q1

September 29, 2019

1 Assignment 3

1.1 Question

Dataset: 1. “Diabetes.arff” file contains the dataset. 2. Each row has 9 comma separated values where first 8 values represent asingle datapoint (8 dim vector values). Ignore the 9th value.

Questions: There are two parameters in DBSCAN algorithm: a. Eps: radius length b. minPts: minimum number of points required to form a cluster. 1. Implement DBSCAN algorithm and find number of clusters formed for eps = 2 and minPts = 5 2. For any one cluster, show its core point and border points.

1.2 The Solution

1.2.1 Importing the libraries

```
In [1]: from scipy.io import arff
        import pandas as pd
        import math
        from collections import OrderedDict
        from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

1.2.2 Import data file in form of DataFrame

```
In [2]: df = pd.read_csv('diabetes.csv', header=None)
        del df[8]
        df.head()
```

```
Out[2]:            0          1          2          3          4          5  \
        0  35.294118  74.371859  59.016393  35.353535   0.000000  50.074516
        1   5.882353  42.713568  54.098361  29.292929   0.000000  39.642325
        2  47.058824  91.959799  52.459016   0.000000   0.000000  34.724292
        3   5.882353  44.723618  54.098361  23.232323  11.111111  41.877794
        4   0.000000  68.844221  32.786885  35.353535  19.858156  64.232489

                   6          7
        0  23.441503  48.333333
        1  11.656704  16.666667
        2  25.362938  18.333333
```

```
3   3.800171   0.000000
4  94.363792  20.000000
```

1.2.3 Normalizing the features with StandardScaler

```
In [3]: scaler = StandardScaler()
        df = pd.DataFrame(scaler.fit_transform(df))
        df.head()
```

```
Out[3]:          0         1         2         3         4         5         6  \
        0  0.639947  0.848324  0.149641  0.907270 -0.692891  0.204013  0.468492
        1 -0.844885 -1.123396 -0.160546  0.530902 -0.692891 -0.684422 -0.365061
        2  1.233880  1.943724 -0.263941 -1.288212 -0.692891 -1.103255  0.604397
        3 -0.844885 -0.998208 -0.160546  0.154533  0.123302 -0.494043 -0.920763
        4 -1.141852  0.504055 -1.504687  0.907270  0.765836  1.409746  5.484909

                 7
        0  1.425995
        1 -0.190672
        2 -0.105584
        3 -1.041549
        4 -0.020496
```

1.2.4 Function to calculate Euclidean distance between two points

```
In [4]: def dist(pointX, pointY):
            disSquare = 0
            for i in range(len(pointX)):
                if (i == 8):
                    break

                disSquare += (pointX[i] - pointY[i]) ** 2

            return math.sqrt(disSquare)
```

1.2.5 Function to get the neighbours(points in the epsilon neighbourhood) of a point

```
In [5]: def getNeighbours(database, point, eps, idx):
            neighbours = []

            for i in range(len(database)):
                if i == idx:
                    continue

                if (dist(point, database[i]) <= eps):
                    neighbours.append(i)

            return neighbours
```

1.2.6 Part 1 : Implementing the DBSCAN Clustering Algorithm

Running the algorithm

```
In [6]: eps = 2
        minPts = 5
        cur_cluster_label = -1
        database = []

        for index, row in df.iterrows():
            rowX = []
            for x in row:
                rowX.append(x)
            rowX.append('Undefined')
            rowX.append(-1)
            database.append(rowX)


        for idx in range(len(database)):
            if (database[idx][-1] != -1):
                continue

            neighbours = getNeighbours(database, database[idx], eps, idx)
            if (len(neighbours) + 1 < minPts):
                database[idx][-2] = 'Noise'
                continue

            cur_cluster_label += 1
            database[idx][-2] = 'Core'
            database[idx][-1] = cur_cluster_label

            for x in neighbours:
                if database[x][-2] == 'Noise':
                    database[x][-1] = cur_cluster_label
                    database[x][-2] = 'Border'

                if database[x][-1] != -1:
                    continue

                database[x][-1] = cur_cluster_label
                database[x][-2] = 'Border'
                neighboursY = getNeighbours(database, database[x], eps, x)

                if (len(neighboursY) + 1 >= minPts):
                    for y in neighboursY:
                        database[x][-2] = 'Core'
                        neighbours.append(y)

        clusters = {}
        for idx in range(len(database)):
            if database[idx][-1] not in clusters:
                clusters[database[idx][-1]] = []

            clusters[database[idx][-1]].append(idx)

        clusters = OrderedDict(sorted(clusters.items(), key=lambda x: x[0]))
```

Printing the clusters formed along with outliers present if any

```
In [7]: for i in clusters.keys():
            if i == -1:
                print("Outliers" + " -> " + str(clusters[i]))
                print("Clusters:")
            else:
                print(str(i) + " -> " + str(clusters[i]))
                print(" ")
```

