Unsupervised learning

Jordi Villà i Freixa

Universitat de Vic - Universitat Central de Catalunya
Study Abroad

jordi.villa@uvic.cat

course 2023-2024

Índex

Preliminary note

The material in these slides is strongly based on [1]. When other materials are used, they are cited accordingly.
Mathematical notation follows as good as it can a good practices proposal from the Beijing Academy of Artificial Intelligence.

What to expect?

In this session we will discuss:

- Unsupervised learning
- The Expectation-Maximization Algorithm
- Principal component analysis

Unsupervised learning

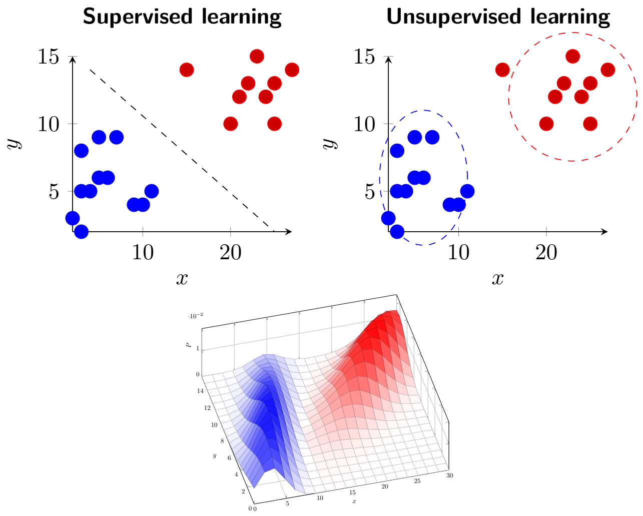


Figure 1: Supervised-unsupervised learning, and actual pdf behind the data

Unsupervised learning

is the process of inferring hidden patterns from historical data. We try here to find similarities, differences, patterns and structure in data by itself, without human intervention. Within this set of techniques, one can find many different real life applications:

- data exploration, preparation or visualization,
- customer segmentation,
- anomaly detection,
- target marketing campaigns,
- recommender systems, and many more.

We use them, in general, for clustering and association problems. We exect less accurate results. Some algorithms: K-Means, Gaussian Mixture models, Frequent Pattern (FP) Growth, Principal Component Analysis, Hierarchical Clustering Analysis, etc.

- Unsupervised learning is useful for data analysis when we do not know what are we looking for in the data. finding similarities, creating groups, etc. for example, by categorizing elements by their activity (cells with respect to their RNA expression, customer segmentation, etc).
- Data does not need to be labelled, and such datais much easier and faster to obtain.
- We can now find unknown paterns and useful insights in data.
- Last, but equally important, it reduces human biass and error.

Generating data

Exercise 1

Generate bivariate data Generate data from a collection of three different bivariate distributions with arbitrary means and covariances.

Joint probability distribution

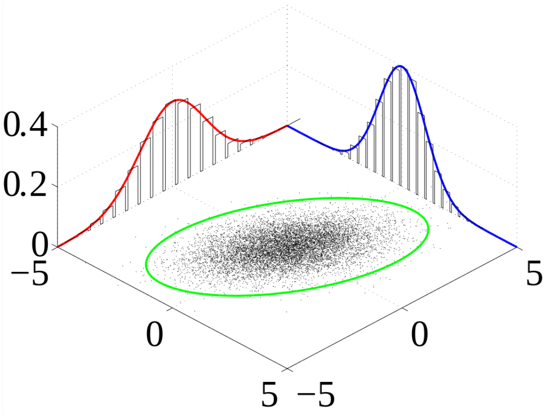


Figure 2: Multivariate normal sample

Maximum likelihood and Expectation-Maximization algorithm

We already saw how to use the K-means algorithm, based on the *distances* of the different data points. Now we will use another method that is based on assuming a given distribution of the data.

Maximum likelihood estimation (MLE) is an approach to density estimation for a dataset by searching accross probability distributions and their parameters. However, it needs that *latents* (or hidden) variables not to exist in the training dataset. It requires the dataset to be complete, or fully observed: that all variables that are relevant to the problem are present.

