other principal components z_i.

Now we simply need to remember that a symmetric matrix, such as Σ, can be diagonalized by an orthonormal matrix! Moreover the matrix corresponding to the coordinate change performing the diagonalisation is orthogonal and build out of the eigenvectors of Σ (placed column-wise).

Exercise: check that the covariance matrix in the original coordinates is given by $\frac{1}{n} \hat{X}^T \hat{X}$.

A real symmetric matrix is diagonalisable by an orthogonal matrix, see e.g. **`https://en.wikipedia.org/wiki/Symmetric_matrix`**.

PCA the receipt in any dimensions

$$z_k = \hat{x}_1 {\varphi^1}_k + ... + \hat{x}_p {\varphi^p}_k \text{ , in matricial notation } \underbrace{\boldsymbol{z}}_{p} = \underbrace{\hat{\boldsymbol{x}}}_{p} \underbrace{\boldsymbol{\varphi}}_{p \times p}$$

(1) Normalize the dataset $\Rightarrow \hat{\boldsymbol{X}}$ (making the mean value of each variable zero);

(2) Compute the covariance matrix of the dataset $\Rightarrow \boldsymbol{\Sigma} = \frac{1}{n} \hat{\boldsymbol{X}}^T \hat{\boldsymbol{X}}$;

(3) Compute the eigenvectors of this covariance matrix;

(4) The coordinate change matrix $\boldsymbol{\varphi}$ is the matrix made out of the eigenvectors of $\boldsymbol{\Sigma}$ put in columns. If the normalized variables of one sample are given by $(\hat{x}_1, ... ,\hat{x}_p)$, then principal components $(z_1, ... ,z_p)$ for this sample can be compute as follows: $\boldsymbol{z} = \hat{\boldsymbol{x}}\boldsymbol{\varphi}$.

(5) We can change at once the matrix of the dataset to the principal components by:

$$\underbrace{\boldsymbol{Z}}_{n \times p} = \underbrace{\hat{\boldsymbol{X}}}_{n \times p} \underbrace{\boldsymbol{\varphi}}_{p \times p}$$

$\boldsymbol{Z}$ is a matrix containing line by line all your samples, and the columns contain the values of the principal component variables for each sample.

What is the relative weight of original variables in the PCs?

$$z_k = \hat{x}_1 {\varphi^1}_k + \ldots + \hat{x}_p {\varphi^p}_k \text{ , in matricial notation } \boldsymbol{z} = \hat{\boldsymbol{x}}\boldsymbol{\varphi}$$

It is given by the $\boldsymbol{p}$ loading vectors: ${\varphi^1}_k$ the loading vector of $\boldsymbol{x}_1$, ... , ${\varphi^p}_k$ the loading vector of $\boldsymbol{x}_p$.

The coordinate change matrix φ contains the eigenvectors of $\boldsymbol{\Sigma}$ and the loading vectors:

eigen vectors

$$\boldsymbol{\varphi} = \begin{pmatrix} {\varphi^1}_1 & \cdots & {\varphi^1}_p \\ \vdots & \ddots & \vdots \\ {\varphi^p}_1 & \cdots & {\varphi^p}_p \end{pmatrix}$$

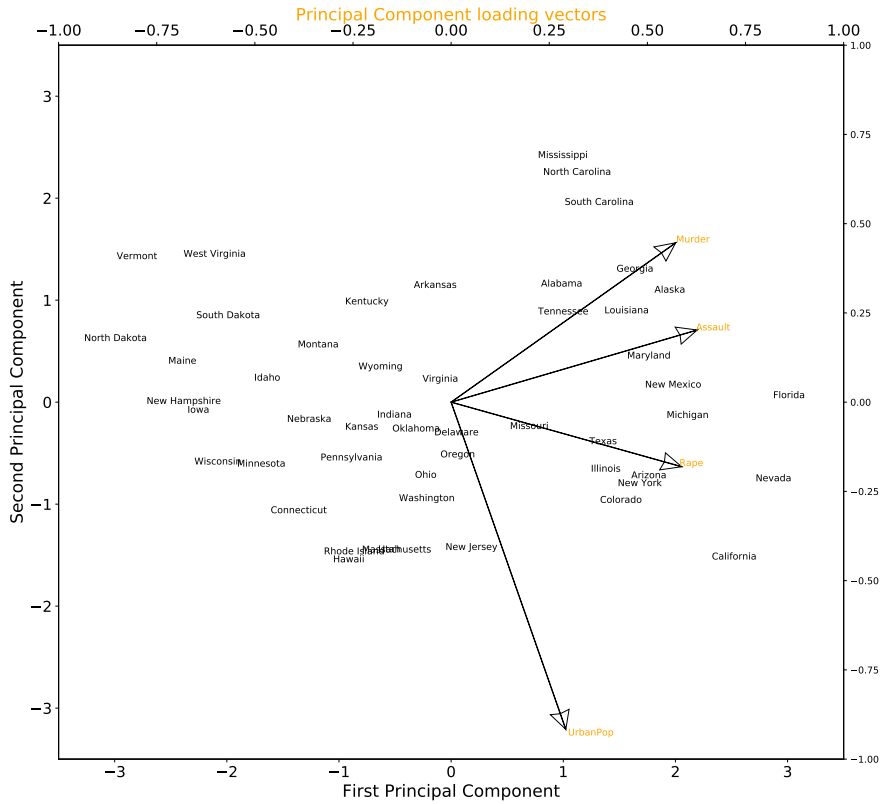
loading vectors

$$\boldsymbol{\varphi} = \begin{pmatrix} {\varphi^1}_1 & \cdots & {\varphi^1}_p \\ \vdots & \ddots & \vdots \\ {\varphi^p}_1 & \cdots & {\varphi^p}_p \end{pmatrix}$$

Rem: Typically one represents the data in terms of the two first PCs, $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$.