```
Outliers -> [4, 8, 9, 13, 43, 45, 58, 75, 145, 177, 182, 193, 220, 228, 231, 247, 254, 342, 349
Clusters:
0 -> [0, 1, 2, 3, 5, 6, 10, 11, 12, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29

1 -> [7, 15, 78, 222, 261, 266, 269, 300, 332, 336, 347, 430, 435, 468, 484, 533, 535, 589, 60

2 -> [49, 60, 81, 426, 494, 522, 706]
```

1.2.7 Part 2 : For any one cluster, show its core point and border points

Printing the Core and Border points for Cluster 0

```
In [8]: print("Index\tPoint Type")
        for i in clusters[0]:
            print(str(i) + "\t" + database[i][-2])
```

```
Index        Point Type
0        Core
1        Core
2        Core
3        Core
5        Core
6        Core
10        Core
11        Core
12        Border
14        Core
16        Core
17        Core
```

```
18        Core
19        Core
20        Core
21        Core
22        Core
23        Core
24        Core
25        Core
26        Core
27        Core
28        Border
29        Core
30        Core
31        Core
32        Core
33        Core
34        Core
35        Core
36        Core
37        Core
38        Core
39        Border
40        Core
41        Core
42        Core
44        Core
46        Core
47        Core
48        Core
50        Core
51        Core
52        Core
53        Core
54        Core
55        Core
56        Core
57        Core
59        Core
61        Core
62        Core
63        Core
64        Core
65        Core
66        Core
67        Border
68        Core
69        Core
70        Core
```

