Machine Learning for researchers (300001030)
Unit 2. Unsupervised Methods

Javier Vales Alonso

Doctorate transversal activity

2020

Technical University of Cartagena

How to study this unit?

1. Do a first reading of the unit's slides. Try to focus on the general ideas and do the first review over the maths involved.
2. Next, read and run the notebook section by section. Leave the questions indicated there for later
3. Do a more in-depth review of the maths with the slides and the notebook side by side. Try to understand all the developments involved. In case of doubts, read the suggested references (see Syllabus) or contact the teacher.
4. Finally, answer the exercises in the notebook and submit them back through AV

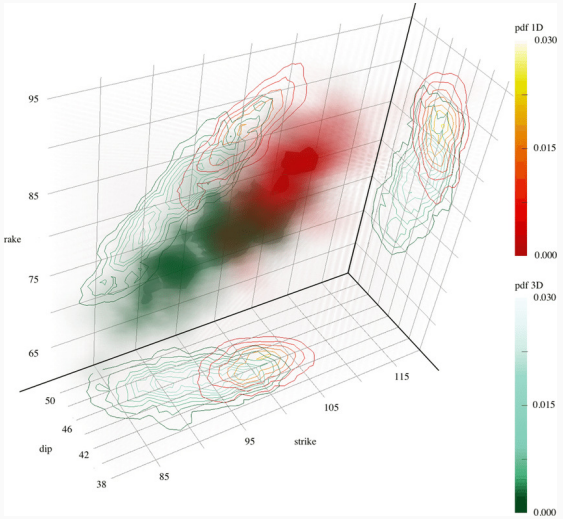
Introduction

Why unsupervised learning?

Labeling data is usually a costly, lengthy, difficult, and *highly underestimated* process. For example, consider a data set with millions of faces, which we would like to classify according to the "sentiment" they express (e.g., surprise, joy, sorrow, etc.). Selecting the prevalent sentiment shown can be cumbersome, and would be an error-prone process if not done with high-quality standards. The probable result is either obtaining a data set with junk or using vast resources to construct it.

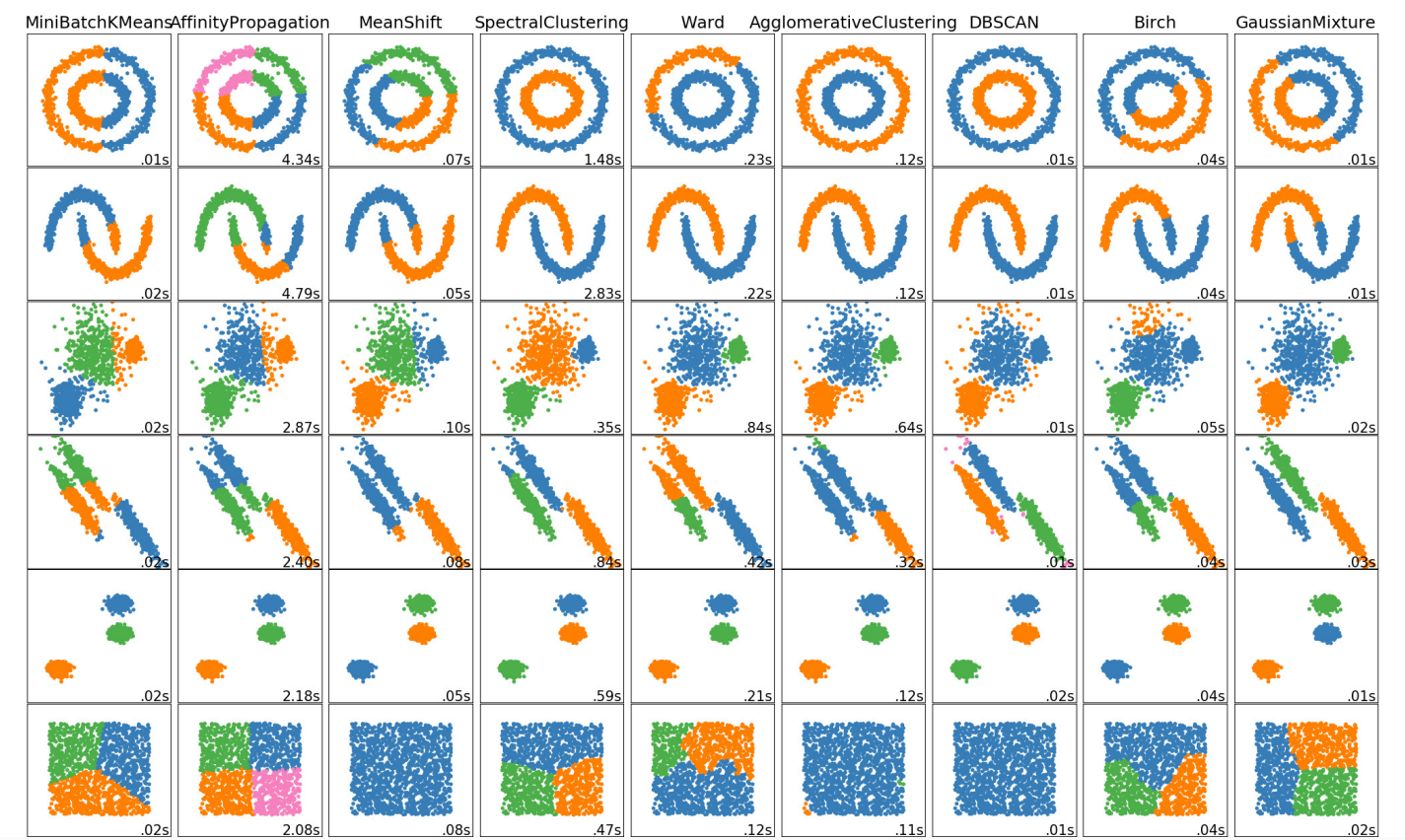
Unsupervised learning is an alternative when data is available, but we lack proper labeling. Its main goals are discovering similarities (patterns), detecting anomalies, and data preprocessing (including automatic labeling).

