Introduction to Artificial Intelligence (ENSIMAG) Intelligent Systems (MOSIG)

Some models for unsupervised and supervised learning

Original Slides by Clovis Galiez
Lecture: Sergi Pujades

2021-2022

Outline

- Unsupervised and supervised learning
- Unsupervised learning
    - EM
    - K-Means
    - PCA
    - t-SNE
- Supervised models
    - General setting
    - Logistic regression
    - SVM
    - Random forest

Supervised and unsupervised learning

Supervised and unsupervised learning

Make sense of the data

Example

Single nucleotide polymorphisms, frequently called SNPs (pronounced "snips"), are the most common type of genetic variation among people

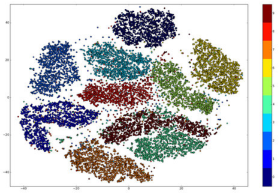
Patient	SNP1	SNP2	SNP3	...
A	0	0	1	...
B	1	0	1	...
C	1	0	0	...
...				

Example

Single nucleotide polymorphisms, frequently called SNPs (pronounced "snips"), are the most common type of genetic variation among people

Patient	SNP1	SNP2	SNP3	...
A	0	0	1	...
B	1	0	1	...
C	1	0	0	...
...				

$\rightarrow$

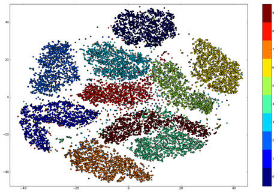


Example

Single nucleotide polymorphisms, frequently called SNPs (pronounced "snips"), are the most common type of genetic variation among people

Patient	SNP1	SNP2	SNP3	...
A	0	0	1	...
B	1	0	1	...
C	1	0	0	...
...				

$\rightarrow$



Techniques

Dimensionality reduction, clustering.

"Slightly" different data

From genotypes:

Patient	Status	SNP1	SNP2	SNP3	SNP4	...
A	**0**	0	0	0	1	...
B	**1**	1	0	0	1	...
C	**1**	1	0	0	0	...
...						

...

"Slightly" different data

From genotypes:

Patient	Status	SNP1	SNP2	SNP3	SNP4	...
A	**0**	0	0	0	1	...
B	**1**	1	0	0	1	...
C	**1**	1	0	0	0	...
...						

...

- discover a function $f(\text{SNP1}, \text{SNP2}, \text{SNP3}, \text{SNP4}, ...) = \text{Status}$ that predicts the status of a **(future)** patient,
- explain which SNP is important.

"Slightly" different data

From genotypes:

Patient	Status	SNP1	SNP2	SNP3	SNP4	...
A	**0**	0	0	0	1	...
B	**1**	1	0	0	1	...
C	**1**	1	0	0	0	...
...						

...

- discover a function $f(\mathrm{SNP1}, \mathrm{SNP2}, \mathrm{SNP3}, \mathrm{SNP4}, \ldots) = $ Status that predicts the status of a **(future)** patient,
- explain which SNP is important.

Techniques

Dimensionality reduction, regularization, supervised learning.

Learning: Make sense of data

Two cases:

- Data is only a set of points: $\mathcal{D} = (x_1, ...x_n)$ where $x_i \in \mathbb{E}^D$ is some (vector) space.

Learning: Make sense of data

Two cases:

- Data is only a set of points: $\mathcal{D} = (x_1, ...x_n)$ where $x_i \in \mathbb{E}^D$ is some (vector) space.
- Data is labelled: $\mathcal{D} = ((x_1, y_1), ...(x_n, y_n))$ where $y_i \in Y$ can be de discrete or continuous space.

Learning: Make sense of data

Two cases:

- Data is only a set of points: $\mathcal{D} = (x_1, ...x_n)$ where $x_i \in \mathbb{E}^D$ is some (vector) space. **Unsupervised learning**
- Data is labelled: $\mathcal{D} = ((x_1, y_1), ...(x_n, y_n))$ where $y_i \in Y$ can be de discrete or continuous space.**Supervised learning**

Technically, the inference methods algorithm are quite different.

Unsupervised learning

Goal

The goal is to make sense of the data.

