Machine learning - ENS - 2021

Unsupervised learning: K-means and Principal Component analysis

Francis Bach

February 19, 2021

To talk about estimation of "hidden" parameters, French speaking people and English speaking people use different terms which can lead to some confusions. Within a supervised framework, English people would prefer to use the term *classification* whereas the French use the term *discrimination*. Within an unsupervised context, English people would rather use the term *clustering*, whereas French people would use *classification* or *classification non-supervisée*. In the following we will only use the English terms.

1 K-means

K- means clustering is a method of vector quantization. K-means clustering is an algorithm of alternate minimization that aims at partitioning n observations into K clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype to the cluster (see Figure 1).

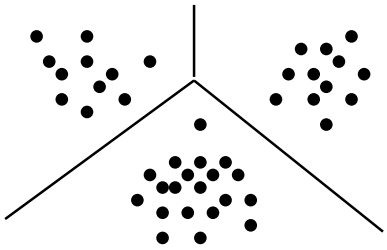

Figure 1: Clustering on a 2D point data set with 3 clusters.

1.1 Notations and notion of Distortion

We will use the following notations:

- $x_i \in \mathbb{R}^p$, $i \in \{1, ..., n\}$ are the observations we want to partition.
- $\mu_k \in \mathbb{R}^p$, $k \in \{1, ..., K\}$ are the means where μ_k is the center of the cluster k. We will denote μ the associated matrix.

- z_i^k are indicator variables associated to x_i such that $z_i^k = 1$ if x_i belongs to the cluster k, $z_i^k = 0$ otherwise. z is the matrix which components are equal to z_i^k.

Finally, we define the *distortion* $J(\mu, z)$ by:

$$J(\mu, z) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k \|x_i - \mu_k\|^2.$$

1.2 Algorithm

The aim of the algorithm is to minimize $J(\mu, z)$. To do so we proceed with an alternating minimization :

- Step 0 : We choose a vector μ
- Step 1 : we minimize J with respect to z : $z_i^k = 1$ if $\| x_i - \mu_k \|^2 = \min_s \| x_i - \mu_s \|^2$, in other words we associate to x_i the nearest center μ_k.
- Step 2 : we minimize J with respect to μ : $\mu_k = \frac{\sum_i z_i^k x_i}{\sum_i z_i^k}$.
- Step 3 : we come back to step 1 until convergence.
- The step of minimization with respect to z is equivalent to allocating the x_i in the Voronoi cells which centers are the μ_k.
- During the step of minimization with respect to μ, μ_k is obtained by setting to zero the k-th coordinate of the gradient of J with respect to μ. Indeed we can easily see that :

$$\nabla_{\mu_k} J = -2 \sum_i z_i^k (x_i - \mu_k)$$

1.3 Convergence and Initialization

We can show that this algorithm converges in a finite number of iterations. Therefore the convergence could be local, thus it introduces the problem of initialization.

A classic method is use of random restarts. It consists in choosing several random vectors μ, computing the algorithm for each case and finally keeping the partition which minimizes the distortion. Thus we hope that at least one of the local minimum is close enough to a global minimum.

One other well known method is the K-means++ algorithm, which aims at correcting a major theoretic shortcomings of the K-means algorithm : the approximation found can be arbitrarily bad with respect to the objective function compared to the optimal clustering.

The K-means++ algorithm addresses this obstacles by specifying a procedure to initialize the cluster centers before proceeding with the standard K-means optimization iterations. With the K-means ++ initialization, the algorithm is guaranteed to find a solution that is $O(\log K)$ competitive to the optimal K-means solution.

The intuition behind this approach is that it is a clever thing to well spread out the K initial cluster centers. At each iteration of the algorithm we will build a new center. We will repeat the algorithm until we have K centers. Here are the steps of the algorithm :

- Step 0 : First initiate the algorithm by choosing the first center uniformly at random among the data points.
- Step 1: For each data point x_i of your data set, compute the distance between x_i and the nearest center that has already been chosen. We denote this distance $D_{\mu_t}(x_i)$ where μ_t is specified to recall that we are minimizing over the current chosen centers.
- Step 2: Choose one new data point at random as a new center, but now using a weighted probability distribution where a point x_i is chosen with probability proportional to $D_{\mu_t}(x_i)^2$.
- Step 3 : Repeat Step 1 and Step 2 until K centers have been chosen.

We see that we have now built K vectors with respect to our first intuition which was to well spread out the centers (because we used a well chosen weighted probability). We can now use those vectors as the initialization of our standard K-means algorithm.

More details can be found on the K-means++ algorithm in [A].

[A] Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms.

1.4 Choice of K

It is important to point out that the choice of K is not universal. Indeed, we see that if we increase K, the distortion J decreases, until it reaches 0 when $K = n$, that is to say when each data point is the center of its own center. To address this issue one solution could be to add to J a penalty over K. Usually it takes the following form :

$$J(\mu, z, K) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k \|x_i - \mu_k\|^2 + \lambda K$$

But again the choice of the penalty is arbitrary.

1.5 Other problems

We can also point out that K-means will work pretty well when the width of the different clusters are similar, for example if we deal with spheres. But clustering by K-means could also be disappointing in some cases such as the example given in Figure 2.

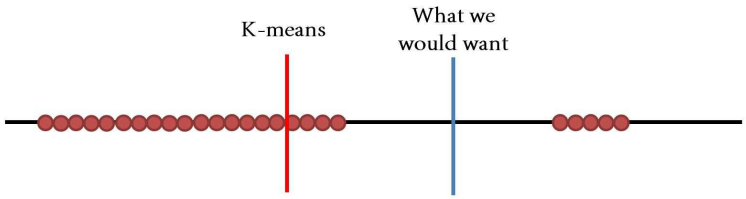

Figure 2: Example where K- means does not provide a satisfactory clustering result

Using Gaussian mixtures provides a way to avoid this problem.

2 Principal component analysis

Framework: $x_1, \dots, x_N \in \mathbb{R}^d$
Goal: put points on a closest affine subspace

2.1 Analysis view

$$\text{Find } w \in \mathbb{R}^d \text{ such that } \text{Var}(x^T w) \text{ is maximal, with } ||w|| = 1$$

With centered data, *i.e.* $\frac{1}{N}\sum_{n=1}^{N} x_n = 0$, the empirical variance is:

$$\widehat{\text{Var}}(x^T w) = \frac{1}{N}\sum_{n=1}^{N}(x_n^T w)^2 = \frac{1}{N} w^T (X^T X) w$$

where $X \in \mathbb{R}^{N \times d}$ is the design matrix. In this case: w is the eigenvector of $X^T X$ with largest eigenvalue. It is not obvious *a priori* that this is the direction we care about.
If more than one direction is required, one can use *deflation*:

1. Find w
2. Project x_n onto the orthogonal of $\text{Vect}(w)$
3. Start again

2.2 Synthesis view

$$\min_{w} \sum_{n=1}^{N} d(x_n, \{\lambda w, \lambda \in \mathbb{R}\})^2 \text{ with } w \in \mathbb{R}^D, \; ||w|| = 1.$$

Advantage: if one wants more than 1 dimension, replace $\{\lambda w, \lambda \in \mathbb{R}\}$ by any subspace.

Read `https://en.wikipedia.org/wiki/Principal_component_analysis` up to (and including) Section 7.