Examples of unsupervised learning



- Density estimation.
- Whereas one of the aims of supervised learning aims is computing the posterior distribution $p(\boldsymbol{x}|t)$ (the distribution of points according to its class), density estimation methods aim at inferring the prior distribution $p(\boldsymbol{x})$.
- Determining the prior $p(\boldsymbol{x})$ is useful for a variety of applications: data visualization, anomaly detection, enhanced posterior probability computation, etc.

Examples of unsupervised learning (II)

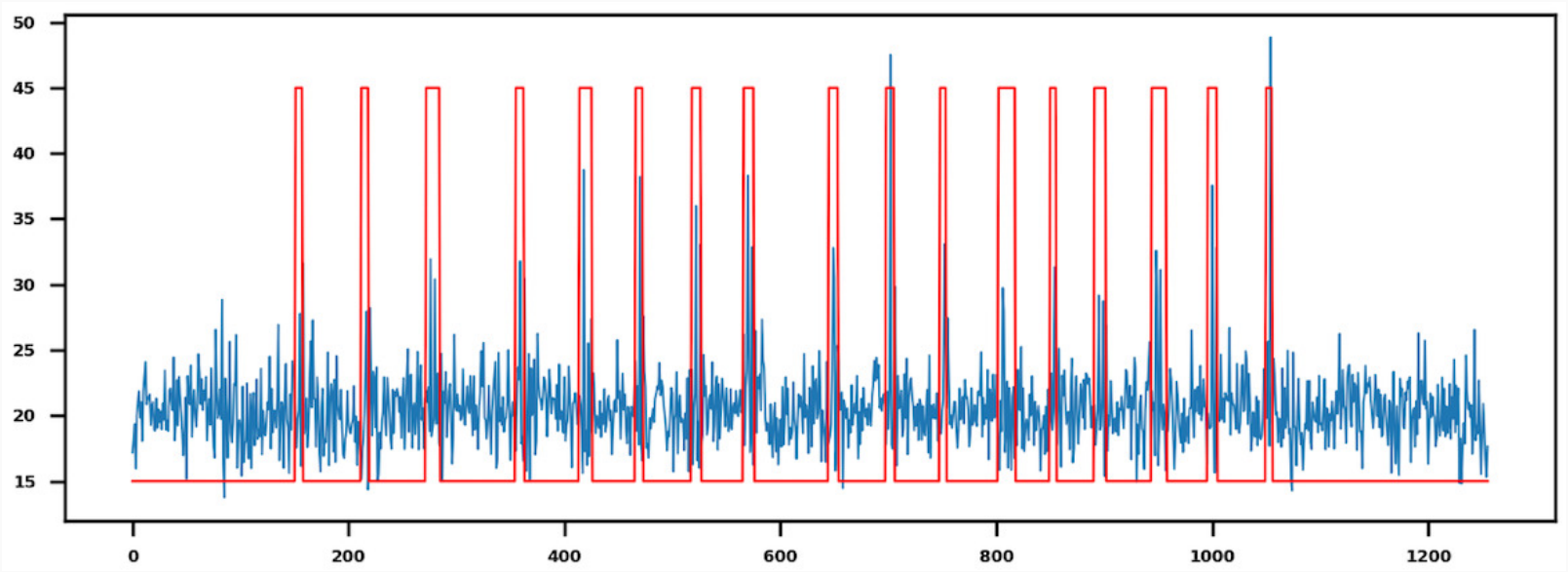


- Data clustering.
- Group instances by commonality.
- Essential tool for data mining in many scientific fields, e.g. economics and finance, astronomy, biology, etc.

Examples of unsupervised learning (III)

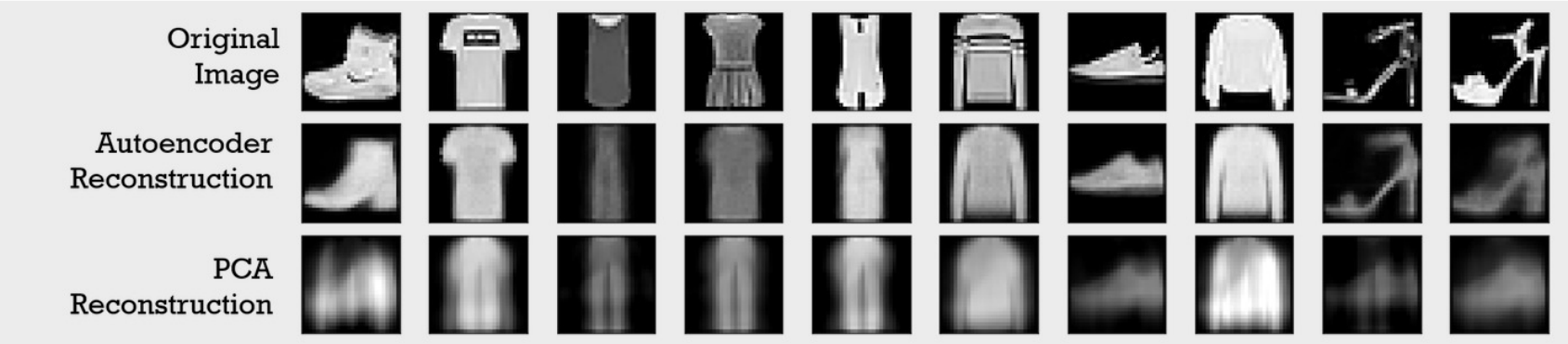
- Image segmentation and general data "quantization".
- Similar pixels are grouped or *clustered* together.
- Useful for image preprocessing before applying other computer vision methods and data compression.

