GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

CLUSTER ANALYSIS

I. CLUSTER ANALYSIS
II. THE K-MEANS ALGORITHM
III. CHOOSING K
IV. DBSCAN CLUSTERING

I. CLUSTER ANALYSIS

	continuous	categorical
supervised	**???**	**???**
unsupervised	**???**	**???**

	continuous	categorical
supervised	**regression**	**classification**
unsupervised	**dimension reduction**	**clustering**

Q: What is a cluster?

Q: What is a cluster?

A: A group of similar **data points.**

Q: What is a cluster?

A: A group of similar **data points.**

The concept of similarity is central to the definition of a cluster, and therefore to cluster analysis.

In general, greater similarity between points leads to better clustering.

Q: What is the purpose of cluster analysis?

Q: What is the purpose of cluster analysis?

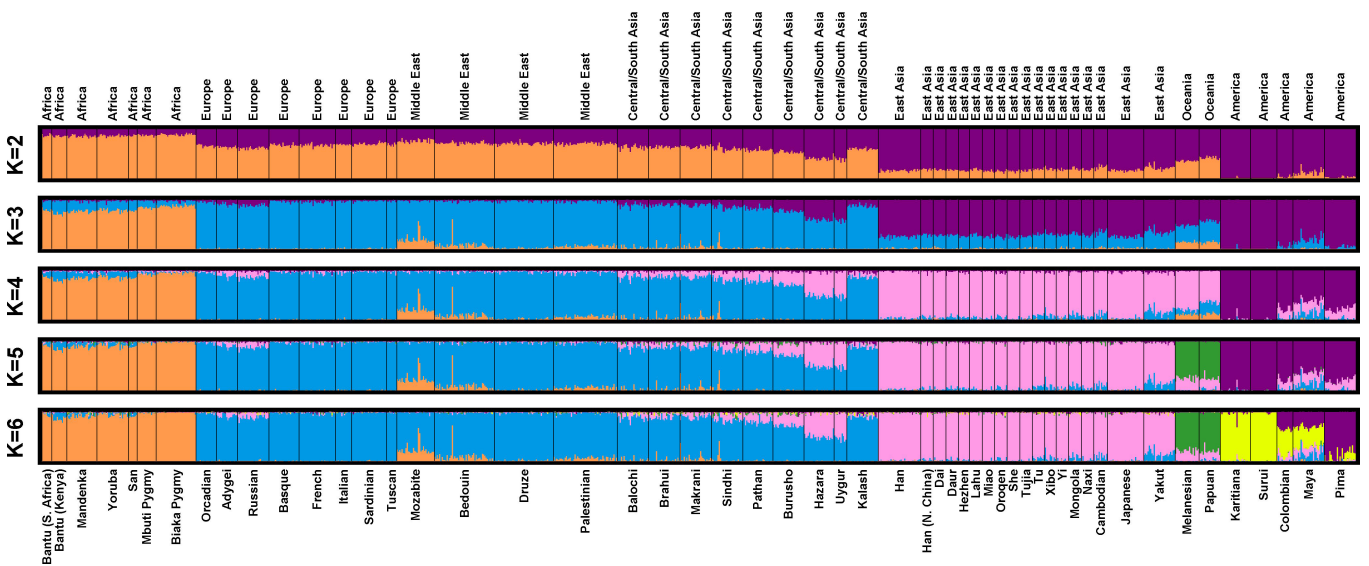
A: To enhance our understanding of a dataset by dividing the data into groups.

Clustering provides a *layer of abstraction* from individual data points.

