Problem III: DBSCAN

1452669, Yang LI, April 8

Data Preprocessing

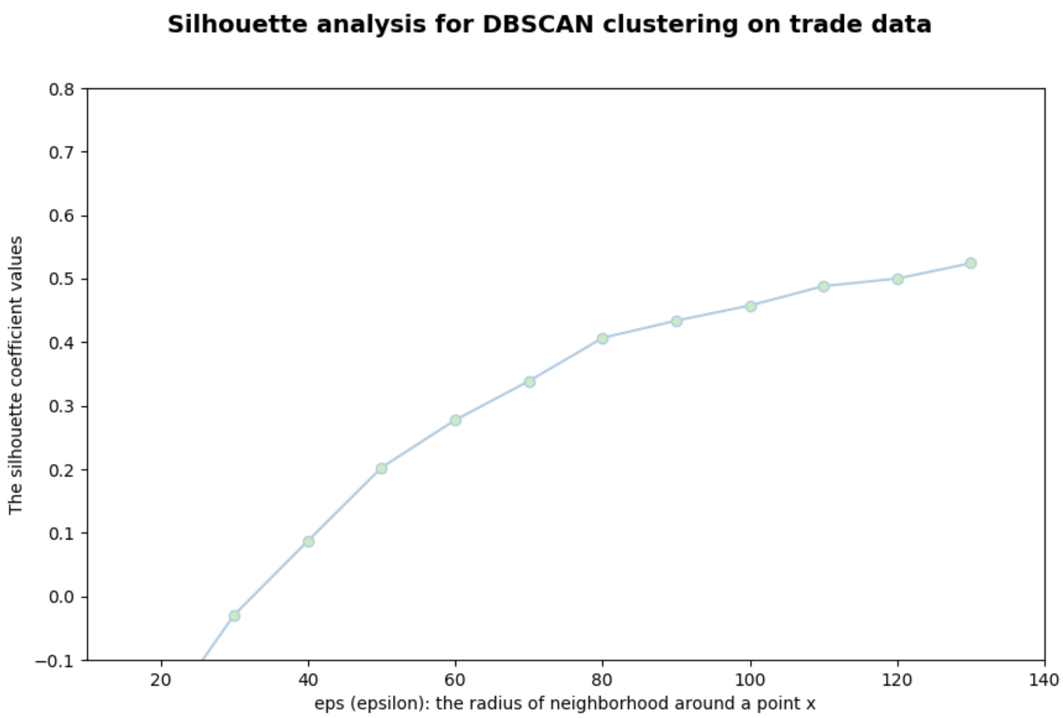
Since the quality of DBSCAN depends on the distance measure used in the function `regionQuery`. The most common distance metric, as I did, used is Euclidean Distance, Especially for high-dimensional data, this metric can be rendered almost useless due to the Curse of Dimensionality, making it difficult to find an appropriate value for eps.

As a response, I use standardization for data preprocessing. `sklearn` provides a class `StandardScaler` to do a Z-score standardization, which has mean 0 and standard deviation 1. Thus, the standard score of a raw score as follows:

$z = \frac{x-\mu}{\sigma}$

DBSCAN

1. Find the ε (eps) neighbors of every point, and identify the core points with more than minPts neighbors.
2. Find the connected components of *core* points on the neighbor graph, ignoring all non-core points.
3. Assign each non-core point to a nearby cluster if the cluster is an ε (eps) neighbor, otherwise assign it to noise.



Detailed result data listed in the following table:

eps	silhouette score	eps	silhouette score
10	-0.31939086805466427	80	0.4067160039619981
20	-0.1942397862954418	90	0.43392130605710816
30	-0.029129862530624537	100	0.4576647790706404
40	0.08708130699171691	110	0.4883338136442235
50	0.20268451888941394	120	0.5001686210200447
60	0.2775394743004492	130	0.5243956411873608
70	0.3387778176593896		

Compared with the clustering of kNN using LSH, the result are in the following table (duplicate 100 times):

k	correctness
2	0.792
3	0.514
4	0.444
5	0.484
6	0.654
7	0.798
8	0.878
9	0.93
10	0.94
11	0.97
12	1
13	1

K-means vs DBSCAN

As for the clustering result, K-means's silhouette score is much better than DBCSAN may due to that DBSCAN is not entirely deterministic and have Curse of Dimensionality in our high dimension dataset.