Examples of unsupervised learning (IV)



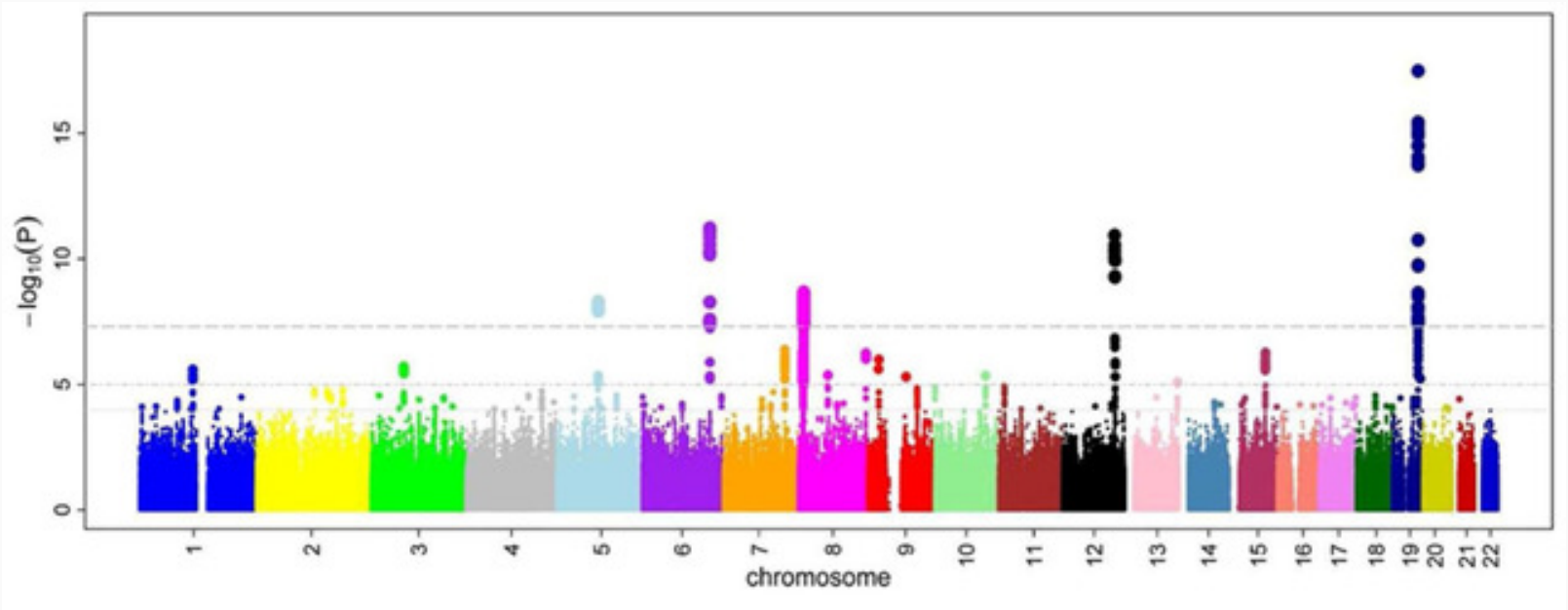
- Outlier and novelties detection.
- Find points in the data set (including time series) which seems different that the rest
- Useful in many fields, e.g., predictive maintenance, forecasting, sensor data pre-analysis, etc.

Examples of unsupervised learning (V)



- Autoencoder.
- Find efficient coding of data.
- Generate artificial data.
- Useful for dimensionality reduction, feature learning, generative models and outlier detection among other usages.

Examples of unsupervised learning (VI)



- Association rule learning.
- Discovering unapparent relations between variables.
- Useful for market studies like basket analysis, bioinformatic applications like genome-wide association study, etc.
- Related to sequence mining, though the later has into account the temporal order, while association rule learning hasn't.

Clustering

K-means

As in the previous unit, let us assume a data set consisting of N instances/observations $\{x_1, x_2, \ldots, x_N\}$ in a D-dimensional input space.

Clustering algorithms aim at identifying groups (clusters) of "similar" data points. K-means is a clustering algorithm whose intuitive idea is grouping together "close" data points while separating "far" ones. The algorithm assumes a fixed number of K clusters (a common way to select K is the elbow method, which is discussed in slide 17).

Henceforth, let $\boldsymbol{\mu}_k$ be a D-dimensional point representing the k^{th} cluster center (or *centroid* for short), for $k{=}1, 2, \ldots, K$.

K-means (II)

Besides, let $\boldsymbol{r}_{nk}$ be binary variables with value 1 if data point $\boldsymbol{x}_n$ is assigned to cluster k, or 0 otherwise, for $n{=}1, \ldots, N$ and $k{=}1, \ldots, K$.

To characterize how good a cluster assignment is, a *distortion measure* is defined:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} \boldsymbol{r}_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2 \tag{1}$$

The goal is to find the values $\{\boldsymbol{r}_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$ minimizing cost J.

