GENERAL ASSEMBLY

DBSCAN

Christoph Rahmede, Data Science Immersive

AGENDA

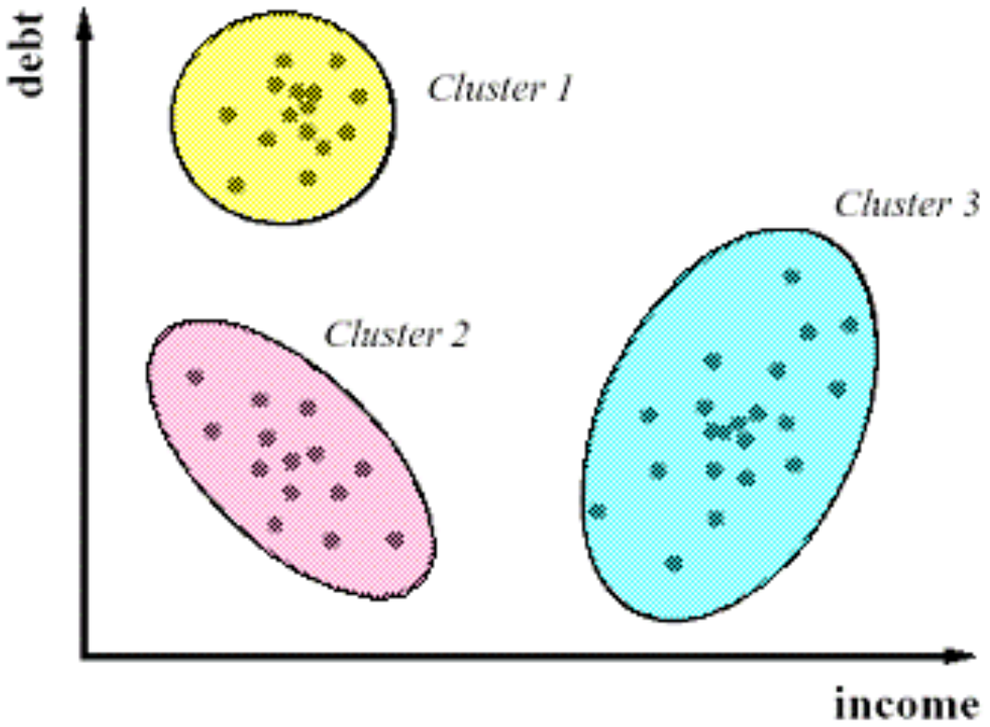
- What is DBSCAN?
- How does DBSCAN work?
- How does DBSCAN compare to K-Means and Hierarchical Clustering?
- Implementation
- Code Example

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 one of the most widely used and applicable clustering algorithms - given that it takes minimum predefined input and can discover clusters of any shape, not just the sphere-like (convex) clusters that k-means often computes. In 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 maximal 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 min_samples or MinPts.

HOW DOES DBSCAN WORK?