A **biplot** shows the original variables on the PC plot

Representation of your data points in ***2d*** using two first PCA and of the loading vectors

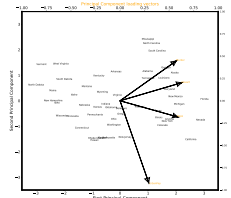


The loading vectors associated with $x_1, x_2, \ldots$ are $\varphi^1{}_i, \varphi^2{}_i, \ldots$ They tell us the importance of the original variable for each PC.

Unsupervised learning
└ Low dimensional representation of high dimensional data sets
└ Principal component analysis
└ A **biplot** shows the original variables on the PC plot

A **biplot** shows the original variables on the PC plot

Representation of your data points in $2d$ using two first PCA and of the loading vectors



The loading vectors associated with $x_1, x_2, ...$ are $\varphi^1_1, \varphi^2_1, ...$ They tell us the importance of the original variable for each PC.

For more details about this dataset and biplots : see chapter 10 of the book ISLR and exercises (ex_unsupervised.ipynb).

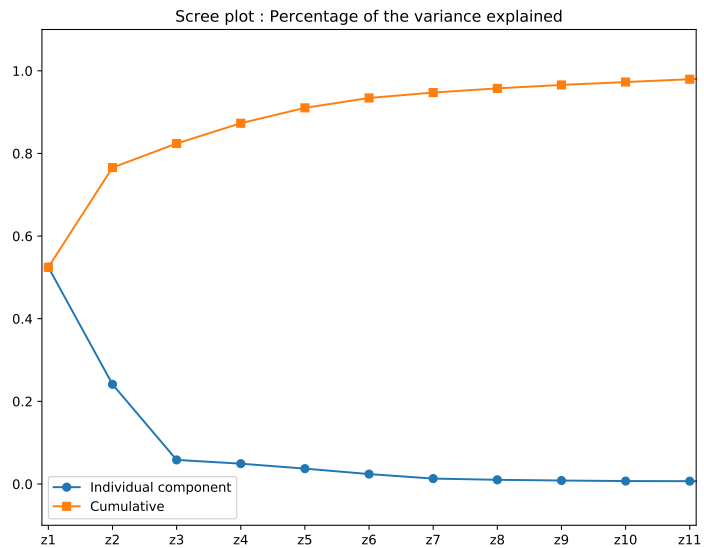
To read for instance the importance of 'UrbanPop' on z_1 (the first principal component), you need to project the loading vector 'UrbanPop' on z_1. We can see that the first PC is essentially an equal mix of the three crime variables and that the second PC is mostly the UrbanPop variable.

PCA : How many component to keep?

The variance 'explained' (or encoded) in the jth principal component is $\sigma^2_{z_j}$. Therefore the proportion of the variance explained by this component is

$$\text{PVE}_j = \frac{\sigma^2_{z_j}}{\sum_i \sigma^2_{z_i}}.$$

How many components should we keep? Draw a scree plot, showing the percentage of the variance explained as a function of the PCs included and decide according to it (and the problem at hand).

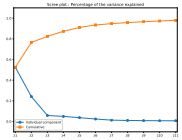


PCA : How many component to keep?

The variance 'explained' (or encoded) in the jth principal component is $\sigma_{z_j}^2$. Therefore the proportion of the variance explained by this component is

$$\text{PVE}_j = \frac{\sigma_{z_j}^2}{\sum_i \sigma_{z_i}^2}.$$

How many components should we keep? Draw a scree plot, showing the percentage of the variance explained as a function of the PCs included and decide according to it (and the problem at hand).



From the scree plot, we can decide in advance that we need to keep as many PCs as needed to explain e.g. 70 percent of the variance of the dataset. We can also inspect the scree plot and see if there is an "elbow" in the plot and keep PCs up to the elbow. In the case of our gut dataset, we would then keep 3 PCs (i.e. z_1, z_2, z_3).

Questions - check you can answer this easily!

1. What are the dimensions of:

The data matrix expressed in terms of the principal component coordinates Z ?
The principal component coordinates z?
The coordinate change matrix φ ?
The normalized data matrix $\hat{\boldsymbol{X}}$?
The covariance matrix (in original coordinates) Σ?

How to write $\Sigma_z = \frac{1}{n}\varphi^T \hat{X}^T \hat{X} \varphi$ with indices?

2. Why is the second PC associated with the eigenvector with the second highest eigenvalue? Of which matrix PCs are eigenvectors?