The **expectation-maximization (EM)** algorithm is an approach for performing MLE when latent variables are present. It is common to use it in estimating the density when data is missing, such as clustering in the **Gaussian Mixture Model**.

The EM algorithm

Let us assume a *mixture model*: unspecified combinadtion of multiple probability distribution functions. In our case, we will deal with the *Gaussian mixture model (GMM)*, where the parameters to estimate are the mean and the standard deviation for each Gaussian (normal) distribution in the mixture.

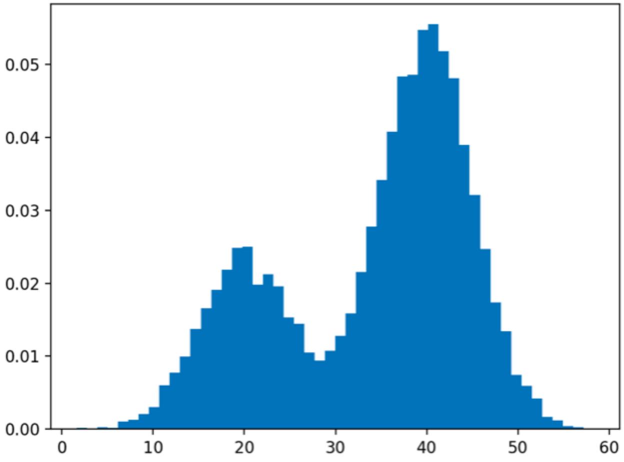
1. E-step: Estimate the missing variables in the dataset: the expected value for each latent variable.
2. M-step: Maximize the parameters of the model in the presence of the data, using MLE.

A one dimensional example for EM

Adapted from here

```
# example of a bimodal constructed from two gaussian p
from numpy import hstack
from numpy.random import normal
from matplotlib import pyplot
# generate a sample
X1 = normal(loc=20, scale=5, size=3000)
X2 = normal(loc=40, scale=5, size=7000)
X = hstack((X1, X2))
# plot the histogram
pyplot.hist(X, bins=50, density=True)
pyplot.show()
```

A one dimensional example for EM



A one dimensional example for EM

```
from numpy import hstack
from numpy.random import normal
from sklearn.mixture import GaussianMixture
# generate a sample
X1 = normal(loc=20, scale=5, size=3000)
X2 = normal(loc=40, scale=5, size=7000)
X = hstack((X1, X2))
# reshape into a table with one column
X = X.reshape((len(X), 1))
# fit model
model = GaussianMixture(n_components=2, init_params='random
model.fit(X)
yhat = model.predict(X) # predict latent values
print(yhat[:100])       # check latent value for first few
print(yhat[-100:])      # check latent value for last few p
```

The maths behind

$$\mu_k = \frac{\sum_{i=1}^{n} p_i^{(t)}(k)\mathbf{x}_i}{\sum_{i=1}^{n} p_i^{(t)}(k)}$$

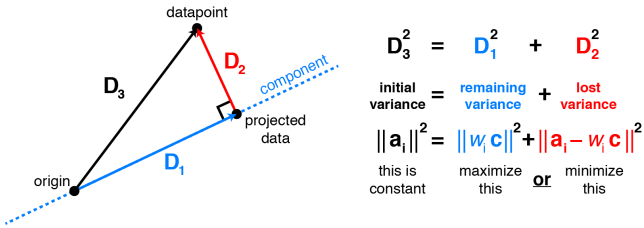
$$\mathrm{Cov}_k = \frac{\sum_{i=1}^{n} p_i^{(t)}(k)(\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T}{\sum_{i=1}^{n} p_i^{(t)}(k)}$$

where we estimate the pdf by using the previous iteration expectation and covariace:

$$p_i^{(t)}(k) \propto w_k^{(t-1)} \phi_k(\mathbf{x}_i | \mu_k^{(t-1)}, \mathrm{Cov}_k^{(t-1)})$$

