

# Unsupervised Machine Learning with Python

# Section 5.1: DBSCAN Algorithm

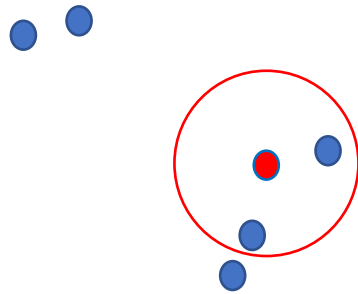
# DBSCAN : What is it?

- DBSCAN is an acronym for Density Based Spatial Clustering of Applications with Noise
- DBSCAN is density based clustering approach grouping points closely clustered together and classifying points in low density regions as noise
- User specifies density (a radius and minimum number of points) for a cluster to exist
- See [UnsupervisedML\\_Resources.pdf](#) for links to additional resources

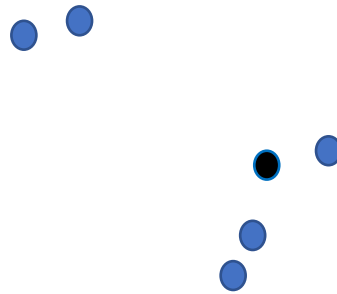
# DBSCAN: Core Points and Noise Points

Specify minimum number of points (minpts=3 in example) and radius  $\varepsilon$

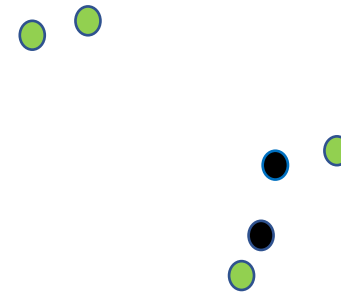
- (A) Find neighbours of a data point (all points within distance of  $\varepsilon$ )
- (B) If number of neighbours is at least minpts (including data point), then trial point is CORE and we will build cluster from this point
- (C) If data point doesn't have minpts neighbours, then it is a NOISE point



**A: Focus on red point and count neighbours in  $\varepsilon$  ball**



**B: Number of neighbours is at least minpts=3 so point is CORE – label as black**



**C: Initial analysis shows 2 CORE and 4 NOISE points**

# DBSCAN: Building a Cluster from Core Point

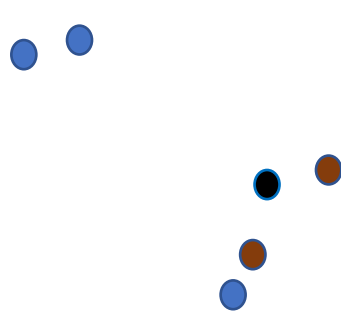
Build cluster starting from CORE point

(D) Investigate neighbours of CORE point

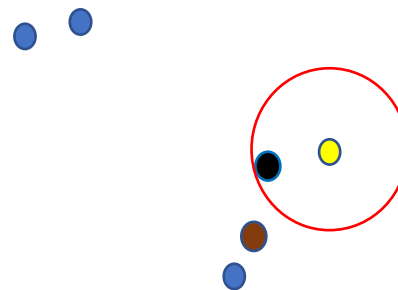
(E) If neighbour is not a core point, then it is a BORDER point (ADD TO CLUSTER)

(F) If neighbour is CORE point (ADD TO CLUSTER), then repeat steps (D), (E), (F) until one runs out of core points

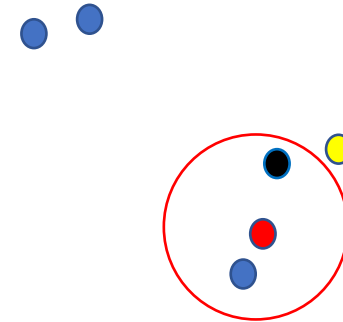
Start new cluster from CORE point that is not part of existing cluster