3. Why didn't I use the gut dataset to do a biplot?

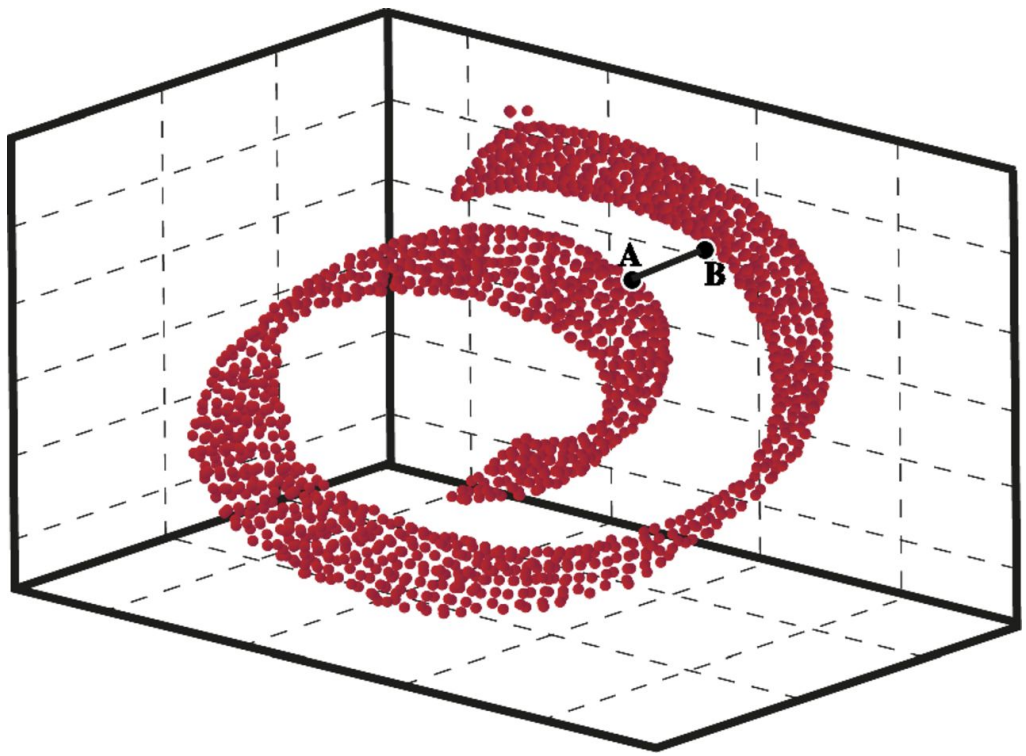
4. What is the difference between covariance and correlation?

5. If I perform twice a PCA on a dataset, will I get the exact same results?

More questions

Why is it interesting to build a variable whose variance is as high as possible? Is it always the most relevant property to look at?

PCA is good a preserving large pairwise distances from high to low dimensions, but is not good at conserving local structures in the data.
Let try something else!



(t)SNE

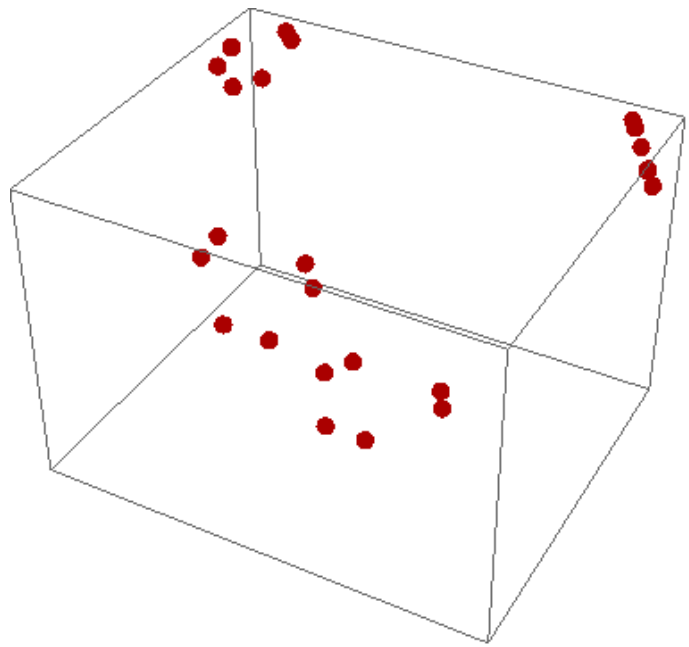
1. Objectives

2. Low dimensional representation of high dimensional data sets
    1. Principal component analysis
    2. (t)SNE

3. Clustering methods
    1. k-means clustering
    2. Dendrogram

SNE

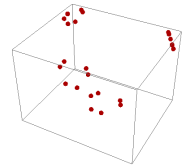
SNE, Stochastic neighborhood embedding, is a method that aims at preserving the local structure of the data. More precisely, it aims at matching the distribution of distances between points in high and low dimensional spaces via conditional probabilities.



We will measure all pairwise distances in the original high dimensional space, and associate a weight (between 0 and 1) to each of them. Then the idea is to do the same in $2d$ and find a way to preserve this "pairwise distance" structure.

SNE

SNE, Stochastic neighborhood embedding, is a method that aims at preserving the local structure of the data. More precisely, it aims at matching the distribution of distances between points in high and low dimensional spaces via conditional probabilities.



We will measure all pairwise distances in the original high dimensional space, and associate a weight (between 0 and 1) to each of them. Then the idea is to do the same in $2d$ and find a way to preserve this "pairwise distance" structure.

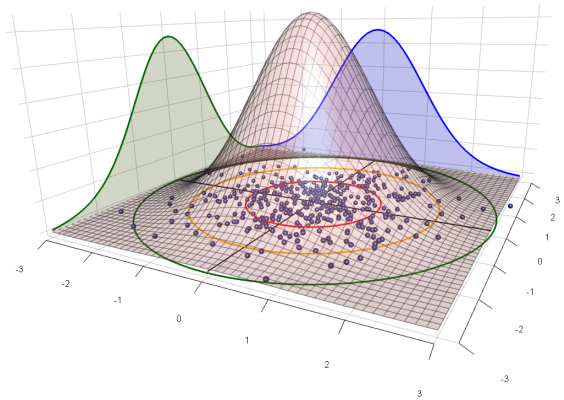
You start with your data matrix $\boldsymbol{X}$, then you will do the following:
(a) you start with the first sample ($\boldsymbol{X}[0, :]$ in python) which is a p-dimensional vector.
(b) you compute the distance between this sample and all other samples ($\boldsymbol{X}[1, :], \boldsymbol{X}[2, :], \ldots$) . This amounts to compute n distances in a p dimensional space.
(c) Let us call these distances $d_{01}, d_{02}, \ldots d_{0n}$. The distribution of these numbers give an idea of how close $\boldsymbol{X}[0, :]$ is to the other samples.
(d) repeat the procedure for all other samples.

