Clustering by k-Means

Thomas Bonald
Telecom ParisTech
thomas.bonald@telecom-paristech.fr

January 2019

In this note, we present the k-means clustering algorithm and some of its variants. We consider n data samples $x_1, \dots, x_n$ of $\mathbb{R}^d$, which we would like to group into k clusters so that samples in the same cluster tend to be close for the Euclidian distance. The parameter k is given (not learned).

1 Cost function

Let $C_1, \dots, C_k$ be a partition of the n samples into k clusters. The quality of the clustering can be assessed through the following cost function:

$$\sum_{j=1}^{k} \sum_{i \in C_j} ||x_i - \mu_j||^2,$$

where μ_j is the center of cluster j:

$$\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} x_i.$$

This cost function is the *squared error* induced by vector quantization whereby the n data samples are replaced by their respective cluster centers, as illustrated by Figure 1.

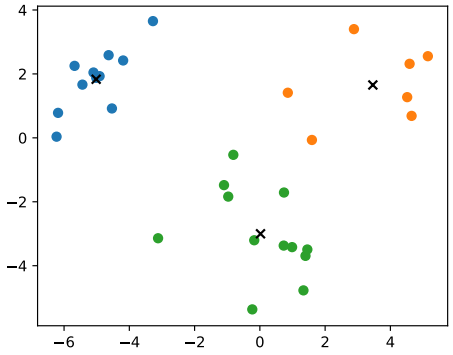

Figure 1: Clustering of $n = 30$ data samples of $\mathbb{R}^2$ into $k = 3$ clusters.

Viewing each data sample as a point particle of unit mass, the cost function may also be interpreted as the total *moment of inertia* of the system, where

$$\sum_{i \in C_j} ||x_i - \mu_j||^2$$

is the moment of inertia of cluster j.

Minimizing the cost function over all partitions $C_1, \ldots, C_k$ is an NP-hard problem. The k-means algorithm provides an approximate solution.

2 The k-means algorithm

The idea of the k-means algorithm is very simple: starting from some arbitrary location of the cluster centers, find the partition induced by the nearest cluster of each data sample, update the cluster centers and iterate. The pseudo-code of the algorithm is given below.

The algorithm stops when the clusters remain unchanged. The cost function decreases at each iteration, unless the clusters remain unchanged. This follows on writing the optimization problem as:

$$\min_{C_1,\ldots,C_k;\mu_1,\ldots,\mu_k} \sum_{j=1}^{k} \sum_{i \in C_j} ||x_i - \mu_j||^2 \tag{1}$$

and observing that the k-means algorithm in fact performs alternating minimization in the clusters $C_1, \ldots, C_k$ and in the cluster centers $\mu_1, \ldots, \mu_k$. This shows that the cost function cannot increase, so the algorithm converges. Note that the convergence point does not minimizes the cost function in general.

Algorithm 1: k-means

Input: Data samples $x_1, \ldots, x_n$; number of clusters k
Output: $C_1, \ldots, C_k$, clustering of samples $1, \ldots, n$ in k clusters

Sample random values $\mu_1, \ldots, \mu_k$ from $x_1, \ldots, x_n$
while no convergence **do**
    `// Update clusters`
    $C_1, \ldots, C_k \leftarrow \emptyset$
    for $i = 1$ **to** n **do**
        $j \leftarrow \arg\min_l ||x_i - \mu_l||$
        $C_j \leftarrow C_j + \{i\}$
    `// Update cluster centers`
    for $j = 1$ **to** k **do**
        $\mu_j \leftarrow 0$
        $n_j \leftarrow 0$
        for i **in** C_j **do**
            $\mu_j \leftarrow \mu_j + x_i$
            $n_j \leftarrow n_j + 1$
        $\mu_j \leftarrow \mu_j / n_j$

The quality of the clustering strongly depend on the initial values of the cluster centers. This is why the algorithm is generally run multiple times for different initial values. The best clustering (i.e., that of minimum cost) is returned.

k-means++ To improve the quality of the clustering, it is interesting in practice to choose the initial cluster centers far from each other. An option is the following:

- select the first cluster center uniformly at random among the n data samples,
- select the following cluster centers at random, with a probability proportional to the square distance to the closest current cluster center.

