THINKFUL

# Density-Based Clustering

DATA SCIENCE

# Warm Up

Consider the following questions:

- What comes to mind when you think of “density” in a non-data context?
- In the context of density, what characteristics would we expect clusters to have? What characteristics would we expect the space between clusters to have?

# Agenda

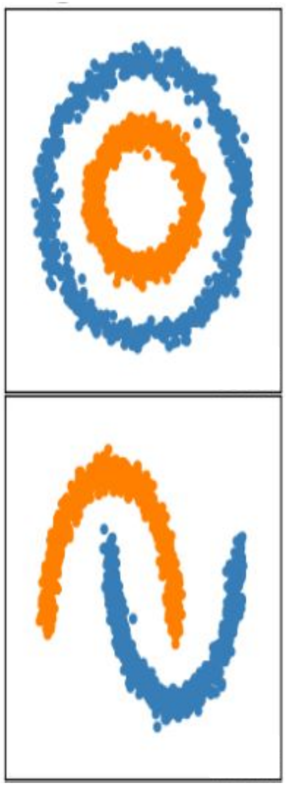
- Overview of density-based clustering
- Walkthrough of DBSCAN algorithm
- Advantages and disadvantages

# Overview

Density-based clustering is built on the premise that clusters consist of highly dense regions of observations separated by less dense/sparse regions

- Density in a scientific context = mass/volume; this definition can be extended to data-driven contexts in which mass = number of data points and volume = size of neighborhood around a given point
- Definition of a neighborhood is flexible and can be predicated upon arbitrary dissimilarity measures (i.e. all points with dissimilarity measure < x)
- Although the neighborhoods may be constrained in shape (e.g. a neighborhood built on Euclidean distance will be spherical), the shapes of the clusters themselves can be very flexible

# Overview



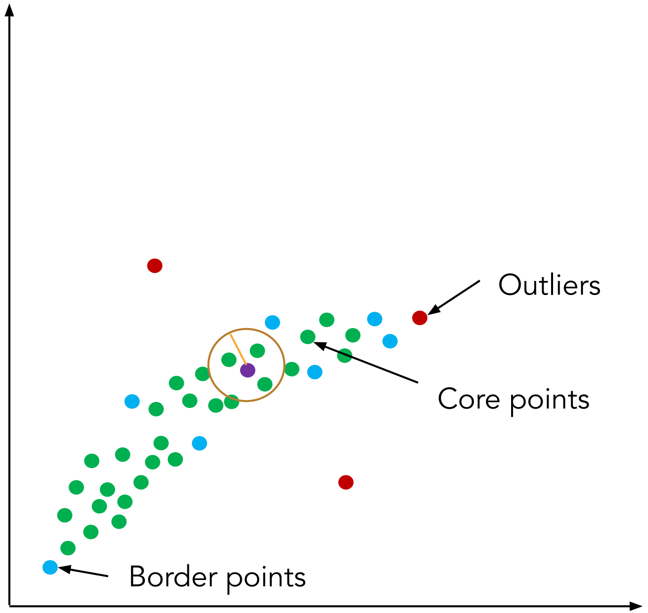
# Terminology

The most well-known density-based clustering technique is DBSCAN, which relies on the following parameters:

- $\varepsilon$: the radius of our neighborhoods around a point *p*
- *minPts*: the minimum number of points in a neighborhood required to form a cluster

In addition, it introduces the following key terminology:

- *Neighborhood*: all points in the feature space that are a distance of $\varepsilon$ or less from a given point *p*
- *Core point*: a point *b* with at least *minPts* other observations in its neighborhood
- *Border point*: a point *p* that has less than *minPts* other observations in its neighborhood, but is itself in the neighborhood of a core point
- *Outlier*: a point *o* that is neither a core point nor a border point




Purple point: the point of interest, *p*
Orange circle: the neighborhood of *p*
Radius of orange circle: $\varepsilon$
MinPts: 3

UP NEXT

# DBSCAN Algorithm

# DBSCAN Overview

The DBSCAN algorithm can be summarized as follows:

