CS412: Introduction to Data Mining **Fall 2015**

Assignment 5

Due: 12/7/2015 11:59pm

General Instruction

- Errata: After the assignment is released, any further corrections of errors or clarifications will be posted at the Errata page at Piazza. Please watch it.
- Feel free to talk to other members of the class while doing the homework. We are more concerned that you learn how to solve the problem than that you solve it entirely on your own. You should, however, write the solution yourself.
- Please use Piazza first if you have questions about the homework. Also feel free to send us e-mails and come to office hours.
- For each question, you should show the necessary calculation steps and reasoning—not only final results. Keep the solution brief and clear.
- For a good balance of cognitive activities, we label each question with an activity type:
  - **L1 (Knowledge)** Definitions, propositions, basic concepts.
  - **L2 (Practice)** Repeating and practicing algorithms/procedures.
  - **L3 (Application)** Critical thinking to apply, analyze, and assess.

Assignment Submission

- Please submit your work before the due time. **We do NOT accept late submission!**
- Please submit your answers electronically via Compass. Contact CITES/TAs if you have technical difficulties in submitting the assignment.
- Please **type** your answers in an **Answer Document**, and submit it in PDF. **Handwritten answers or hand-drawn pictures are not acceptable**. Your answers to all questions (including mini-MP) should be included in one Answer Document.
- Please **DO NOT** zip the Answer Document (PDF) so that the graders can read it directly on Compass. Compress other files into a single zip file. Overall, you need to submit one Answer Document (PDF file), named as `hw5_netid.pdf`, and one `zip` file, named as `hw5_netid.zip`.
- If scripts are used, you should submit the source code, and use file names to identify the questions or sub-questions being answered. E.g., `question1_netid.py` is the python code for Question 1; and `question1a_netid.py` that for sub-question 1(a). You can submit separate files for sub-questions or a single file for the entire question.

1 Conceptual Questions (10 points)

a. (L3, 2') Given the same parameters, does DBSCAN always output the same cluster assignment or not? Briefly explain.

Answer

Not always. DBSCAN is not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed. Fortunately, this situation does not arise often, and has little impact on the clustering result.

b. (L3, 2') With improper choice of initial cluster centers, K-Means can give us a very bad clustering output. Could you propose a method to alleviate the issue?

Answer

Run K-Means several times with different initial cluster centers, and choose the best one with regard to some particular quality measure like the total distances between each point and its centroid.

c. (L1, 2') Between k-Medoids (PAM) and k-Means, which is more efficient? Briefly explain.

Answer

K-Means is more efficient. PAM: $O(k(n-k)^2$ for each iteration vs. K-Means: O(tkn) where normally, k, t << n). Thus, K-Means complexity is linear while PAM is quadratic, which means K-Means is more efficient than PAM.

d. (L1, 2') OPTICS is superior to DBSCAN because we do not have to set the paramenter ϵ. True or False? Briefly explain.

Answer

This question was removed because OPTICS was not covered in the lecture! Students will automatically have the two points if submitting the homework! Anyway, the answer is False. We still have to set ϵ but OPTICS give us hints to choose better (and bigger) ϵ through the graphical output

e. (L1, 2') Is it true that Lazy learners (e.g., kNN) is more efficient than non-lazy learners (e.g., decision trees)?

Answer

kNN involves little overhead for training but more time for predictions (e.g., finding nearest neighbors) than other non-lazy classifiers.

2 k-Nearest Neighbors (12 points)

You will use k-nearest-neighbor classifier to predict labels for new data points, and investigate under what situations kNN works well. The set of labeled data points are given in Tabel 1.

Purpose

- Understand how kNN works and its pros and cons.

Requirements

- No need to use distance-weighted labels of nearest neighbors.
- Show your calculations for questions asking for a number.

id	x	y	**label**
1	0	0	+1
2	0	1	+1
3	1	0	-1
4	1	1	-1
5	1.5	0.5	+1
6	2	0.5	-1

Table 1: Data for kNN

a. (L2, 2') Plot the given labeled data in the x-y plane, and highlight the regions where data will be classified as '+1' by a 1NN classifier (i.e, kNN for $k = 1$).