The corresponding algorithm is known as k-means++; it differs from k-means through the choice of the initial cluster centers only.

Mini-batch k-means. The complexity of k-means is in $O(nk)$ per iteration. This can be reduced to $O(bk)$ per iteration by using batches of size b. Specifically, b samples among n are chosen at random and the cluster centers are updated by smooting average based on these samples.

Algorithm 2: Mini-batch k-means

Input: Data samples $x_1, \dots, x_n$; batch size b; number of clusters k
Output: $C_1, \dots, C_k$, clustering of samples $1, \dots, n$ in k clusters

Sample random values $\mu_1, \dots, \mu_k$ from $x_1, \dots, x_n$
$n_1, \dots, n_k \leftarrow 0$
while no convergence **do**
    Sample a mini-batch $y_1, \dots, y_b$ from $x_1, \dots, x_n$
    `// Find clusters for the batch`
    $B_1, \dots, B_k \leftarrow \emptyset$
    for $i = 1$ **to** b **do**
        $j \leftarrow \arg\min_l ||y_i - \mu_l||$
        $B_j \leftarrow B_j + \{i\}$
    `// Update cluster centers`
    for $j = 1$ **to** k **do**
        $\mu_j \leftarrow n_j \mu_j$
        for i **in** B_j **do**
            $\mu_j \leftarrow \mu_j + y_i$
            $n_j \leftarrow n_j + 1$
        $\mu_j \leftarrow \mu_j / n_j$
Find the partition $C_1, \dots, C_k$ induced by the cluster centers

3 Soft k-means

While the usual k-means algorithm returns a partition of the data samples (hard clustering), it can easily be adapted to get a distribution of each sample over the clusters (soft clustering). This may be viewed as a relaxation of the problem (1) into:

$$\min_{p_1,\dots,p_n;\mu_1,\dots,\mu_k} \sum_{j=1}^{k} \sum_{i=1}^{n} p_{ij} ||x_i - \mu_j||^2$$

where p_{ij} is the probability that sample i belongs to cluster j, with

$$\forall i = 1, \dots, n, \quad \sum_{j=1}^{k} p_{ij} = 1.$$

Observe that the cluster centers are then given by:

$$\forall j = 1, \dots, k, \quad \mu_j = \frac{\sum_{i=1}^{n} p_{ij} x_i}{\sum_{i=1}^{n} p_{ij}}.$$

The alternating minimization in cluster distributions $p_1, \ldots, p_n$ and cluster centers $\mu_1, \ldots, \mu_k$ leads to the usual k-means algorithm. To enforce soft clustering, we can interpret data samples as noisy versions of their corresponding cluster centers.

Specifically, assume that the distribution of samples around each cluster center has a Gaussian distribution with variance σ^2, independently in each dimension. Then the probability that sample i has been generated by cluster j is given by the *softmax* function:

$$p_{ij} \propto e^{-\frac{||x_i - \mu_j||^2}{2\sigma^2}}.$$

The parameter σ controls the spread of the clusters, in terms of typical distance to the cluster center. The pseudo-code of the algorithm is shown below.

Algorithm 3: Soft k-means

Input: Data samples $x_1, \ldots, x_n$; distance σ; number of clusters k
Output: $p_1, \ldots, p_n$, distributions of the samples over the k clusters