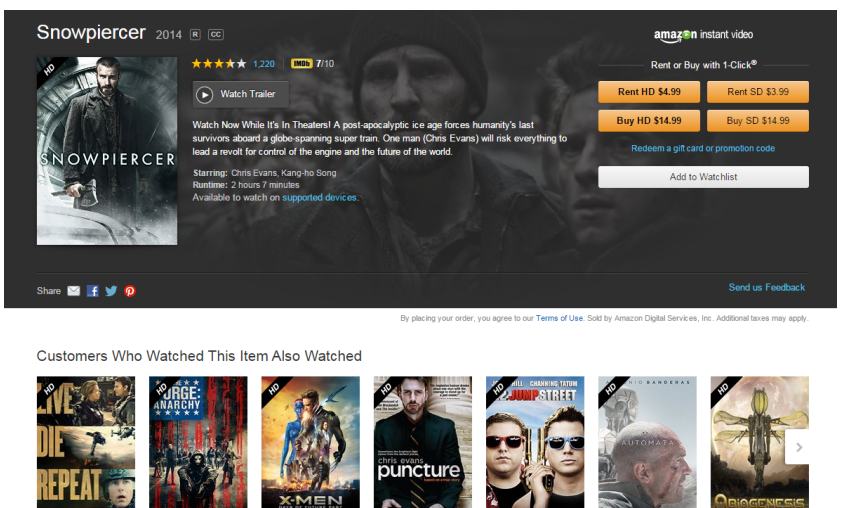
The goal is to extract and enhance the natural structure of the data

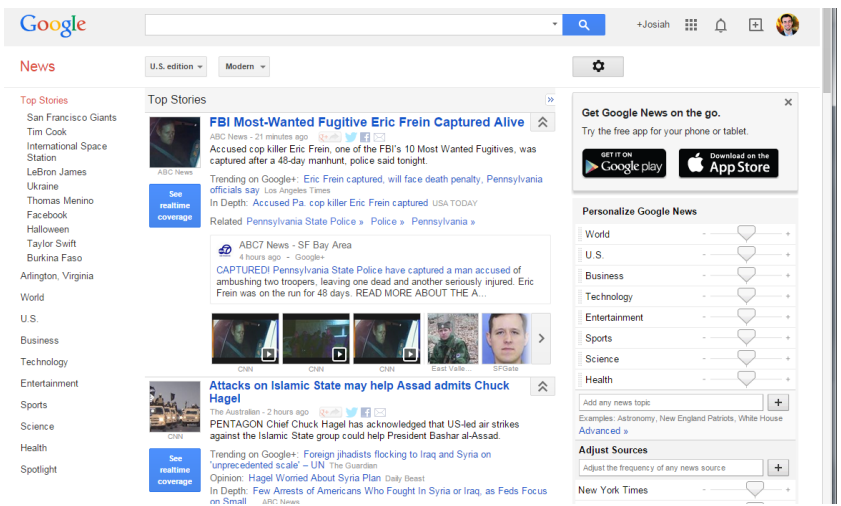
Clustering can be useful in a wide variety of domains, including genetics, consumer internet and business.

Clustering can be useful in a wide variety of domains, including genetics, consumer internet and business.

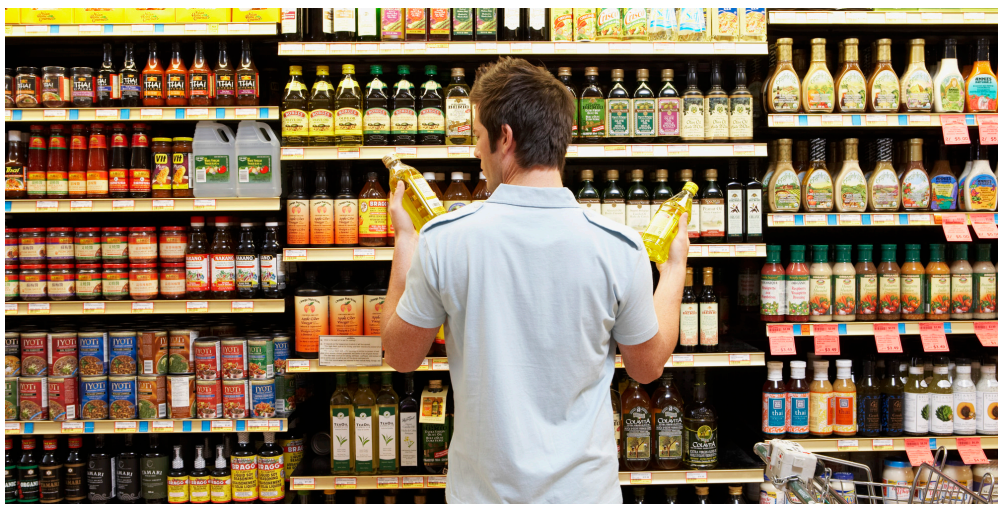


Clustering can be useful in a wide variety of domains, including genetics, consumer internet and business.





