An Exploration of the DBSCAN Algorithm

Matthew Morgan, Jace Ritchie

Introduction

It is sometimes difficult to know how many clusters are necessary to adequately describe a dataset

Many popular algorithms require the number of clusters a priori

We propose using the DBSCAN algorithm as a way to cluster data without knowing the most desirable number of groups beforehand

Methodology: DBSCAN (Ester et al., 1996)

The DBSCAN algorithm requires two parameters:

- ε is the minimum distance between points in a cluster
- *MinPts* is the minimum number of points within ε of each other to make a group

There is a possibility of creating no groups, or having unclustered (noise) points

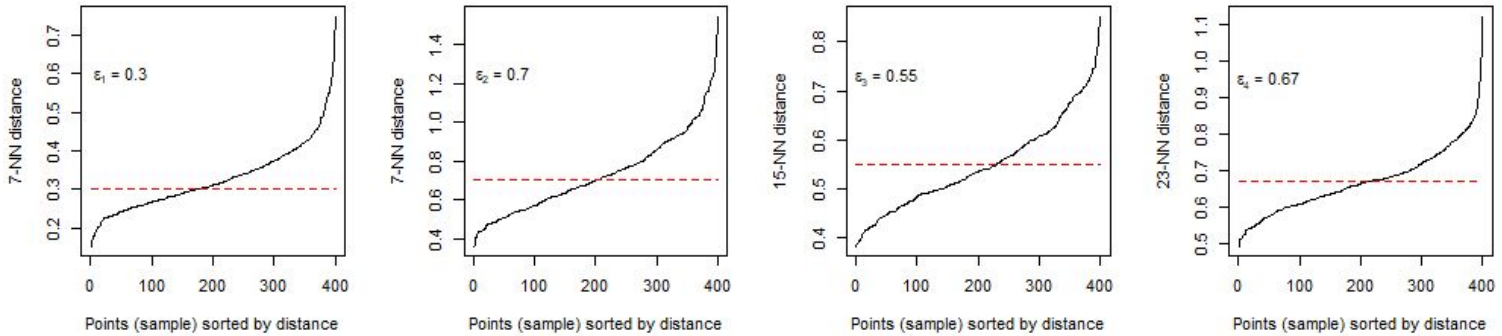
DBSCAN algorithm:

1. Start at a random, unclustered point that hasn't been selected yet
2. If there are *MinPts* points within ε of the point, make a cluster and keep adding points that are within ε of any point in the cluster
3. Repeat 1-2 until all points have been considered

Choosing ε and *MinPts*

Sander et al. (1998) recommend setting *MinPts* to twice the number of variables in the dataset

Schubert et al. (2017) suggest setting ε to the inflection point on a *MinPts* - 1 nearest neighbors distance plot



Methodology: *k*-means (Rencher and Christensen 2012)

The k-means algorithm requires:

- The number of clusters *k* **or** a set of initial (distinct) cluster centroids
- If only *k* is specified, a random set of cluster centroids are used initially

k-means algorithm:

1. Partition the objects into k clusters
2. Assign each object to closest cluster centroid (using Euclidean distance)
    a. Recalculate cluster centroids
3. Repeat Step 2 until no more reassignments occur

Simulation Study: Setup

10,000 iterations

4 “true” underlying groups

Data generated from 4 multivariate normal distributions

100 data points

Shared covariance matrix $\underset{p_s \times p_s}{\Sigma} = \begin{bmatrix} 5 & .5 & \dots & .5 \\ .5 & 5 & \dots & .5 \\ \vdots & \vdots & & \vdots \\ .5 & .5 & \dots & 5 \end{bmatrix}$

Distinct $p_s \times 1$ mean vectors $\boldsymbol{\mu}_{1s}, \boldsymbol{\mu}_{2s}, \boldsymbol{\mu}_{3s}$, and $\boldsymbol{\mu}_{4s}$ for $s \in 1, \dots, 4$

Data was standardized after it was generated

Simulation Study: Setup (Mean Vectors)

The first set of mean vectors used to generate the data were of length $p_1 = 4$ with:

- $\boldsymbol{\mu}_{11}$ = (5, 6, 7, 8), a baseline mean vector
- $\boldsymbol{\mu}_{21}$ = (−5, −6, −7, −8), a negatively collinear mean vector
- $\boldsymbol{\mu}_{31}$ = (15, 18, 21, 24), a positively collinear mean vector
- $\boldsymbol{\mu}_{41}$ = (0, 0, 0, 0), a non-collinear mean vector

Simulation Study: Setup (Mean Vectors)

The second set of mean vectors used to generate the data were of length $p_2 = 4$ with:

- $\boldsymbol{\mu}_{21}$ = (5, 6, 7, 8), a baseline mean vector
- $\boldsymbol{\mu}_{22}$ = (3, 4, 5, 6), an overlapping mean vector
- $\boldsymbol{\mu}_{23}$ = (7, 8, 9, 10), an overlapping mean vector
- $\boldsymbol{\mu}_{24}$ = (0, 0, 0, 0), a non-overlapping mean vector

Simulation Study: Setup (Mean Vectors)

The third set of mean vectors used to generate the data were of length $p_3 = 8$ with:

- $\boldsymbol{\mu}_{31}$ = (5, 6, 7, 8, 9, 10, 11, 12), a baseline mean vector
- $\boldsymbol{\mu}_{32}$ = (−5, −6, −7, −8, -9, -10, -11, -12), a negatively collinear mean vector
- $\boldsymbol{\mu}_{33}$ = (15, 18, 21, 24, 27, 30, 33, 36), a positively collinear mean vector
- $\boldsymbol{\mu}_{34}$ = (0, 0, 0, 0, 0, 0, 0, 0), a non-collinear mean vector

Simulation Study: Setup (Mean Vectors)

The fourth set of mean vectors used to generate the data were of length $p_4 = 12$ with:

- $\boldsymbol{\mu}_{41}$ = (5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16), a baseline mean vector
- $\boldsymbol{\mu}_{42}$ = (−5, −6, −7, −8, -9, -10, -11, -12, -13, -14, -15, 16), a negatively collinear mean vector
- $\boldsymbol{\mu}_{43}$ = (15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48), a positively collinear mean vector
- $\boldsymbol{\mu}_{44}$ = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), a non-collinear mean vector

Simulation Study: Setup

For each simulation study setting *s*, values of *MinPts* and ε were chosen for the DBSCAN algorithm

The DBSCAN algorithm was implemented on the generated and standardized dataset

Checked whether the DBSCAN algorithm found 4 clusters

If the DBSCAN algorithm had found 4 clusters, the percent of the data points left unclustered was recorded

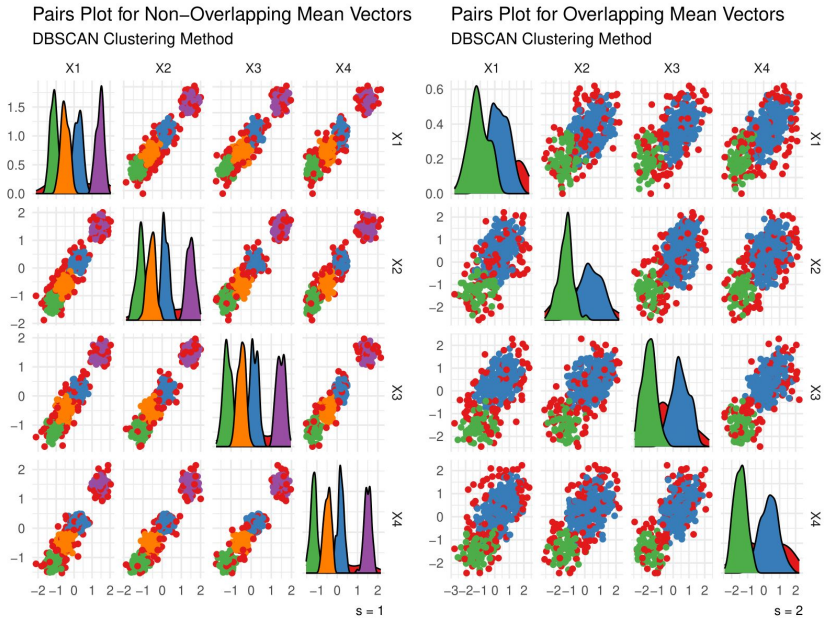
Then, on the set of data points that the DBSCAN algorithm had clustered, *k*-means was implemented to cluster that same set of data points into 4 clusters

Finally, the accuracy of group identification, as measured by the proportion of data points assigned to the correct group was recorded

Simulation Study: Results

s	Found Groups	Percent Noise	Accuracy DBSCAN	Accuracy *k*-means
1	0.9113	0.2067	0.8991	0.7228
2	0.0037	0.0008	0.0012	0.0017
3	1	0.0523	1	0.8802
4	1	0.0225	1	0.7892

Simulation Study: Results



Application: Dataset

A dataset from the PGA 2014 season was used to form clusters

Useful because it has natural groups of players and courses - each player played courses 4 times

SG stands for strokes gained, a numerical score for overall performance for a certain activity

CP is the players score on a standardized common par of 75

The variables used were:

- SG while putting
- SG from tee to green
- SG on approach
- SG around the green
- CP
- Course slope
- Each player's average driving distances
- The number of hits in regulation onto the green

Application: Results

Many of the clusters were composed of one or two players across many courses

Most of the 4 observations of each course per player were in one cluster

Cluster 4 seems to be the “average player on an average course” cluster

Cluster	Noise	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Observations	182	49	40	44	343	43	34	26	16	40	35	24	26	45	47	48	44
Players	19	1	1	1	9	1	1	1	1	1	1	1	2	1	1	2	1
Courses	18	13	11	13	18	11	11	9	5	13	12	7	9	12	14	13	12

Discussion

DBSCAN is very effective for identifying correct groups given distinct underlying mean vectors and scales well to high dimensional data

Both algorithms underperformed on data with overlapping mean vectors

DBSCAN seemed to find natural groupings among the noise in the real world data set

Next steps include testing DBSCAN on data generated with different settings than used here and using DBSCAN clusters as starting centroids for *k*-means when we desire to cluster all points

Questions?

References

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Rencher, A. C. and Christensen, W. F. (2012). *Cluster Analysis*, page 532–539. Wiley.

Sander, J., Ester, M., Kriegel, H.-P., and Xu, X. (1998). Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery*, 2:169–194.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017). Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Trans. Database Syst.*, 42(3).