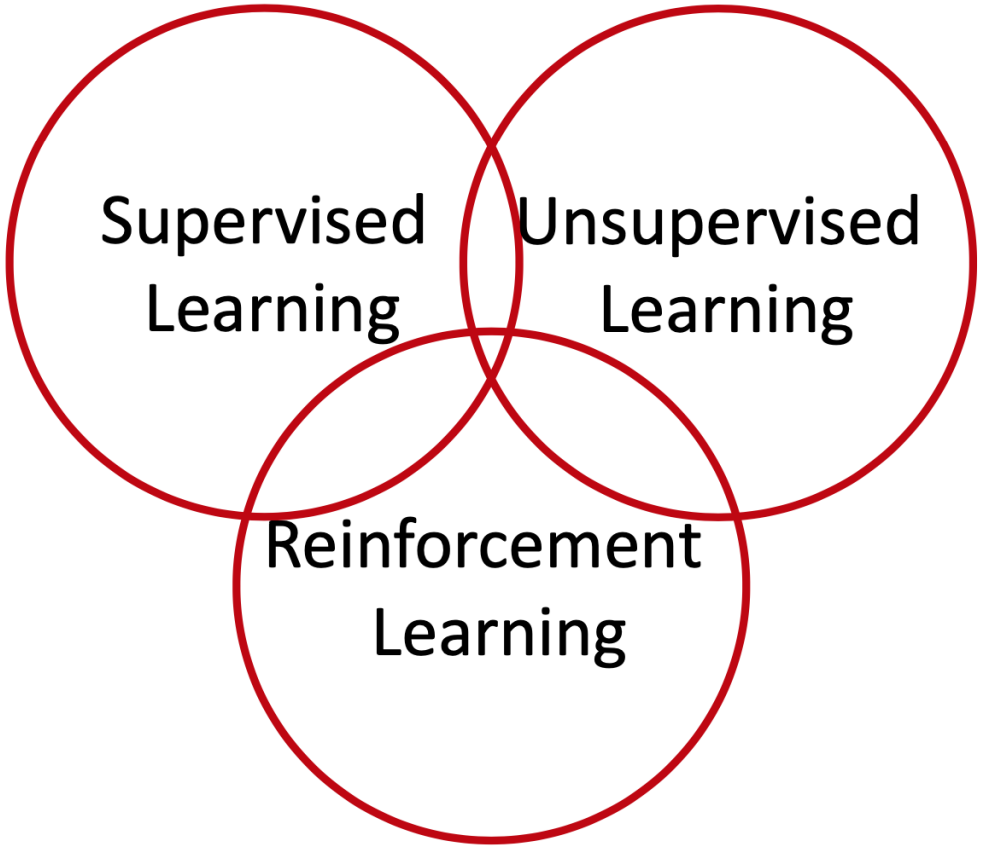
University of Salford MANCHESTER

Artificial Intelligent – Week 10

Dr Salem Ameen

S.A.AMEEN1@SALFORD.AC.UK

Week 10 - Machine Learning



Week 10 - Machine Learning

University of Salford MANCHESTER

❑ Key types of Machine Learning problems :

- **Supervised machine learning**: Learn to predict target values from labelled data.
- **Unsupervised machine learning: Find structure in *unlabelled data***
  - Find groups of similar instances in the data (clustering)
  - Finding unusual patterns (outlier detection)
- **Reinforcement Learning: RL** is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences.

Week 10 – Revision

University of Salford MANCHESTER

	Supervised vs Unsupervised	Regression vs Classification	Parametric vs Non-Parametric	Generative vs Discriminative
Linear Regression	Supervised	Regression	Parametric	Discriminative
Logistic Regression	Supervised	Classification	Parametric	Discriminative
k-NN	Supervised	either	Non-Parametric	Discriminative
Decision Tree	Supervised	either	Non-Parametric	Discriminative
Clustering				
PCA				