K-means (III)

J can be minimized performing repeatedly the next two phases:

1. Minimize J by selecting the best assignment to clusters $\{\boldsymbol{r}_{nk}\}$ keeping centroids $\{\boldsymbol{\mu}_k\}$ fixed. It is done trivially, by selecting for each point $\boldsymbol{x}_n$ the cluster with the closer centroid $\boldsymbol{\mu}_k$,

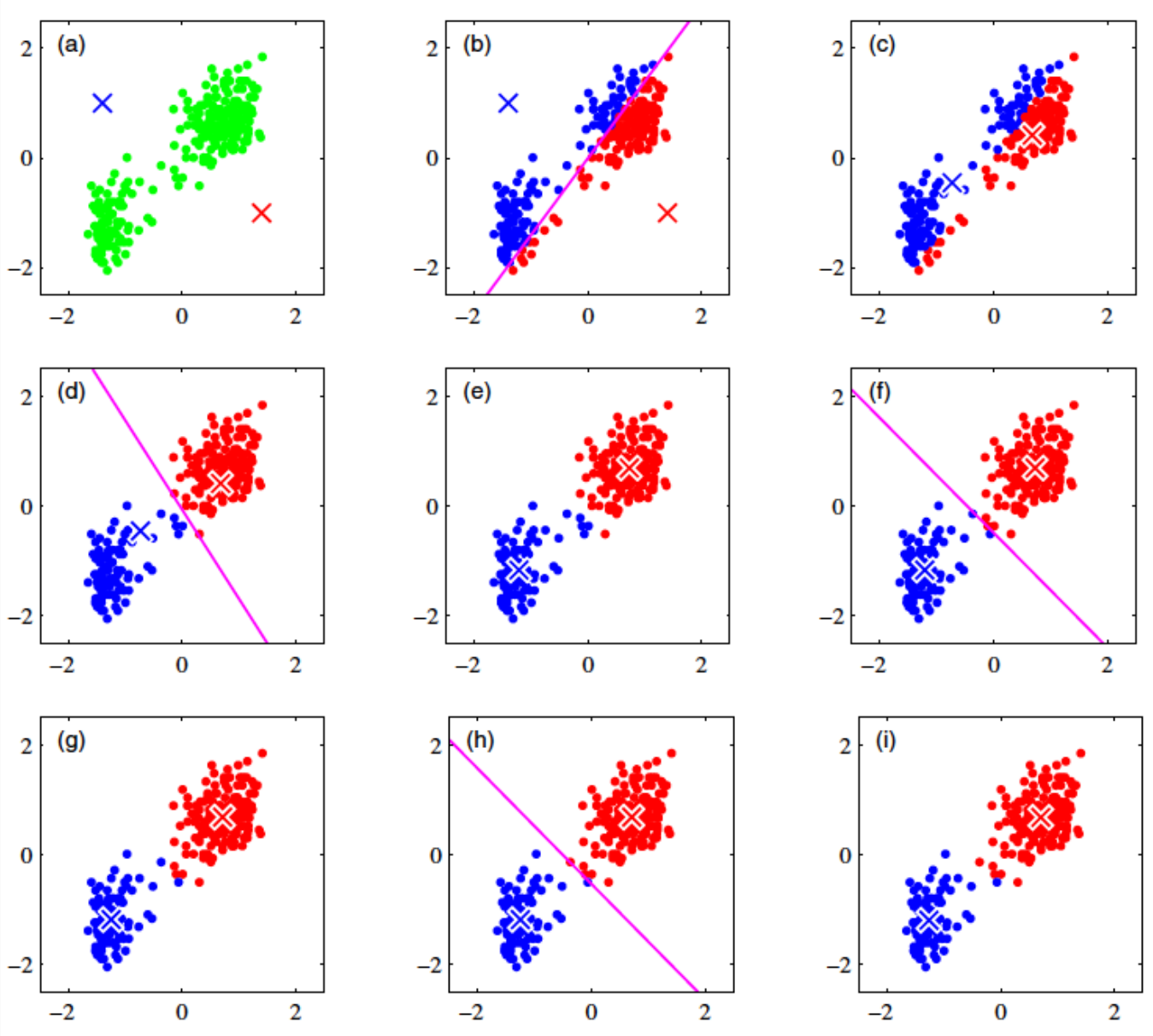
$$\boldsymbol{r}_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

2. Minimize J by selecting the best centroids $\{\boldsymbol{\mu}_k\}$ keeping $\{\boldsymbol{r}_{nk}\}$ fixed. Let the derivative of J with respect to $\boldsymbol{\mu}_k$ be 0,

$$2\sum_{n=1}^{N} \boldsymbol{r}_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k) = 0 \implies \boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} \boldsymbol{r}_{nk}\boldsymbol{x}_n}{\sum_{n=1}^{N} \boldsymbol{r}_{nk}} \tag{3}$$

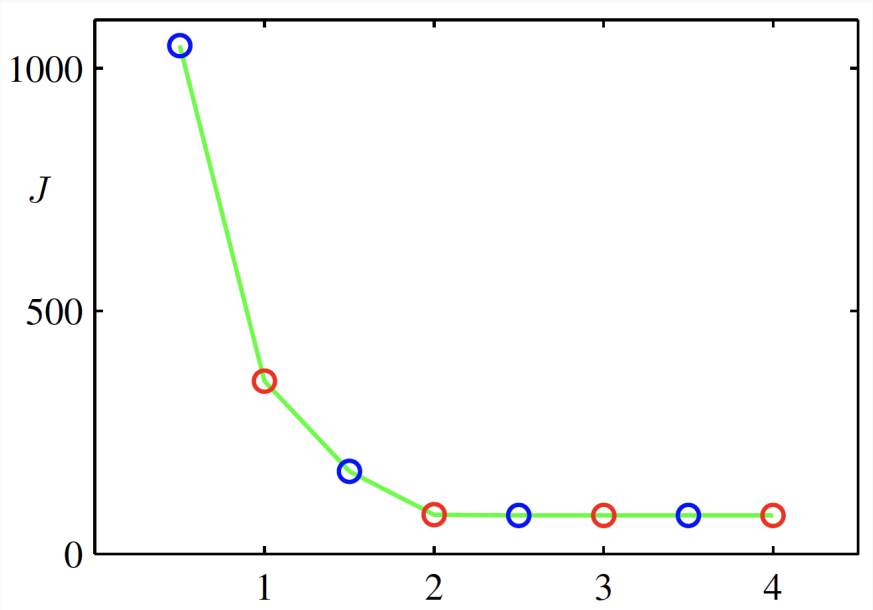
That is, $\boldsymbol{\mu}_k$ is the *mean* of the points assigned to the k^{th} cluster.