Goal

The goal is to make sense of the data.

For this we define a model $\mathcal{M}(\theta)$ that have some parameters θ, and we try to get the model fit to the data.
How fit?

Goal

The goal is to make sense of the data.

For this we define a model $\mathcal{M}(\theta)$ that have some parameters θ, and we try to get the model fit to the data.
How fit? We describe the fit through a loss function:

$$\arg\min_{\theta} \mathcal{L}(x_1, ..x_n; \theta)$$

A loss can be thought as a kind of metric between the model and the real underlying model generating the data.

Goal

The goal is to make sense of the data.

For this we define a model $\mathcal{M}(\theta)$ that have some parameters θ, and we try to get the model fit to the data.
How fit? We describe the fit through a loss function:

$$\arg\min_{\theta} \mathcal{L}(x_1, ..x_n; \theta)$$

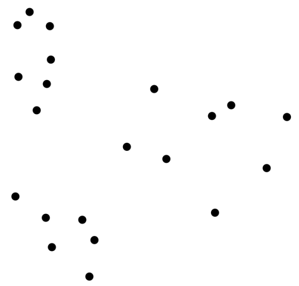
A loss can be thought as a kind of metric between the model and the real underlying model generating the data.

Typical example

Clustering and Gaussian Mixture models, Dimensionality reduction.

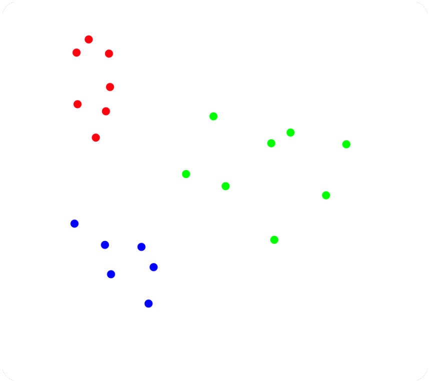
Clustering

Goal: find groups of similar data points.



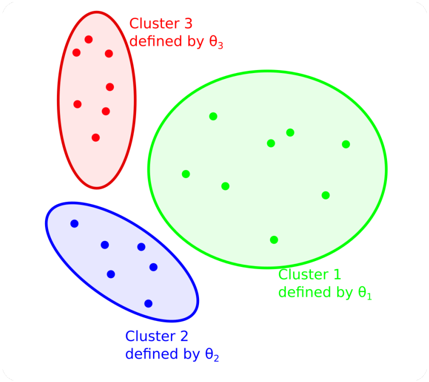
Clustering

Goal: find groups of similar data points.



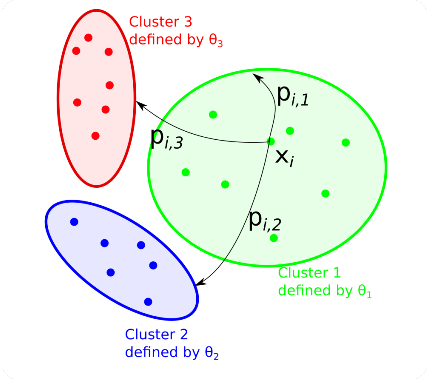
Clustering

Goal: find groups of similar data points.



Clustering

Goal: find groups of similar data points.



Clustering, formally

Formally:

- x_i data points
- $p_{i,k}$ probability of x_i to belong to a cluster k
- a parametric model for each cluster: $\mathcal{M}(\theta_k)$

Clustering, formally

Formally:

- x_i data points
- $p_{i,k}$ probability of x_i to belong to a cluster k
- a parametric model for each cluster: $\mathcal{M}(\theta_k)$

There are several usual variation around this idea. Some examples:

- one can force the $p_{i,k}$ to be in the set $\{0, 1\}$ (hard clustering), or rather to be continuous in $[0, 1]$ (soft clustering).
- for $\mathcal{M}(\theta_k)$, one can have a multivariate Gaussian $\mathcal{N}(\mu_k, I)$ generative model (example: k-means).