SNE

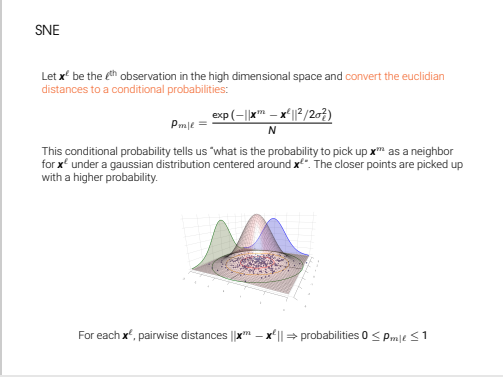
Let $\boldsymbol{x}^\ell$ be the ℓ^{th} observation in the high dimensional space and convert the euclidian distances to a conditional probabilities:

$$p_{m|\ell} = \frac{\exp\left(-||\boldsymbol{x}^m - \boldsymbol{x}^\ell||^2/2\sigma_\ell^2\right)}{N}$$

This conditional probability tells us "what is the probability to pick up $\boldsymbol{x}^m$ as a neighbor for $\boldsymbol{x}^\ell$ under a gaussian distribution centered around $\boldsymbol{x}^\ell$". The closer points are picked up with a higher probability.



For each $\boldsymbol{x}^\ell$, pairwise distances $||\boldsymbol{x}^m - \boldsymbol{x}^\ell|| \Rightarrow$ probabilities $0 \leq p_{m|\ell} \leq 1$



Let us focus on the first sample $\boldsymbol{X}[0, :]$ and the distances $d_{01}, d_{02}, \ldots d_{0n}$.

The idea is to assume that the distances to the first sample are distributed according to a gaussian distribution with a variance σ_1 (more later on how to choose σ_1). Then you can convert the distances $d_{01}, d_{02}, \ldots$ to probabilities $p_{1|0}, p_{2|0}, \ldots$ which are (obviously) all numbers between 0 and 1 and which tell you how likely it is to pick sample 1, sample 2,.. as a neighbor for the first sample (sample 0) assuming the distribution of pairwise distances is gaussian with σ_1..

For sample 0, we call the probabilities $p_{\ell|0}$. We can build up all conditional probabilities $p_{\ell|m}$. The goal will be to preserve those distributions in 2d.

SNE

Let us have a closer look at

$$p_{m|\ell} = \frac{\exp\left(-||\boldsymbol{x}^m - \boldsymbol{x}^\ell||^2/2\sigma_\ell^2\right)}{N}$$

- N is a normalisation factor $N = \sum_{m \neq \ell} \exp\left(-||\boldsymbol{x}^m - \boldsymbol{x}^\ell||^2/2\sigma_\ell^2\right)$
- $p_{m|\ell}$ is the probability that $\boldsymbol{x}^\ell$ would pick $\boldsymbol{x}^m$ as neighbor, if neighbors were picked in proportion to their probability under a Gaussian centered at $\boldsymbol{x}^\ell$ with variance σ_ℓ^2.
- σ_ℓ^2 is a variance that essentially characterize the **effective number of neighbors**. It is found by asking that the perplexity is fixed:

$$\text{Perp}(P_\ell) = 2^{H(P_\ell)}, \text{ where } H(P_\ell) = -\sum_m p_{m|\ell} \log_2 p_{m|\ell}\,.$$

  $H(P_\ell)$ is the Shannon entropy of P_ℓ.

So far, what we have done is only to characterize the high dimensional dataset.

The method gives a "weight" to each pairwise distance. The smaller the distance is, the highest the weight will be. The higher the weight is, the most important it is to conserve the pairwise distance in the low dimensional representation.

SNE

Let us have a closer look at

$$p_{m|\ell} = \frac{\exp\left(-||\boldsymbol{x}^m - \boldsymbol{x}^\ell||^2/2\sigma_\ell^2\right)}{N}$$

- N is a normalisation factor $N = \sum_{m \neq \ell} \exp\left(-||\boldsymbol{x}^m - \boldsymbol{x}^\ell||^2/2\sigma_\ell^2\right)$
- $p_{m|\ell}$ is the probability that $\boldsymbol{x}^\ell$ would pick $\boldsymbol{x}^m$ as neighbor, if neighbors were picked in proportion to their probability under a Gaussian centered at $\boldsymbol{x}^\ell$ with variance σ_ℓ^2.
- σ_ℓ^2 is a variance that essentially characterize the **effective number of neighbors**. It is found by asking that the perplexity is fixed:

$$\text{Perp}(P_\ell) = 2^{H(P_\ell)}, \text{ where } H(P_\ell) = -\sum_m p_{m|\ell} \log_2 p_{m|\ell}\,.$$

$H(P_\ell)$ is the Shannon entropy of P_ℓ.