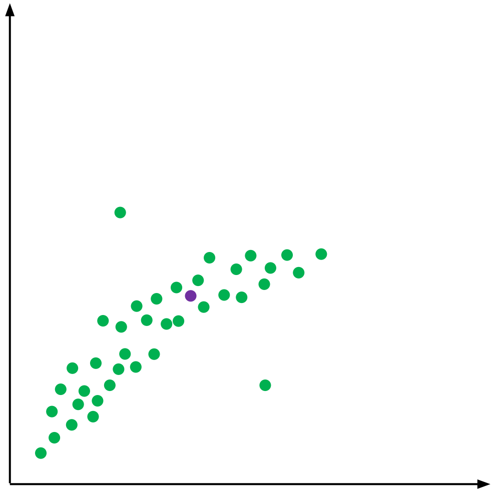
1. Pick an arbitrary point *k* that has not yet been classified as a core point, boundary point, or outlier
2. Determine how many other observations are in the neighborhood of k (with neighborhood determined by the choice of ε).
3. If this number is less than minPts, label it an outlier and go back to step 1.

# DBSCAN Overview

4. If this number is greater than minPts:
    a. Label k as a core point and start a new cluster
    b. Add all of the observations in the neighborhood of k to the cluster
    c. Visit all of the observations in the neighborhood of k
    d. If a neighboring observation is also a core point, its neighbors are added to the cluster and visited (this process can be repeated many times, jumping from core point to core point).
    e. If a neighboring observation is an outlier, it is relabeled as a boundary point and its neighbors are not visited.
5. Eventually, all of the points that are density-reachable from our initial point k will be added to the cluster; at this point, start over from step 1. Repeat until every point has been classified.

# DBSCAN Example

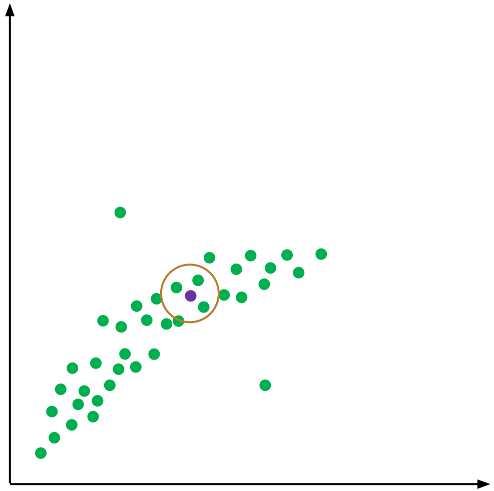
Pick a point that has not been classified



Purple point: the point of interest

# DBSCAN Example

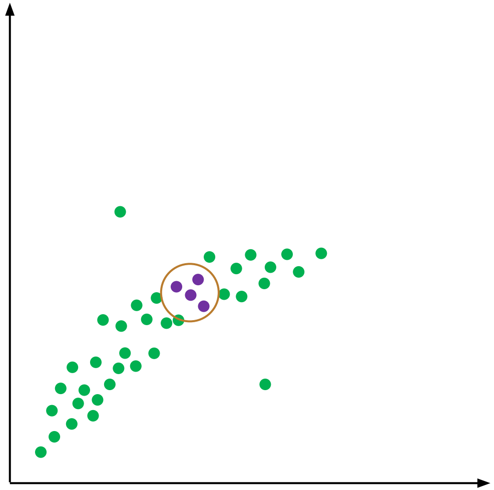
Examine the point's neighborhood



Purple point: the point of interest, *p*
Orange circle: the neighborhood of *p*
Radius of orange circle: $\varepsilon$
MinPts: 3