K-means example for $K=2$



K-means example for K=2 (II)

The first step is called Expectation (E) and second Maximization (M). This terminology will be justified in the next section. With this algorithm J converges since at each step, the cost is non-decreasing, as can be seen in the example below.



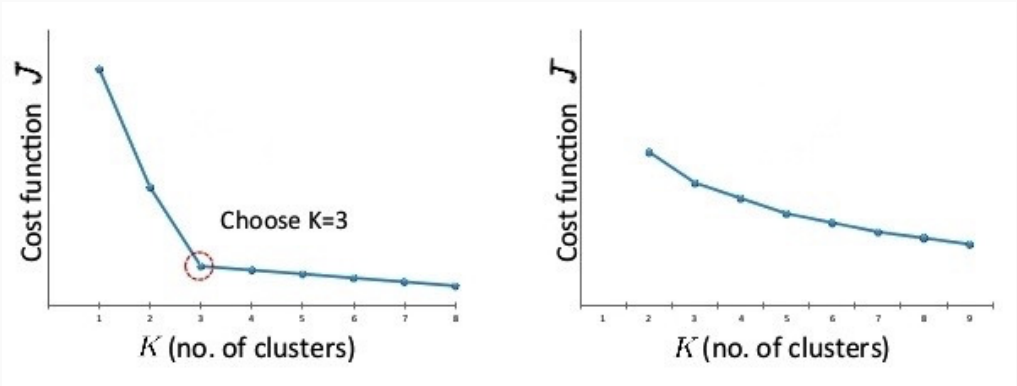
- E steps are marked in blue and M steps in red
- Algorithm converges after the 3^{rd} M step

Implementation notes

- The initial centroids $\{\boldsymbol{\mu}_k\}$ are chosen randomly.
- The algorithm runs until no further changes occur, and no improvement is achieved in J.
- The solution is guaranteed to be a *local* optimum.
- Usually, the algorithm is run several times with different initial centroids, and the best solution selected.
- Since Euclidean distance is used, data standardization (applying a linear transformation to get 0 mean and 1 variance) is necessary to avoid that clustering is focused on some particular directions.

Implementation notes (II)

- Arbitrary distances (e.g., supporting also categorical variables) can be used. In this case the algorithm is called K-methoids. Some alternative distances have been mentioned in Unit 1.
- A frequently used method to determine a suitable value for K is the *elbow criterion*: it selects the value K such that J has marginal improvements (e.g., a slope drop of more than 50%) afterwards (left figure). It can't be applied when the loss improvement is steady (right figure).



Gaussian mixture model

K-means is limited to "spherical shape" clusters. To overcome this restriction, the *Gaussian mixture model (GMM)* allows clusters to be conformed as arbitrary D-dimensional Gaussian[1] variables $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

A GMM is a superposition of K-dimensional Gaussian variables, and has distribution:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{4}$$

where the $\{\pi_k\}$ are non-negative and satisfy $\sum_{k=1}^{K} \pi_k = 1$.

[1] See Multi-variate Gaussian Distribution appendix in AV.

Gaussian mixture model (II)

The GMM model can also be understood by assuming a latent (hidden) variable z, whose value $k \in \{1, \ldots, K\}$ selects the k^{th} Gaussian variable $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ from which the variable $\boldsymbol{x}$ is subsequently drawn.

Given a data set of N points $\{\boldsymbol{x}_n\}$, the goal is computing the parameters $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ which maximizes its likelihood:

$$\sum_{n=1}^{N} ln(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)) \tag{5}$$

A method for finding maximum likelihood solutions for the GMM is called the *expectation-minimization* (EM) algorithm.

EM algorithm

As the K-means, EM assumes a fixed K value, and proceeds sequentially performing the following steps:

1. Set initial parameters $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$.
2. (**Expectation**). With the current parameters, evaluate the *responsibilities* $\{\gamma_{nk}\}$ – the probability that the n^{th} data point has been generated by cluster k:

$$\gamma_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_j, \Sigma_j)} \tag{6}$$

   The name of this step comes from the fact that the responsibilities can be interpreted as the *expectations* of random variables $\{z_{nk}\}$, assigning value 1 to the n^{th} point if it belongs to cluster k, or 0 otherwise.