Answer
See Figure 1.

b. (L2, 2') Suppose we use a 3NN classifier based on 100 (labeled) data points uniformly distributed in the d-dimensional unit cube. For $d = 2$ (i.e, the unit square), What is the size of the smallest cube (square in this case) centered at q that covers at least 3 neighbors of q in expectation? Show your steps.

Answer
Each side of the required cube has length $0.03^{1/d}$.

c. (L2, 2') Calculate the size asked in (b) for $d = 100$.

Answer
See answer for (b).

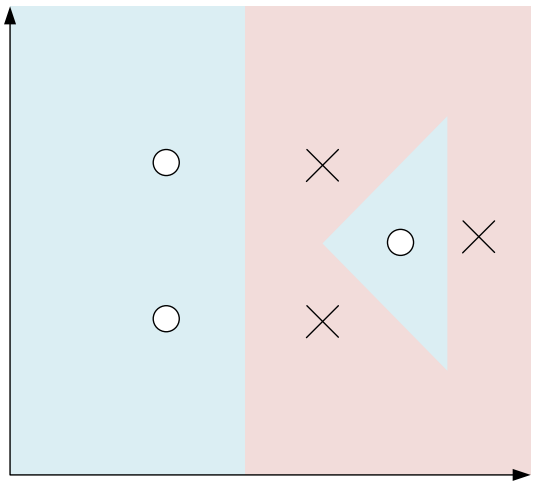


Figure 1: Decision regions

d. (L3, 2') According to your calculations in (b) and (c), do you see any problem in applying a kNN classifier to data in high-dimensional spaces, say, $d = 100$? Briefly explain. (*Hint: consider the underlying principle of why kNN works generally.*)

Answer

The nearest neighbors of any query point q becomes far away from q, hence less representative of q and might not be good to estimate q's label.

e. (L3, 2') What are the pros and cons of using a large k for the kNN classifier?

Answer

Larger k makes the classifier more robust to noises, but make the classification result less specific to the query point q; for k large enough, the class for all querying points will be the same.

f. (L3, 2') What are the pros and cons of using the kNN classifier, compared to other "non-lazy" classifiers (e.g, decision trees and Naive Bayes)?

Answer

Lazy classifiers needs much less time for training, but needs more time for predicting on new data.

3 Perceptrons (12 points)

You will design neurons and neural networks to implement some specified functions.

Purpose

- Understand how neurons and neural network approximates functions.
- See the different modeling capacities of single perceptrons and multilayer neural networks (multiplayer perceptrons).

Requirements

- Show the neurons and neural networks using diagrams.

a. (L2, 2') Given $x_1, x_2 \in \{0, 1\}$, design a neuron that implements the logical operation *AND*. That is, the neuron takes x_1 and x_2 as input and computes $y = x_1$ AND x_2 as output ($y \in \{0, 1\}$) (*Hint: use the activation function threshold*$(a) = 1$ *if* $a \geq 0$ *and 0 otherwise.*)

Answer
$w_1 = w_2 = 1, b = -2$. We assume that the threshold activation function is used throughout the question.

b. (L2, 2') Design a neuron that outputs $y = x_1$ OR x_2.

Answer
$w_1 = w_2 = 1, b = -1$.

c. (L2, 2') Design a neuron that takes a single input $x \in \{0, 1\}$ and outputs $y =$ NOT x.

Answer
$w = -1, b = 0$.

d. (L3, 3') Design a neural network that takes $x_1, x_2 \in \{0, 1\}$ as input and computes $y = x_1$ XOR x_2 as output (x_1 XOR $x_2 = 1$ if and only if $x_1 \neq x_2$). (*Hint: you can reuse the neurons you designed in (a)–(c)*)

Answer
See Figure 2.

e. (L3, 2') Is it possible to implement XOR a single perceptron? Explain your answer.

Answer
No, because the data points are not linearly separable in the case of XOR.