Week 10 - Machine Learning

University of Salford MANCHESTER

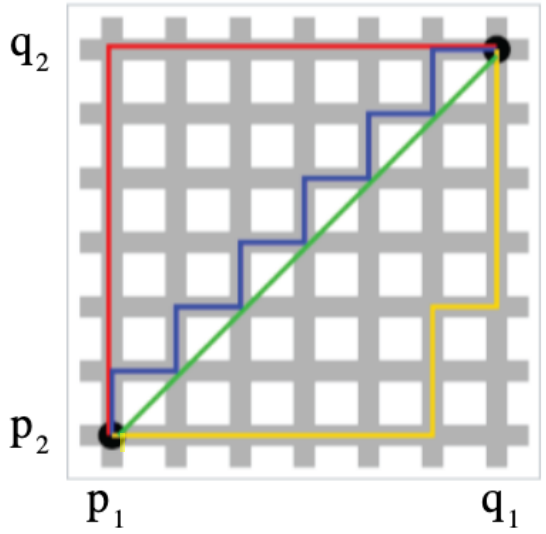
Learning Objectives

- Understand the common distance metrics (e.g., Euclidean, Manhattan)
- Understand how different clustering algorithms work (e.g., k-means, Hierarchical, DBScan)
- Explain the trade-offs between the clustering approaches
- Quantitatively describe the quality clusters' fit, according to different metrics

Week 10 - Machine Learning

1. Distance Metrics

In the picture below, we are concerned with measuring the distance between two points, **p** and **q**.



Week 10 - Machine Learning

University of Salford MANCHESTER

Euclidean Distance:

The Euclidean distance measures the shortest path between the two points, navigating through all dimensions:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

Manhattan Distance:

The Manhattan distance measures the cumulative difference between the two points, across all dimensions.

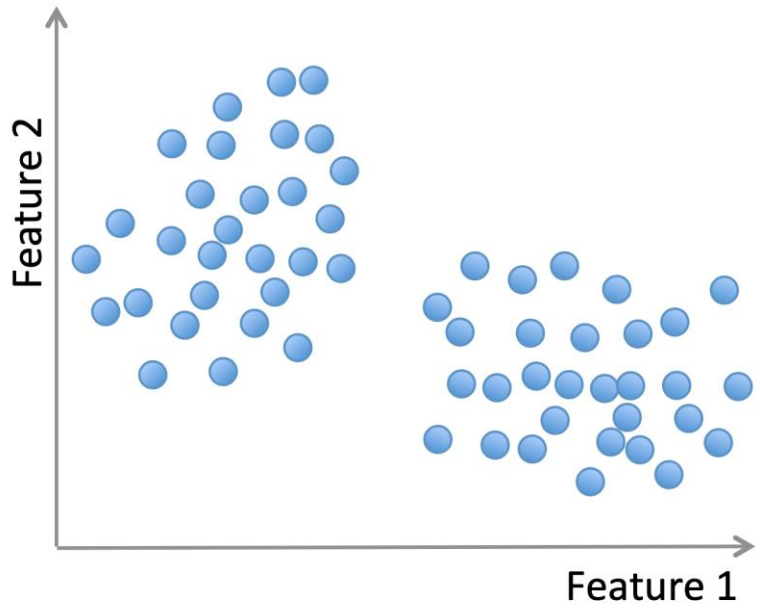
$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|$$

Cosine Similarity:

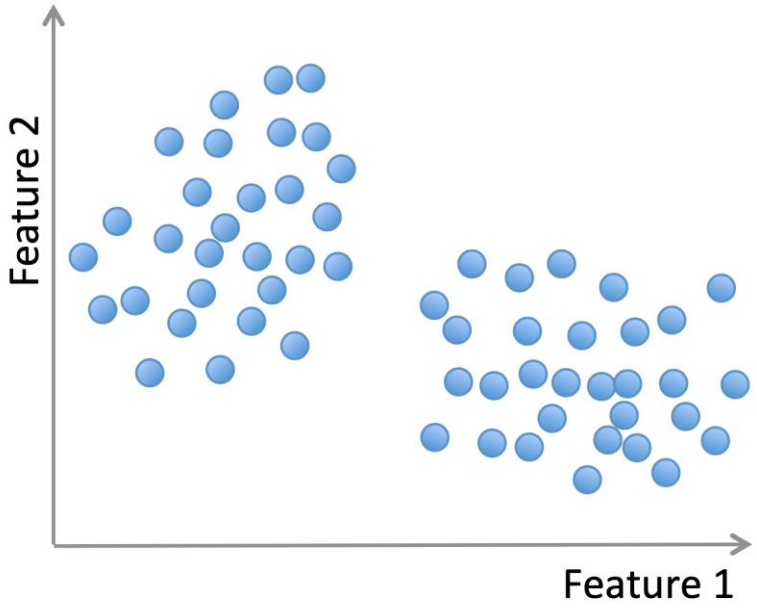
Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space.