Sample random values $\mu_1, \ldots, \mu_k$ from $x_1, \ldots, x_n$
while no convergence **do**
    `// Update cluster distributions`
    for $i = 1$ **to** n **do**
        $s \leftarrow 0$
        for $j = 1$ **to** k **do**
            $p_{ij} \leftarrow e^{-\frac{||x_i - \mu_j||^2}{2\sigma^2}}$
            $s \leftarrow s + p_{ij}$
        for $j = 1$ **to** k **do**
            $p_{ij} \leftarrow p_{ij}/s$
    `// Update cluster centers`
    for $j = 1$ **to** k **do**
        $\mu_j \leftarrow 0$
        $n_j \leftarrow 0$
        for i **in** C_j **do**
            $\mu_j \leftarrow \mu_j + p_{ij} x_i$
            $n_j \leftarrow n_j + p_{ij}$
        $\mu_j \leftarrow \mu_j / n_j$

When $\sigma \to 0$ (small distance to the cluster center), this becomes:

$$p_{ij} = \begin{cases} 1 & \text{if } j = l, \\ 0 & \text{otherwise,} \end{cases}$$

where l is the index for which the distance $||x_i - \mu_l||$ is minimum (assuming this index is unique). This is the usual k-means algorithm (hard clustering). When $\sigma \to +\infty$ (large distance to the cluster center), the distributions $p_1, \ldots, p_n$ become uniform over the k clusters.

Another popular approach, known as fuzzy c-means, is based on the Pareto distribution:

$$p_{ij} \propto \frac{1}{||x_i - \mu_j||^\alpha},$$

for some parameter $\alpha \geq 0$. When $\alpha \to +\infty$, this boils down to the usual k-means algorithm, while the limiting case $\alpha \to 0$ corresponds to uniform distributions.

4 Weighted k-means

Now assume some positive weights $w_1, \ldots, w_n$ are assigned to the n data samples, capturing their relative importance. Namely, the weight of each data sample in the final clustering must be proportional to its weight. If the weights are integers, these can be interpreted as the multiplicity of each data sample (e.g., $w_1 = 2$ is equivalent to have 2 copies of the data sample x_1). These can be useful in practice when the samples represent populations of users or items.

The cost function becomes:

$$\sum_{j=1}^{k} \sum_{i \in C_j} w_i ||x_i - \mu_j||^2,$$

where μ_j is the center of mass of cluster j:

$$\mu_j = \frac{\sum_{i \in C_j} w_i x_i}{\sum_{i \in C_j} w_i}.$$

This can be interpreted as the total *moment of inertia* of the system, where each data sample is viewed as a point particle of mass equal to its weight. The corresponding algorithm is very similar, the variable n_j representing the *mass* of cluster j instead of its size in number of samples (see the pseudo-code below). Observe that the initial cluster centers are selected at random among the data samples in proportion to their weights. For integer weights, this guarantees that the algorithm behaves like the standard k-means algorithm where each sample x_i is replaced by w_i identical copies of it. For weighted k-means++, the successive initial cluster centers are selected in proportion to the product of the weight of each data sample and its square distance to the closest existing cluster center.

Algorithm 4: Weighted k-means

Input: Data samples $x_1, \ldots, x_n$; sample weights $w_1, \ldots, w_n$; number of clusters k
Output: $C_1, \ldots, C_k$, clustering of samples $1, \ldots, n$ in k clusters

Sample random values $\mu_1, \ldots, \mu_k$ from $x_1, \ldots, x_n$ in proportion to the weights $w_1, \ldots, w_n$
while no convergence **do**
    `// Update clusters`
    ...
    `// Update cluster centers`
    for $j = 1$ **to** k **do**
        $\mu_j \leftarrow 0$
        $n_j \leftarrow 0$
        for i **in** C_j **do**
            $\mu_j \leftarrow \mu_j + w_i x_i$
            $n_j \leftarrow n_j + w_i$
        $\mu_j \leftarrow \mu_j / n_j$

Further reading

- The first version of the k-means algorithm, proposed by Lloyd in 1957 and published in 1982 [Lloyd, 1982].
- A survey on k-means and other clustering algorithms [Jain, 2010].
- Some convergence properties of the k-means algorithm [Bottou and Bengio, 1995].

References

[Bottou and Bengio, 1995] Bottou, L. and Bengio, Y. (1995). Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems*.

[Jain, 2010] Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*.

[Lloyd, 1982] Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*.