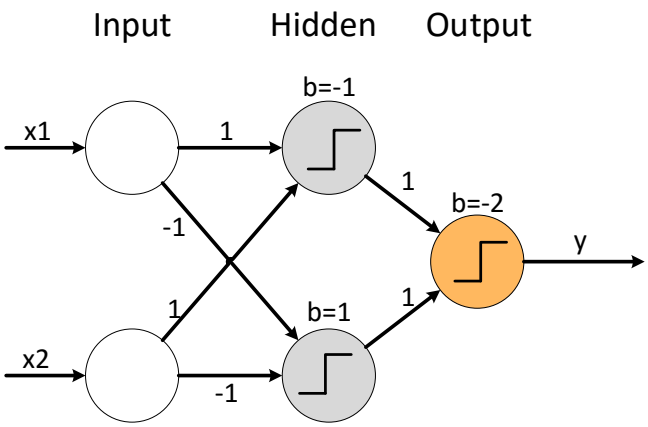


Figure 2: A neural network that implements XOR

4 Hierarchical Agglomerative Clustering and B-Cubed Evaluation (8 points)

Purpose

- Understand how AGNES and B-Cubed work.

Requirements

- In sub-question a, only draw the dendrogram.
- In sub-question b, only list the members of each cluster.
- In sub-question c, show detailed calculations of B-Cubed precision and recall.

Suppose we have 13 data points as listed and plotted in Figure 3. The ground truth (the correct clustering) is also provided.

a. (L2, 4') Draw the dendrogram using AGNES. Please use *single link* and *Euclidean distance* as the dissimilarity measure.

Answer
As in Figure 4:

Point	x	y	Ground-truth cluster
P1	1	3	C1
P2	1	2	C1
P3	2	1	C1
P4	2	2	C1
P5	2	3	C1
P6	3	2	C1
P7	4	3	C1
P8	6	3	C2
P9	4	5	C2
P10	5	4	C2
P11	5	5	C2
P12	6	4	C2
P13	6	5	C2