Clustering, formally

Formally:

- x_i data points
- $p_{i,k}$ probability of x_i to belong to a cluster k
- a parametric model for each cluster: $\mathcal{M}(\theta_k)$

There are several usual variation around this idea. Some examples:

- one can force the $p_{i,k}$ to be in the set $\{0, 1\}$ (hard clustering), or rather to be continuous in $[0, 1]$ (soft clustering).
- for $\mathcal{M}(\theta_k)$, one can have a multivariate Gaussian $\mathcal{N}(\mu_k, I)$ generative model (example: k-means).

The loss associating the set of parameters $p_{i,k}, \theta_k$ for $i \in 1, ...n$ and $k \in 1, ...K$ (note that K is not necessarily bounded: $K \in [|0, \infty|]$) is chosen consistently: if the model is naturally probabilistic, the loss is often the **negative log-likelihood** of the data.

Gaussian mixture

A typical example of (soft) clustering is the Gaussian mixture:

- $\mathcal{M}(\mu_k, \Sigma_k) = \mathcal{N}(\mu_k, \Sigma_k)$.
- $\forall i, \exists! k$ s.t. $p_{i,k} = 1$, the others are null: $p_{i,k' \neq k} = 0$. We note by z_i this specific k.

Loss: $\mathcal{L}(\mu, \Sigma, z; x) = \sum_i \log p(x_i | \mu_{z_i}, \Sigma_{z_i})$. (where $p_{i,z_i} = 1$).

Gaussian mixture

A typical example of (soft) clustering is the Gaussian mixture:

- $\mathcal{M}(\mu_k, \Sigma_k) = \mathcal{N}(\mu_k, \Sigma_k)$.
- $\forall i, \exists! k$ s.t. $p_{i,k} = 1$, the others are null: $p_{i,k' \neq k} = 0$. We note by z_i this specific k.

Loss: $\mathcal{L}(\mu, \Sigma, z; x) = \sum_i \log p(x_i | \mu_{z_i}, \Sigma_{z_i})$. (where $p_{i,z_i} = 1$).
How to minimize this loss?

Gaussian mixture

A typical example of (soft) clustering is the Gaussian mixture:

- $\mathcal{M}(\mu_k, \Sigma_k) = \mathcal{N}(\mu_k, \Sigma_k)$.
- $\forall i, \exists! k$ s.t. $p_{i,k} = 1$, the others are null: $p_{i,k' \neq k} = 0$. We note by z_i this specific k.

Loss: $\mathcal{L}(\mu, \Sigma, z; x) = \sum_i \log p(x_i | \mu_{z_i}, \Sigma_{z_i})$. (where $p_{i,z_i} = 1$).
How to minimize this loss?

Out-of-the-box optimization algorithms like (stochastic) gradient descent would struggle since it is not convex, not smooth (z_i are discrete).

Specific algorithm

EM Algorithm

EM framework for unsupervised classification (clustering)

2 facts: Estimating the best θ when the z_i are fixed can often be tractable (or approximated).
Estimating the best z when the θ_k are fixed can often be tractable (or approximated).

EM framework for unsupervised classification (clustering)

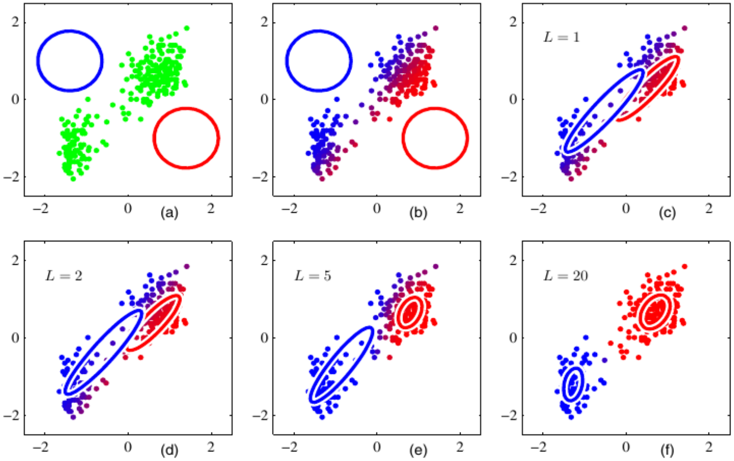
2 facts: Estimating the best θ when the z_i are fixed can often be tractable (or approximated).
Estimating the best z when the θ_k are fixed can often be tractable (or approximated). But both at the same time is hard!