PCA

1. Get some data
2. Substract the mean from each data dimension. Better standardize: $z = \frac{\mathbf{x} - \bar{\mathbf{x}}}{\sigma}$. Calculate the covariance matrix XX^T.
3. Diagonalize and obtain unit eigenvectors $c^T XX^T c - \lambda c^T c$ via SVD: $XX^T = UD^2U$
4. Choosing components and forming a feature vector
5. Recast the data along the chosen PC axes. $\mathbf{x}_i^{\mathrm{PC}} = U_k U_k^T \mathbf{x}_i$



datapoint
D_3
D_2
component
projected data
D_1
origin
$D_3^2 = D_1^2 + D_2^2$
initial variance = remaining variance + lost variance
$\|\mathbf{a}_i\|^2 = \|w_i\,\mathbf{c}\|^2 + \|\mathbf{a}_i - w_i\,\mathbf{c}\|^2$
this is constant
maximize this
or
minimize this

Figure 3: Maximizing variance in principal component space is equivalent to minimizing least-squares reconstruction error. Taken from this post

EX2

Exercise 2

EX2. PCA example Get the *Iris* dataset from the UCI repository.

1. Using the **kdeplot** method of **seaborn**, generate a figure for the kernel density plots of the variable *Petal.length* for each of the three species of *Iris sp.* in the dataset.
2. Show a scatter plot of the variable Petal.length vs the variable Sepal.length and comment on the correlation you see.
3. In order to analyze the possible correlatoons between the different data in the file, perform a PCA and obtain the principal component matrix U as well as the diagonal matrix D^2. What can you say in terms of the variance?
4. Project the data on the first component and plot the kernel density estimate of the PCA-combined data.

See also implementations in R of hierarchical clustering in this famous dataset.

Hierarchical clustering

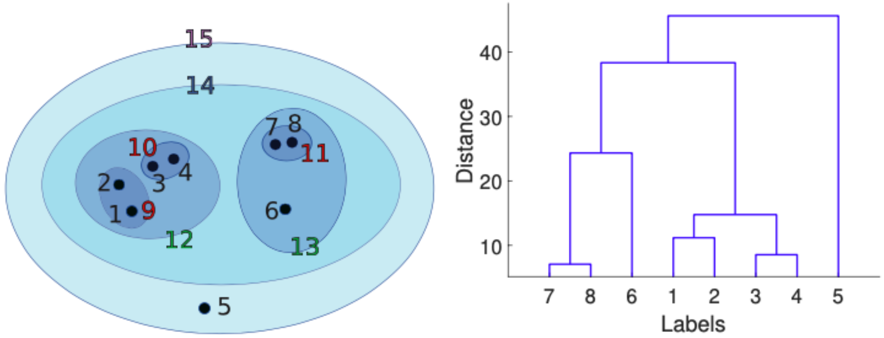


Figure 4: Left, a cluster hierarchy of 15 clusters. Right, the corresponding dendrogram[1].

Hierarchical clustering

In hierarchical cluster analysis (or HCA) one seeks to build a hierarchy of clusters with one of these two strategies:

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

Merges and splits are determined in a **greedy** manner. Results are usually presented in a dendrogram. Any valid measure of distance can be used. In fact, the observations themselves are not required: all that is used is a matrix of distances. Except for some specific cases, the algorithms cannot be guaranteed to find the optimum solution.

HCA flavours

Different types of HCA can be done, depending on the way the distance between points (typically the Euclidean distance) and between clusters I and J (how the clusters are linked: *linkage criterion*) is computed:

- Single linkage: The closest distance between the clusters

$$d_{\min} = \min_{i \in I, j \in J} \mathrm{dist}(\mathbf{x}_i, \mathbf{x}_j)$$

- Complete linkage: The farthest distance between the clusters

$$d_{\max} = \max_{i \in I, j \in J} \mathrm{dist}(\mathbf{x}_i, \mathbf{x}_j)$$

- Group average: The mean distance between clusters
- Ward's minimal variance: distance computed as the additional amount of "variance" that would be introduced if two clusters would be merged.