EM algorithm (II)

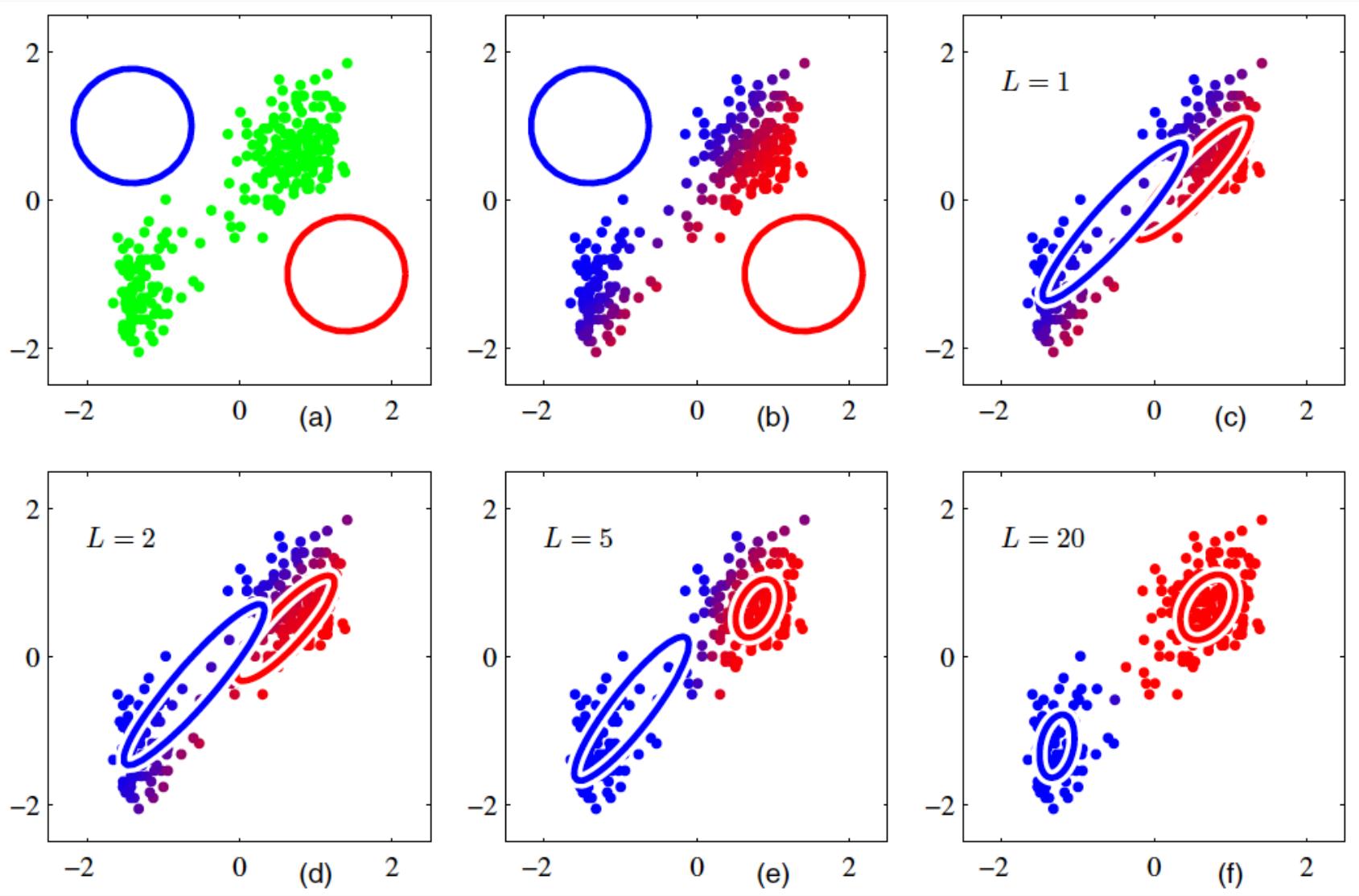
3. (**Maximization**). Set $N_k = \sum_{n=1}^{N} \gamma_{nk}$ and re-estimate the parameters:

$$\begin{aligned} \boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \boldsymbol{x}_n \\ \boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T \\ \pi_k^{\text{new}} &= \frac{N_k}{N} \end{aligned} \tag{7}$$

   This step should maximize the likelihood of the data set, hence its name.

4. Compute likelihood from eq. (5). If it or the parameters have changed, go to step 2.

GMM example for $K=2$



Implementation notes (III)

- Same implementation ideas as in K-means can be used.
- Due to operation similarities, the steps in K-means are also called expectation-maximization in the literature.
- The EM algorithm can be applied to any other model based on latent variables, either discrete or continuous (see Bishop 9.4 and ch. 12), and both for maximizing ML and MAP estimators.
- In contrast with single Gaussian fitting, GMM fitting may encounter singularities of the likelihood function, leads to useless results. Different heuristics are employed to deal with such situations.

Outlier detection

Outlier detection

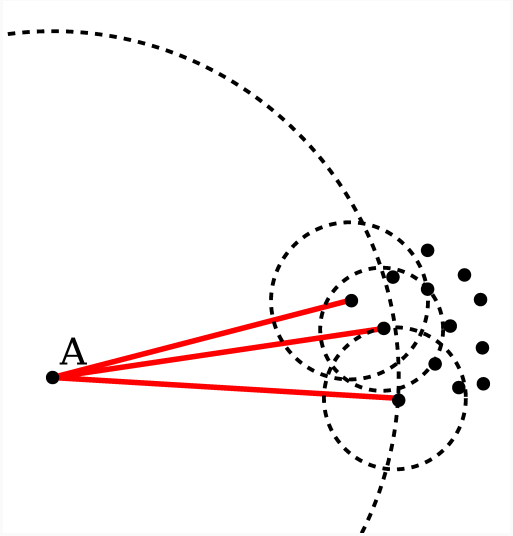
Many times it is possible to collect data coming from natural or artificial sources. For example, sensors in engines, time-series from financial charts, allele configurations in DNA chains, pictures of people or their voice, etc.

In such situations, *Outlier detection* is concerned with determining which data has significant differences with the rest, based on the principle that the amount of regular data outnumbers the number of outliers and a labeled data set is not available.