# DBSCAN Example

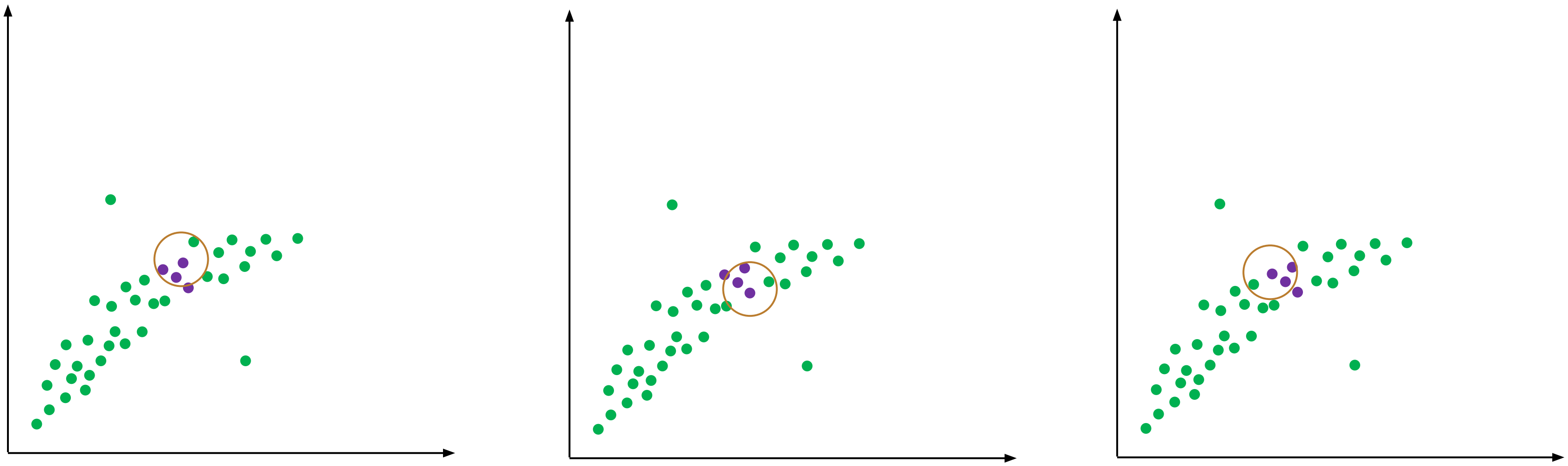
Since the number of observations in the neighborhood exceeds *minPts*, a new cluster is formed and the observations in the neighborhood are added to the cluster



