GENERAL ASSEMBLY

DBSCAN

Joseph Nelson, Data Science Immersive

AGENDA

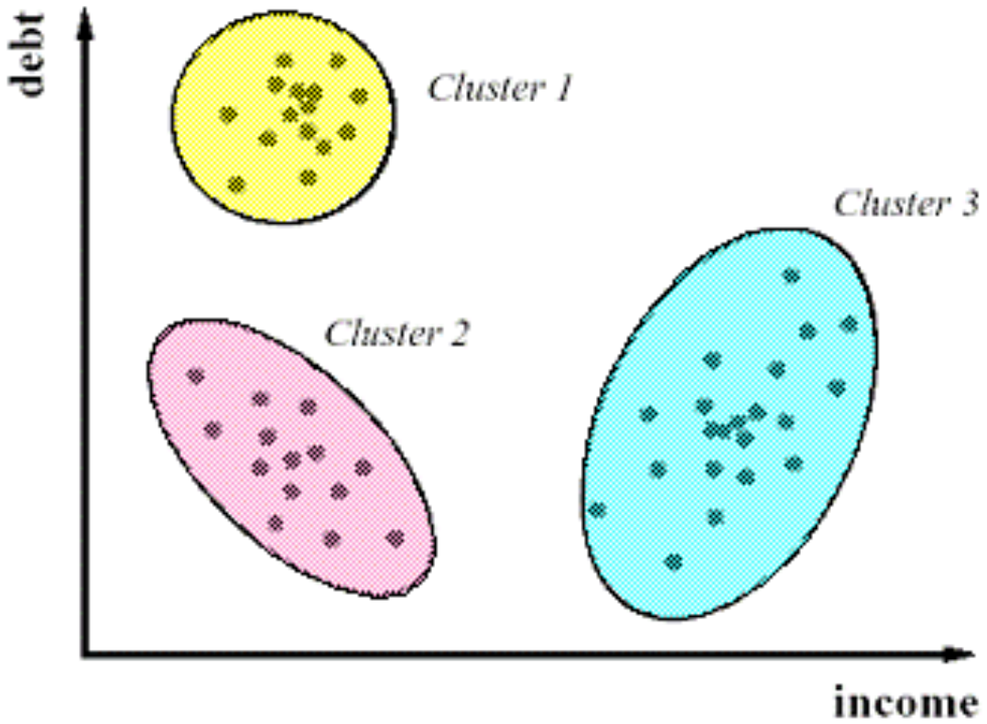
- What is DBSCAN?
- How does DBSCAN work?
- How does DBSCAN compare to K-Means and Hierarchical Clustering?
- Implementation
- Code Example

WHAT IS DBSCAN?

- Review: what is clustering?

WHAT IS DBSCAN?

- Review: what is clustering?
- Clustering is an unsupervised learning technique we employ to group “similar” data points together
- With unsupervised learning, remember: there is no **clear** objective, there is no “right answer” (hard to tell how we’re doing), there is no response variable, just observations with features, and labeled data is not required



WHAT IS DBSCAN?

- DBSCAN: Density-based Spatial Clustering of Applications with Noise
- For DBSCAN, clusters of **high density** are separated by clusters of **low density**

WHAT IS DBSCAN?

- DBSCAN: Density-based Spatial Clustering of Applications with Noise
- For DBSCAN, clusters of **high density** are separated by clusters of **low density**
- DBSCAN is the most widely used and applicable clustering algorithm - given that it takes minimum predefined input and can discover clusters of any shape, not just the sphere-like clusters that k-means often computes. This way, we can discover less pre-defined patterns and glean some more useful insights.

HOW DOES DBSCAN WORK?

- DBSCAN is a **density based clustering algorithm**, meaning that the algorithm finds clusters by seeking areas of the dataset that have a higher density of points than the rest of the dataset.
- Given this, unlike in our previous examples, after you apply DBSCAN there may be data points that aren't assigned to any cluster at all!
- When we use DBSCAN, it requires two input parameters - **epsilon**, which is the minimum distance between two points for them to be considered a cluster, and the **minimum number of points** necessary to form a cluster, which we'll call the minimum points.

HOW DOES DBSCAN WORK?