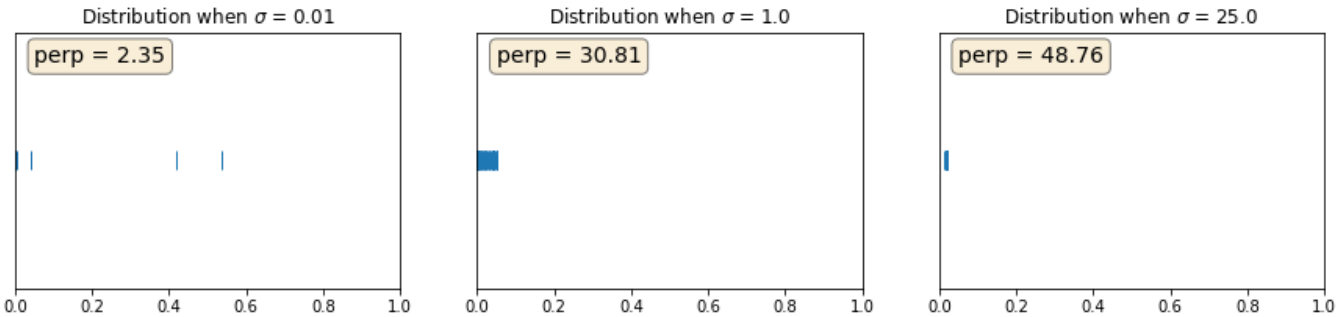
So far, what we have done is only to characterize the high dimensional dataset.

The method gives a "weight" to each pairwise distance. The smaller the distance is, the highest the weight will be. The higher the weight is, the most important it is to conserve the pairwise distance in the low dimensional representation.

The way to fix σ_ℓ with the perplexity is a bit subtle. You just know that we fix a different σ for each sample in such a way that all samples have a similar number of 'close' neighbors.

SNE

Some examples of discrete distributions and their perplexities (50 samples):



The perplexity (and entropy) for the first distribution is low because a lot of the ‘points’ have (almost) zero probability. Only a few points have finite probability and therefore the ‘unknown’ is not large.

For the second distribution, the perplexity (and entropy) increases because no ’point’ has zero probability.

In the third distribution, the perplexity increases because all points have almost same probability, ie equal change of being picked.

SNE

We want to find, for each $\boldsymbol{x}^\ell$, the corresponding point $\boldsymbol{y}^\ell$ in the low dimensional space (think a 2d space) in such a way that the $\boldsymbol{p}_{m|\ell}$ are preserved as much as possible for all ℓ in the low dimensional space.

For that, you need to construct these probability distributions in low dimension too:

$$q_{m|\ell} = \frac{\exp\left(-||\boldsymbol{y}^m - \boldsymbol{y}^\ell||^2\right)}{\sum_{m \neq \ell} \exp\left(-||\boldsymbol{y}^m - \boldsymbol{y}^\ell||^2\right)}$$

The idea is that you start by putting the points $\boldsymbol{y}^\ell$ (in 2d) associated with points $\boldsymbol{x}^\ell$ (in the high dimensional space) at random.

Then, step by step, we will move the points $\boldsymbol{y}^\ell$ in such a way that the distributions of pairwise distances in high dimension $\boldsymbol{p}_{m|\ell}$ and in 2d, $\boldsymbol{q}_{m|\ell}$, are the most 'similiar' to each other.

For that we need a manner to quantify the similarity between probability distributions...

SNE

A specific number to compare probability distributions is given by the **Kullback-Leibler divergence**:

$$KL(P||Q) = \sum_{\ell} \sum_{m \neq \ell} p_{m|\ell} \log \frac{p_{m|\ell}}{q_{m|\ell}}$$

The Kullback-Leibler divergence measures the (dis)similarity between two probability distributions. We want to minimize the KL between the high and low dimensional probability distributions.

Rem: The Kullback-Leibler diverenge is asymmetric: the cost is high if you represent two close points (in high d) by points that are far apart in 2d; however the cost is not high if you represent two far apart points (in high d) by close points (in 2d).

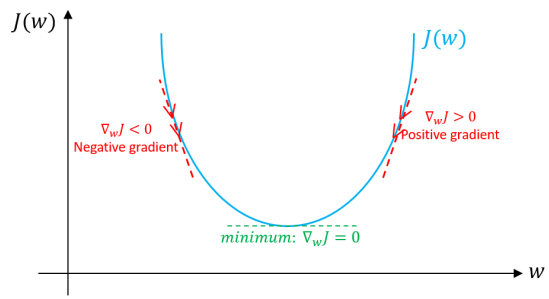
SNE

How to do in practice?

- Choose the y^ℓ using a uniform random distribution.
- Compute the cost function $C = KL(P||Q)$
- Compute the gradient of this cost function $\frac{\delta C}{\delta y_\ell} = 2\sum_m (p_{m|\ell} - q_{m|\ell} + p_{\ell|m} - q_{\ell|m})(y_\ell - y_m)$.
- update the y^ℓ

$$Y^{(t)} = Y^{(t-1)} - \eta \frac{\delta C}{\delta \boldsymbol{y}} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$$

where $Y^{(t)}$ indicates the solution at iteration t, η the learning rate, and $\alpha(t)$ represents the momentum at iteration t.



Why adding the momentum term? https://distill.pub/2017/momentum/

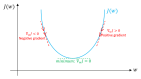
SNE

How to do in practice?

- Choose the y^ℓ using a uniform random distribution.
- Compute the cost function $C = KL(P||Q)$
- Compute the gradient of this cost function $\frac{\delta C}{\delta y_\ell} = 2\sum_m (p_{m|\ell} - q_{m|\ell} + p_{\ell|m} - q_{\ell|m})(y_\ell - y_m)$.
- update the y^ℓ

$$Y^{(t)} = Y^{(t-1)} - \eta\frac{\delta C}{\delta Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$$

where $Y^{(t)}$ indicates the solution at iteration t, η the learning rate, and $\alpha(t)$ represents the momentum at iteration t.