Purple points: the members of the new cluster
Green points: not yet categorized

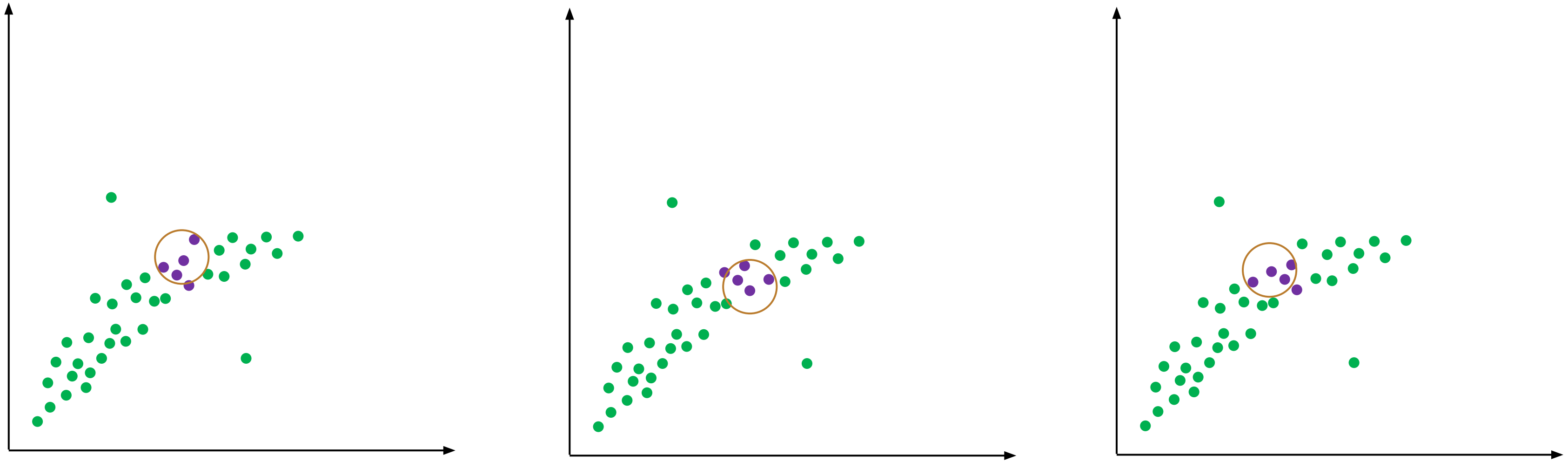
# DBSCAN Example

Examine the neighborhoods of the newly added neighbors to our initial point



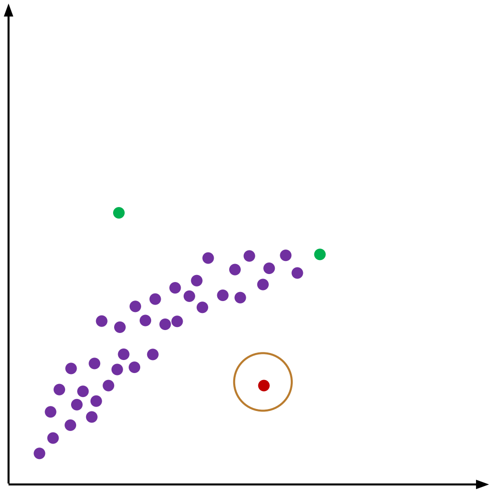
# DBSCAN Example

All of the neighbors have greater than *minPts* observations in their neighborhoods, so they are labeled core points; their neighboring observations are added to the cluster and have their neighborhoods visited



# DBSCAN Example

This process is repeated until no more core points are found; after the last round of boundary points are added, we have our final cluster. We then start over with a new point (in this case, an outlier)



# Advantages & Disadvantages

Advantages

- Doesn’t require any pre-specified number of clusters
- Like k-medoids and hierarchical clustering, works with arbitrary dissimilarity measures and categorical data
- Can identify clusters with highly flexible, non-standard shapes
- Relatively resistant to noise
- Can identify and label outliers
- Reasonably good time complexity: naive implementation would be $O(n^2)$, but well-optimized versions can achieve $O(n*\log(n))$ average case complexity for most practical use cases (i.e. data that is not extremely high-dimensional)

1/2

# Advantages & Disadvantages

Disadvantages

- Results can be very sensitive to parameter choices
- Choosing the right values of ε and minPts can be difficult
- Has difficulty when there are clusters of varying density

# Summary

- Density-based clustering defines clusters as regions of high density (number of points/area) separated by regions of low density
- The most popular technique for density-based clustering is DBSCAN, which has two parameters defining the size of the neighborhood and the minimum number of points needed to form a cluster
- DBSCAN can produce clusters with very flexible shapes and variable sizes, and it also scales significantly better than most other algorithms
- However, choosing optimal parameter values can be tricky

## EXERCISE

1. Jupyter notebook
2. Data set: Starbucks locations in the U.S.
3. Choose a reasonably sized subset of the locations in the U.S.
4. Build a DBSCAN model using an initially specified set of values for eps and minPts (we do not need to subset; details as to why are provided in the exercise)
5. Plot the results on a map
6. Tweak the values of eps and minPts to produce more meaningful clustering results and replot on map

THINKFUL

# Thank You


# Density-Based Clustering

# Warm Up

Consider the following questions:

- What comes to mind when you think of “density” in a non-data context?
- In the context of density, what characteristics would we expect clusters to have? What characteristics would we expect the space between clusters to have?

# High Level Agenda

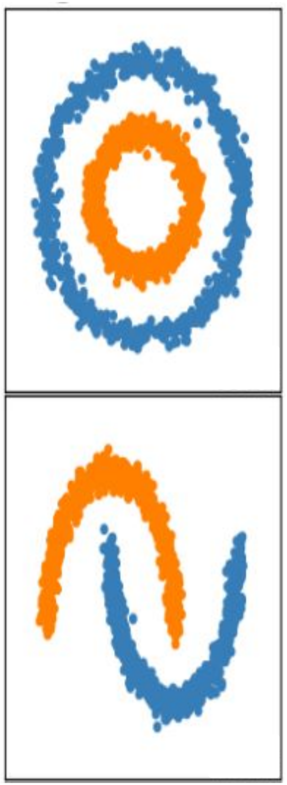
- Overview of density-based clustering
- Walkthrough of DBSCAN algorithm
- Advantages and disadvantages

# Overview

# Overview

Density-based clustering is built on the premise that clusters consist of highly dense regions of observations separated by less dense/sparse regions

- Density in a scientific context = mass/volume; this definition can be extended to data-driven contexts in which mass = number of data points and volume = size of neighborhood around a given point
- Definition of a neighborhood is flexible and can be predicated upon arbitrary dissimilarity measures (i.e. all points with dissimilarity measure < x)
- Although the neighborhoods may be constrained in shape (e.g. a neighborhood built on Euclidean distance will be spherical), the shapes of the clusters themselves can be very flexible