EM framework for unsupervised classification (clustering)

2 facts: Estimating the best θ when the z_i are fixed can often be tractable (or approximated).
Estimating the best z when the θ_k are fixed can often be tractable (or approximated). But both at the same time is hard! The idea of EM algorithms is to break this dependency, by running iteratively these steps:

- **E(xpectation) step**: We fix a $\theta^{(t)}$ and this steps computes the distribution $z|\theta^{(t)}, x$
  This allows to estimate the loss as a function of θ as:
  $\mathcal{L}^{(t)}(\theta) = \mathbb{E}_{z|\theta^{(t)}}[\mathcal{L}(\theta, z; x)]$
- **M(aximization) step:** We optimize $\theta^{(t+1)} = \arg\min_\theta \mathcal{L}^{(t)}(\theta, z)$ with some optimization algorithm or analytical solution.

EM for Gaussian Mixture, visually



[Source: Bishop 06]

Exercise

In the specific case of isotropic Gaussian of variance 1 and that all cluster are equiprobable, compute the estimated loss as output of the E-Step.

Exercise

In the specific case of isotropic Gaussian of variance 1 and that all cluster are equiprobable, compute the estimated loss as output of the E-Step.

Solution

$$\log p(z_i = k|\theta^{(t)}, x) = -\frac{1}{2}||x_i - \mu_k^{(t)}||^2 + o(1)$$

Exercise

In the specific case of isotropic Gaussian of variance 1 and that all cluster are equiprobable, compute the estimated loss as output of the E-Step.

Solution

$$\log p(z_i = k|\theta^{(t)}, x) = -\frac{1}{2}||x_i - \mu_k^{(t)}||^2 + o(1)$$
$$p(z_i = k|\theta^{(t)}, x) = \frac{\exp(-\frac{1}{2}||x_i - \mu_k^{(t)}||^2)}{\sum_{k'} \exp(-\frac{1}{2}||x_i - \mu_{k'}^{(t)}||^2)}$$

Exercise

In the specific case of isotropic Gaussian of variance 1 and that all cluster are equiprobable, compute the estimated loss as output of the E-Step.

Solution

$\log p(z_i = k|\theta^{(t)}, x) = -\frac{1}{2}||x_i - \mu_k^{(t)}||^2 + o(1)$

$p(z_i = k|\theta^{(t)}, x) = \frac{\exp(-\frac{1}{2}||x_i - \mu_k^{(t)}||^2)}{\sum_{k'} \exp(-\frac{1}{2}||x_i - \mu_{k'}^{(t)}||^2)}$

$\mathcal{L}^{(t)}(\mu) = \sum_k \sum_i p(z_i = k|\theta^{(t)}, x).||x_i - \mu_k||^2$

Exercise

In the specific case of isotropic Gaussian of variance 1 and that all cluster are equiprobable, compute the estimated loss as output of the E-Step.

Solution

$\log p(z_i = k|\theta^{(t)}, x) = -\frac{1}{2}||x_i - \mu_k^{(t)}||^2 + o(1)$

$p(z_i = k|\theta^{(t)}, x) = \frac{\exp(-\frac{1}{2}||x_i - \mu_k^{(t)}||^2)}{\sum_{k'} \exp(-\frac{1}{2}||x_i - \mu_{k'}^{(t)}||^2)}$

$\mathcal{L}^{(t)}(\mu) = \sum_k \sum_i p(z_i = k|\theta^{(t)}, x).||x_i - \mu_k||^2 =$

$\sum_k \sum_i \frac{\exp(-\frac{1}{2}||x_i - \mu_k^{(t)}||^2)}{\sum_{k'} \exp(-\frac{1}{2}||x_i - \mu_{k'}^{(t)}||^2)}.||x_i - \mu_k||^2$

Exercise

In the specific case of isotropic Gaussian of variance 1 and that all cluster are equiprobable, compute the estimated loss as output of the E-Step.