**D: Look  $\epsilon$  balls around neighbours of original CORE point**



**E: yellow is BORDER point since it doesn't have minpts=3 neighbour points**

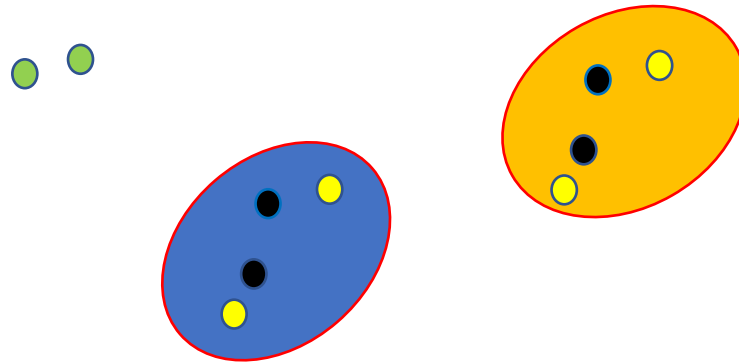


**F: red is a CORE point, so repeat steps D, E, F**

# DBSCAN: Summary

In this example

- Green are NOISE
- Black are CORE
- Yellow are BORDER
- There are 2 clusters and the 2 NOISE points



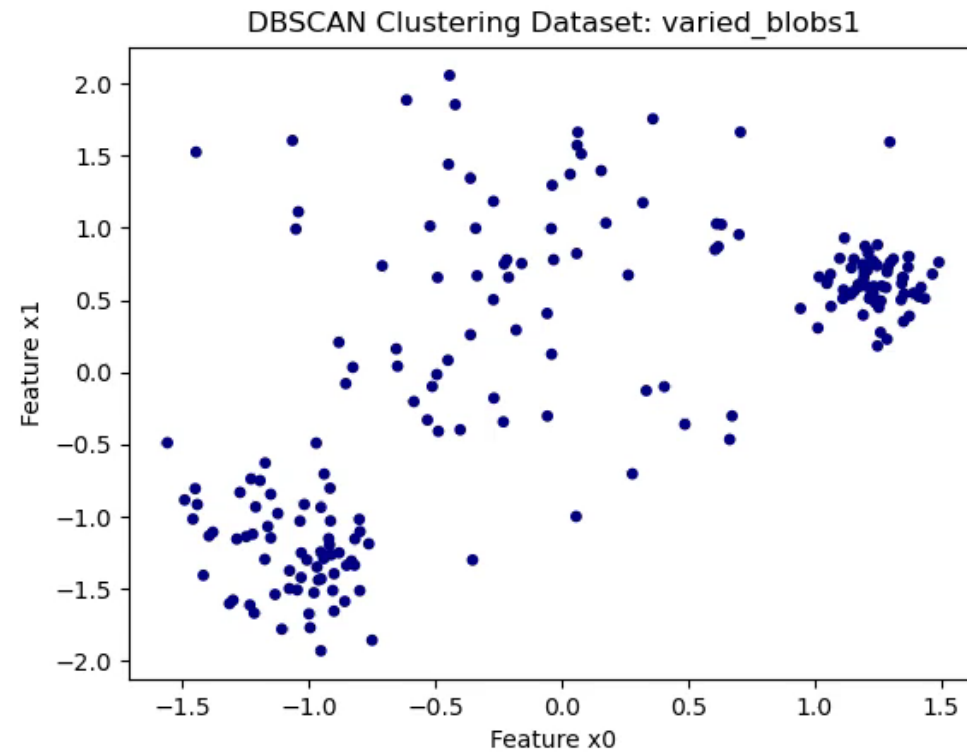
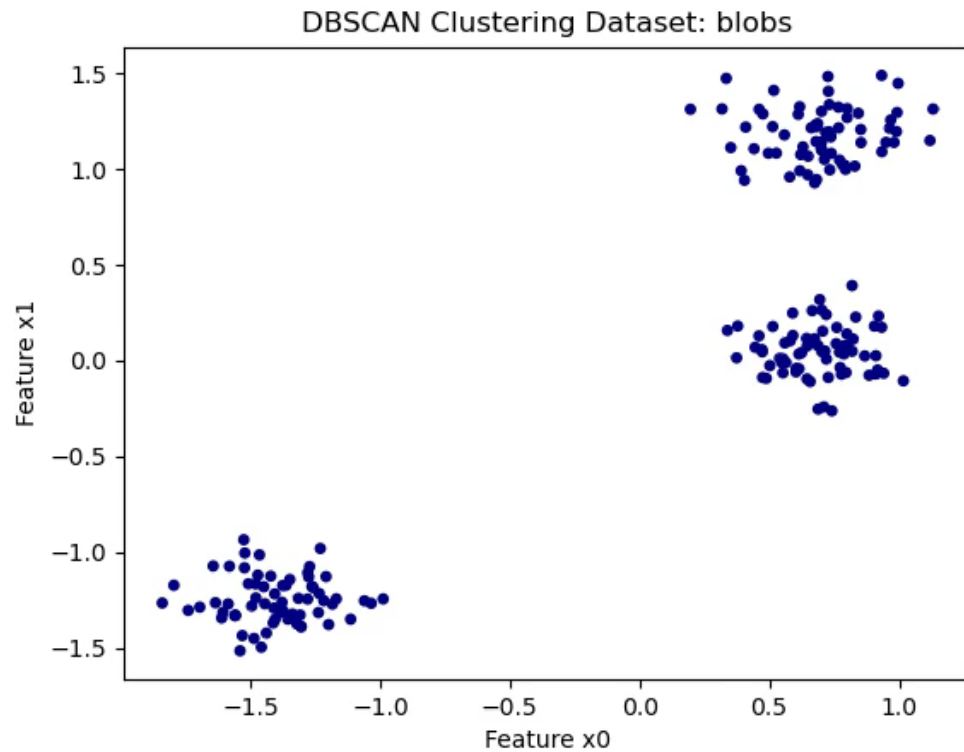
# DBSCAN Algorithm

