University of Southern Denmark
IMADA
Ricardo Campello

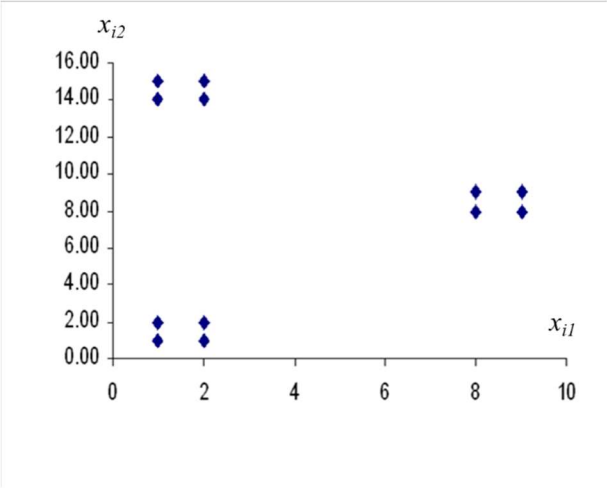
DM583: Data Mining

Exercise 3: Partitioning Clustering and Evaluation (k-Means, SSE and Silhouette)

Exercise 3-1 Standard k-Means

a) Perform k-means in the above dataset starting from prototypes [6 6], [4 6] and [5 10]. You can use R as a calculator to assist e.g. during distance and centroid computations, but otherwise you are asked to do it "manually" in a detailed and step-by-step fashion. Use standard *Euclidean* distance (not squared).

Observation $\mathbf{x}_i$	x_{i1}	x_{i2}
1	1	2
2	2	1
3	1	1
4	2	2
5	8	9
6	9	8
7	9	9
8	8	8
9	1	15
10	2	15
11	1	14
12	2	14



b) Would you expect the clustering result (partition) to change if you were to use *squared Euclidean* distance instead? Discuss it conceptually, justifying your answer on a theoretical basis.

c) Repeat item (a), now using function `kmeans()` from package `stats` in R with argument `centers` to explicitly provide the specified initial prototypes as input.

Exercise 3-2 k-Means with Recursive Centroid Updates

Suppose that at a certain iteration during execution of k-Means, the cluster prototypes of all k clusters have just been updated. Specifically for the ith cluster (C_i), its prototype has just been updated as the centroid ($\mathbf{v}_i$) of the $|C_i|$ observations currently belonging to that cluster ($|C_i|$ denotes the ith cluster's size or cardinality). Now, in the next step of the algorithm, the distances between every observation in the dataset and such updated centroids will be recomputed. Let's suppose that observations $\mathbf{x}_j$, $\mathbf{x}_l$, and $\mathbf{x}_m$ will no longer be closer to $\mathbf{v}_i$, they will become closer to the updated centroid of another cluster, whereas observations $\mathbf{x}_p$ and $\mathbf{x}_q$, which are currently part of other clusters, will have $\mathbf{v}_i$ as their closest centroid. Assuming that these are the only 5 observations to be moved from/to the ith cluster in the next step, derive an equation to update its centroid $\mathbf{v}_i$ from its current value, using only these 5 observations as well as the cluster size $|C_i|$ prior to such an update.

Exercise 3-3 Parallel / Distributed k-Means

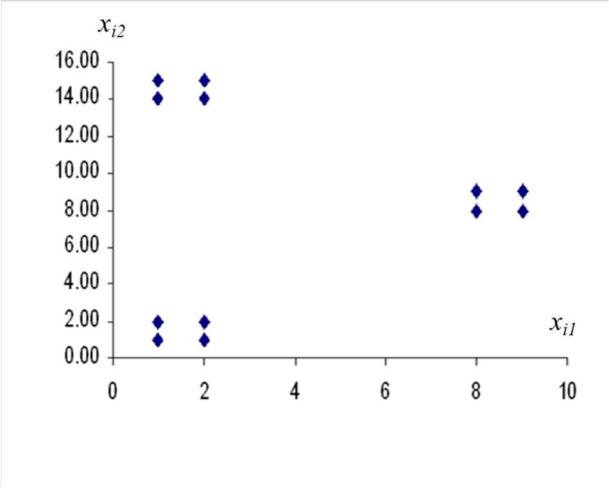
a) Suppose that we have a data collection $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ with N observations distributed across s local processing/storage sites with independent (separated, non-shared) memory, each of which holding exclusive access to (and only to) a subset of the data. For instance, assuming $s = 3$ sites and a dataset with $N = 9$ observations, Site 1 could have access only to observations $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$, Site 2 could have access only to observations $\{\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7\}$, and Site 3 could have access only to observations $\{\mathbf{x}_8, \mathbf{x}_9\}$. We want to perform exact k-Means (for a given choice of k) on the whole dataset $\mathbf{X}$ and obtain the same result (no approximations) as if the algorithm had simultaneous and unrestricted access to any data observation, which is not the case. For simplicity, we assume that we have a central processing node that can communicate *summary cluster statistics* to and from the local processing/storage sites. However, for privacy and/or data transmission bandwidth constraints, data observations are NOT allowed to be transmitted across the network (neither between two local sites nor between a local site and the central processing node). Describe an algorithm that solves the problem.