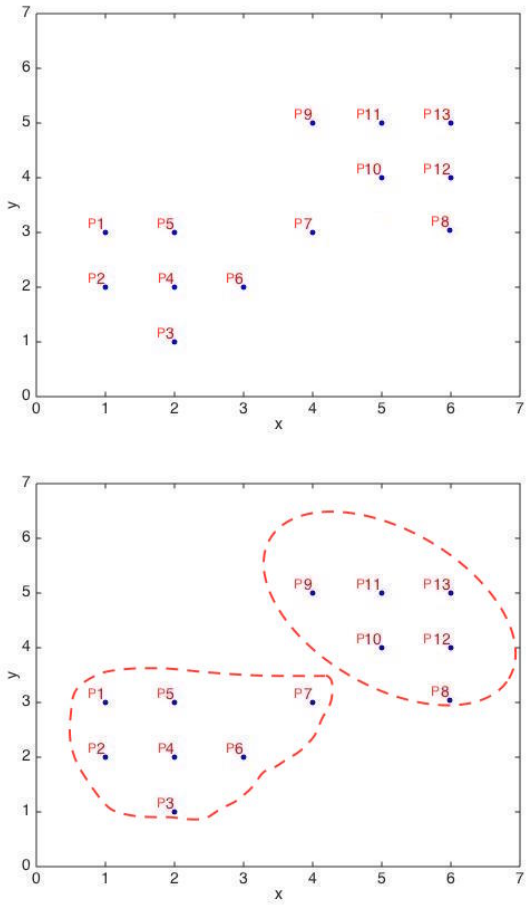


Figure 3: Data and ground truth

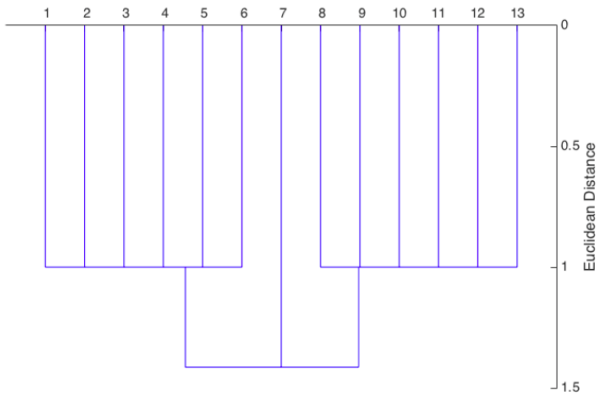


Figure 4: Dendrogram

b. (L2, 2') If we want to cluster the dataset into 3 groups based on the dendrogram, what are the members of each of the 3 groups?

   - C1: P1, P2, P3, P4, P5, P6
   - C2: P7
   - C3: P8, P9, P10, P11, P12, P13

c. (L2, 2') Based on the given ground truth, what are the *B-Cubed* precision and recall of the output?

Point i	1	2	3	4	5	6	7	8	9	10	11	12	13
P_i	6/6	6/6	6/6	6/6	6/6	6/6	1/1	6/6	6/6	6/6	6/6	6/6	6/6
R_i	6/7	6/7	6/7	6/7	6/7	6/7	1/7	6/6	6/6	6/6	6/6	6/6	6/6

$$Precision = (1/13) * \sum_{i=1}^{13} P_i = 1/13 * (13) = 1$$

$$Recall = (1/13) * \sum_{i=1}^{13} P_i = 1/13 * (6 * 6/7 + 1/7 + 6 * 6/6) = 79/91$$

5 K-Means (8 points)

Purpose

- Understand how k-Means works.

Requirements

- In sub-question a and b, for each iteration of k-Means:
  - Annotate the data points in figure 3 to show which points belong to which clusters, e.g., draw a red circle at each point belonging to cluster 1, and green and blue circles for cluster 2 and cluster 3. (The file for the figure is provided in `data.zip`.)

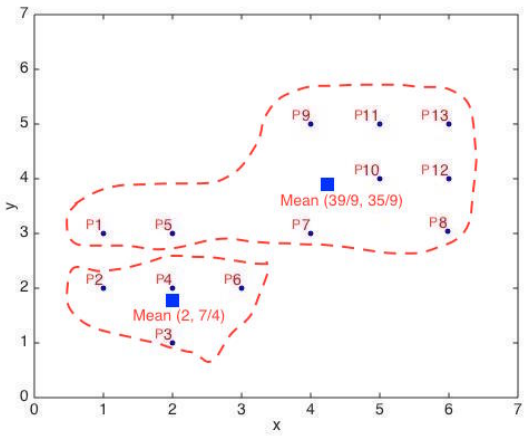
  - Plot the mean of each cluster with the same color you use for data points in this cluster, but in different shape to differentiate them from data points.
  - Show the coordinates of the cluster centers in each iteration.
  - Do not include scanned pictures. You might use annotation or image processing tools such as Mac Preview or Microsoft Paint to annotate the file.
  - Do not show numerical distances in each iteration.
- In sub-question c, any reasonable method with brief explanation is acceptable.

We will use the same data as question 4 (figure 3).

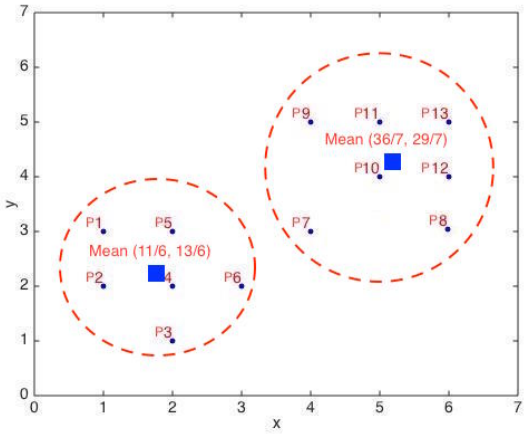
a. (L2, 4') Perform k-means using Euclidean distance with $k = 2$ and the initial cluster centers $P1$ and $P2$.