- Assume  $M$  data points  $\{X_i\}$
- Specify minpts and radius  $\epsilon$
- (1) Loop over all data points  $\{X_i\}$ 
  - If  $X_i$  is unvisited, then find neighbours else go to next point
  - If number of neighbours less than minpts, label as NOISE and go to next point
  - Label  $X_i$  as CORE point and start new cluster
  - $S$  is set of neighbours of  $X_i$
  - Loop over points  $Y$  in  $S$ 
    - If  $Y$  is previously labeled as NOISE, then relabel as BORDER and go to next  $Y$
    - If  $Y$  was visited before, then go to next  $Y$
    - If  $Y$  is not Core, then label as BORDER and go to next  $Y$
    - If  $Y$  is CORE point, then label as CORE and add its neighbours to  $S$

# DBSCAN: Example

Example:

- sklearn blobs and varied\_blobs1 datasets with 200 points
- Use minpts = 4 and  $\varepsilon = 0.3$





# DBSCAN: Choice of minpts and $\epsilon$

Choosing minpts:

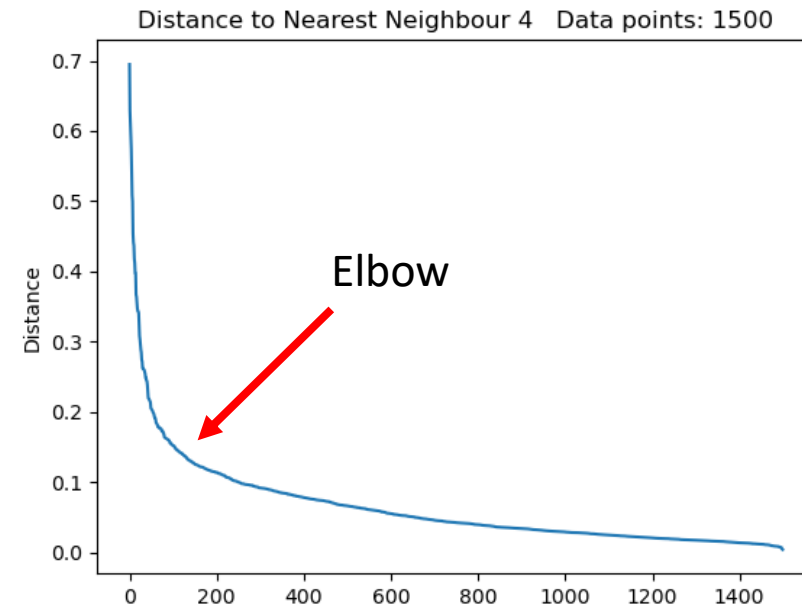
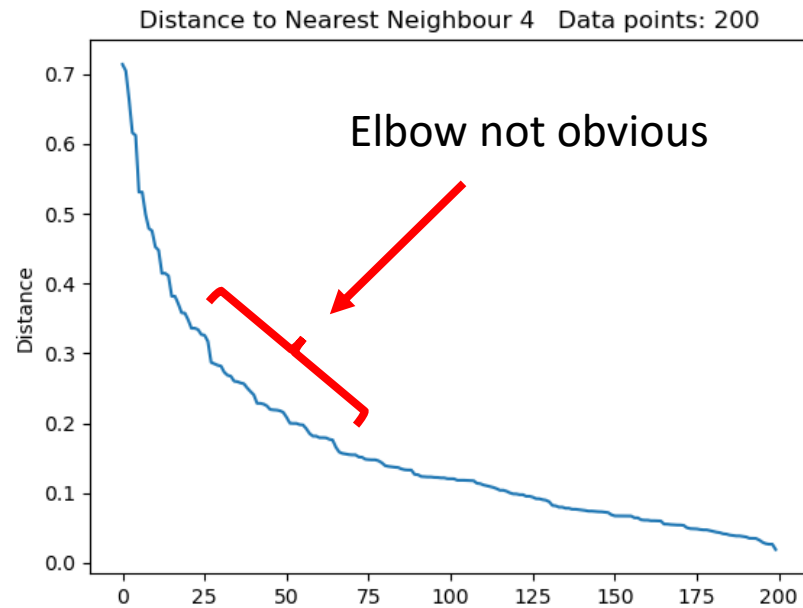
- Rule of thumb is minpts related to number of dimensions  $d$
- Suggested values  $\text{minpts} \geq d+1$  or  $\text{minpts} \geq 2d$

Choosing  $\epsilon$ :

- If  $\epsilon$  is large (clusters can have low density)  $\rightarrow$  large clusters
- If  $\epsilon$  is small (clusters must have high density)  $\rightarrow$  small clusters and noise points

# Example: Choosing $\epsilon$ using Elbow Method

- Choose minpts=4 and “varied\_blobs1” dataset
- For each data point compute distance to  $k = 4$  nearest neighbour
- Sort distances in descending order and plot
- Elbow is point where curve starts to level off
- For more details, see Wikipedia page for Elbow Method (citation in Resources file)

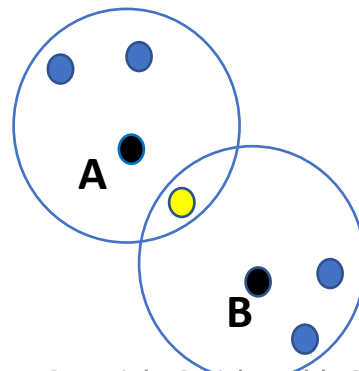


# DBSCAN: Complexity

- Assume:  $M$  data points in  $d$  dimensions
- Worst case scenario requires  $O(M^2)$  operations as  $M \rightarrow \infty$
- Memory requirement is  $O(M)$  storage as  $M \rightarrow \infty$
- Number of operations and storage depend linearly on dimension  $d$
- Can reduce number of operations by using more sophisticated approaches for identifying points in neighbourhood - see Wikipedia page for DBSCAN for more details