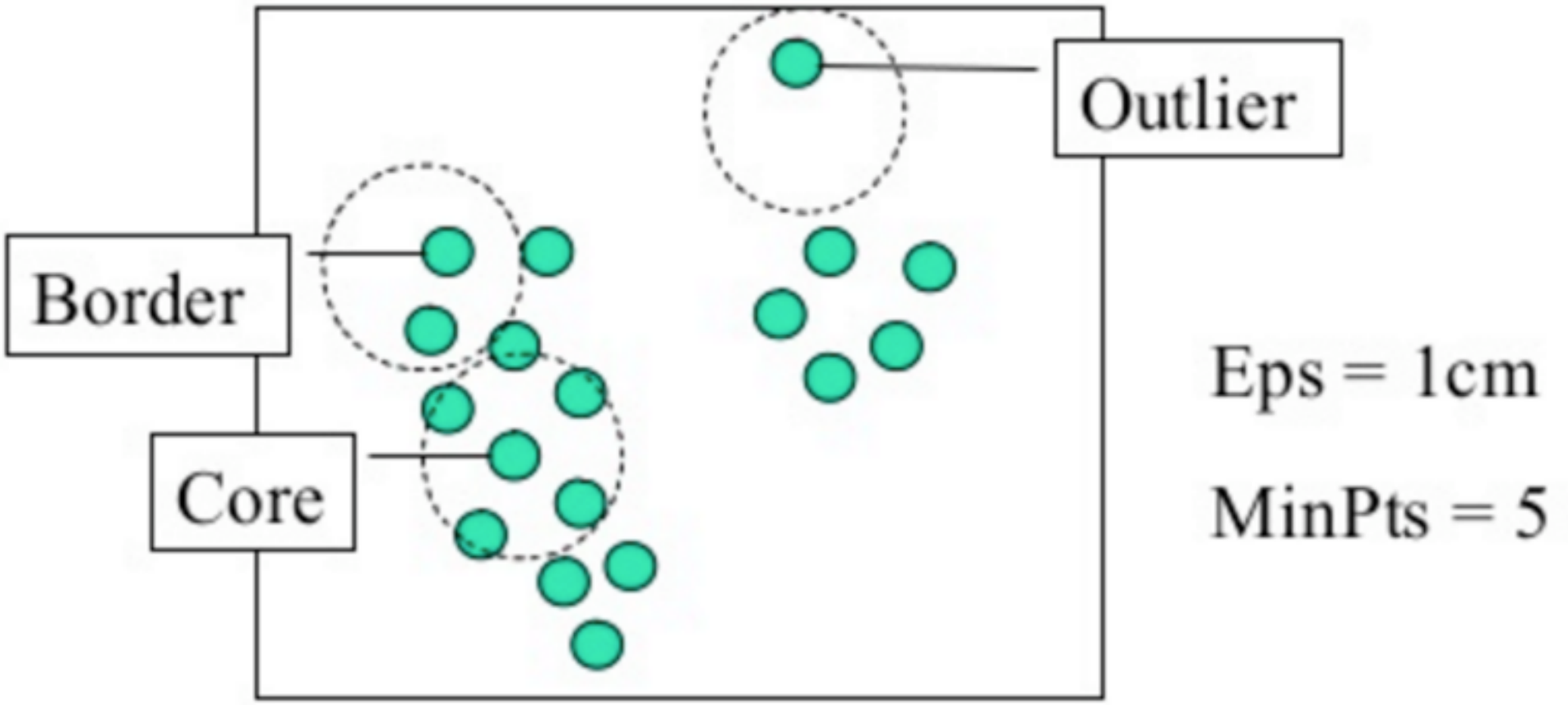
- DBSCAN Algorithm:
- 1. Choose an “epsilon” and “min_samples”
- 2. Pick an arbitrary point, and check if there are at least “min_samples” points within distance “epsilon”
- If yes, add those points to the cluster and check each of the new points
- If not, choose another arbitrary point to start a new cluster
- 3. Stop once all points have been checked

CHARACTERISATION OF POINTS

- A point is a **core point** if at least min_samples points are within distance epsilon (including itself)
- A point is a **border point** if it is within distance epsilon of a core point but has less than min_samples points within distance epsilon
- A **noise point** is a point that is neither a core point nor a border point, i.e. it does not have min_samples points within distance epsilon nor is it within distance epsilon of a core point

HOW DOES DBSCAN WORK?

- Visually:



HOW DOES DBSCAN WORK?

- Algorithm visualisation:
- 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 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 min_samples we provided it and cluster all of the points in a neighbourhood, first passing the min_samples requirement and then clustering each of the points within epsilon distance to form the clusters. Once one cluster is formed, the algorithm 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 COMPARE TO K-MEANS AND HIERARCHICAL CLUSTERING?

- K-means forms clusters of convex shape whereas DBSCAN performs especially well with unevenly distributed, non-linearly separable clusters
- DBSCAN is **density based** and assigns clusters based on the number of points in a given area, whereas k-means assigns clusters based on distance from a **central point**, and hierarchical clustering uses a similarity measure
- K-means is sensitive to outliers whereas DBSCAN is more robust
- K-means assigns evenly sized clusters whereas DBSCAN and hierarchical clustering can have clusters of varying size
- K-means and hierarchical clustering assign a cluster label to each data point whereas DBSCAN might not assign any label to some of the points

DBSCAN - ADVANTAGES AND DISADVANTAGES

- ADVANTAGES – Class?
- DISADVANTAGES – Class?

DBSCAN - ADVANTAGES AND DISADVANTAGES

- ADVANTAGES –
- No need to fix the number of clusters
- Clusters can be any size or shape
- Only two parameters to tune (epsilon and min_samples)
- Robust to outliers
- DISADVANTAGES –
- Not completely deterministic for border points
- Quality of clustering depends on distance measure
- Sensitive to parameter choice
- If the density varies strongly, it might not be possible to find a suitable epsilon for the whole dataset

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 as 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 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?

- `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 neighbourhood to form the cluster, and the labels are the cluster labels (the labels assigned may differ between different runs).
- Comprehensive question: how many labels should we expect to receive?

HOW DO WE IMPLEMENT DBSCAN?

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…