Solution

$\log p(z_i = k|\theta^{(t)}, x) = -\frac{1}{2}||x_i - \mu_k^{(t)}||^2 + o(1)$

$p(z_i = k|\theta^{(t)}, x) = \frac{\exp(-\frac{1}{2}||x_i - \mu_k^{(t)}||^2)}{\sum_{k'} \exp(-\frac{1}{2}||x_i - \mu_{k'}^{(t)}||^2)}$

$\mathcal{L}^{(t)}(\mu) = \sum_k \sum_i p(z_i = k|\theta^{(t)}, x).||x_i - \mu_k||^2 =$

$\sum_k \sum_i \frac{\exp(-\frac{1}{2}||x_i - \mu_k^{(t)}||^2)}{\sum_{k'} \exp(-\frac{1}{2}||x_i - \mu_{k'}^{(t)}||^2)}.||x_i - \mu_k||^2$

What $\mu_k^{(t+1)}$ minimizes the loss?

Specific case: K-Means

If we make the following two restrictions:

- it supposes isotropy: $\forall k, \Sigma_k = I$ (as in previous exercise)
- artificially pushes the conditional probability of $p(z|\theta^{(t)})$ to 0 or 1 in the E-step.

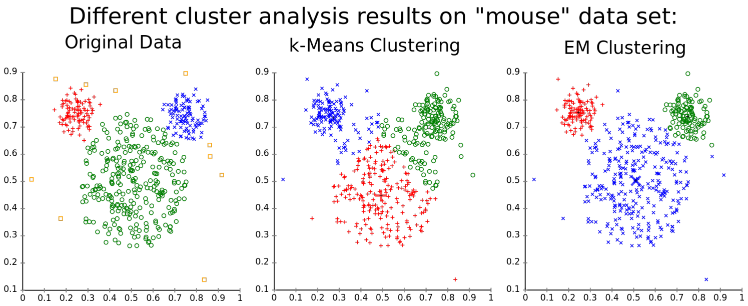
In the end, K-means simply consists of iteratively repeating:

- assign z_i to the closest center (defined by the $\theta_k^{(t)}$)
- define new centers $\theta_k^{(t+1)}$ as the barycenters of respectively the $\{x_i|z_i = k\}$

Until convergence.
The initialization is done with random arbitrary center.

Guarantees

There is a guarantee that the loss decreases along iteration, but it won't reach a global minimum in general.



[Source: Wikipedia]

Hierarchical clustering

Another useful family of clustering is called hierarchical clustering: it allows to vary the number of clusters in a hierarchical way.

Hierarchical clustering

Another useful family of clustering is called hierarchical clustering: it allows to vary the number of clusters in a hierarchical way.

Principle

There are many variants, but the common core principle is to greedily split (top-down clustering) or aggregate (bottom-up clustering) the data into smaller and smaller clusters.

Hierarchical clustering

Another useful family of clustering is called hierarchical clustering: it allows to vary the number of clusters in a hierarchical way.

Principle

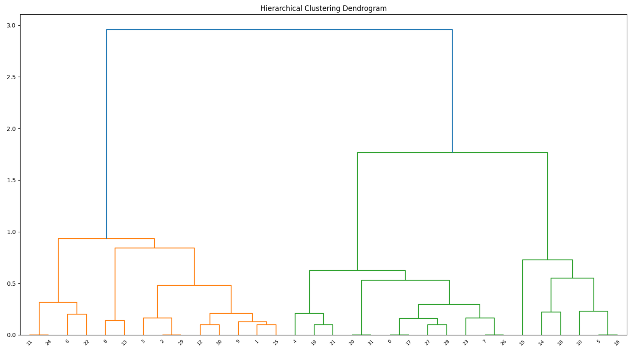
There are many variants, but the common core principle is to greedily split (top-down clustering) or aggregate (bottom-up clustering) the data into smaller and smaller clusters.

Here is the Ward's variant (bottom-up). It starts with every data point being a cluster (singleton) and then iteratively:

- find the pair of cluster which minimizes a loss if we fuse the two clusters (in general the variance, i.e. the sum of squared Euclidean distance from the barycenter)
- fuse this pair into a new cluster

Hierarchical clustering

The result of the clustering can be visualize as a tree (called a dendrogram):



The y-axis represents the loss corresponding to the fusions of clusters.