Clustering can be useful in a wide variety of domains, including genetics, consumer internet and business.



http://i.huffpost.com/gen/1563531/thumbs/o-GROCERY-STORE-facebook.jpg

There are many kinds of classification procedures. For our class, we will be focusing on K-means clustering, which is one of the most popular clustering algorithms.

K-means is an iterative method that partitions a data set into k clusters.

II. K-MEANS CLUSTERING

Q: How does the algorithm work?

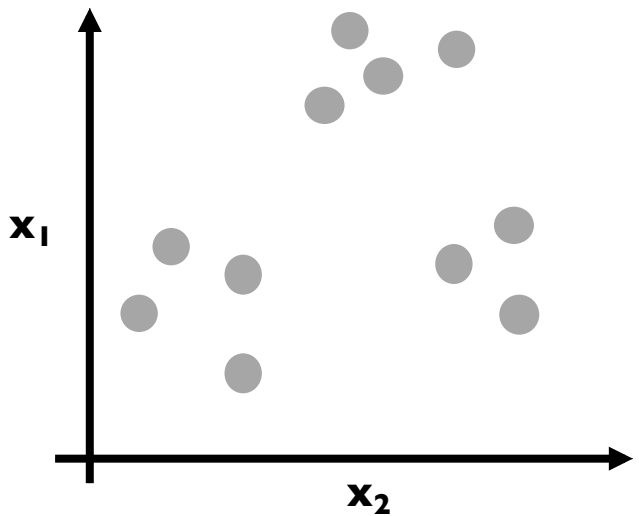
1) choose k initial centroids (note that k is an input)

2) for each point:
   - assign point to nearest centroid

3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

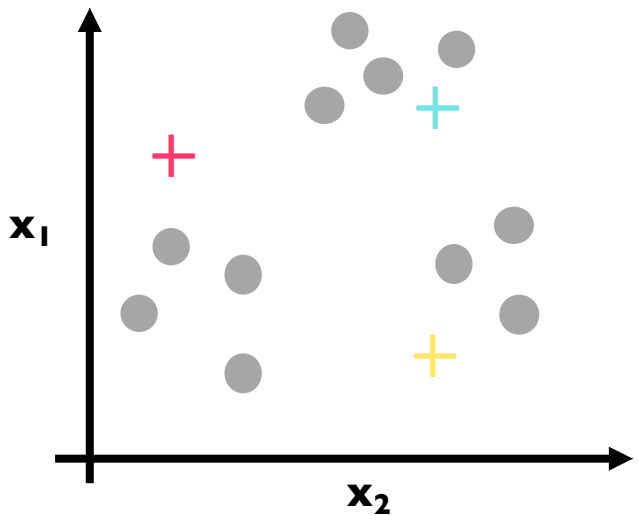
2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2–3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

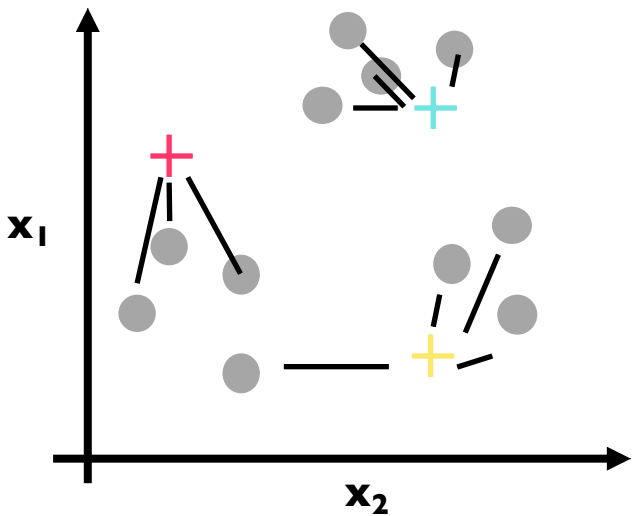
2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