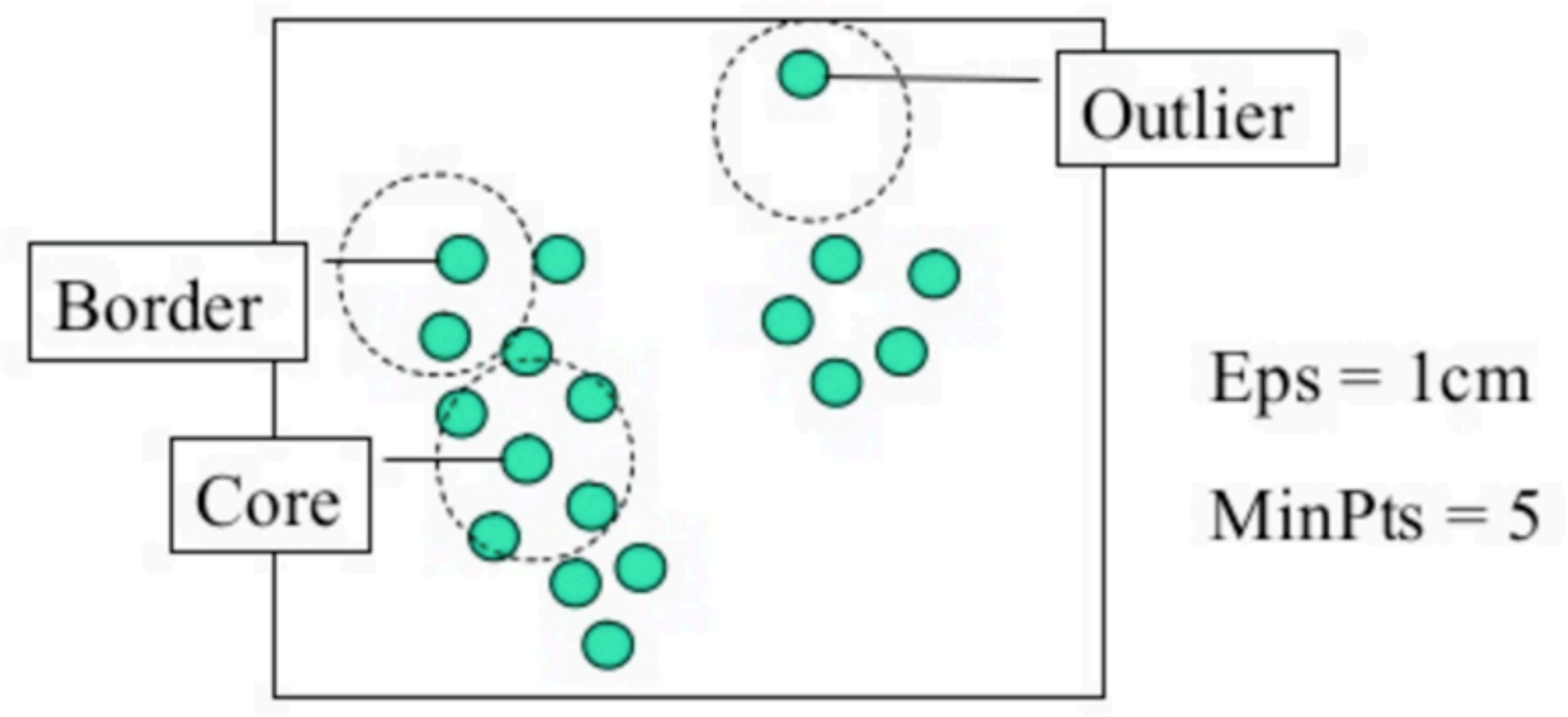
- DBSCAN Algorithm:
- 1. Choose an “epsilon” and “min_samples”
- 2. Pick an arbitrary point, and check if there are at least “min_samples” points within distance “epsilon”
- If yes, add those points to the cluster and check each of the new points
- If no, choose another arbitrary point to start a new cluster
- 3. Stop once all points have been checked

HOW DOES DBSCAN WORK?

- BRAVE VOLUNTEER: Whiteboard example

HOW DOES DBSCAN WORK?

- Visually:



HOW DOES DBSCAN WORK?

- DBSCAN algorithm in words/review of concept:
- VOLUNTEER EXPLANATION?

HOW DOES DBSCAN WORK?

- DBSCAN algorithm in words/review of concept:
- DBSCAN will take the epsilon and minimum points we provided it and cluster all of the points in a neighborhood, first passing the minimum points requirement and then clustering each of the points within epsilon distance to form the clusters. Once one cluster is formed, the algorithm then moves to a new datapoint, and seeks to find related points to form yet another cluster; this will continue until DBSCAN simply runs out of points!

HOW DOES DBSCAN WORK?

- Algorithm visualization:
- http://www.naftaliharris.com/blog/visualizing-dbscan-clustering/
- Let’s play with this a bit^^^

- Independently, select the “Pimpled Smiley” distribution of points. What is an optimal epsilon? What about minimum number of points?