Dimensionality reduction: why?

Most of the time, modern data is high dimensional: data points are vectors $x \in \mathbb{E}^D$ with, say, $D > 10$, and $\mathbb{E}$ some data space (e.g. for instance $\mathbb{R}$, or $\mathbb{N}$).

Reducing the dimension consists into finding a (not necessarily linear) projection from $\mathbb{E}^D$ to $\mathbb{E}^d$ with $d < D$.

Dimensionality reduction: why?

Most of the time, modern data is high dimensional: data points are vectors $x \in \mathbb{E}^D$ with, say, $D > 10$, and $\mathbb{E}$ some data space (e.g. for instance $\mathbb{R}$, or $\mathbb{N}$).

Reducing the dimension consists into finding a (not necessarily linear) projection from $\mathbb{E}^D$ to $\mathbb{E}^d$ with $d < D$.

Reducing the dimensionality is useful for several reasons:

- Visualization purposes: look at the data in 2D or 3D
- Computing reasons: reduce the memory footprint
- Data science reason: denoise and remove the redundancy arising from correlated components
- Reduce risks of overfitting (more details in follow-up lectures) when in presence of big data with few labelled instances.

Dimensionality reduction: methods

Several methods exist for dimensionality reduction. We will detail two commonly used:

- Principal Component Analysis (PCA): based on linear algebra, represent well **global variation of the data**, deterministic, get a nice statistical interpretation
- t-SNE: non-linear, represent well **local tendencies** (closely related points), very powerful in practice but non-deterministic

Principal Component Analysis (PCA)

We represent a dataset of N data points of $\mathbb{E}^D$ in a matrix X of dimension $N \times D$.

Principal Component Analysis (PCA)

We represent a dataset of N data points of $\mathbb{E}^D$ in a matrix X of dimension $N \times D$.

For reducing the data down to d dimensions, PCA finds the linear projector M (matrix of dimension $D \times d$) minimizing the following distortion: $\sum_{i=1}^{N} ||x'_i - M^\top x'_i||^2$ where x'_i is the centred version of x_i:

$x'_{i,k} = x_{i,k} - \frac{1}{N} \sum_{j=1}^{N} x_{j,k}$.

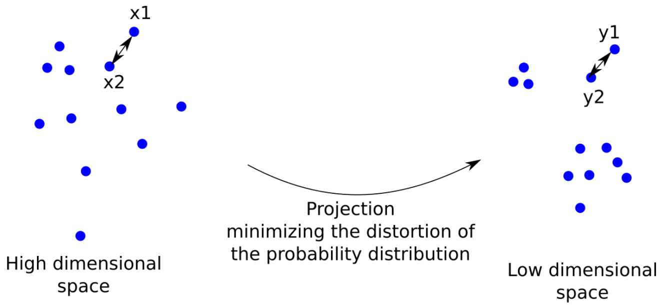
Principal Component Analysis (PCA)

We represent a dataset of N data points of $\mathbb{E}^D$ in a matrix X of dimension $N \times D$.
For reducing the data down to d dimensions, PCA finds the linear projector M (matrix of dimension $D \times d$) minimizing the following distortion: $\sum_{i=1}^{N} ||x'_i - M^\top x'_i||^2$ where x'_i is the centred version of x_i:
$x'_{i,k} = x_{i,k} - \frac{1}{N} \sum_{j=1}^{N} x_{j,k}$.