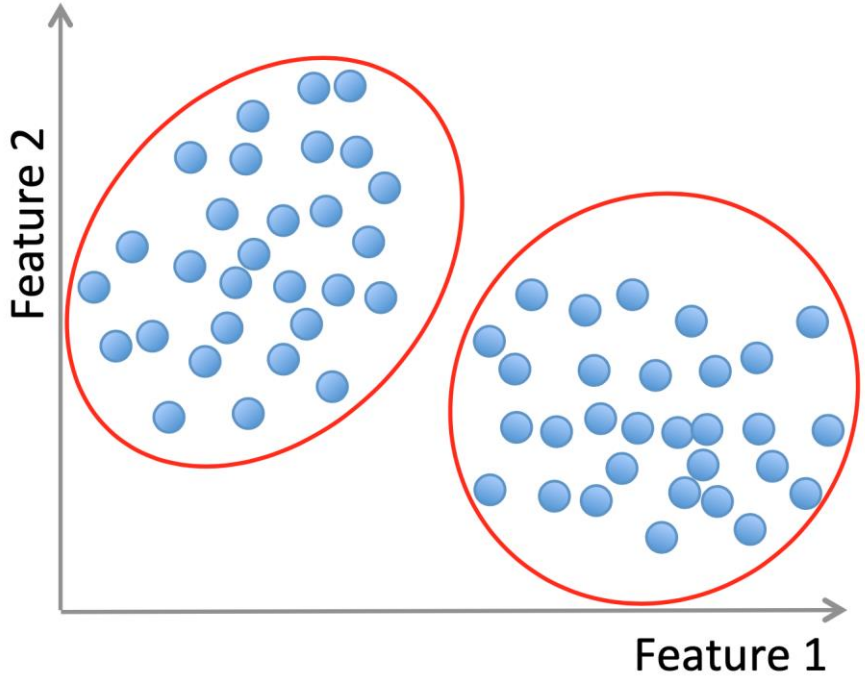
$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

Week 10 – Unsupervised learning



Week 10 – Unsupervised learning





Week 10 Unsupervised learning / Clustering

University of Salford MANCHESTER

What are the Uses of Clustering?