Why adding the momentum term? https://distill.pub/2017/momentum/

We want to minimize the dissimilarity between the distributions of pairwise distances in high and low d. For that we need to minimize the cost function (which is the Kullback-Leibler divergence between the distributions).ÃÂ

If you imagine that the cost function $C(w)$ depends only on one parameter w, you need to compute de derivative of the cost function with respect to this parameter $\frac{\partial C(w)}{\partial w}$ and update the parameter w to $w - \eta\frac{\partial C(w)}{\partial w}$ where η is called the learning rate. And then you proceed iteratively until you have find a minimal of $C(w)$. Techniques such as stochastic gradient descent exist to avoid getting stuck in a local minima.

tSNE

SNE is not good enough... difficult to optimize the cost function and there is 'crowding problem': SNE tends to clump all points close by and moderately far apart points together.

tSNE is a adaptation of SNE that

- replace the Gaussian distribution in the low dimensional space by a Student's t-distribution: $q_{m|\ell} = \frac{(1+||y_\ell - y_m||^2)^{-1}}{\sum_{n \neq \ell}(1+||y_n - y_\ell||^2)^{-1}}$
- symmetrize $p_{m|\ell}$

The gradient formula now takes the following form:

$$\frac{\delta C}{\delta y_\ell} = 4 \sum_m (p_{m|\ell} - q_{m|\ell})(y_\ell - y_m)(1 + ||y_\ell - y_m||^2)^{-1}$$

Example : Gene expression patterns and developing mouse brain anatomy are closely related

t-SNE PCA

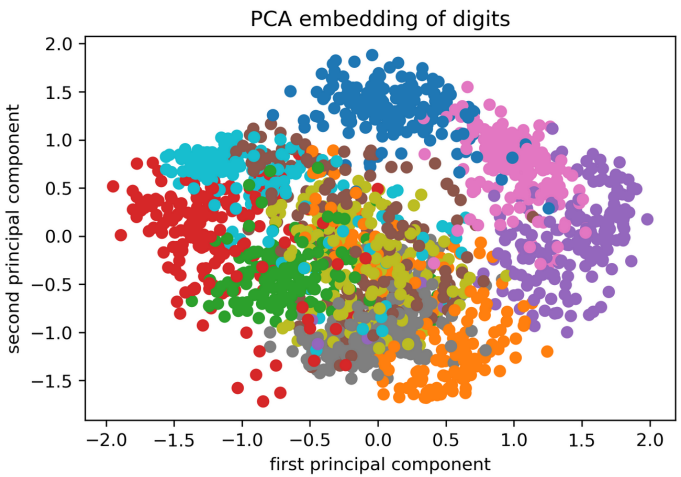
Left: t-SNE applied to gene expression data from different spatial regions in the brain of the mouse (about 2000 genes).

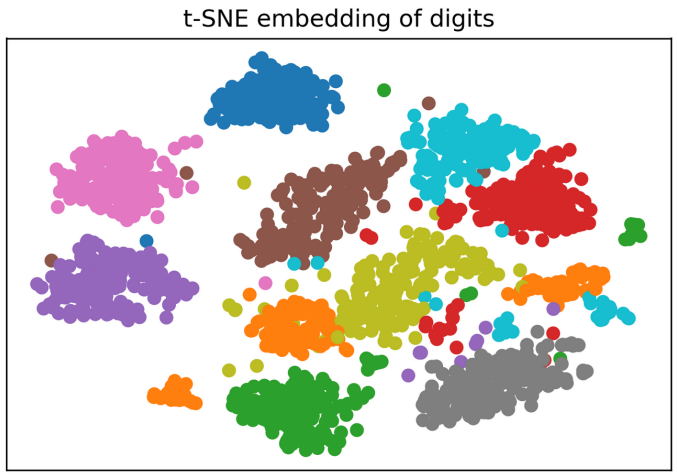
Middle: idem but PCA

Right: Brain anatomic regions - same color code for all figures.

[Ji, BMC Bioinformatics 2013]

Example : Digits recognition





Each color corresponds to one digit. t-SNE is clearly much better at recognizing digit images.

Clustering methods

1. Objectives

2. Low dimensional representation of high dimensional data sets
    1. Principal component analysis
    2. (t)SNE

3. Clustering methods
    1. k-means clustering
    2. Dendrogram

Clustering methods

Methods to find subgroups, or *clusters* in a data set. The points in a cluster must be similar, which implies that we need to define what we mean by 'similar' and by 'different'.

We will first consider *k-means clustering* which is a simple clustering technique.

Clustering methods

Methods to find subgroups, or *clusters* in a data set. The points in a cluster must be similar, which implies that we need to define what we mean by 'similar' and by 'different'.

We will first consider *k-means clustering* which is a simple clustering technique.

Clustering methods are covered in the reference book in the section 10.3.

The book can be downloaded here: **`http://faculty.marshall.usc.edu/gareth-james/ISL/`**

k-means clustering

1. Objectives

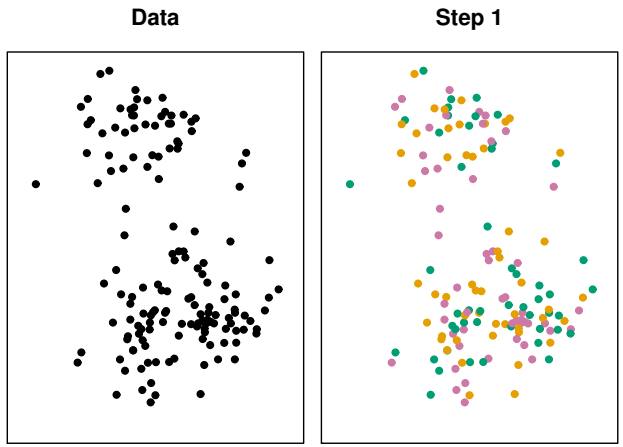
2. Low dimensional representation of high dimensional data sets
    1. Principal component analysis
    2. (t)SNE