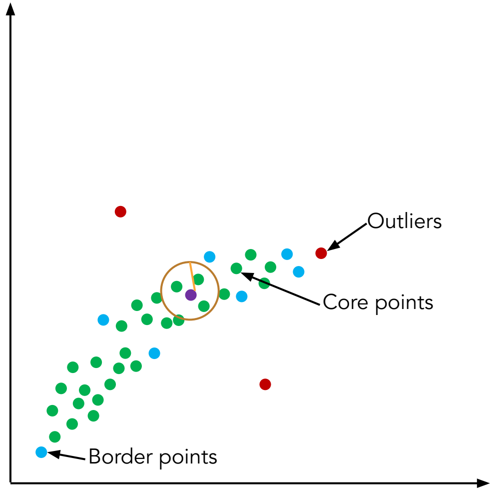
# Terminology

The most well-known density-based clustering technique is DBSCAN, which relies on the following parameters:

- $\varepsilon$: the radius of our neighborhoods around a point *p*
- *minPts*: the minimum number of points in a neighborhood required to form a cluster

In addition, it introduces the following key terminology:

- *Neighborhood*: all points in the feature space that are a distance of $\varepsilon$ or less from a given point *p*
- *Core point*: a point *b* with at least *minPts* other observations in its neighborhood
- *Border point*: a point *p* that has less than *minPts* other observations in its neighborhood, but is itself in the neighborhood of a core point
- *Outlier*: a point *o* that is neither a core point nor a border point




Purple point: the point of interest, *p*
Orange circle: the neighborhood of *p*
Radius of orange circle: $\varepsilon$
MinPts: 3

# DBSCAN Algorithm

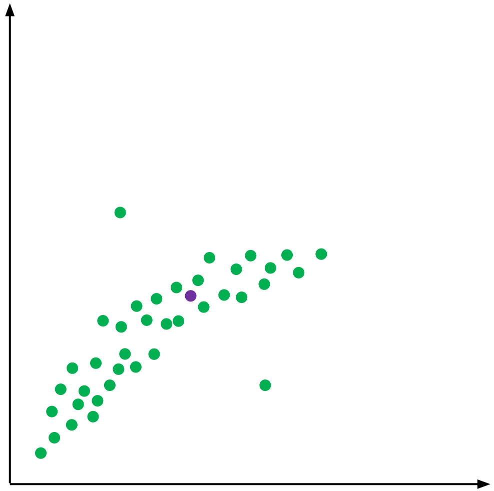
# DBSCAN Overview

The DBSCAN algorithm can be summarized as follows:

1. Pick an arbitrary point *k* that has not yet been classified as a core point, boundary point, or outlier
2. Determine how many other observations are in the neighborhood of *k* (with neighborhood determined by the choice of $\varepsilon$).
3. If this number is less than *minPts*, label it an outlier and go back to step 1.
4. If this number is greater than *minPts*:
   - Label *k* as a core point and start a new cluster
   - Add all of the observations in the neighborhood of *k* to the cluster
   - Visit all of the observations in the neighborhood of *k*
   - If a neighboring observation is also a core point, its neighbors are added to the cluster and visited (this process can be repeated many times, jumping from core point to core point).
   - If a neighboring observation is an outlier, it is relabeled as a boundary point and its neighbors are not visited.
5. Eventually, all of the points that are density-reachable from our initial point *k* will be added to the cluster; at this point, start over from step 1. Repeat until every point has been classified.

# DBSCAN Example

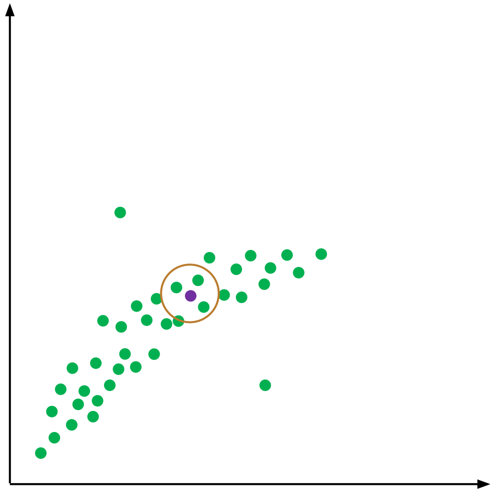
Pick a point that has not been classified