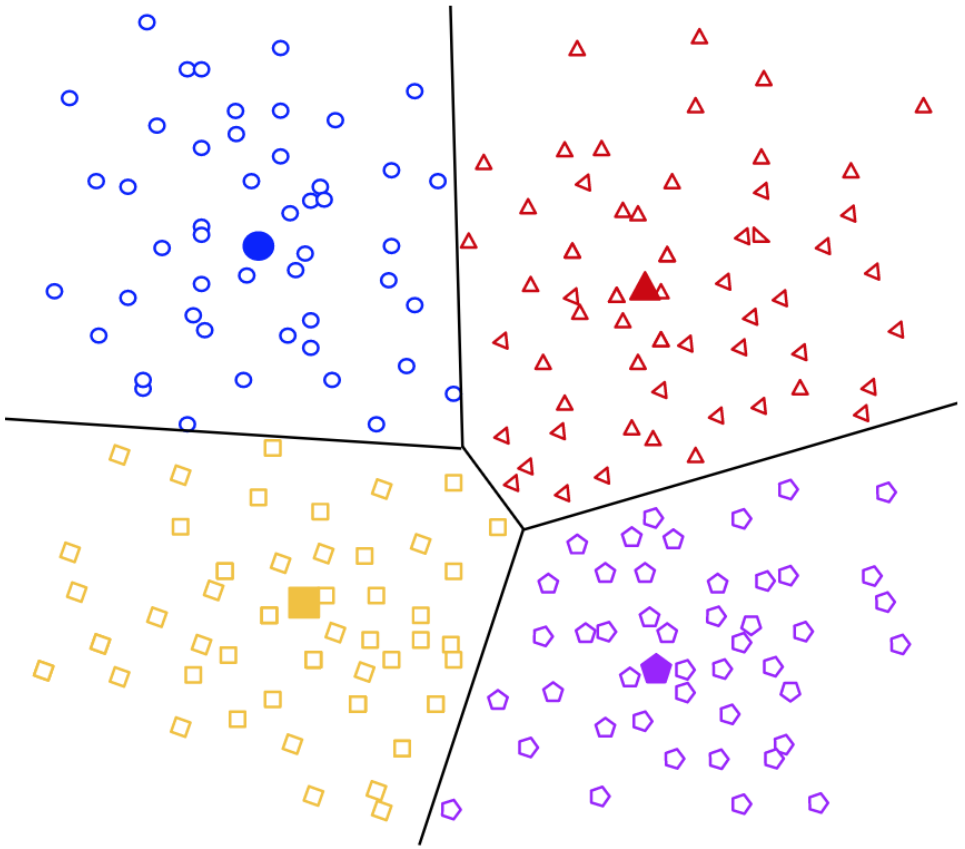
Clustering has a myriad of uses in a variety of industries. Some common applications for clustering include the following:

- anomaly detection
- market segmentation
- social network analysis
- search result grouping
- medical imaging
- image segmentation