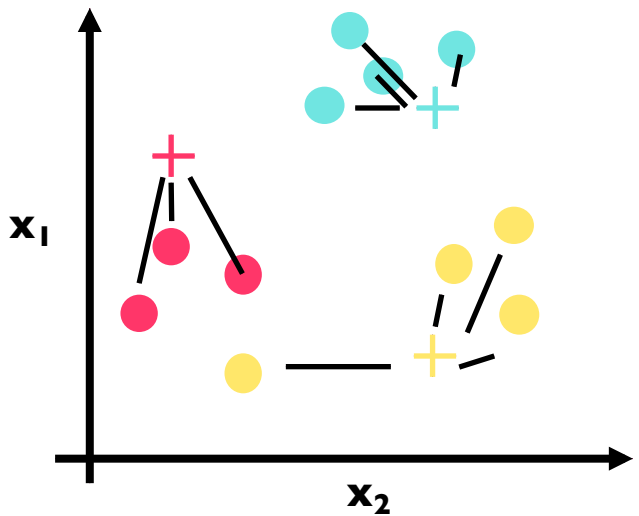
2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2–3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

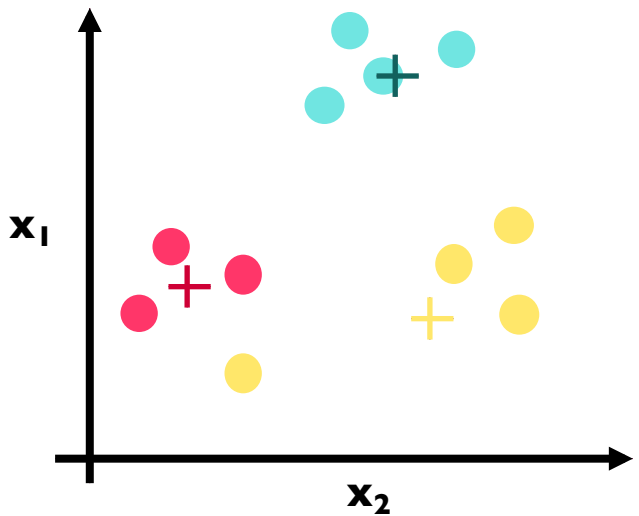
2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

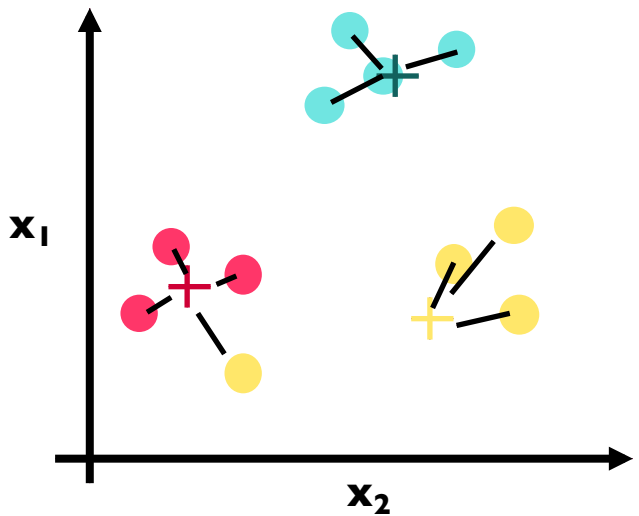
2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2–3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