Answer

Iteration 1:



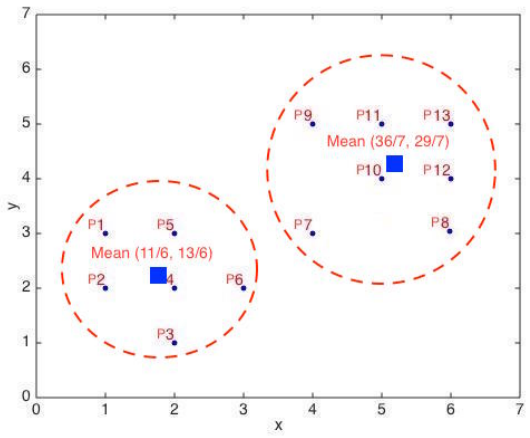
Iteration 2:



Algorithm terminated as no change was made.

b. (L2, 2') Perform k-means using Euclidean distance with $k = 2$ and the initial cluster centers $P2$ and $P12$. Answer

Iteration 1:



Algorithm terminated as no change was made.

c. (L3, 2') As you can see in sub-questions a and b, choices of initial cluster centers can affect the speed of the clustering process. Assume $k = 2$ and the data are 2-dimensional, suggest a general and reasonable way to select initial cluster centers to speed up the clustering process. Briefly justify your choice.

Answer
One way is to choose initial points very far from each other (such as the left-most point vs. the right-most point) so the algorithm can speed up the process to move the centroids to the converged positions.

6 Machine Problem (50 points)

Purposes

- Get deeper understanding and working experience of DBSCAN algorithm and B-Cubed evaluation through implementation and result visualization.
- Get hands-on experience with cluster analysis using Weka.

General Requirements

- This is the second and the last MP (not a mini one) we have in this course. It consists of several programming tasks, which usually take more time than written assignments, so please start early.

- This is an individual assignment, which means it is OK to discuss with your classmates and TAs regarding the methods, but it is not OK to work together or share code.
- Libraries or programs of cluster analysis algorithms can be found on-line, but you are prohibited from using these resources directly. This means you can not include external libraries, or modify existing programs, since the purpose of this MP is to help you go through DBSCAN step by step.
- You can use Java/C++/Python as your programming language. No other languages are allowed.
- Each sub question will contain detailed requirements. Please note that most of them require you to not only submit code but also write discussions, report results, or draw graphs. If you submit code without answering those questions, you will receive 0 point.
- Put all your codes in a separate folder with the name `NetId_assign5_codes`. Do not use sub-folders inside this folder. All of your codes should have been successfully compiled before submission. Do not include files other than the codes you write. Put a single `readme.txt` file in the code folder to briefly describe your codes and how to run them.
- Put the results you generate in another folder with the name `NetId_assign5_results`.
- Compress folder `NetId_assign5_codes` and `NetId_assign5_results` into a single zip file, and name it `NetId_assign5.zip`. Submit both this zip file and Answer Document `hw5_nedid.pdf` through Compass2g. Please do not zip the PDF file!
- If you copy source code from other students or external sources, you will receive a serious penalty. We will run plagiarism detection software. Please do not cheat.

Description of Data

- You can find `data-hw5.zip` from the course website. It contains three files: `data.txt`, `data.arff`, and `truth.txt`.
- The description of `data.txt` is as follows:
  - The first line contains an integer n as the number of data points.
  - Each of the n following lines corresponds to a 2-dimensional data point, which contains two floating-point numbers as the coordinates of a point. They are separated by a comma.
- File `data.arff` is a Weka-friendly version of `data.txt`.
- File `truth.txt` is the ground truth of clustering for the data in `data.txt`. It is hand-labeled by domain experts. The description of `truth.txt` is as follows:
  - The first line contains an integer n as the number of data points.
  - The second line contains an integer m as the number of clusters.