Week 10 Unsupervised learning / Types of Clustering

University of Salford MANCHESTER

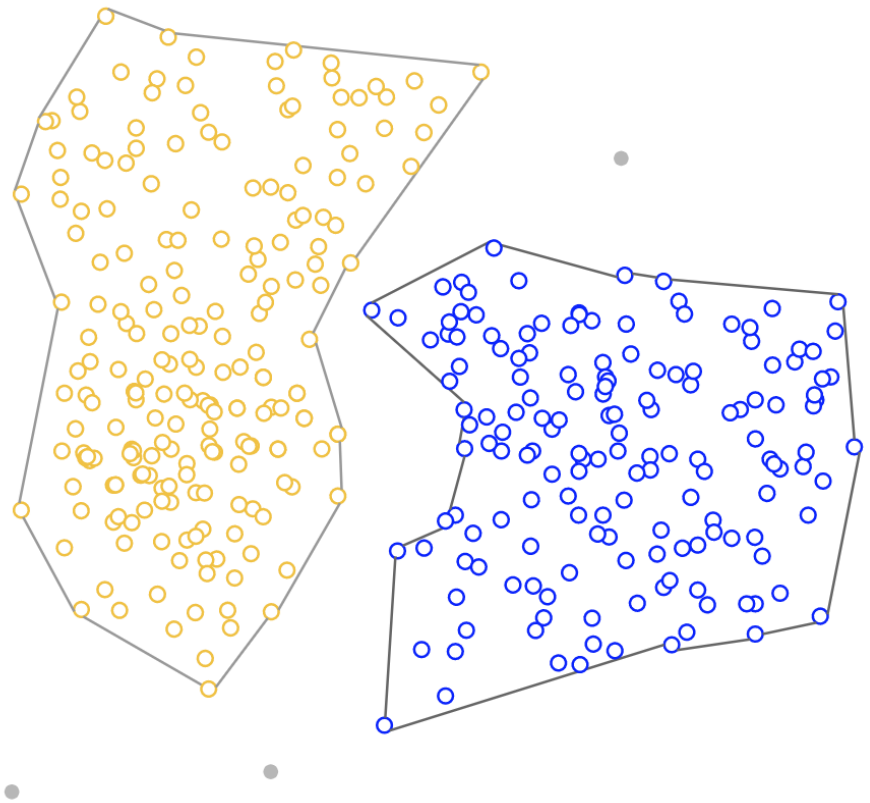
Centroid-based Clustering



Week 10 Unsupervised learning / Types of Clustering

University of Salford MANCHESTER

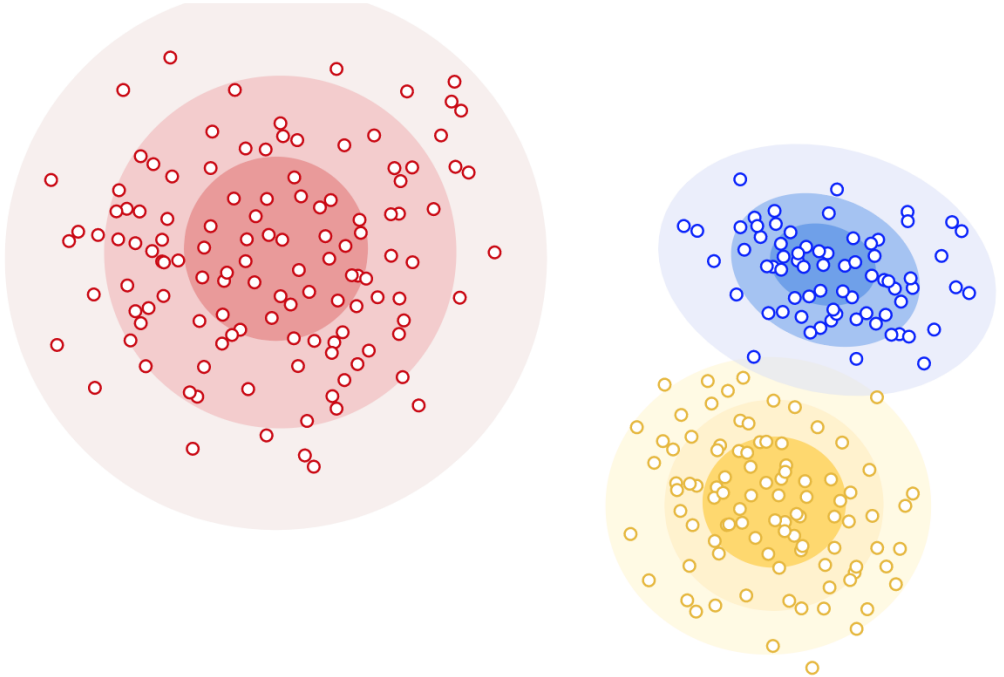
Density-based Clustering



Week 10 Unsupervised learning / Types of Clustering

University of Salford MANCHESTER

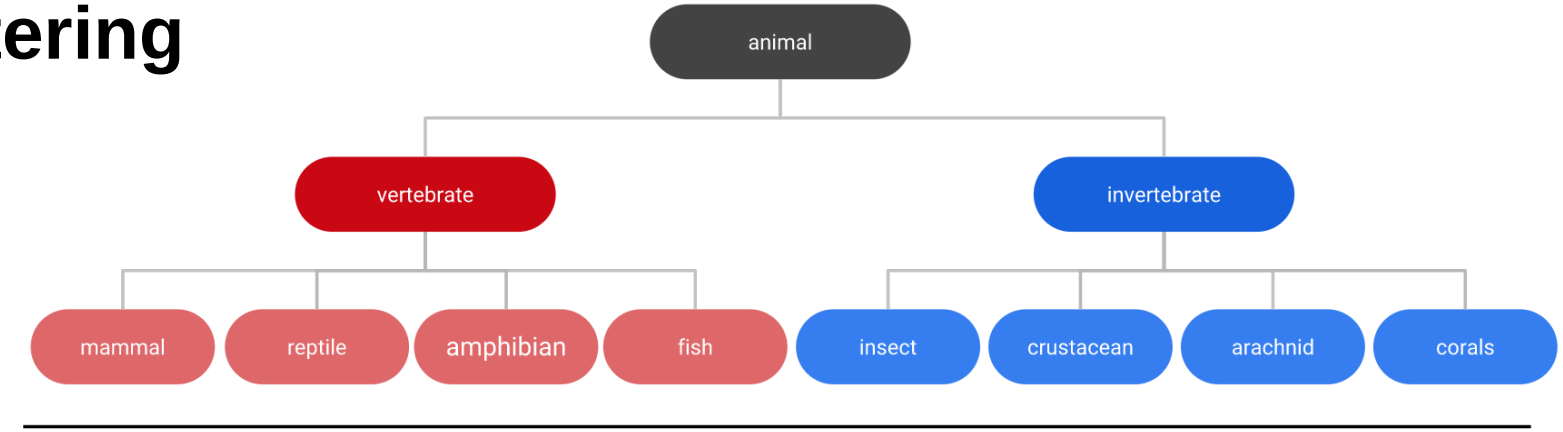