Thus, DBSCAN cannot cluster data sets well with large differences in densities, since the minPts-ε combination cannot then be chosen appropriately for all clusters.

In the other part, DBSCAN does not require one to specify the number of clusters in the data a priori. It can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster. Due to the MinPts parameter, the so-called single-link effect (different clusters being connected by a thin line of points) is reduced. Thus, DBSCAN has a notion of noise, and is robust to outliers and can be used with databases that can accelerate region queries.

Performance

Time & Space Complexity in Theory

- time complexity
  - average case: $O(nlgn)$
  - worst case: $O(n^2)$
- space complexity: $O(n^2)$ to store distance matrix

Benchmark in Practice

```
Timer unit: 1e-06 s

Total time: 6.06212 s
File: /Users/Yang/Developer/420235DataMining/hw1/q3/dbscan.py
Function: dbscan at line 10

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    10                                           @profile
    11                                           def dbscan(df, random_vip, knns):
    12         1        207.0    207.0      0.0      silhouette_avgs = []
    13         1          3.0      3.0      0.0      hits = []
    14         1          1.0      1.0      0.0      x_min = 10
    15         1          1.0      1.0      0.0      x_max = 140
    16        14         40.0      2.9      0.0      for eps in numpy.arange(x_min, x_max, 10):
    17        13        125.0      9.6      0.0          logging.info("DBSCAN: eps = {}".format(eps))
    18        13     155072.0  11928.6      2.6          X = StandardScaler().fit_transform(df.T)
    19        13    5831575.0 448582.7     96.2          db = DBSCAN(eps=eps).fit(X)
    20        13        347.0     26.7      0.0          core_samples_mask = numpy.zeros_like(db.labels_, dtype=bool)
    21        13         64.0      4.9      0.0          core_samples_mask[db.core_sample_indices_] = True
    22        13         53.0      4.1      0.0          cluster_labels = db.labels_
    23        13        453.0     34.8      0.0          n_clusters = len(set(cluster_labels)) - (
    24        13        394.0     30.3      0.0              1 if -1 in cluster_labels else 0)
    25        13      72657.0   5589.0      1.2          silhouette_avg = silhouette_score(X, db.labels_)
    26        13         40.0      3.1      0.0          silhouette_avgs.append(silhouette_avg)
    27        13         25.0      1.9      0.0          logging.info("For n_clusters =", n_clusters,
    28        13          9.0      0.7      0.0                       "The average silhouette_score in DBSCAN is :",
    29        13        156.0     12.0      0.0                       silhouette_avg)
    30
    31        13         10.0      0.8      0.0          hit = 0
    32        13        211.0     16.2      0.0          no = cluster_labels[df.columns.get_loc(random_vip)]
    33        78         67.0      0.9      0.0          for neighbor in knns:
    34        65        243.0      3.7      0.0              if cluster_labels[df.columns.get_loc(neighbor)] == no:
    35        36         29.0      0.8      0.0                  hit += 1
    36                                                       else:
    37        29         24.0      0.8      0.0                  logging.debug(
    38        29         25.0      0.9      0.0                      "DBSCAN: vipno: {} is not in the same cluster.".format(
    39        29        149.0      5.1      0.0                          neighbor))
    40        13         11.0      0.8      0.0          logging.info(
    41        13         10.0      0.8      0.0              "For k = {} in kNN, there has {} in the same cluster in DBSCAN.".format(
    42        13         96.0      7.4      0.0                  len(knns), hit))
    43        13         17.0      1.3      0.0          hits.append(hit)
    44
    45                                               # plot_silhouette(x_min, x_max, silhouette_avgs)
    46
    47         1          4.0      4.0      0.0      return silhouette_avgs.index(max(silhouette_avgs)) / 10, hits
```

