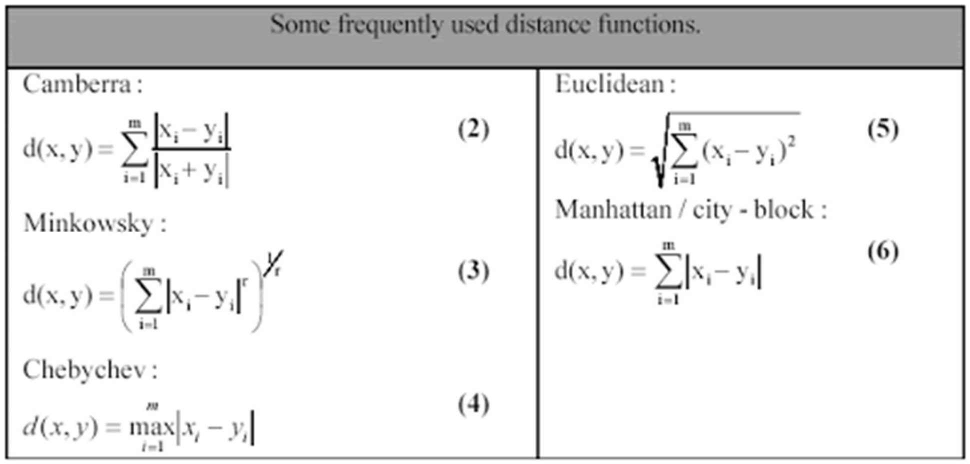
Standard k-means clustering algorithm

The k-means clustering problem is to partition a given set of n observations (which are m-dimensional vectors) into k (k<=n) sets (called clusters) such that the sum of squares of within-cluster ‘distances’ (i.e. distance from mean (also called centroid) of the cluster which the observation belongs to) of all observations is minimized.

The distance function may vary, but the most commonly used is the Euclidean distance. Some other distance functions are Chebyshev distance, Manhattan distance, Minkowski distance and Camberra Distance.



Some frequently used distance functions.	
Camberra: $d(x,y)=\sum_{i=1}^{m}\frac{\lvert x_i-y_i\rvert}{\lvert x_i+y_i\rvert}$ (2)	Euclidean: $d(x,y)=\sqrt{\sum_{i=1}^{m}(x_i-y_i)^2}$ (5)
Minkowsky: $d(x,y)=\left(\sum_{i=1}^{m}\lvert x_i-y_i\rvert^r\right)^{1/r}$ (3)	Manhattan / city - block: $d(x,y)=\sum_{i=1}^{m}\lvert x_i-y_i\rvert$ (6)
Chebychev: $d(x,y)=\max_{i=1}^{m}\lvert x_i-y_i\rvert$ (4)	

The popular algorithm for this problem is the standard k-means clustering algorithm which uses iterative refinement. It is also called ‘naïve k-means’, because there exist much faster alternatives.

The algorithm has 3 steps:

1. *Initialization:* An initial set of k means is chosen. A variety of initialization methods are used. The Forgy method (randomly choose k-observations as means) and Random Partition method (randomly assign a cluster to each observation) are the most common, but give poor performance. The k-means++ method gives far better performance.
2. *Assignment:* We assign each observation to the cluster whose mean has the minimum distance from the observation.
3. *Update:* Recalculate the means of all clusters.

Steps 2 and 3 are repeated till no observation changes clusters or till some maximum number of iterations have occurred.

Note that the algorithm does not guarantee to find the global optimum. The result depends on initialization. Also, the clustering may vary with different initializations. Hence, a careful initialization is crucial for the success of this algorithm. One such good initialization technique is k-means++, which is discussed below.

k-means++ algorithm

The standard k-means algorithm (with Forgy or Random Partition initializations) has at least two major theoretic shortcomings:

1. The worst-case running time of the algorithm is super-polynomial in the input size.
2. The approximation found can be arbitrarily bad with respect to the optimal clustering.

The k-means++ algorithm addresses the second obstacle by initializing the k means in such a way that the solution found is guaranteed to be O(log k) competitive to the optimal k-means solution.

The intuition behind this approach is that spreading out the k initial cluster centers is a good thing because points in different clusters are much more separated than points in same cluster.

The algorithm has 3 steps:

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. Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$.

Steps 2 and 3 are repeated till k-centers have been chosen. After that, the standard k-means clustering algorithm is executed with these k means.

Although the initial selection in the algorithm takes extra time, the standard k-means clustering algorithm itself converges very quickly after this seeding and thus the algorithm actually lowers the computation time.