Distribution-based Clustering

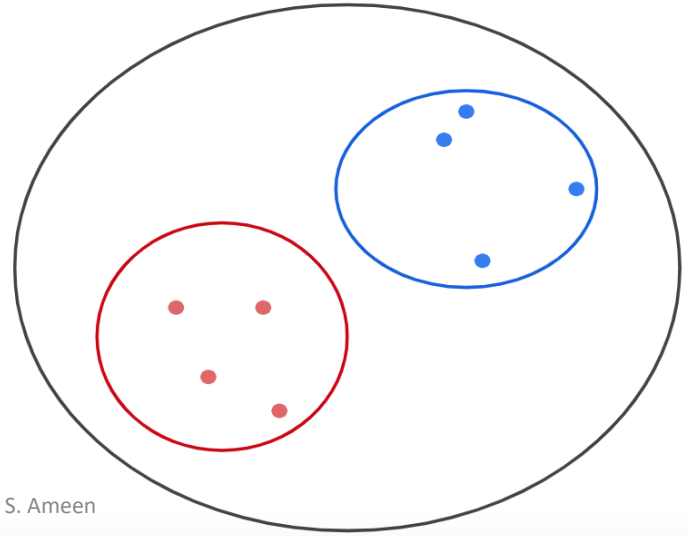


Week 10 Unsupervised learning / Types of Clustering

University of Salford MANCHESTER

Hierarchical Clustering





Week 10 Unsupervised learning / Types of Clustering

University of Salford MANCHESTER

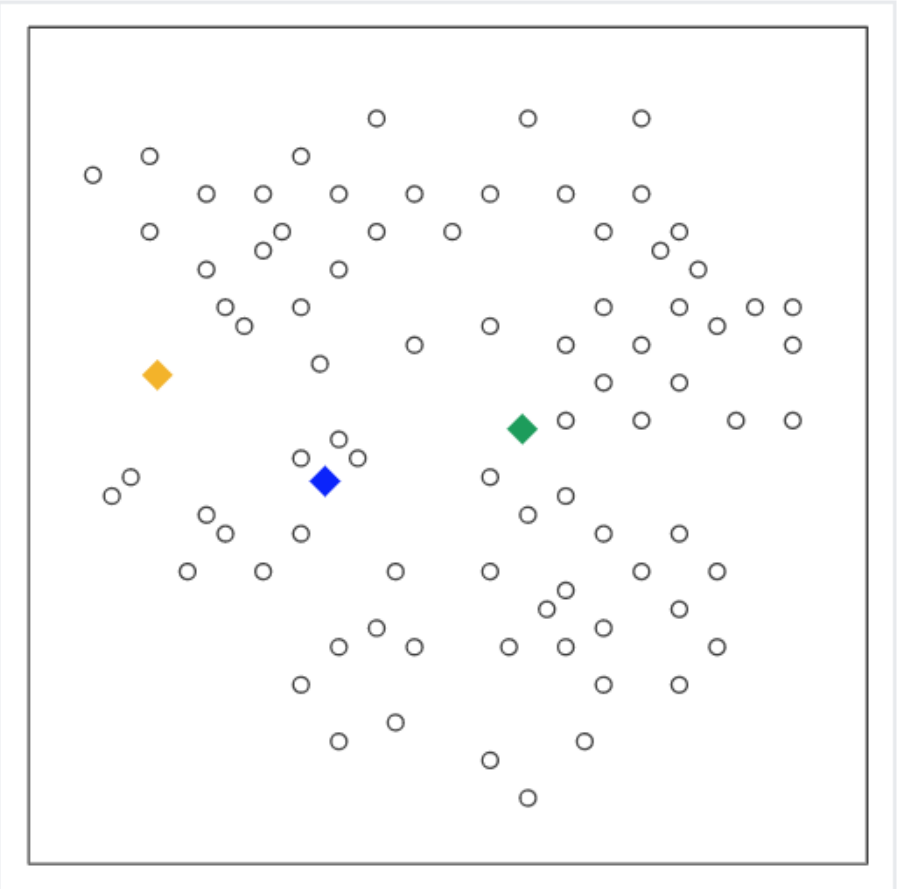
k-means

Week 10 Unsupervised learning/k-means Clustering Algorithm

University of Salford MANCHESTER

Step One