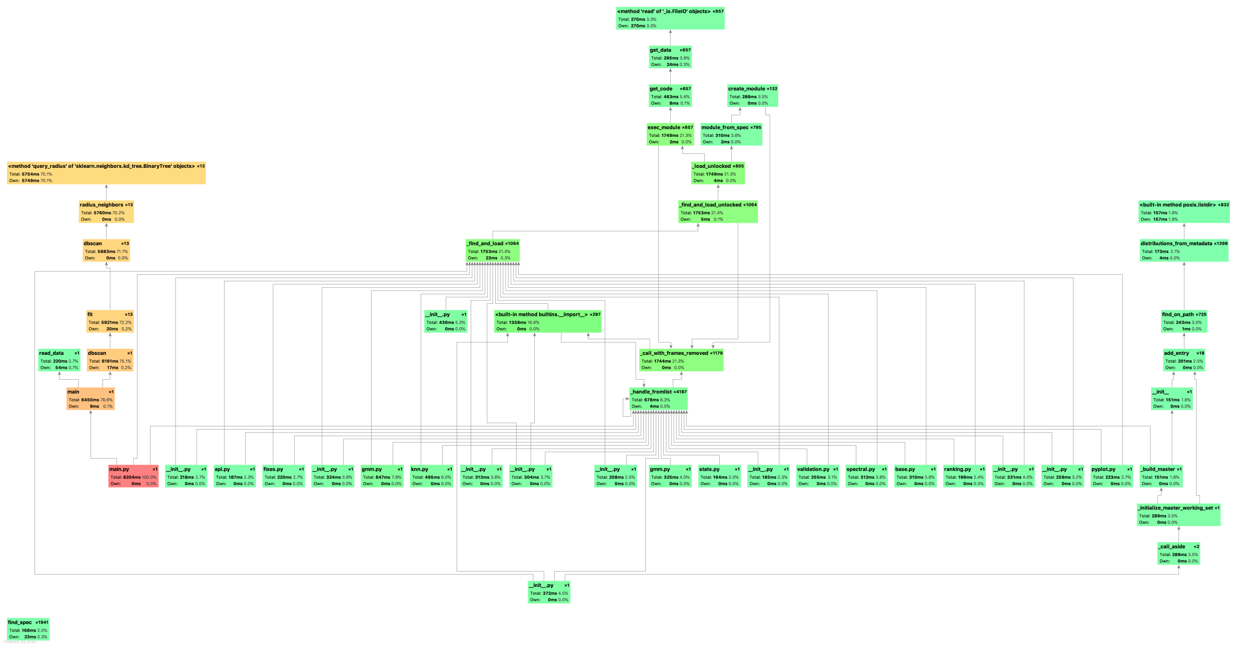
```
Line #    Mem usage    Increment   Line Contents
================================================
    10  140.027 MiB  140.027 MiB   @profile
    11                             def dbscan(df, random_vip, knns):
    12  140.027 MiB    0.000 MiB       silhouette_avgs = []
    13  140.027 MiB    0.000 MiB       hits = []
    14  140.027 MiB    0.000 MiB       x_min = 10
    15  140.027 MiB    0.000 MiB       x_max = 140
    16  163.449 MiB  -20.383 MiB       for eps in numpy.arange(x_min, x_max, 10):
    17  163.449 MiB  -17.156 MiB           logging.info("DBSCAN: eps = {}".format(eps))
    18  165.840 MiB   -1.168 MiB           X = StandardScaler().fit_transform(df.T)
    19  165.863 MiB  -24.191 MiB           db = DBSCAN(eps=eps).fit(X)
    20  165.863 MiB  -35.484 MiB           core_samples_mask = numpy.zeros_like(db.labels_, dtype=bool)
    21  165.863 MiB  -35.480 MiB           core_samples_mask[db.core_sample_indices_] = True
    22  165.863 MiB  -35.484 MiB           cluster_labels = db.labels_
    23  165.863 MiB  -35.480 MiB           n_clusters = len(set(cluster_labels)) - (
    24  165.863 MiB  -35.484 MiB               1 if -1 in cluster_labels else 0)
    25  163.449 MiB  -40.984 MiB           silhouette_avg = silhouette_score(X, db.labels_)
    26  163.449 MiB  -20.383 MiB           silhouette_avgs.append(silhouette_avg)
