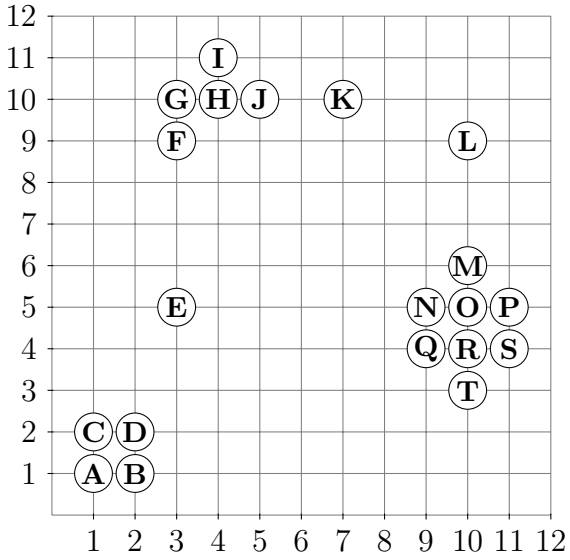
Exercise 8 : Clustering Algorithms, Density Estimation

Exercise 8-1 : Density Estimation

Given the following data set :



Estimate the density around each point in the dataset, using the discrete Kernel

$$\hat{f}(x) = \frac{k}{nV_k(x)}$$

based on Manhattan distance (L_1)

(a) with a fixed $k = 2$,

(b) with a fixed $k = 4$,

(c) with a fixed volume based on radius $\varepsilon = 1$,

(d) with a fixed volume based on radius $\varepsilon = 2$.

Explain what your choices are in computing the density estimate regarding

(a) including or excluding the point itself,

(b) ties in the neighborhood.

Note that using the Manhattan distance results in estimators that slightly differ from those discussed in the lecture.

What do you observe ?

Exercise 8-2 : Properties of DBSCAN

Discuss the following questions or statements on DBSCAN :

— For $minPts = 2$, what about border points ?

— The result of DBSCAN is deterministic for core and noise points, but not for border points.

— A cluster in DBSCAN can contain less than $minPts$ objects.

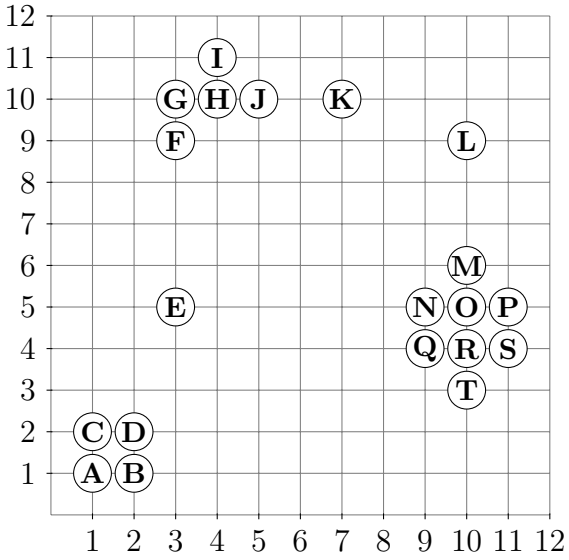
— If the dataset has n objects, DBSCAN computes always exactly n neighborhood range queries.

— On uniformly distributed data, DBSCAN will typically put everything in one cluster or everything in noise. k-means will typically partition the uniformly distributed data in k approximately equal-size partitions.

— What is the relationship of DBSCAN with $minPts = 2$ to single-linkage clustering ?

Exercise 8-3 : Shared Nearest Neighbors

Given the following data set :



(a) Compute the pairwise shared-nearest-neighbor-similarities SNN_5 of the objects M, N, O, P, Q, R, S, and T.
Use Manhattan-distance L_1 to obtain the neighbors and neighborhoodsize 5.
The query point is a member of its neighborhood.

$$L_1(x, y) = |x_1 - y_1| + |x_2 - y_2|$$

(b) Give parameters ε and minpts such that the SNN variant of DBSCAN (Ertöz et al., 2003) identifies the 8 points as "dense" and connects them into a single cluster.

Exercise 8-4 : Tools : Precision, Recall

(a) Install `imblearn`, and load python packages : `datasets, metrics` from `sklearn` and `train-test-split` from `sklearn.model-selection`, `KNeighborsClassifier` from `sklearn.neighbors`, also `precision-recall-fscore-support` from `sklearn.metrics`, `matplotlib.pyplot` and `make-imbalance` from `imblearn.datasets`.

(b) Load `make-moons` dataset with 200 samples.

(c) Check how this dataset is balance, then make that imbalance by putting 90 percent of data in observation class and 10 percent in target class.

(d) split dataset to train and test part with 20 percent for test data.

(e) Train the K-Neighbors Classifier model with different number of neighbours from 1 to 5, and calculate `train accuracy, test accuracy` and `precision, recall, fscore` of model by using `precision-recall-fscore-support` function with "macro" averaging.

(f) Plot the results.