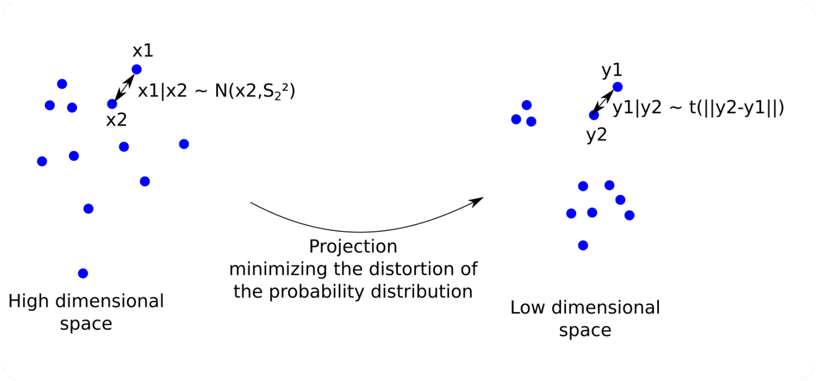
Eckart–Young–Mirsky Theorem

It turns out that with some linear algebra, the columns of M are the top-d eigenvectors of $X'^T X'$ (symmetric positive matrix of dimension $D \times D$).

t-distributed Stochastic Neighbor Embedding (t-SNE)


x1
x2
y1
y2
High dimensional space
Projection minimizing the distortion of the probability distribution
Low dimensional space

t-distributed Stochastic Neighbor Embedding (t-SNE)



Formally, t-SNE

t-SNE focuses on local similarity between points, by finding a projection that minimizes the distortion (measured by the Kullback-Leibler divergence) between the high and low dimensional space, equipped with probability distributions describing pair of points.

[1]We normalize the probabilities for pairs:

- In high dimension $h_{ij} \propto p(x_i|x_j) + p(x_j|x_i)$ and $\sum_{ij} h_{ij} = 1$
- In low dimension $l_{ij} \propto p(y_i|y_j)$ and $\sum_{ij} l_{ij} = 1$

Formally, t-SNE

t-SNE focuses on local similarity between points, by finding a projection that minimizes the distortion (measured by the Kullback-Leibler divergence) between the high and low dimensional space, equipped with probability distributions describing pair of points. More specifically,

- it endows the high dimensional space with $x_i|x_j \sim \mathcal{N}(x_j, \sigma_j^2 I)$
- the target low dimensional space with a t-distributed model: $y_i|y_j \sim t(y_i|y_j)$

The σ_j are set so that each $x|x_j$ has the same user defined entropy for all j.

[1] We normalize the probabilities for pairs:

- In high dimension $h_{ij} \propto p(x_i|x_j) + p(x_j|x_i)$ and $\sum_{ij} h_{ij} = 1$
- In low dimension $l_{ij} \propto p(y_i|y_j)$ and $\sum_{ij} l_{ij} = 1$

Formally, t-SNE

t-SNE focuses on local similarity between points, by finding a projection that minimizes the distortion (measured by the Kullback-Leibler divergence) between the high and low dimensional space, equipped with probability distributions describing pair of points. More specifically,

- it endows the high dimensional space with $x_i|x_j \sim \mathcal{N}(x_j, \sigma_j^2 I)$
- the target low dimensional space with a t-distributed model: $y_i|y_j \sim t(y_i|y_j)$

The σ_j are set so that each $x|x_j$ has the same user defined entropy for all j.
The loss[1] to be minimized is:

$$KL(H||L) = \sum_{i \neq j} h_{ij} \log \frac{h_{ij}}{l_{ij}}$$

[1]We normalize the probabilities for pairs:

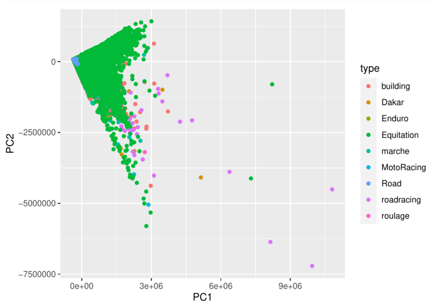
- In high dimension $h_{ij} \propto p(x_i|x_j) + p(x_j|x_i)$ and $\sum_{ij} h_{ij} = 1$
- In low dimension $l_{ij} \propto p(y_i|y_j)$ and $\sum_{ij} l_{ij} = 1$

PCA vs. t-SNE illustration

The same **accelerometer data** has been used for PCA and t-SNE, and no information about the clusters (walking, motoracing, etc.) have been provided (unsupervised):