# DBSCAN: Notes

- Approach identifies clusters that are arbitrarily shaped
- Number of clusters is not pre-defined
- Must specify minpts and radius  $\epsilon$  so tuning required
- Single minpts &  $\epsilon$ , so method performs poorly if clusters have varying densities
  - OPTICS is a variant of DBSCAN that can handle varying densities
- Division into clusters is not unique as a BORDER point may belong to more than one cluster group
  - Consider example of minpts = 4 where yellow BORDER point can belong to cluster based on CORE A or CORE B - BORDER point assigned to first cluster that is created



# Unsupervised Machine Learning with Python

# Section 5.2: DBSCAN Code Design

# DBSCAN Code Design

- This section presents a design of the DBSCAN Clustering code
- Design is based on algorithm described in Section 5.1
- Stop video here, if you would like to do code design yourself

# DBSCAN Code Design

- (1) Derive dbscan class from clustering\_base class
- (2) Introduce self.list\_label, which is label for each data point
  - “unvisited” if point has not yet been assigned a label
  - “core”
  - “noise”
  - “border”
- (3) Assign cluster label = -1 to all data points initially
  - Points in first cluster assigned label 0, points in second cluster assigned label 1, ...
  - Noise points continue to have label = -1



# dbscan class: Principal Variables

Variable	Type	Description
self.time_fit	float	Time for clustering
self.X	2d numpy array	Contains the dataset Number of rows = number of dimensions for data Number of cols = number of data points Example: 2 dimensions and 5 data points $\begin{bmatrix} 1 & 1.1 & 0.8 & 0.6 & 0.6 \\ 0.9 & 1.0 & 0.7 & 0.5 & 0.5 \end{bmatrix}$
self.clustersave	list of 1d numpy arrays	self.clustersave[i][j] is cluster assignment for iteration i, data point j Example: for 3 iterations: $\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 0 & -1 \end{bmatrix}$
self.list_label	list of strings	Label for each data point: one of “unvisited”, “core”, “border” or “noise”
self.animation	boolean	If True, then store update for clustersave for each data point for animation purposes. If false, clustersave has 2 entries: initial assignment and final assignment.

# dbscan class – Key Methods

Method	Input	Description
<code>__init__</code>	<code>minpts</code> (integer) <code>epsilon</code> (float) <code>animation</code> (boolean)	Constructor for class
<code>initialize_algorithm</code>		Initialize variables for the algorithm: <code>self.clustersave</code> , <code>self.list_label</code> Return: nothing
<code>fit</code>	<code>X</code> (2d numpy array)	Performs dbscan clustering Return: nothing
<code>neighbours</code>	<code>idx</code> (integer)	Finds all points within distance <code>epsilon</code> of point with index <code>idx</code> Return: list of indices
<code>extend_cluster</code>	<code>cluster_number</code> (integer) <code>list_neighbour</code> (list)	This function builds cluster with label <code>cluster_number</code> starting with data points in <code>list_neighbour</code> Return: nothing
<code>update_cluster_assignment</code>	<code>cluster_number</code> (integer) <code>idx</code> (integer)	Update <code>self.clustersave</code> : point <code>idx</code> is assigned label <code>cluster_number</code> Return: nothing
<code>add_points</code>	<code>list_cluster</code> (list) <code>in_cluster</code> (list) <code>list_idx</code> (list)	Add points to <code>list_cluster</code> from <code>list_idx</code> if they have not already been considered Return: <code>list_cluster</code> , <code>in_cluster</code>

# Unsupervised Machine Learning with Python

# Section 5.3: DBSCAN Code Walkthrough

# DBSCAN Clustering: Code Walkthrough

Code located at:

- UnsupervisedML/Code/Programs

Files to Review	Description
dbscan.py	Class for DBSCAN clustering
driver_dbscan.py	Driver for DBSCAN clustering

Course Resources at:

- <https://github.com/satishchandrareddy/UnsupervisedML/>
- Stop video if you would like to implement code yourself first