Algorithm for HCA

Algorithm 1: Greedy Agglomerative Clustering

```
Input: Distance function; linkage function $d$; number of clusters $K$.
Output: The labels set for the tree
1 Initialize the set of $n$ cluster identifiers $I$; Initialize the corresponding
   label sets $L_i$; Initialize a distance marix $D = [d_{ij}]$;
2 for $k = n + 1$ to $2n - K$ do
3     Find $i$ and $j < i$ in $I$ such that $d_{ij}$ is minimal; Create a new label
       set $L_k$; Add the new identifier $k$ to $I$ and remove the old
       identifiers $i$ and $j$ from $I$; Update the distance matrix $D$ with
       respect to $i$ and $j$;
4 return $L_i$
```

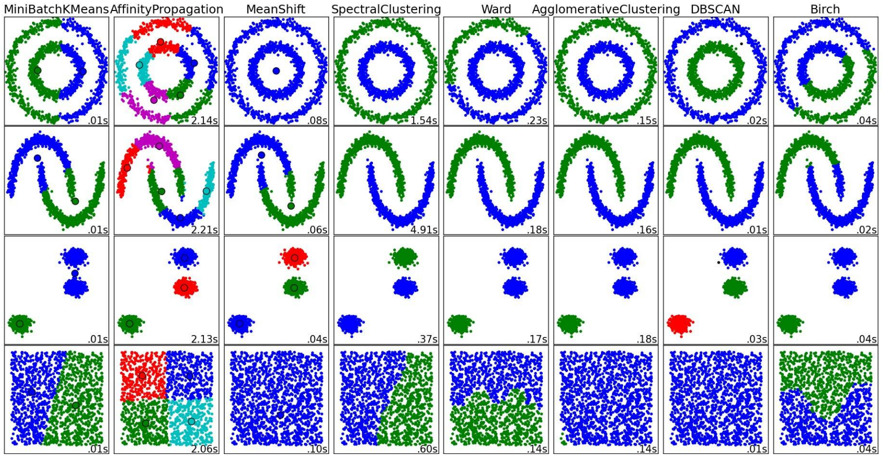


Figure 5: Overview of clustering methods from the scikit site

Density-Based Clustering of Applications with Noise (DBSCAN)

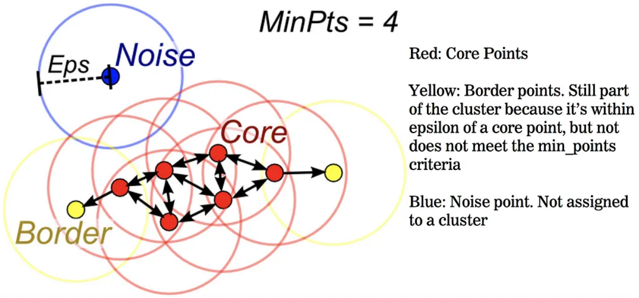

MinPts = 4
Eps
Noise
Core
Border
Red: Core Points
Yellow: Border points. Still part of the cluster because it's within epsilon of a core point, but not does not meet the min_points criteria
Blue: Noise point. Not assigned to a cluster

Figure 6: Schema of DBSCAN clustering for the distinction of hiugh to low density data clusters. Obtained from the Medium web site. See also a good explanation of the difference between HCA and DBSCAN here.

Exercise on clustering

Exercise 3

Testing DBSCAN Try to use HCA and DBSCAN on the *Iris* dataset. You can use as a template the work done on the Breast Cancer Wisconsin (Diagnostic) Data Set. Which one of the two methods can provide better clustering of the data? What oher methods would perform better to separate the three species?

Dirk P. Kroese, Zdravko Botev, Thomas Taimre, and Radislav: Vaisman.
Data Science and Machine Learning: Mathematical and Statistical Methods.
Machine Learning & Pattern Recognition. Chapman & Hall/CRC, 2020.

FACULTAT DE CIÈNCIES, TECNOLOGIA I ENGINYERIES
UVIC | UVIC·UCC