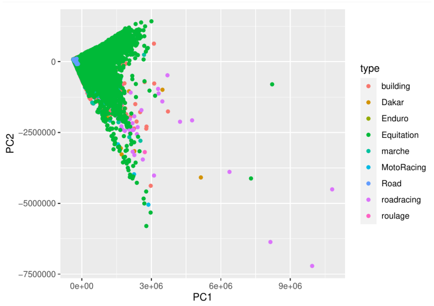
PCA vs. t-SNE illustration

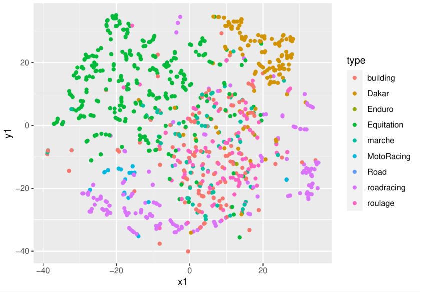
The same **accelerometer data** has been used for PCA and t-SNE, and no information about the clusters (walking, motoracing, etc.) have been provided (unsupervised):



PCA vs. t-SNE illustration

The same **accelerometer data** has been used for PCA and t-SNE, and no information about the clusters (walking, motoracing, etc.) have been provided (unsupervised):





Summary comparison tSNE vs PCA

Comparison between the methods

PCA has a 2 big advantages compared to t-SNE:

- It is **deterministic**
- the axis are **interpretable** as they are a linear combination of the variables (cf. stat lectures).
- no parameter to tune (target entropy in case of t-SNE)

t-SNE has the advantage at looking only at local scale, which is often relevant, and is **non-linear** projection method.

Other types of unsupervised learning

There are other methods for unsupervised learning that we will develop in the next lectures:

- Auto-encoders (AE)
- Generative Adversial Networks (GAN)

Metrics for unsupervised learning methods

There are two groups of metrics to evaluate the unsupervised methods:

- When labels are available:
  - AMI: Adjusted Mutual Information
  - Confusion Matrix
- When labels are not available (fully unsupervised)
  - Silhouette Index
  - Variance Ratio Criterion (Calinski-Harabasz index)

Unsupervised Metrics: Silhouette Index

For a data point $i \in C_I$ we compute the *mean intra-cluster distance*:

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i, j)$$

Unsupervised Metrics: Silhouette Index

For a data point $i \in C_I$ we compute the *mean intra-cluster distance*:

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i, j)$$

and the *mean nearest-cluster distance*:

$$b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i, j)$$

Unsupervised Metrics: Silhouette Index

For a data point $i \in C_I$ we compute the *mean intra-cluster distance*:

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i, j)$$

and the *mean nearest-cluster distance*:

$$b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i, j)$$

The silhouette index is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Unsupervised Metrics: Silhouette Index

For a data point $i \in C_I$ we compute the *mean intra-cluster distance*:

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i, j)$$

and the *mean nearest-cluster distance*:

$$b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i, j)$$

The silhouette index is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Best value is 1. Worst value is -1. 0 values indicate overlapping clusters. Negative values mean a different cluster is more similar.

Unsupervised Metrics: Calinski-Harabasz index

We note μ the center of all points and μ_I the center of each cluster.

Unsupervised Metrics: Calinski-Harabasz index

We note μ the center of all points and μ_I the center of each cluster.
We define the inter-group variance

$$B = \sum_{k=1}^{K} |C_k| \, ||\mu_k - \mu||$$

Unsupervised Metrics: Calinski-Harabasz index

We note μ the center of all points and μ_I the center of each cluster. We define the inter-group variance

$$B = \sum_{k=1}^{K} |C_k| \, ||\mu_k - \mu||$$

the intra-group variance as

$$W_k = \sum_{i \in C_k} ||x_i - \mu_k||$$

.

Unsupervised Metrics: Calinski-Harabasz index

We note μ the center of all points and μ_I the center of each cluster.
We define the inter-group variance

$$B = \sum_{k=1}^{K} |C_k| \, ||\mu_k - \mu||$$

the intra-group variance as

$$W_k = \sum_{i \in C_k} ||x_i - \mu_k||$$

.
The Calinski-Harabasz index is:

$$S_{CH} = \frac{(N-K)B}{(K-1)\sum_{k=1}^{K} W_k}$$