Purple point: the point of interest

# DBSCAN Example

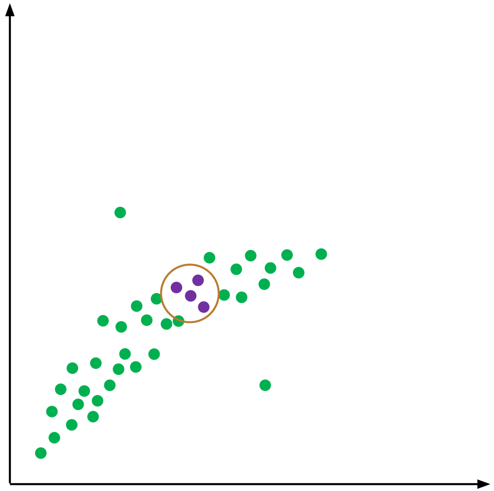
Examine the point's neighborhood



Purple point: the point of interest, *p*
Orange circle: the neighborhood of *p*
Radius of orange circle: $\varepsilon$
MinPts: 3

# DBSCAN Example

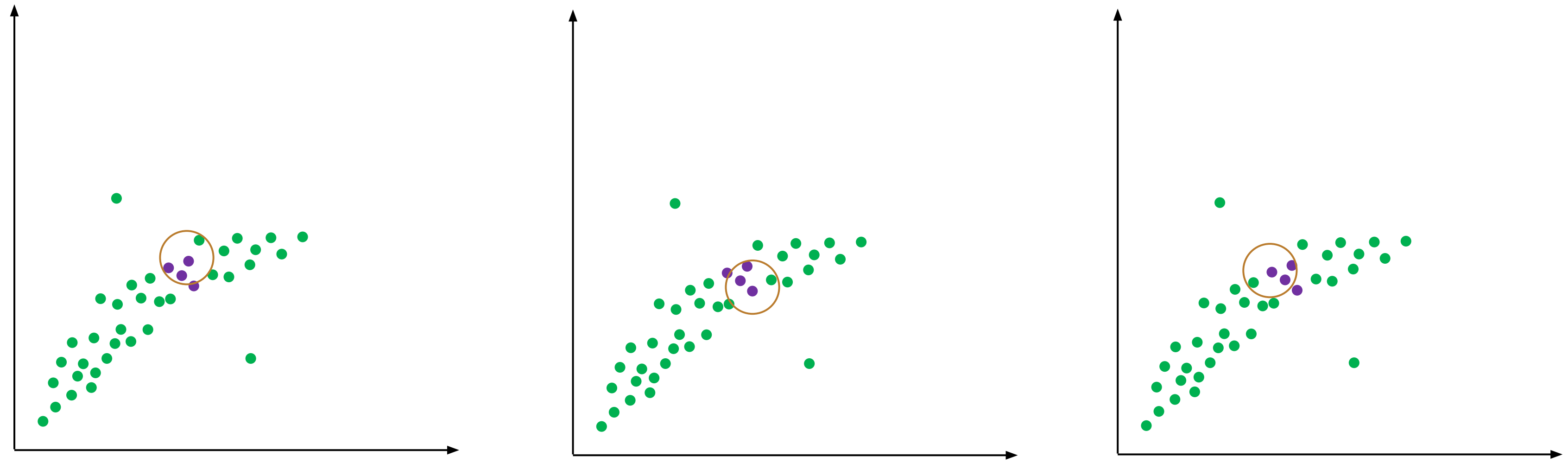
Since the number of observations in the neighborhood exceeds *minPts*, a new cluster is formed and the observations in the neighborhood are added to the cluster