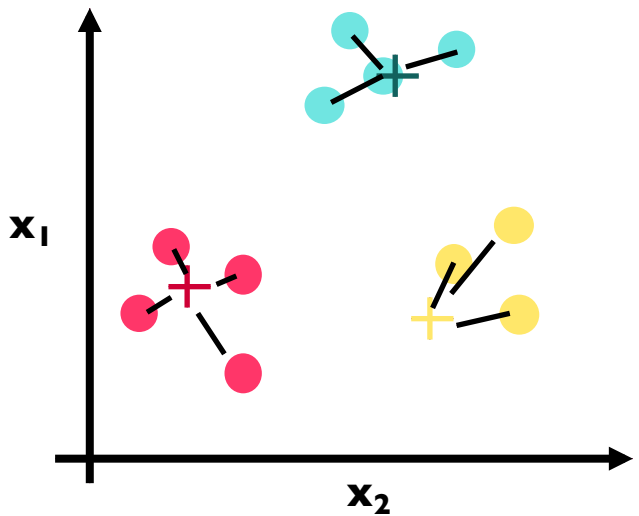
2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

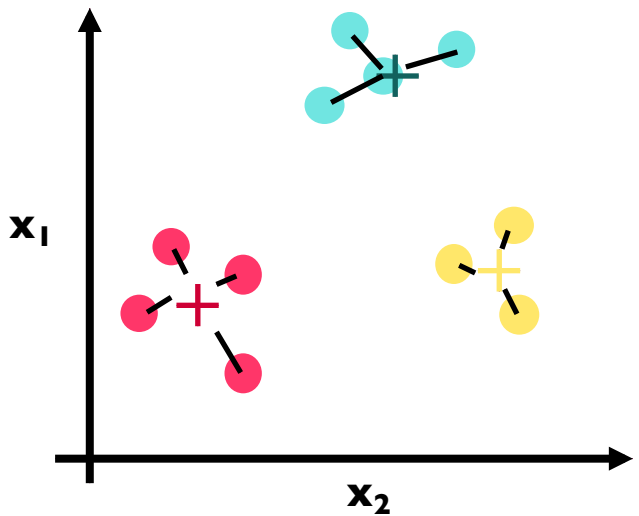
2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2–3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

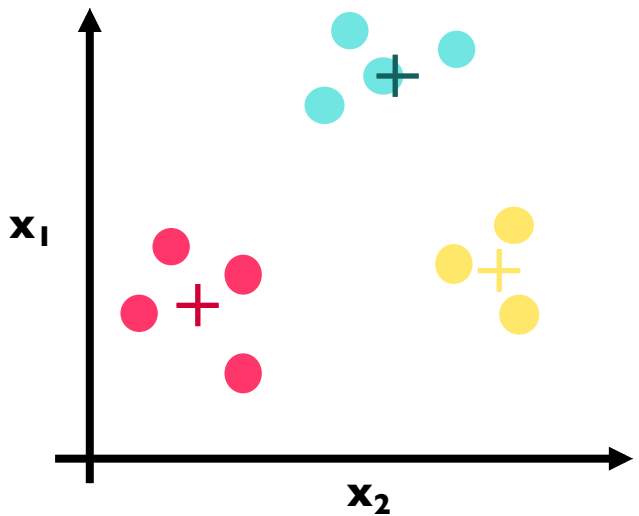
2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2–3 until stopping criteria met

Q: How do you choose the initial centroid positions?

Q: How do you choose the initial centroid positions?

A: There are several options:

- randomly (but may yield divergent behavior)
- perform alternative clustering task, use resulting centroids as initial k-means centroids
- start with global centroid, choose point at max distance, repeat (but might select outlier)

Q: How do you determine which centroid a given point is most similar to?

Q: How do you determine which centroid a given point is most similar to?

The similarity criterion is determined by the measure we choose.

In the case of k-means clustering, the similarity metric is the Euclidian distance:

Q: How do we re-compute the positions of the centers at each iteration of the algorithm?

A: By calculating the centroid (i.e., the geometric center)

This is done by taking the average of each index of vectors

Centroid of [1, 4, 2] and [6, 4, 2] is

[(1 + 6) / 2, (4 + 4) / 2, (2 + 2) / 2] == [3.5, 4, 2]

We iterate until some stopping criteria are met; in general, suitable convergence is achieved in a small number of steps.