Outlier detectors rely either on *local* methods (which consider similarity among neighbors) or *global* ones (which aim at describing data with a generative model and then analyze which data fit less into that model). In the next sections, one example of each kind is introduced.

Local Outlier Factor

Local Outlier Factor (LOF) is a *local density* method, performed over the K nearest neighbors, whose distance is used to estimate the point density. When that density is much lower than that of the neighbors, a point is considered an outlier.

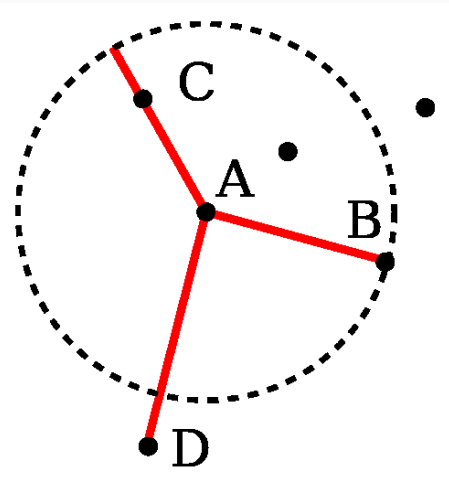


Point A is considered an outlier since it has a much lower density than its neighbors

Local Outlier Factor (II)

Let k-distance($\boldsymbol{x}$) be the distance of a point $\boldsymbol{x}$ to its k^{th} nearest neighbor, and let $N_K(\boldsymbol{x})$ be the set of neighbors[2] at distance less or equal than K-distance($\boldsymbol{x}$):

Let the *reachability* between two points $\boldsymbol{x}, \boldsymbol{x}'$ be $r_K = \max\{K\text{-distance}(\boldsymbol{x}), d(\boldsymbol{x}, \boldsymbol{x}')\}$.
Reachability forces to consider all points of a cluster at the same "distance" as happens with points B and C in the figure.



[2] More than K points may be in this set when several neighbors lie in the boundary defined at distance K-distance($\boldsymbol{x}$).

Local Outlier Factor (III)

Then, the *local reachability density* of $\boldsymbol{x}$ is defined as:

$$\rho_K(\boldsymbol{x}) = \left(\frac{\sum_{\boldsymbol{x}' \in N_K(\boldsymbol{x})} r_K(\boldsymbol{x}, \boldsymbol{x}')}{|N_K(\boldsymbol{x})|} \right)^{-1} \tag{8}$$

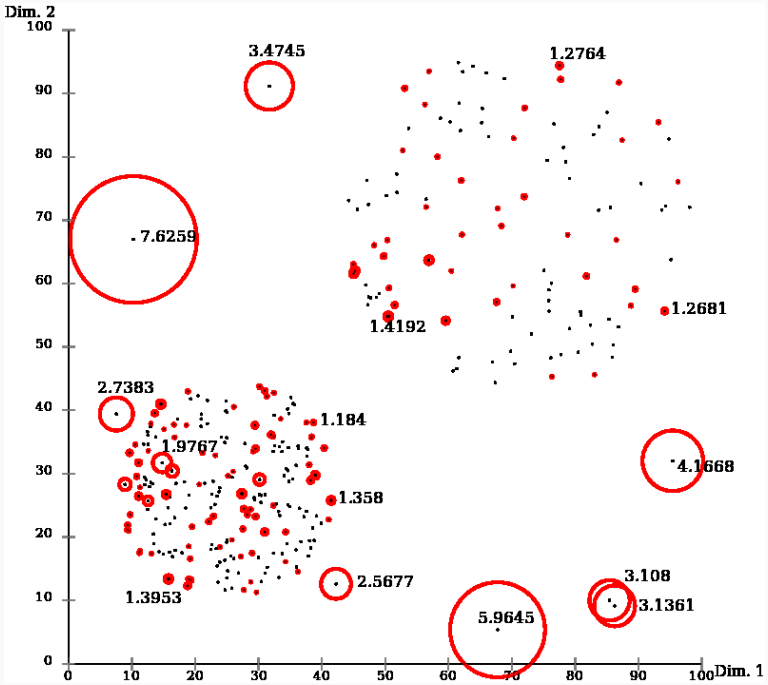
and represents the inverse of the distance at which $\boldsymbol{x}$ can be reached *from* its neighbors.

Finally, it is possible to compare the local density with that of the neighbors by computing:

$$\mathrm{LOF}_K(\boldsymbol{x}) = \frac{\sum_{\boldsymbol{x}' \in N_K(\boldsymbol{x})} \rho_K(\boldsymbol{x}')}{|N_K(\boldsymbol{x})| \rho_K(\boldsymbol{x})} \tag{9}$$

A ratio greater than a selected $\mathrm{LOF}_{\mathrm{critical}}$ identifies an outlier.

Local Outlier Factor (IV)



This example shows points with LOF > 1. As an advantage over global models, LOF can identify points, which, in its vicinity, constitute outliers. The critical ratio can be difficult to determine. In some contexts, it can be close to 1, while in others even higher than 2.

Distance based

The core idea of these methods is determining how much unexpected an observation $\boldsymbol{x}$ is, given a previously observed data set $\{\boldsymbol{x}_n\}$.