    27  163.449 MiB  -20.383 MiB           logging.info("For n_clusters =", n_clusters,
    28  163.449 MiB  -20.383 MiB                        "The average silhouette_score in DBSCAN is :",
    29  163.449 MiB  -20.383 MiB                        silhouette_avg)
    30
    31  163.449 MiB  -20.383 MiB           hit = 0
    32  163.449 MiB  -20.383 MiB           no = cluster_labels[df.columns.get_loc(random_vip)]
    33  163.449 MiB  -74.992 MiB           for neighbor in knns:
    34  163.449 MiB -101.914 MiB               if cluster_labels[df.columns.get_loc(neighbor)] == no:
    35  162.098 MiB -101.914 MiB                   hit += 1
    36                                         else:
    37  163.449 MiB    0.000 MiB                   logging.debug(
    38  163.449 MiB    0.000 MiB                       "DBSCAN: vipno: {} is not in the same cluster.".format(
    39  163.449 MiB    0.000 MiB                           neighbor))
    40  163.449 MiB  -20.383 MiB           logging.info(
    41  163.449 MiB  -20.383 MiB               "For k = {} in kNN, there has {} in the same cluster in DBSCAN.".format(
    42  163.449 MiB  -20.383 MiB                   len(knns), hit))
    43  163.449 MiB  -20.383 MiB           hits.append(hit)
    44
    45                                 # plot_silhouette(x_min, x_max, silhouette_avgs)
    46
    47  160.223 MiB   -3.227 MiB       return silhouette_avgs.index(max(silhouette_avgs)) / 10, hits
```

