Math 123 Homework 5

Hanyuan Zhu

October 25, 2018

Question1

(a) As $\epsilon \to \infty$ and $MinPts \leq n$ for dataset $\{x_i\}_{i=1}^n$, the entire dataset become one cluster.

As $\epsilon \to +0$, and $MinPts \leq 1$, each point will become a single-point cluster; while the $MinPts \geq 2$, All points will become noisy points.

(b) As $MinPts \to \infty$, all points will be noises. No cluster will form.

As $MinPts \to +0$, all points are core points to form clusters, and no noise.

Question2

(a) Below is my function for DBSCAN algorithm. You can call it by DBSCAN(dataBase, eps, minPts), where dataBase is $n \times 2$ array, eps is the threshold of radius of corepoints neighorhood, and minPts is the minimum points in neighorhood qualified for corepoint.

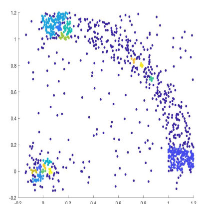
It returns a $3 \times n$ matrix. The first two elements of each column representing a point, and the third element is the label of cluster of the point. The label of all noise point is -1 here.

```
function [DB]=DBSCAN(dataBase, eps, minPts)

C = 0; % Cluster Counter

%% Prepare the Dataset
labelDB = zeros(length(dataBase),1);% Create a label set for DB
DB = [dataBase labelDB]';% transpose the DB, to make each vertical
    component of dataset as a point. for the "for loop" . Union the
    label and dataset
% NOTICE : Each column of DB is a point now

%% Preprocess the neigbhorhood set
DBdistance = pdist2(dataBase,dataBase);
NC = {};
toSearch = [];
```

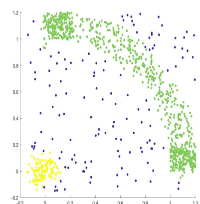
```
%% to generate neighborhood matrix [[size of neighborhood,
    corepoint=1],[list
%of neigbhorhood index]]
for idx = 1:length(DBdistance)
    aa = find(DBdistance(:,idx)<eps & DBdistance(:,idx)>0);
    if length(aa) >= minPts
        NC{end+1} = {[length(aa),1],aa};
        toSearch(end+1) = idx;
    else
        NC{end+1} = {[length(aa),0],aa};
        DB(3,idx) = -1;
    end
end


%% Here only search all core points
while ~isempty(toSearch)
    idx = toSearch(1);
    toSearch(1) = [];

    if DB(3,idx)
        continue
    end

    C=C+1; % if P satisfies all conditions above, label lable it as a
        new cluster

    DB(3,idx)=C;
    seedSet= NC{idx}{2};

    while ~isempty(seedSet)
        idx = seedSet(1); % the index of such element in DB
        seedSet(1) = [];
        if DB(3,idx) == -1 %check if it is noise first. to update the
            label of noise point to border point
            DB(3,idx) = C;
        end

        if DB(3,idx) %skip all labelled point in neigbhorhood
            continue
        end

        seedSet = union(seedSet,NC{idx}{2});% adding the neighorhood of
            the idx core point to the seedset
        DB(3,idx) = C; % update the cluster label of idx point
    end

end
    scatter(DB(1,:),DB(2,:),25,DB(3,:),'filled')
end
```

(b) I run the following code block to to try different values of eps and minPts

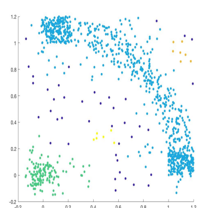
```
load("DBSCAN_Data.mat");

%fixing the minPts for various eps.
DBSCAN(X,0.02,5);
DBSCAN(X,0.05,5);
DBSCAN(X,0.1,5);
DBSCAN(X,0.2,5);

%fixing the eps for various minPts.
DBSCAN(X,0.05,2);
DBSCAN(X,0.05,5);
DBSCAN(X,0.05,10);
DBSCAN(X,0.05,50);
DBSCAN(X,0.05,51);
```



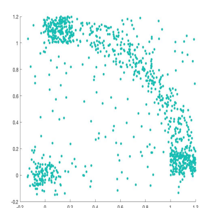
(a) eps=0.02



(b) eps=0.05

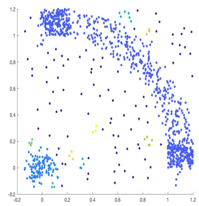


(a) eps=0.1

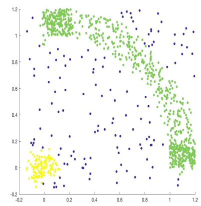


(b) eps=0.2

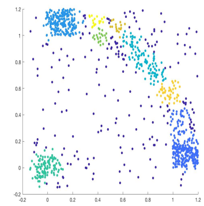
Figure 1: Fixing the mintPts = 5



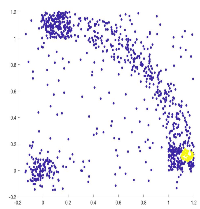
(a) minPts = 2



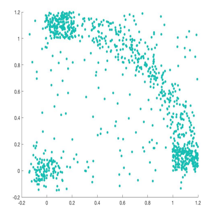
(b) minPts = 5



(c) minPts = 10



(a) minPts = 50



(b) minPts = 51

Figure 2: Fixing the eps = 0.05

Clearly, the relatively best partition result is at eps = 0.05 and minPts = 5.