II. CLUSTER VALIDATION

In general, k-means will converge to a solution and return a partition of k clusters, even if no natural clusters exist in the data.

In general, k-means will converge to a solution and return a partition of k clusters, even if no natural clusters exist in the data.

We will look at two validation metrics useful for partitional clustering, cohesion **and** separation**.**

Cohesion **measures clustering effectiveness within a cluster.**

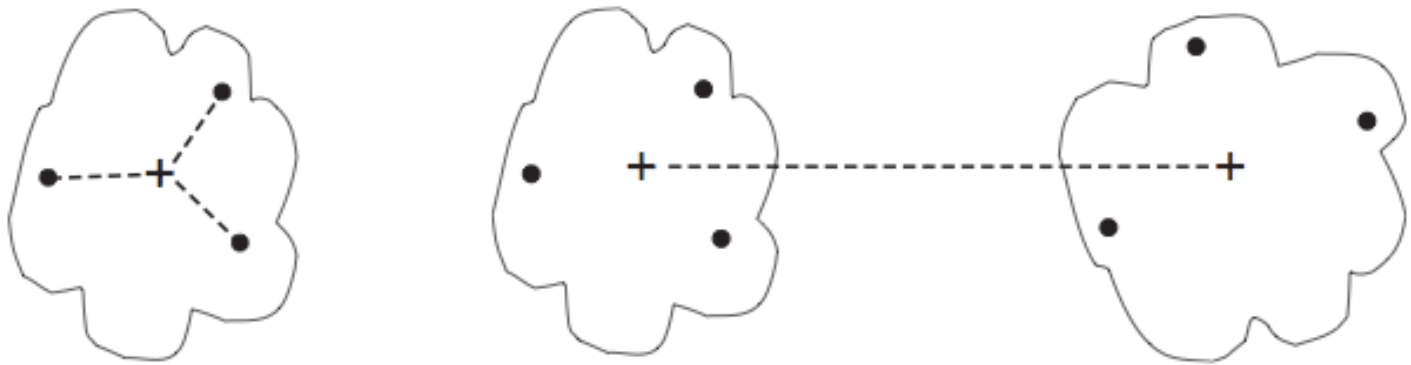
$$\hat{C}(C_i) = \sum_{x \in C_i} d(x, c_i)$$

Cohesion **measures clustering effectiveness within a cluster.**

$$\hat{C}(C_i) = \sum_{x \in C_i} d(x, c_i)$$

Separation **measures clustering effectiveness between clusters.**

$$\hat{S}(C_i, C_j) = d(c_i, c_j)$$



(a) Cohesion. (b) Separation.

Figure 8.28. Prototype-based view of cluster cohesion and separation.

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf

One useful measure than combines the ideas of cohesion and separation is the

silhouette coefficient

The silhouette coefficient can take values between -1 and 1.

In general, we want separation to be high and cohesion to be low. This corresponds to a value of SC close to +1.

A negative silhouette coefficient means the cluster radius is larger than the space between clusters, and thus clusters overlap.

Ultimately, cluster validation and clustering in general are suggestive techniques that rely on human interpretation to be meaningful.

Strengths:

K-means is a popular algorithm because of its computational efficiency and simple and intuitive nature.

Strengths:

K-means is a popular algorithm because of its computational efficiency and simple and intuitive nature.

Weaknesses:

However, K-means is highly scale dependent, and is not suitable for data with widely varying shapes and densities.

IV. DBSCAN CLUSTERING

DBSCAN: Density-Based Spatial Clustering of Applications with Noise

For DBSCAN, clusters are areas of high density separated by areas of low density.

DBSCAN Algorithm:

1. Choose “epsilon” and “min_samples”
2. Pick an arbitrary point, and check if there are at least “min_samples” points within the 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

Visualization: Uniform Points

DBSCAN **Advantages**:

Clusters can be any shape or size

No need to choose the number of clusters

DBSCAN **Disadvantages**:

More parameters to tune

Doesn't work with clusters of varying density

Note: Not every point is necessarily assigned to a cluster!