– Each of the n following lines corresponds to a 2-dimensional data point in `data.txt`, which contains three numbers, where the first two are the coordinates of the point, and the last one is an integer $c \in \{1 \dots m\}$ indicating the point belongs to cluster c, and $c = 0$ if the point is an outlier. The three numbers are separated by commas. Please note that the name/index of each cluster is unimportant. It is just a notation to indicate which points belong to the same cluster.

Step 0: $(0')$ **Normalizing Data.**

The first step you need to do is preprocessing data. In this assignment, you should min-max normalize each dimension of the data. The formula is as follows:

$$x\_normalized = \frac{x - minX}{maxX - minX}$$

We also provide you with the min-max-normalized data in file `data-normalized-hw5.zip` from the course website.

Step 1: $(23', L3)$ **Implementing DBSCAN.**

In this step, you need to implement DBSCAN:

- Name: `dbscan.java`, or `dbscan.py`, or `dbscan.cpp`
- Input
  - Dataset file, e.g., `/home/mike/mike_assign5/mike_assign5_data/data.txt`
  - Output file, e.g., `/home/mike/mike_assign5/mike_assign5_results/step1.txt`
  - Parameter $MinPts$, e.g, 25
  - Parameter ϵ, e.g, 0.065
- The structure of output file:
  - The first line contains $minPts$
  - The second line contains ϵ
  - The third line contains the number of clusters
  - Each of the n following lines corresponds to a 2-dimensional data point in `data.txt`, which contains three numbers, where the first two are the coordinates of the point, and the last one is integer $c \in \{1 \dots m\}$ if the point belongs to cluster c, or $c = 0$ if the point is an outlier. The three numbers are separated by comma characters.

As it is hard for humans to interpret the output, you need to visualize it. In particular, do a scatterplot for all the data points, and color the points according to the clustering. For example, all outlier points are black, all the points belonging to cluster 1 are red, all the points belonging to cluster 2 are blue, etc. You can create the plot using MATLAB, Excel or any tools you like. You do not need to include the code for creating the graph.

Requirements:

- Put your DBSCAN source code to folder `NetId_assign5_codes`

- Run your program on the given data file with $MinPts = 25$ and $\epsilon = 0.065$ so that we can check the correctness of your code by looking at the output file. Name the output file as `step1.txt`, and put it to folder `NetId_assign5_results`.
- Include in your Answer Document two scatterplots, one for your output and one for the ground truth.
- *Question to ponder A: How long does it take for your machine to run DBSCAN on the given data? What is the time complexity of DBSCAN in the worst case scenario? Can you describe a worst case scenario?*
- *Question to ponder B: Usually, a good clustering algorithm should put similar points to the same cluster, and dissimilar points to different clusters. Based on that intuition, what do you think of the clustering result? Given the same MinPts, should we increase or decrease ϵ to get more intuitive clustering results?*