HOW DOES DBSCAN WORK?

- Algorithm visualization:
- http://www.naftaliharris.com/blog/visualizing-dbscan-clustering/
- Let’s play with this a bit^^^

- Now, independently, select the “Pimpled Smiley” distribution of points. What is an optimal epsilon? What about minimum number of points? (5-10min)

HOW DOES DBSCAN COMPARE TO K-MEANS AND HIERARCHICAL?

- Class?

HOW DOES DBSCAN COMPARE TO K-MEANS AND HIERARCHICAL?

- Whereas k-means can be thought of as a "general" clustering approach, DBSCAN performs especially well with unevenly distributed, non-linear clusters.
- The fundamental difference with DBSCAN lies in the fact that it is **density based** rather than k-means, which calculates clusters based on distance from a **central point**, or hierarchical clustering. When choosing epsilon in the minimum points in DBSCAN, a selection of < 2 will result in a linkage cluster - essentially the same result as if you were to perform a hierarchical clustering. To diversify the DBSCAN, we therefore must give it a significant amount of points to form a cluster.
- DBSCAN is density based, which means that it determines clusters based on the number of points in a certain area
- By choosing too few points for DBSCAN, i.e. less than two, we'll effectively get a straight line if we connect the points, just like linkage clustering.

HOW DOES DBSCAN COMPARE TO K-MEANS AND HIERARCHICAL?

- DBSCAN can be useful to us when we have a lot of dense data. If we used k-means on this data, the algorithm would effectively give us just one large cluster! However with DBSCAN, we can actually break down this cluster into smaller groups to see their attributes.
- ADVANTAGES – Class?
- DISADVANTAGES – Class?

HOW DOES DBSCAN COMPARE TO K-MEANS AND HIERARCHICAL?

- DBSCAN can be useful to us when we have a lot of dense data. If we used k-means on this data, the algorithm would effectively give us just one large cluster! However with DBSCAN, we can actually break down this cluster into smaller groups to see their attributes.
- ADVANTAGES –
- Clusters can be any size or shape
- No need to choose number of clusters
- DISADVANTAGES –
- More parameters to tune
- Doesn’t work with clusters of varying density
- **NOTE:** Not every point is assigned to a cluster!

HOW DO WE IMPLEMENT DBSCAN?

- To implement DBSCAN in Python, we first import it from sklearn:
- `from sklearn.cluster import DBSCAN`
- Next, assuming that we are using the classic Iris dataset, we define X as the data and y are the class variables
- `X, y = iris.data, iris.target`
- Next, we call DBSCAN from sklearn:
- `db = DBSCAN(eps=0.3, min_samples=10).fit(X)`
- Given the above input, what have we said about our clusters?

HOW DO WE IMPLEMENT DBSCAN?

- To implement DBSCAN in Python, we first import it from sklearn:
- `from sklearn.cluster import DBSCAN`
- Next, assuming that we are using the classic Iris dataset, we define X as the data and y are the class variables
- `X, y = iris.data, iris.target`
- Next, we call DBSCAN from sklearn:
- `db = DBSCAN(eps=0.3, min_samples=10).fit(X)`
- Given the above input, what have we said about our clusters?
- Here, we've set epsilon to a standard value of .3 and set the minimum number of points at 10, and then fit the model to our data X.

HOW DO WE IMPLEMENT DBSCAN?

- As a general rule when choosing the minimum points - you should always aim to have the minimum number of points be greater or equal to the amount of dimensions in your data, plus one. This typically will give the algorithm a good estimation of how to evaluate the clusters. Calculating epsilon is a bit trickier and uses a method called the k-distance, which can help visualize the best epsilon.

HOW DO WE IMPLEMENT DBSCAN?

- `core_samples = db.core_sample_indices_`
- `labels = db.labels_`
- The DBSCAN algorithm in Python returns two items - **the core samples** and the **labels**. The core samples are the points which the algorithm initially finds and searches around the neighborhood to form the cluster, and the labels are simply the cluster labels.
- Comprehensive question: how labels should we expect to receive?

HOW DO WE IMPLEMENT DBSCAN?

- Comprehensive question: how labels should we expect to receive?

Attributes:

core_sample_indices_ : array, shape = [n_core_samples]

Indices of core samples.

components_ : array, shape = [n_core_samples, n_features]

Copy of each core sample found by training.

labels_ : array, shape = [n_samples]

Cluster labels for each point in the dataset given to fit(). Noisy samples are given the label -1.

- From the documentation
  http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

CODING IMPLEMENTATION

- To the repo…