3. Clustering methods
    1. k-means clustering
    2. Dendrogram

k-means clustering

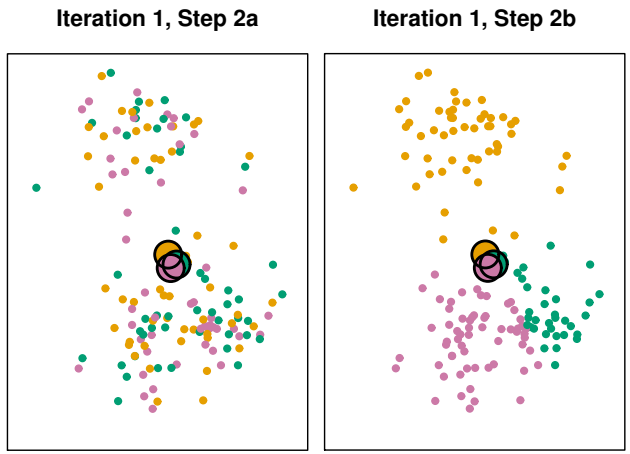
Step 1

Randomly assign a number, from 1 to *K*, to each of the observations. These serve as initial cluster assignments for the observations.



Step 2

(a) For each of the *K* clusters, compute the cluster centroid.
(b) Assign each observation to the cluster whose centroid is closest.



Iterate step 2 until the cluster assignments stop changing.ÃÂ

RemÂ : The cluster number is represented here by a color.

k-means clustering

The centroid of a cluster is essentially the center of the cluster. The jth component of the centroid of the cluster k can be computed as follows:

$$\bar{x}^{(k)}{}_j = \frac{1}{|C_k|} \sum_{\ell \in C_k} x^{\ell}{}_j$$

where $|C_k|$ is the number of samples in the cluster k and the sum goes over all samples in this cluster k.

k-means clustering

If I do twice the algorithm, will I obtain the same clusters?

If I do twice the algorithm, will I obtain the same clusters?

No, the algorithm depends on the initial random assignment.

k-means clustering

We need a measure of how tightly grouped all the points in the cluster are. For that, we will define a number $s(\ell)$ for each sample. This number measures how well the point belong to its cluster and it is based on two numbers:

- $a(\ell)$ tells how close the sample ℓ is to other samples from the same cluster

$$a(\ell) = \frac{1}{|C_{(\ell)}| - 1} \sum_{m \in C_\ell \, , \, m \neq \ell} d(m, \ell)$$

- $b(\ell)$ tells how close is sample ℓ from the closest neighboring cluster.

$$b(\ell) = \min_{m \neq \ell} \frac{1}{|C_m|} \sum_{m \in C_m} d(\ell, m)$$

The silhouette value of one sample is :

$$s(\ell) = \frac{b(\ell) - a(\ell)}{\max(a(\ell), b(\ell))} \quad \text{if } |C_\ell| > 1.$$

If $|C_\ell| = 1$, we define $s(\ell) = 0$.

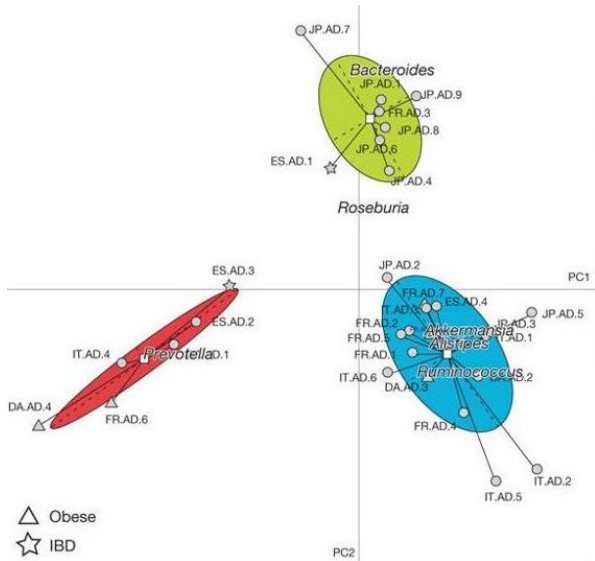
The silhouette index is defined as the mean value of the silhouette value for all points. It is by construction a number between -1 and 1. A good clustering will give a number close to 1.

How to assess the quality of clustering?

The silhouette index is by construction a number between -1 and 1.

A number close to one means a good clustering. Computing the silhouette indices for various ***k*** helps finding most appropriate number of clusters.

see : **`https://en.wikipedia.org/wiki/Silhouette_(clustering)`**



⇒ The 2d representation of the dataset suggests that you can belong to one of the three types (green, blue or red) of gut bacterial composition. They are called ‘enterotypes’.

ted talk (J. Raes KUL) : https://www.youtube.com/watch?v=Af5qUxl1ktI

Dendrogram

1. Objectives

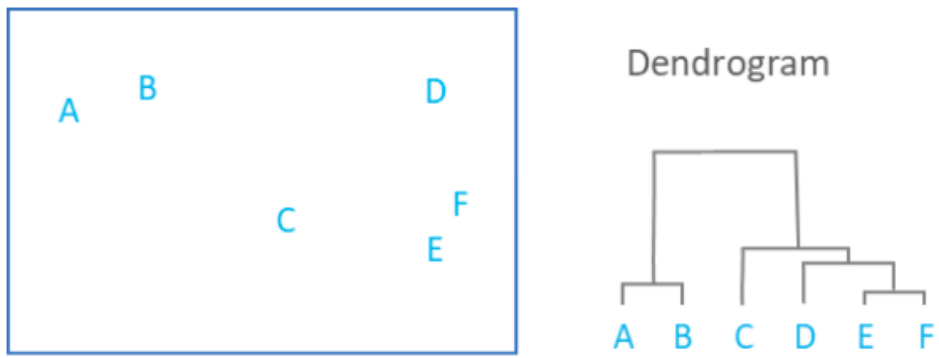
2. Low dimensional representation of high dimensional data sets
    1. Principal component analysis
    2. (t)SNE