Answer

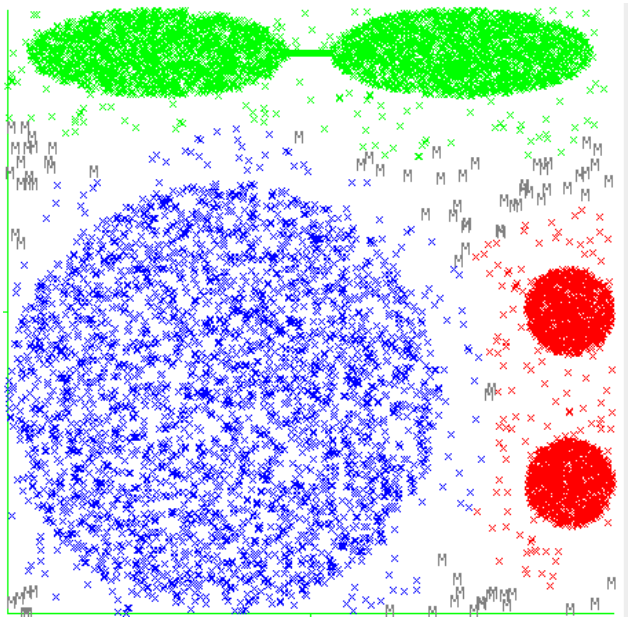


Figure 5: Output in step 1

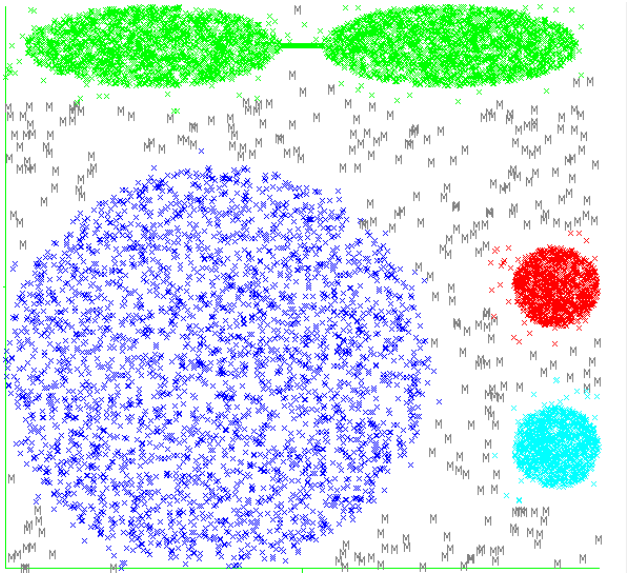


Figure 6: Truth

Question A: Any running time is acceptable. Both $O(nlogn)$ and $O(n^2)$ are accepted. Any reasonable description of any case that require computation $nlogn$ or n^2 are accepted. For example: we need to visit each data point, in each visit we need to search for neighbors in the entire dataset, so the complexity is $O(n^2)$

Question B: The clustering result merge two clusters into a bigger cluster, which looks not intuitive. In order to separate them, we need to decrease ϵ given that $MinPts$ is fixed.

Step 2: $(10', L3)$ **Parameter tuning for DBSCAN.**

Given $MinPts = 25$, find two new ϵ such that the numbers of clusters in the corresponding results are different from each other and from the ϵ in step 1.

Requirements:

- Run DBSCAN two times with the two ϵ described above, and create two output files with the format similar to `step1.txt`. Name them as `step2a.txt` and `step2b.txt`, and put them to folder `NetId_assign3_results`
- Create a table in the answer document to report the number of clusters, and the number of outliers corresponding to each of the two new ϵ and the ϵ in step 1 (your table should have three rows excluding the header).
- Draw two scatter plots corresponding to the two new ϵ. Include them in the Answer Document.
- *Question to ponder C: Which one of three ϵ is the best? Could you suggest a general way to tune parameters $MinPts$ and ϵ for DBSCAN?*

Answer

Any chosen ϵ that result in different numbers of clusters are acceptable. Answer to

question C: MinPts usually relies on domain knowledge while ϵ can be tuned using OPTICS.

Step 3: $(7', L3)$ **Implementing B-Cubed evaluation.**

In this step, you need to implement B-Cubed evaluation:

- Name: `bcubed.java`, or `bcubed.py`, or `bcubed.cpp`
- Input: two parameters:
  - Clustering output file, e.g., `/home/mike/mike_assign5/mike_assign5_results/step1.txt`
  - Ground truth file, e.g., `/home/mike/mike_assign5/mike_assign5_data/truth.txt`
- Output: Two lines in `stdout`
  - The first line contains *precision*
  - The second line contains *recall*

Requirements:

- Put your B-Cubed source code to folder `NetId_assign5_codes`
- Run B-Cubed with the outputs from Step 1 and Step 2. Create a table to report the *precision* and *recall* corresponding to each of the outputs.
- *Question to ponder D: Can you suggest a way to combine B-Cubed precision and recall into a single measure so that it would be easier to compare different results?*

Answer

The table depends on your choice in step 2. For step 1, precision is 0.856 and recall is 1.00. If you want to have recall < 1.00, you only need to set ϵ sufficiently low.