Screenshot

```
/usr/local/bin/python3.6 /Users/Yang/Developer/420235DataMining/hw1/main.py
INFO:root:DataFrame shape: (2635, 298)
<class 'pandas.core.frame.DataFrame'>
Index: 2635 entries, 10000004 to 40000700
Columns: 298 entries, 13205496418 to 6222021615015662822
dtypes: float64(298)
memory usage: 6.0+ MB
DEBUG:root:DataFrame info: None
INFO:root:random vipno: 1590142206747
INFO:root:vipno in ranked order using kNN(k = 5):
INFO:root:1595150722760
INFO:root:1595151110818
INFO:root:1593140148286
INFO:root:1590142516563
INFO:root:1594140467704
DEBUG:root:DBSCAN: eps = 10
For n_clusters = 1 The average silhouette_score in DBSCAN is : -0.31939086805466427
For k = 5 in kNN, there has 3 in the same cluster in DBSCAN.
INFO:root:DBSCAN: vipno: 1595151110818 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1593140148286 is not in the same cluster.
DEBUG:root:DBSCAN: eps = 20
INFO:root:DBSCAN: vipno: 1595150722760 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1595151110818 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1593140148286 is not in the same cluster.
DEBUG:root:DBSCAN: eps = 30
For n_clusters = 1 The average silhouette_score in DBSCAN is : -0.1942397862954418
For k = 5 in kNN, there has 2 in the same cluster in DBSCAN.
```

```
INFO:root:DBSCAN: vipno: 1595150722760 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1595151110818 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1593140148286 is not in the same cluster.
For n_clusters = 1 The average silhouette_score in DBSCAN is : -0.029129862530624537
INFO:root:DBSCAN: vipno: 1594140467704 is not in the same cluster.
For k = 5 in kNN, there has 1 in the same cluster in DBSCAN.
DEBUG:root:DBSCAN: eps = 40
For n_clusters = 1 The average silhouette_score in DBSCAN is : 0.08708130699171691
For k = 5 in kNN, there has 1 in the same cluster in DBSCAN.
INFO:root:DBSCAN: vipno: 1595150722760 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1595151110818 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1593140148286 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1594140467704 is not in the same cluster.
DEBUG:root:DBSCAN: eps = 50
INFO:root:DBSCAN: vipno: 1595150722760 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1595151110818 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1593140148286 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1590142516563 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1594140467704 is not in the same cluster.
DEBUG:root:DBSCAN: eps = 60
For n_clusters = 1 The average silhouette_score in DBSCAN is : 0.20268451888941394
For k = 5 in kNN, there has 0 in the same cluster in DBSCAN.
INFO:root:DBSCAN: vipno: 1595150722760 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1595151110818 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1593140148286 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1590142516563 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1594140467704 is not in the same cluster.
DEBUG:root:DBSCAN: eps = 70
For n_clusters = 1 The average silhouette_score in DBSCAN is : 0.2775394743004492
For k = 5 in kNN, there has 0 in the same cluster in DBSCAN.
INFO:root:DBSCAN: vipno: 1595150722760 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1595151110818 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1593140148286 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1590142516563 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1594140467704 is not in the same cluster.
DEBUG:root:DBSCAN: eps = 80
For n_clusters = 1 The average silhouette_score in DBSCAN is : 0.3387778176593896
For k = 5 in kNN, there has 0 in the same cluster in DBSCAN.
INFO:root:DBSCAN: vipno: 1595150722760 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1595151110818 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1593140148286 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1590142516563 is not in the same cluster.
INFO:root:DBSCAN: vipno: 1594140467704 is not in the same cluster.
DEBUG:root:DBSCAN: eps = 90
For n_clusters = 1 The average silhouette_score in DBSCAN is : 0.4067160039619981
For k = 5 in kNN, there has 0 in the same cluster in DBSCAN.
DEBUG:root:DBSCAN: eps = 100
For n_clusters = 1 The average silhouette_score in DBSCAN is : 0.43392130605710816
For k = 5 in kNN, there has 5 in the same cluster in DBSCAN.
For n_clusters = 1 The average silhouette_score in DBSCAN is : 0.4576647790706404
For k = 5 in kNN, there has 5 in the same cluster in DBSCAN.
DEBUG:root:DBSCAN: eps = 110
For n_clusters = 1 The average silhouette_score in DBSCAN is : 0.4883338136442235
For k = 5 in kNN, there has 5 in the same cluster in DBSCAN.
DEBUG:root:DBSCAN: eps = 120
For n_clusters = 1 The average silhouette_score in DBSCAN is : 0.5001686210200447
For k = 5 in kNN, there has 5 in the same cluster in DBSCAN.
DEBUG:root:DBSCAN: eps = 130
For n_clusters = 1 The average silhouette_score in DBSCAN is : 0.5243956411873608
For k = 5 in kNN, there has 5 in the same cluster in DBSCAN.

Process finished with exit code 0
```