3. Clustering methods
    1. k-means clustering
    2. Dendrogram

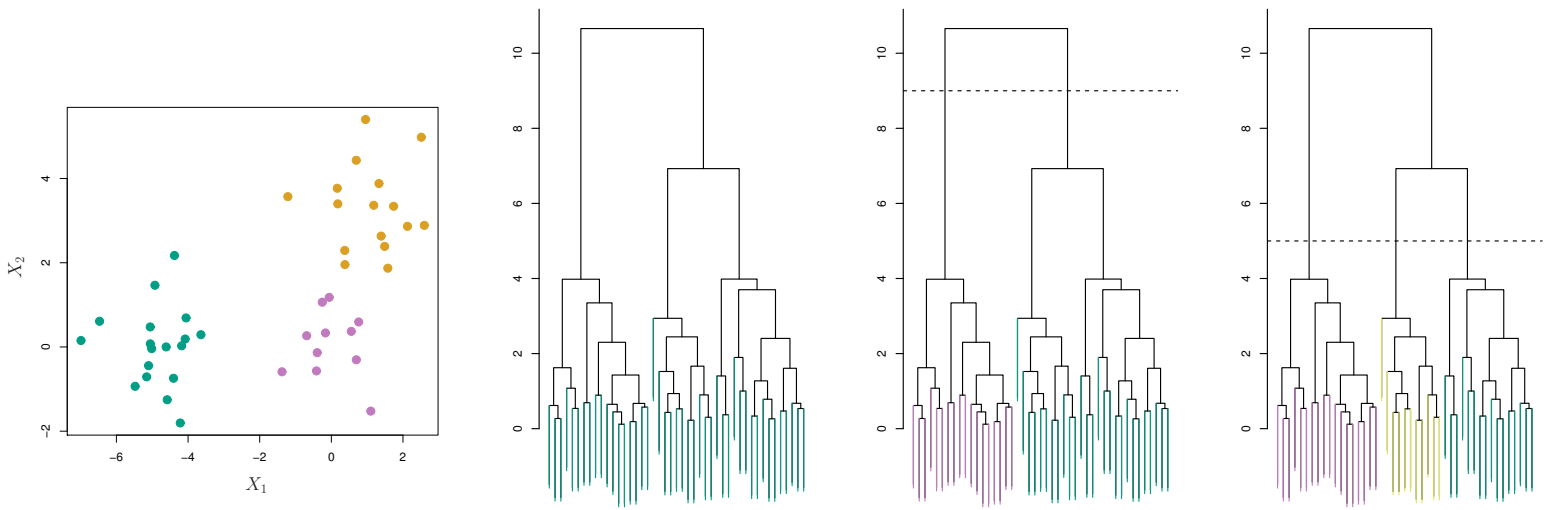
Dendrogram

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters, meaning that clusters can be compared (a cluster can be higher, lower or at the same level as another cluster).
Strategies for hierarchical clustering generally fall into two types:

- Divisive or top down approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.
- Agglomerative or bottom up approachÂ : each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. Example:



Dendrogram : interpretation



Each observation is represented as a 'leaf', in green on the second figure. Leaves are joined when you move upwards in the dendrogram according to their closeness (relative to the chosen measure).

Dendrogram : the algorithm

1. Each point is considered be belong to its own cluster.

2. Compute all pairwise distances between observations. The observations having the minimal distance are fused into one cluster.

3. Next we need to compute pairwise distance between the newly formed cluster and all other observations. For that we will to define how to compute these distances. Here are common choices:

- **Complete**: maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and observations in cluster B, and record the *largest* of these similarities.
- **Single**: minimal intercluster dissimilarity. Idem but with *smallest* distance recorded.
- **Average**: Mean intercluster dissimilarity. Idem but with *averaged* ÃÂÃÂÃÂÃÂ distance recorded.
- **Centroid** : Dissimilarity between the centroid for cluster A and B. Centroid linkage can result in undesired inversion.

4. Proceed similarly until all observations belong to only one cluster. You can then partition the data points by cutting the dendrogram at a given height (measuring the distances between groups of observations).

Clustering methods : caveats

Small decisions with big consequences:

- should the observations be standardized in some way? (e.g. zero mean and standard deviation one)
- for hierarchical clustering: what dissimilarity measure to use? which linkage? where to cut?
- for K-means clustering: how many clusters?

There is no simple right answer.

How to validate the cluster obtained? Techniques exist but there is no consensus.

Clustering is often not robust. For instance, if you remove a set of observations, you hoped to find the same clusters on the kept observations. It is often not the case.

Clustering results should not be taken as the absolute truth about a data set. Rather, they should constitute a starting point for the development of a scientific hypothesis and further study, preferably on an independent data set.

Summary

- Dimensional reduction methods such as PCA and t-SNE allow for a low-dimensional representation of the observations. The methods focus on preserving ‘something’ about the high-dimensional data, a good fraction of the variance and local structures respectively.
- Clustering methods seek for homogeneous subgroups among the observations.