Purple points: the members of the new cluster
Green points: not yet categorized

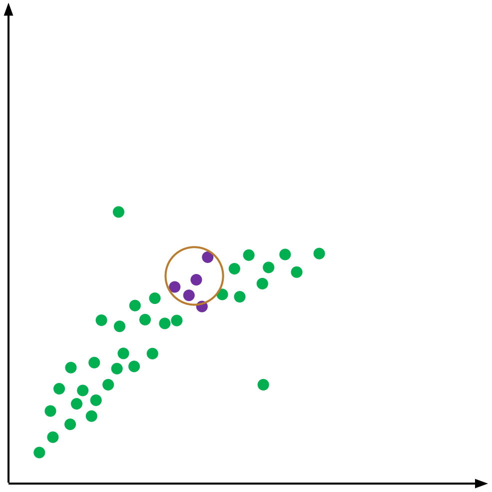
# DBSCAN Example

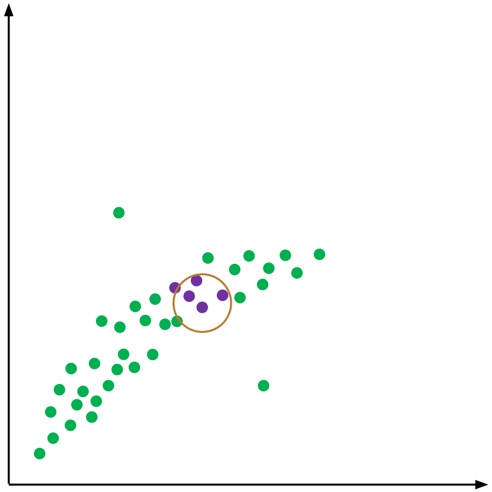
Examine the neighborhoods of the newly added neighbors to our initial point

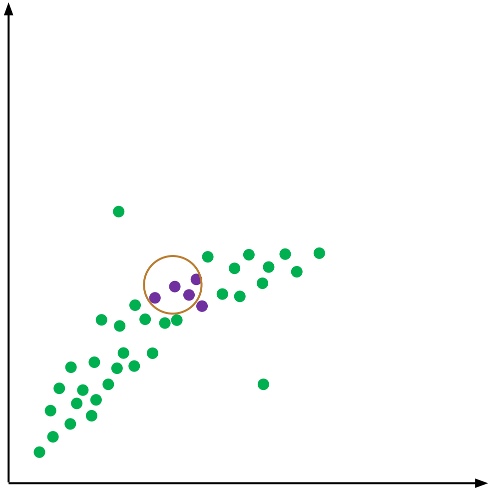


# DBSCAN Example

All of the neighbors have greater than *minPts* observations in their neighborhoods, so they are labeled core points; their neighboring observations are added to the cluster and have their neighborhoods visited

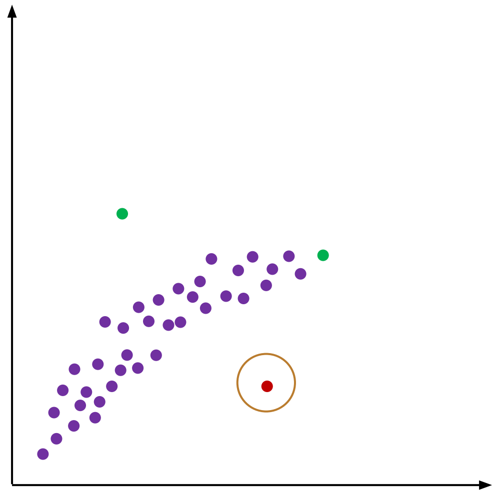






# DBSCAN Example

This process is repeated until no more core points are found; after the last round of boundary points are added, we have our final cluster. We then start over with a new point (in this case, an outlier)



# Advantages and Disadvantages

# Advantages and Disadvantages

## Advantages

- Doesn’t require any pre-specified number of clusters
- Like k-medoids and hierarchical clustering, works with arbitrary dissimilarity measures and categorical data
- Can identify clusters with highly flexible, non-standard shapes
- Relatively resistant to noise
- Can identify and label outliers
- Reasonably good time complexity: naïve implementation would be $O(n^2)$, but well-optimized versions can achieve $O(n*\log(n))$ average case complexity for most practical use cases (i.e. data that is not extremely high-dimensional)

## Disadvantages

- Results can be very sensitive to parameter choices
- Choosing the right values of $\varepsilon$ and *minPts* can be difficult
- Has difficulty when there are clusters of varying density

# Recap

- Density-based clustering defines clusters as regions of high density (number of points/area) separated by regions of low density
- The most popular technique for density-based clustering is DBSCAN, which has two parameters defining the size of the neighborhood and the minimum number of points needed to form a cluster
- DBSCAN can produce clusters with very flexible shapes and variable sizes, and it also scales significantly better than most other algorithms
- However, choosing optimal parameter values can be tricky

# Exercise:
# Density-Based Clustering

- Jupyter notebook
- Data set: Starbucks locations in the U.S.
- Choose a reasonably sized subset of the locations in the U.S.
- Build a DBSCAN model using an initially specified set of values for eps and minPts (we do not need to subset; details as to why are provided in the exercise)
- Plot the results on a map
- Tweak the values of eps and minPts to produce more meaningful clustering results and replot on map