1 k-means clustering

From Wikipedia, the free encyclopedia

In statistics and machine learning, ***k*-means clustering** is a method of cluster analysis which aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean.

The *k*-means clustering algorithm is commonly used in computer vision as a form of **image segmentation**. The results of the segmentation are used to aid border detection and object recognition. In this context, the standard Euclidean distance is usually insufficient in forming the clusters. Instead, a weighted distance measure utilizing pixel coordinates, RGB pixel color and/or intensity, and image texture is commonly used.

The term "*k*-means" was first used by James MacQueen in 1967, though the idea goes back to Hugo Steinhaus in 1956. The standard algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse-code modulation, though it wasn't published until 1982.

2 Description

Given a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$), where each observation is a *d*-dimensional real vector, then *k*-means clustering aims to partition the *n* observations into *k* sets ($k < n$) $\mathbf{S}=\{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS):

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of S_i.

3 Algorithms

Standard algorithm

The most common algorithm uses an iterative refinement technique. Due to its ubiquity it is often called the ***k*-means algorithm**; it is also referred to as **Lloyd's algorithm**, particularly in the computer science community.

Given an initial set of *k* means $\mathbf{m}_1^{(1)},\ldots,\mathbf{m}_k^{(1)}$, which may be specified randomly or by some heuristic, the algorithm proceeds by alternating between two steps:

Assignment step: Assign each observation to the cluster with the closest mean (i.e. partition the observations according to the Voronoi diagram generated by the means).

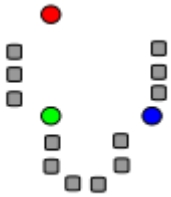
$$S_i^{(t)} = \left\{ \mathbf{x}_j : \left\| \mathbf{x}_j - \mathbf{m}_i^{(t)} \right\| \leq \left\| \mathbf{x}_j - \mathbf{m}_{i^*}^{(t)} \right\| \text{ for all } i^* = 1, \ldots, k \right\}$$

Update step: Calculate the new means to be the centroid of the observations in the cluster.

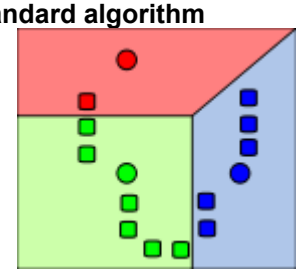
$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

The algorithm is deemed to have converged when the assignments no longer change.

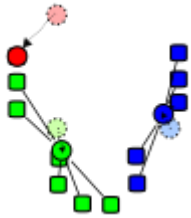
Demonstration of the standard algorithm



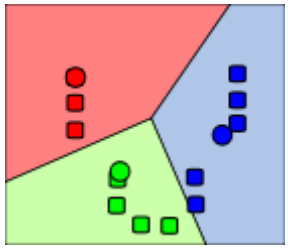
1) *k* initial "means" (in this case *k*=3) are randomly selected from the data set (shown in color).



2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.



3) The centroid of each of the *k* clusters becomes the new means.



4) Steps 2 and 3 are repeated until convergence has been reached.

As it is a heuristic algorithm, there is no guarantee that it will converge to the global optimum, and the result may depend on the initial clusters. As the algorithm is usually very fast, it is common to run it multiple times with different starting conditions.

The two key features of *k*-means which make it efficient are often regarded as its biggest drawbacks:

- The number of clusters *k* is an input parameter: an inappropriate choice of *k* may yield poor results.
- Euclidean distance is used as a metric and variance is used as a measure of cluster scatter.

4 Optimization: K-means++

k-means++ is an algorithm for choosing the initial values for k-means clustering in statistics and machine learning. It was proposed in 2007 by David Arthur and Sergei Vassilvitskii - a way of avoiding the sometimes poor clusterings found by the standard k-means algorithm.

Background

The k-means problem is to find cluster centers that minimize the sum of squared distances from each data point being clustered to its cluster center (the center that is closest to it). Although finding an exact solution to the k-means problem for arbitrary input is NP-hard, the standard approach to finding an approximate solution (often called Lloyd's algorithm or the k-means algorithm) is used widely and frequently finds reasonable solutions quickly.

However, the k-means algorithm has at least two major theoretic shortcomings:

- First, it has been shown that the worst case running time of the algorithm is super-polynomial in the input size.
- Second, the approximation found can be arbitrarily bad with respect to the objective function compared to the optimal clustering.

In a nutshell, k-means++ addresses the second of these 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.

Example Bad Case

To illustrate the potential of the k-means algorithm to perform arbitrarily poorly with respect to the objective function of minimizing the sum squared distance of points to assigned clusters, consider the example of four points in $\mathbb{R}^2$ that form an axis aligned rectangle with the width of the rectangle they form being somewhat larger than its height.

If $k = 2$ and the two initial cluster centers lie at the mid-points of the top and bottom line segments of the rectangle formed by the four data points, the k-means algorithm will converge without moving these cluster centers. Consequently, the two bottom data points are clustered together and the two data points forming the top of the rectangle are clustered together - a suboptimal clustering because the width of the rectangle is greater than the height of the rectangle.

Now, consider stretching the rectangle horizontally to an arbitrary width. The standard k-means algorithm will still cluster the points suboptimally, and by increasing the horizontal distance between the two data points in each cluster, we can make the algorithm do arbitrarily poorly with respect to the k-means objective function.

Initialization Algorithm

With the intuition of spreading the k initial cluster centers away from each other, the first cluster center is chosen uniformly at random from the data points that are being clustered, after which each subsequent cluster center is chosen from the remaining data points with probability proportional to its distance squared to the point's closest cluster center.

The exact algorithm is as follows:

1. Choose one center uniformly at random from among the data points.
2. For each data point x, compute $D(x)$, the distance between x and the nearest center that has already been chosen.
3. Add one new data point as a center. Each point x is chosen with probability proportional to $D(x)^2$.
4. Repeat Steps 2 and 3 until k centers have been chosen.
5. Now that the initial centers have been chosen, proceed using standard k-means clustering.

This seeding method gives out considerable improvements in the final error of *k*-means. Although the initial selection in the algorithm takes extra time, the *k*-means part itself converges very fast after this seeding and thus the algorithm actually lowers the computation time too. The authors tested their method with real and synthetic datasets and obtained typically 2-fold improvements in speed, and for certain datasets close to 1000-fold improvements in error. Additionally, the authors calculate an approximation ratio for their algorithm. The *k*-means++ algorithm guarantees an approximation ratio $O(\log(k))$ where k is the number of clusters used.