b) Run one complete iteration of the algorithm described in the previous item (with standard Euclidean distance, $k = 3$, and initial prototypes given by $\mathbf{v}_1 = [6\ \ 6]$, $\mathbf{v}_2 = [4\ \ 6]$ and $\mathbf{v}_3 = [5\ \ 10]$) on the following dataset, where it is assumed that the observations are distributed across 2 separate processing/storage sites (A and B):

Object $\mathbf{x}_i$	x_{i1}	x_{i2}	Processor/Site
1	1	2	A
2	2	1	B
3	1	1	A
4	2	2	B
5	8	9	A
6	9	8	B
7	9	9	B
8	8	8	A
9	1	15	B
10	2	15	A
11	1	14	B
12	2	14	A

Exercise 3-4 Choice of *k* and Initial Prototypes in k-Means with SSE Evaluation

Consider the same toy dataset used in other exercises:



Use the built-in function `kmeans()` from package `stats` in R with argument `centers` set as an integer k to provide the specified number of clusters as input. In this setting, by default, the k initial cluster prototypes will be selected randomly by the algorithm. In order to minimize the chances that the algorithm gets stuck at local

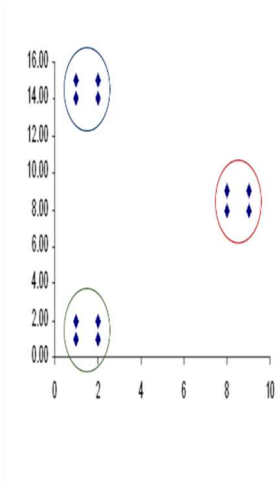
minima due to bad initial prototypes (unlikely in this simple toy example, but very likely in real datasets), you can use argument `nstart` to tell function `kmeans()` to run the algorithm multiple (`nstart`) times, each time from a different random initialization of prototypes, and choose the best clustering result according to the Sum of Squared Errors (SSE) criterion, which is minimized by k-Means for a fixed value of *k*. You can then run the algorithm for k varying within a given range (say, $k = 1, 2, 3, \ldots, 10$) and plot the resulting SSE values as a function of *k*. Does the plot suggest the natural number of clusters in the dataset? Discuss.

Note: You can repeat this exercise for any real dataset of your choice (e.g., `Iris`, etc.) **[OPTIONAL].**

Exercise 3-5 Silhouette Width Criterion (SWC)

a) Consider again the same toy dataset used in other exercises, and the natural clustering solution as indicated by circles in the figure below. Compute the *individual contribution* of the bottom right observation in the blue (top left) cluster – i.e., the 12th observation ([2 14]) – to both the original Silhouette Width Criterion (SWC) as well as to its simplified version, the Simplified SWC (SSWC), using Euclidean distance. You can use R as a calculator to assist e.g. during distance and centroid computations, but otherwise you are asked to do it "manually" in a detailed and step-by-step fashion.

Observation $\mathbf{x}_i$	x_{i1}	x_{i2}
1	1	2
2	2	1
3	1	1
4	2	2
5	8	9
6	9	8
7	9	9
8	8	8
9	1	15
10	2	15
11	1	14
12	2	14



b) If you would write a code in R that repeats the computations in item (a) for all observations and averages the results, you would be able to compute the SWC and SSWC values for the candidate clustering solution in the figure. While writing your own code from scratch is left as optional, in this item you are asked to freely use R (packages, functions, programming) to compute (at least) the original SWC criterion for this solution.

c) Now, merge the two clusters on the left (blue and green circles) into a single cluster and repeat item (b). Compare with the results in item (b). What conclusions can you draw about the quality of the two candidate clustering solutions assessed according to SWC?

d) Repeat Exercise 3.4, now replacing SSE with SWC when choosing the number of clusters, noticing that: (i) you will no longer look for a knee/elbow but rather for a maximum peak at the plot of the SWC as a function of *k*; (ii) SWC is not defined for $k = 1$, so you need to start from $k = 2$ instead.

Note: You can repeat this exercise for any real dataset of your choice (e.g., `Iris`, etc.) **[OPTIONAL].**

Hint: You can use the following given function to compute SWC for a given dataset `dataPoints` and its candidate clustering solution `clusterLabels`, which is based on the built-in `silhouette()` function from package `cluster` (it requires this R package to be installed):

```
SWC <- function(clusterLabels, dataPoints){
   require(cluster)
```

```
   sil <- silhouette(x = clusterLabels, dist = dist(dataPoints))
return(mean(sil[,3]))
}
```