TALLINNA TEHNIKAÜLIKOOL
INFOTEHNOLOOGIA TEADUSKOND

TALLINNA TEHNIKAÜLIKOOL

Home Assignment 1

Student: Mihhail Daniljuk

Student code: 211579IAPM

Tallinn 2022

Exercise 1. Representative based clustering

Comparing my K-medoids clustering algorithm using different distance functions I would say that overall clulstering on my particular dataset had given good results. For comparing clustering results I used only visual representation and almost every distance function contributed to similar results, as shown on Fig.1. Canberra Distance function was the only one that provided strange clustering results (Fig 2.), I suppose that, this is because of taking modulus of different features separately and then computing distance with them.

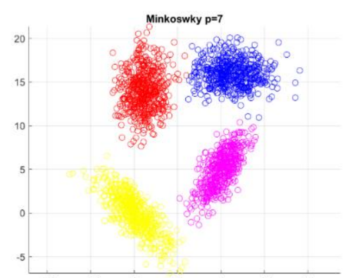

Figure 1.K-medoids Minkowsky p=7

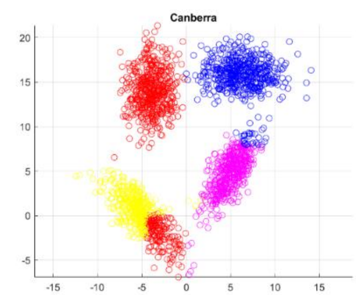

Figure 2.K-Medoids Canberra

Exercise 2. Density based clustering.

I decided to implement DBSCAN as it seemed to be easier. As my test dataset contains 2 close clusters I had to tune my *epsilon* and *theta (minimum number of neighbors)* parameters. Clustering results with diferent *epsilon* values can be seen on Fig. 3 and 4.

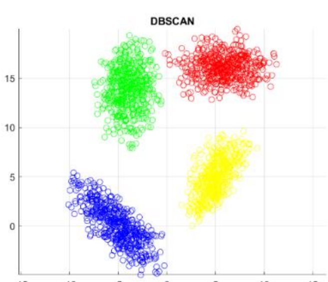

Figure 3.DBSCAN clustering, eps = 0.85

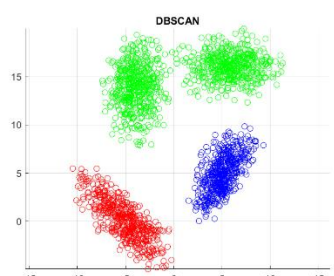

Figure 4. DBSCAN clustering, eps = 1

Exercise 3. Dataset generation

As I've had to tune my DBSCAN parameters I discovered its behaviour and with the theory provided about this clustering method I understood that DBSCAN will unite 2 different clusters if they will pass the threshold level. So I created dataset that contains

3 clusters near each other. DBSCAN fails and K-medoids is doing good. Figures can be seen in matlab, but shortly, result is near to Fig. 4 and Fig. 1.

Exercise 4. Feature selection

For this excercise I created dataset with easy distinguishable clusters in one dimension and much harder in second dimension, computing Fisher's score for both features should give much more greater score for first feature than to the second. Graphs provided in matlab

Exercise 5. Decision tree

My implementation of decision tree classifies rightly about 97% of my dataset, which is easy to classify overall. I tried it only with 2-feature dataset with binary labels, so classifying lines are parallel to the axis and it affects model if the dataset has some spiral or ohter complex form datasets. Implementing crossvalidation helped to really understand if my model works correctly, because firstly my code had bugs inside and I got successful results occasionally. Crossvalidation gave insight how model works on some „random“ samples from initial dataset and helped to find the concrete bug.

I think, that in some places I „hardcoded“ decision tree to work on TWO-dimensional dataset. Figures provided in matlab.

Exercise 6. Research.

It looks like increasing the number of dimensions positively affects Knn classification model. *K* parameeter although behaves in two-ways, before some threshold in number of dimensions, increasing *k* parameeter affects results positively, but after the threshold - negatively. Table with results.

Rightly predicted labels percentage

	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
dim=1	49.4444	49.4444	53.3333	56.6667	53.8889	53.8889	53.3333	55.5556	52.2222	52.7778
dim=2	60	58.8889	60.5556	59.4444	63.8889	65	67.2222	65	66.6667	67.2222
dim=3	75	72.2222	75	77.2222	75.5556	76.6667	76.1111	78.8889	77.2222	78.8889
dim=4	82.2222	78.3333	85.5556	86.6667	85	83.8889	83.8889	84.4444	82.7778	86.1111
dim=5	90.5556	91.1111	91.6667	92.2222	88.8889	90	90	91.1111	87.2222	92.2222
dim=6	90	88.8889	89.4444	89.4444	86.6667	88.3333	86.6667	89.4444	86.1111	87.2222
dim=7	92.7778	93.3333	90.5556	92.7778	88.8889	91.1111	87.7778	87.7778	85.5556	88.3333
dim=8	92.2222	92.2222	88.8889	91.1111	87.2222	90	86.6667	88.3333	83.8889	85.5556

Tabel 1. KNN classifier rightly predicted labels percentage depending on dimensionality and k parameter