Question D: Any reasonable measure with reasonable explanation has full score. One typical choice is F1-score: https://en.wikipedia.org/wiki/F1_score. It is good because when either precision or recall is low, which we don't like, F1-score is low; and when both precision and recall are decent, which we like, F1-score is high. Thus, F1-score is a reasonable measure to combine precision and recall.

Step 4: $(10', L2)$ **Doing cluster analysis by Weka.**

In class, we have demonstrated how to do this. You can watch the online lecture video to review the process. We also provide the procedure below.

(a) Open Weka.

(b) Choose `Explorer`.

(c) Click on `Preprocess` tag.

(d) Click on `Open file`, choose `data.arff` , click on `Choose` and wait until Weka finishes loading the data.

(e) Click on `Cluster` tag. Here you can find common algorithms for cluster analysis, such as OPTICS, DBSCAN, etc.

Requirements:

- Run `SimpleKMeans` with different number of clusters. Report the best number of clusters, and the plot showing the corresponding clustering result. You can generate the plot by right clicking on the entry of the `Result list`, and choose `Visualize cluster assignments`.
- Run DBSCAN with the best *minPts* and ϵ in Step 2. Show the plot for the corresponding clustering result.
- *Question to ponder E: Compare the results from k-Means and DBSCAN. Is k-Means or DBSCAN more suitable for the provided dataset? Why?*

Answer

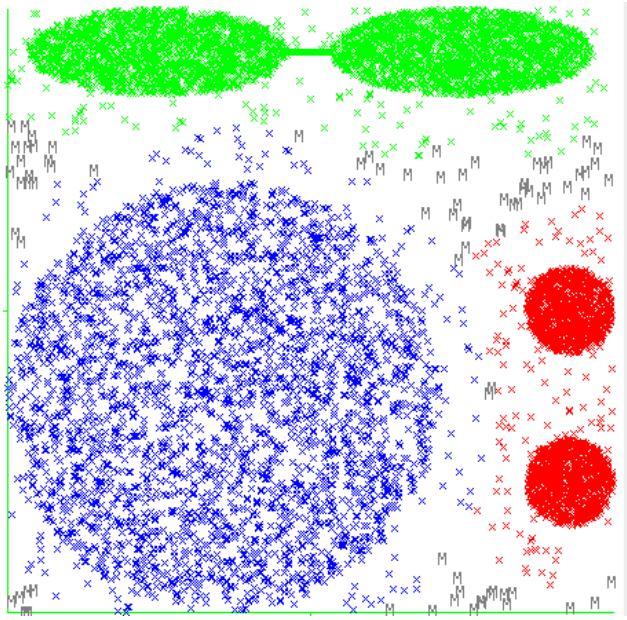


Figure 7: DBSCAN in Weka

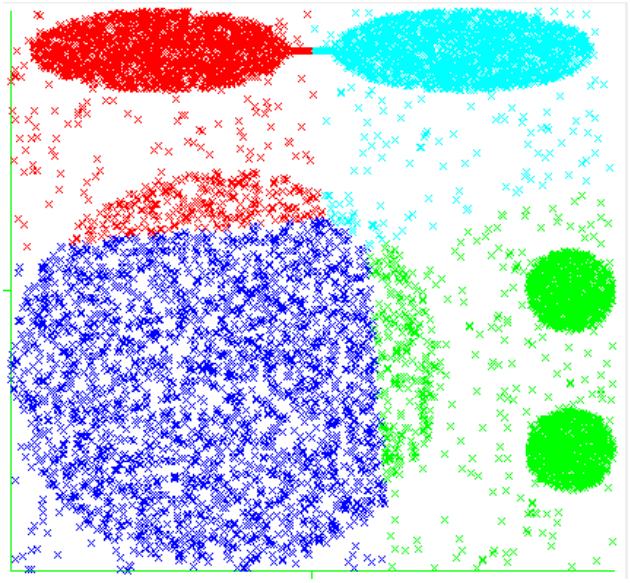


Figure 8: K-Means in Weka

Question E: DBSCAN because the clusters have arbitrary shapes. No matter how hard you tune the parameters for K-Means, you will never get good results, as the clusters here do not have the round shape.