<method 'read' of '_io.FileIO' objects> ×657 Total: 270ms 3.3% Own: 270ms 3.3%
get_data ×657 Total: 295ms 3.6% Own: 24ms 0.3%
get_code ×657 Total: 463ms 5.6% Own: 8ms 0.1%
create_module ×132 Total: 288ms 3.5% Own: 0ms 0.0%
exec_module ×657 Total: 1749ms 21.3% Own: 2ms 0.0%
module_from_spec ×795 Total: 310ms 3.8% Own: 2ms 0.0%
_load_unlocked ×805 Total: 1749ms 21.3% Own: 4ms 0.0%
<method 'query_radius' of 'sklearn.neighbors.kd_tree.BinaryTree' objects> ×13 Total: 5754ms 70.1% Own: 5749ms 70.1%
radius_neighbors ×13 Total: 5760ms 70.2% Own: 0ms 0.0%
_find_and_load_unlocked ×1064 Total: 1753ms 21.4% Own: 5ms 0.1%
<built-in method posix.listdir> ×832 Total: 157ms 1.9% Own: 157ms 1.9%
dbscan ×13 Total: 5883ms 71.7% Own: 0ms 0.0%
_find_and_load ×1064 Total: 1753ms 21.4% Own: 23ms 0.3%
distributions_from_metadata ×1398 Total: 173ms 2.1% Own: 4ms 0.0%
fit ×13 Total: 5921ms 72.2% Own: 20ms 0.2%
__init__.py ×1 Total: 436ms 5.3% Own: 0ms 0.0%
<built-in method builtins.__import__> ×297 Total: 1358ms 16.6% Own: 0ms 0.0%
find_on_path ×720 Total: 243ms 3.0% Own: 1ms 0.0%
read_data ×1 Total: 220ms 2.7% Own: 54ms 0.7%
dbscan ×1 Total: 6161ms 75.1% Own: 17ms 0.2%
_call_with_frames_removed ×1179 Total: 1744ms 21.3% Own: 0ms 0.0%
add_entry ×18 Total: 201ms 2.5% Own: 0ms 0.0%
main ×1 Total: 6450ms 78.6% Own: 9ms 0.1%
_handle_fromlist ×4187 Total: 678ms 8.3% Own: 4ms 0.0%
__init__ ×1 Total: 151ms 1.8% Own: 0ms 0.0%
main.py ×1 Total: 8204ms 100.0% Own: 0ms 0.0%
__init__.py ×1 Total: 218ms 2.7% Own: 0ms 0.0%
api.py ×1 Total: 187ms 2.3% Own: 0ms 0.0%
fixes.py ×1 Total: 220ms 2.7% Own: 0ms 0.0%
__init__.py ×1 Total: 324ms 3.9% Own: 0ms 0.0%
gmm.py ×1 Total: 647ms 7.9% Own: 0ms 0.0%
knn.py ×1 Total: 495ms 6.0% Own: 0ms 0.0%
__init__.py ×1 Total: 313ms 3.8% Own: 0ms 0.0%
__init__.py ×1 Total: 304ms 3.7% Own: 0ms 0.0%
__init__.py ×1 Total: 208ms 2.5% Own: 0ms 0.0%
gmm.py ×1 Total: 325ms 4.0% Own: 0ms 0.0%
stats.py ×1 Total: 164ms 2.0% Own: 0ms 0.0%
__init__.py ×1 Total: 185ms 2.3% Own: 0ms 0.0%
validation.py ×1 Total: 255ms 3.1% Own: 0ms 0.0%
spectral.py ×1 Total: 312ms 3.8% Own: 0ms 0.0%
base.py ×1 Total: 310ms 3.8% Own: 0ms 0.0%
ranking.py ×1 Total: 196ms 2.4% Own: 0ms 0.0%
__init__.py ×1 Total: 331ms 4.0% Own: 0ms 0.0%
__init__.py ×1 Total: 259ms 3.2% Own: 0ms 0.0%
pyplot.py ×1 Total: 223ms 2.7% Own: 0ms 0.0%
_build_master ×1 Total: 151ms 1.8% Own: 0ms 0.0%
_initialize_master_working_set ×1 Total: 289ms 3.5% Own: 0ms 0.0%
_call_aside ×2 Total: 289ms 3.5% Own: 0ms 0.0%
__init__.py ×1 Total: 372ms 4.5% Own: 0ms 0.0%
find_spec ×1941 Total: 168ms 2.0% Own: 23ms 0.3%
Powered by yFiles