```
71          Core
72          Core
73          Core
74          Core
76          Core
77          Core
79          Core
80          Core
82          Core
83          Core
84          Core
85          Core
86          Core
87          Core
88          Core
89          Core
90          Core
91          Core
92          Core
93          Core
94          Core
95          Core
96          Core
97          Core
98          Core
99          Core
100          Border
101          Core
102          Core
103          Core
104          Core
105          Core
106          Border
107          Core
108          Core
109          Core
110          Core
111          Core
112          Core
113          Core
114          Core
115          Core
116          Core
117          Core
118          Core
119          Core
120          Border
121          Core
```

122	Core
123	Core
124	Core
125	Border
126	Core
127	Core
128	Core
129	Core
130	Core
131	Core
132	Core
133	Core
134	Core
135	Core
136	Core
137	Core
138	Core
139	Core
140	Core
141	Core
142	Core
143	Core
144	Core
146	Core
147	Core
148	Core
149	Core
150	Core
151	Core
152	Core
153	Core
154	Core
155	Border
156	Core
157	Core
158	Core
159	Border
160	Core
161	Core
162	Core
163	Core
164	Core
165	Core
166	Core
167	Core
168	Core
169	Core
170	Core

171	Core
172	Border
173	Core
174	Core
175	Core
176	Core
178	Core
179	Core
180	Core
181	Core
183	Core
184	Core
185	Core
186	Border
187	Border
188	Core
189	Core
190	Core
191	Core
192	Core
194	Core
195	Core
196	Core
197	Core
198	Core
199	Core
200	Core
201	Core
202	Core
203	Core
204	Core
205	Core
206	Core
207	Core
208	Core
209	Core
210	Core
211	Core
212	Border
213	Core
214	Core
215	Core
216	Core
217	Core
218	Core
219	Core
221	Border
223	Core

224	Core
225	Core
226	Core
227	Core
229	Core
230	Core
232	Core
233	Core
234	Core
235	Core
236	Core
237	Core
238	Core
239	Core
240	Core
241	Core
242	Core
243	Core
244	Core
245	Border
246	Core
248	Core
249	Core
250	Core
251	Core
252	Core
253	Core
255	Core
256	Core
257	Core
258	Border
259	Border
260	Core
262	Core
263	Core
264	Core
265	Core
267	Core
268	Core
270	Border
271	Core
272	Core
273	Core
274	Core
275	Core
276	Core
277	Core
278	Core

279	Core
280	Core
281	Core
282	Core
283	Core
284	Core
285	Core
286	Border
287	Core
288	Core
289	Core
290	Core
291	Core
292	Core
293	Core
294	Border
295	Core
296	Core
297	Core
298	Core
299	Core
301	Core
302	Core
303	Border
304	Core
305	Core
306	Core
307	Core
308	Core
309	Core
310	Core
311	Core
312	Core
313	Core
314	Core
315	Core
316	Core
317	Core
318	Core
319	Core
320	Core
321	Core
322	Core
323	Core
324	Core
325	Core
326	Core
327	Core

328	Core
329	Core
330	Border
331	Core
333	Core
334	Core
335	Core
337	Core
338	Core
339	Core
340	Core
341	Core
343	Core
344	Core
345	Core
346	Core
348	Core
350	Core
351	Core
352	Core
353	Core
354	Core
355	Core
356	Core
358	Core
359	Core
360	Core
361	Core
363	Core
364	Core
365	Core
366	Core
367	Core
368	Core
369	Core
372	Core
373	Core
374	Core
375	Border
376	Core
377	Core
378	Core
379	Core
380	Core
381	Core
382	Core
383	Core
384	Core

385	Core
386	Core
387	Core
388	Core
389	Core
390	Core
391	Core
392	Core
393	Core
394	Core
395	Border
396	Core
397	Core
398	Core
399	Core
400	Core
401	Core
402	Core
403	Core
404	Core
405	Core
406	Core
407	Core
408	Border
409	Core
410	Core
411	Core
412	Core
413	Core
414	Core
415	Core
416	Core
417	Core
418	Core
419	Core
420	Core
421	Core
422	Core
423	Core
424	Core
425	Core
427	Core
428	Core
429	Core
431	Core
432	Core
433	Core
434	Core

436	Core
437	Core
438	Core
439	Core
440	Border
441	Core
442	Core
443	Core
444	Core
446	Core
447	Core
448	Core
449	Core
450	Core
451	Core
452	Core
454	Core
455	Border
456	Core
457	Core
458	Core
460	Core
461	Core
462	Core
463	Core
464	Border
465	Core
466	Core
467	Core
469	Core
470	Core
471	Core
472	Core
473	Core
474	Core
475	Core
476	Core
477	Core
478	Core
479	Core
480	Core
481	Core
482	Core
483	Core
485	Core
486	Core
487	Border
488	Core

489	Core
490	Core
491	Core
492	Core
493	Core
495	Core
496	Core
497	Core
498	Border
499	Core
500	Core
501	Core
503	Core
504	Core
505	Core
506	Core
507	Core
508	Core
509	Core
510	Core
511	Core
512	Core
513	Core
514	Core
515	Core
516	Core
517	Core
518	Core
520	Core
521	Core
523	Core
524	Core
525	Core
526	Core
527	Core
528	Core
529	Core
530	Core
531	Core
532	Core
534	Core
536	Core
538	Core
539	Core
540	Core
541	Core
542	Core
543	Core

544	Core
545	Core
546	Core
547	Core
548	Core
550	Core
551	Core
552	Core
553	Core
554	Core
555	Core
556	Core
557	Core
558	Core
559	Core
560	Core
561	Core
562	Core
563	Core
564	Core
565	Core
566	Core
567	Core
568	Core
569	Core
570	Core
571	Core
572	Core
573	Core
574	Core
575	Core
576	Core
577	Core
578	Core
580	Core
581	Core
582	Core
583	Core
584	Border
585	Core
586	Core
587	Core
588	Border
590	Border
591	Core
592	Core
593	Core
594	Core

595	Core
596	Border
597	Core
598	Core
599	Core
600	Core
602	Core
603	Core
605	Core
606	Border
607	Core
608	Core
609	Core
610	Core
611	Core
612	Core
613	Core
614	Core
615	Core
616	Core
617	Core
618	Core
620	Core
621	Border
623	Core
624	Core
625	Core
626	Core
627	Core
628	Core
629	Core
630	Core
631	Core
632	Core
633	Core
634	Core
635	Core
636	Core
637	Core
638	Core
639	Core
640	Core
641	Core
642	Core
644	Core
645	Core
646	Core
647	Core

648	Core
649	Core
650	Core
651	Core
652	Core
653	Core
654	Core
655	Core
656	Core
657	Core
658	Core
659	Core
660	Core
662	Border
663	Core
664	Core
665	Core
666	Core
667	Core
668	Core
669	Core
670	Core
671	Core
672	Border
673	Border
674	Core
675	Core
676	Core
677	Core
678	Core
679	Core
680	Core
681	Core
682	Core
683	Core
685	Core
686	Core
687	Core
688	Core
689	Core
690	Core
691	Border
692	Core
693	Core
694	Core
695	Border
696	Core
698	Core

699	Core
700	Core
701	Core
704	Core
705	Core
707	Core
708	Core
709	Core
710	Core
711	Core
712	Core
713	Core
714	Core
715	Border
716	Core
717	Core
718	Core
719	Core
720	Core
721	Core
722	Core
723	Core
724	Core
725	Core
726	Core
727	Core
728	Core
729	Core
730	Core
731	Core
732	Core
733	Core
734	Core
735	Core
736	Core
737	Core
738	Core
739	Core
740	Core
741	Core
742	Core
743	Core
744	Core
745	Core
746	Core
747	Core
748	Core
749	Core

750	Core
751	Core
752	Core
753	Core
754	Core
755	Core
756	Core
757	Core
758	Core
759	Core
760	Core
761	Core
762	Core
763	Border
764	Core
765	Core
766	Core
767	Core