The DBSCAN fails at eps larger than 1.5(for fixing minPts=5), where more and more noise points will be included.

The DBSCAN completely fails at minPts larger than 50(for fixing eps=0.05), where all points are counted as noise.

Question3

(a) To show L is positive semidefinite, that is, to show $yLy^T \geq 0$ for any $y \in \mathbb{R}^{1\times n}$, $y \neq 0$

Because $L = D - W$,

$$\begin{aligned} yLy^T &= \sum_{i=1}^{n}\sum_{j=1}^{n} L_{ij}y_iy_j \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} (D_{ij} - W_{ij})y_iy_j \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} D_{ij}y_iy_j - \sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}y_iy_j \end{aligned} \tag{1}$$

Since $D_{ij} = 0, if i \neq j$ and $D_{ii} = \sum_{j=1}^{n} W_{ij}$, also by relabeling the index, we have $\sum_{i=1}^{n} D_{ii}y_i^2 = \sum_{j=1}^{n} D_{jj}y_j^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}y_j^2$ then

$$
\begin{aligned}
(1) &= \sum_{i=1}^{n} D_{ii} y_i^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} y_i y_j \\
&= \frac{1}{2}(2\sum_{i=1}^{n} D_{ii} y_i^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}) \\
&= \frac{1}{2}(\sum_{i=1}^{n} D_{ii} y_i^2 + \sum_{j=1}^{n} D_{jj} y_j^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}) \\
&= \frac{1}{2}(\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} y_i^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} y_j^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} y_i y_j) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}(y_i^2 + y_j^2 - y_i y_j) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}(y_i - y_j)^2
\end{aligned} \tag{2}
$$

Because $W_{ij} \in [0, 1]$ and $(y_i - y_j)^2 \geq 0$, then

$$
yLy^T = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}(y_i - y_j)^2 \geq 0 \tag{3}
$$

Thus L is positive semidefinite matrix.

(b) Because L has a eigenvalue 0, that is, $v^T L v = 0$, where v is the eigenvector of eigenvalue 0, and $v \neq 0$. Therefore L is not positive definite.

Question4

Because the weight function is determined by the difference of the color (or brightness) of two pixel points, thus when the σ becomes sufficiently large, the color (or brightness) between two pixels are become smaller or even neglectible.

Therefore, as the σ go larger, the wider color range will be shown on the results.

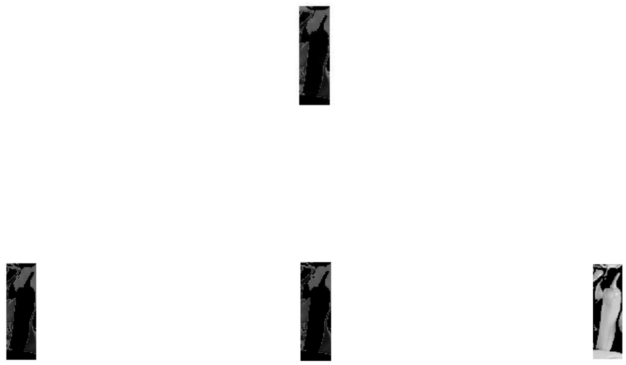


Figure 3: from top to bottom right, σ are 0.2, 0.5, 1 and 5

The code for graph Lapacian

```
load('Ncut_Data.mat')
reshaped = reshape(pepper,[], 1);

% Wdist is the matrix of the 2 norm of x_i and x_j
Wdist = pdist2(reshaped,reshaped);

%set sigma here
sigma = 0.2;
sigM = ones(size(Wdist))*sigma;

% here is the weight matrix
W = arrayfun(@expoweight, Wdist, sigM);

%sum each column of W, to get the D_i. and diagnolize the vector.
D =diag(sum(W,2));

%graph Lapacian = D - W
L = D - W;

%eigenvlaue decomposition
[V E] = eig(L);

% Since the eigenvalue is sorted. the second smallest is E(2,2), and the
% vector is V(:,2)

%processing the data
```

```
imagebyV = arrayfun(@bipartitioN,reshaped,V(:,2));

newpepper = reshape(imagebyV,100,31);

imshow(newpepper)
```