A natural approach, is using the data set to fit a Gaussian variable, and then use some mapping of its distribution $p(\boldsymbol{x})=\mathcal{N}(x|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as the novelty measure. The unbiased estimators for a Gaussian fit are:

$$\begin{aligned} \widehat{\boldsymbol{\mu}} &= \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n \\ \widehat{\boldsymbol{\Sigma}} &= \frac{1}{N-1}\sum_{n=1}^{N}(\boldsymbol{x}_n-\widehat{\boldsymbol{\mu}})(\boldsymbol{x}_n-\widehat{\boldsymbol{\mu}})^T \end{aligned} \tag{10}$$

Distance based (II)

Thus, the logarithm of the distribution is proportional to $\ln p(\boldsymbol{x}) \propto -\frac{1}{2} d^2_{\mathcal{M}}$, where:

$$d_{\mathcal{M}} = [(\boldsymbol{x} - \widehat{\boldsymbol{\mu}})^T \widehat{\boldsymbol{\Sigma}}^{-1} (\boldsymbol{x} - \widehat{\boldsymbol{\mu}})]^{1/2} \tag{11}$$

is the *Mahalanobi's distance*, and it's commonly used as a measure of the novelty of a point compared with a clustered data set or distribution. A multi-cluster data set can be fit to a GMM model using the EM algorithm, and then $d_{\mathcal{M}}$ computed to each of the clusters. An outlier will be beyond a critical distance from *all* clusters.

The critical level can be set with an educated guess or from the fact that, for Gaussian clusters, $d^2_{\mathcal{M}}$ is a scaled Beta variable (see paper).

Autoencoders

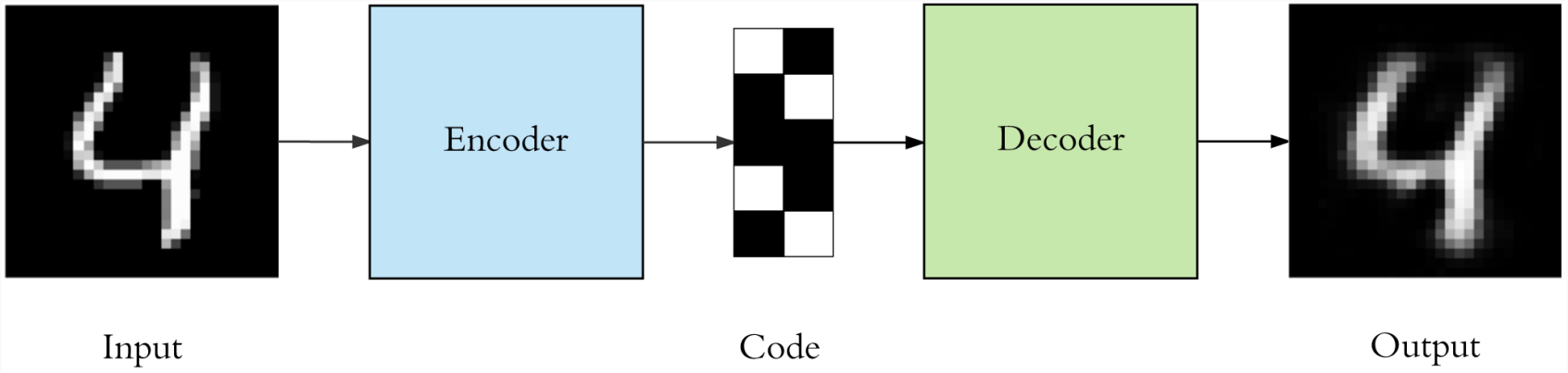
An autoencoder is a parametric structure (most typically an ANN) which transforms the data inputs $\boldsymbol{x}_n$ into a coded form $c(\boldsymbol{x}, \boldsymbol{w_c})$, and then back to the input space $\widetilde{\boldsymbol{x}}_n = d(c(\boldsymbol{x}_n, \boldsymbol{w_c}), \boldsymbol{w_d})$. Since the coding is of (much) lower dimensionality than the input space, some reconstruction noise $\boldsymbol{\epsilon}_n = \boldsymbol{x}_n - \widetilde{\boldsymbol{x}}_n$ appears.

The autoencoder parameters $\boldsymbol{w_c}$, $\boldsymbol{w_d}$ are found by minimizing, e.g., the average modulus data reconstruction noise (J):

$$\min_{\boldsymbol{w_c}, \boldsymbol{w_d}} \frac{1}{2} \sum_{n=1}^{N} \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \tag{12}$$

Autoencoders (II)

Autoencoder structure:



Other uses of autoencoder comprise denoising or generative models (see link, and link).

Autoencoders (III)

Outliers correspond to the data whose reconstruction noise is beyond abound determined statistically to a given confidence level $1 - p$.

For example, assuming a Gaussian reconstruction noise $\mathcal{N}(0, \mathbf{\Sigma})$, then $N\bar{\boldsymbol{\epsilon}}^T \widehat{\mathbf{\Sigma}} \bar{\boldsymbol{\epsilon}} \sim \mathrm{t}^2_{D,N-1}$, being $\bar{\boldsymbol{\epsilon}} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\epsilon}_n$, $\widehat{\mathbf{\Sigma}} = \frac{1}{N-1} \sum_{n=1}^{N} (\boldsymbol{\epsilon} - \bar{\boldsymbol{\epsilon}})(\boldsymbol{\epsilon} - \bar{\boldsymbol{\epsilon}})^T$, and $\mathrm{t}^2_{D,N-1}$ the Hotelling's T-squared distribution. The critical level is the value fulfilling $p(\mathrm{t}^2 > \mathrm{t}_{\mathrm{critical}}) = p$.

Further reading

Further reading

Many other methods to create clusters exists. The figure in slide 6 shows some of the main ones. Density based methods (e.g. DBSCAN) are some of the most used. A reference for these methods can be seen here.

A well-known association rule method is apriori. Link and link provide a gentle introduction to this topic.

Semi-supervised learning uses a small labeled data set to assign labels to points clustered together.