- The algorithm randomly chooses a centroid for each cluster. In our example, we choose a k of 3, and therefore the algorithm randomly picks 3 centroids.



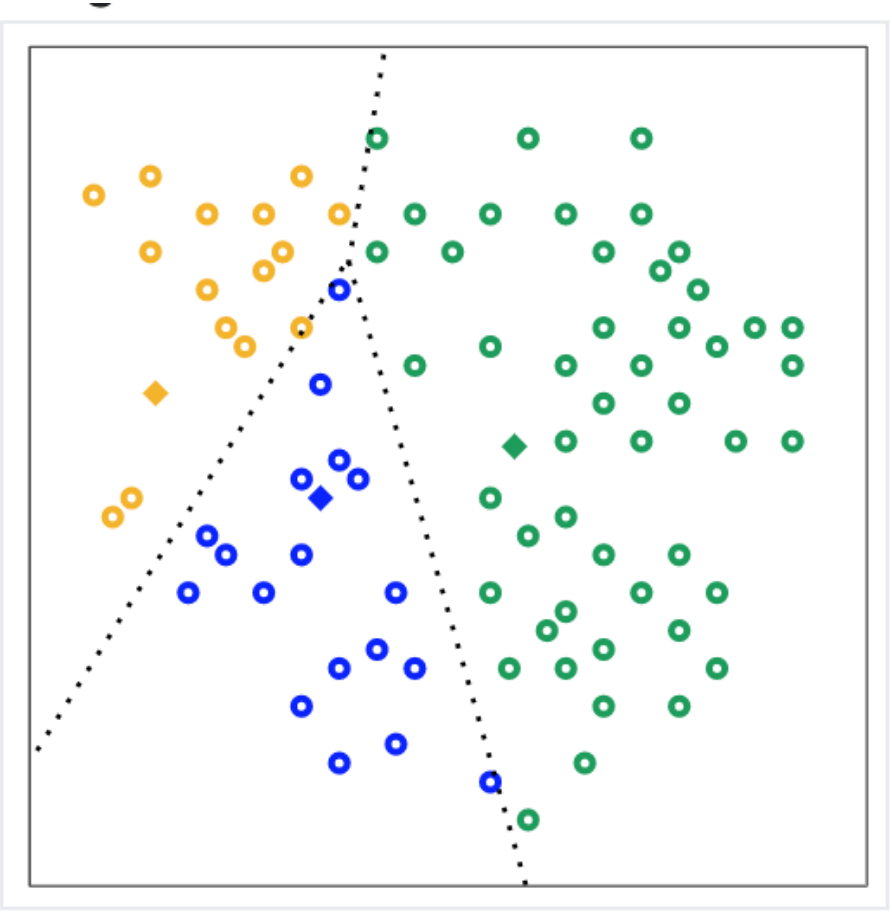
Week 10 Unsupervised learning/k-means Clustering Algorithm

University of Salford MANCHESTER

Step Two

- The algorithm assigns each point to the closest centroid to get k initial clusters.

$$z_i \leftarrow \arg\min_{k=1,\ldots,K} \|\phi(x_i) - \mu_k\|^2$$



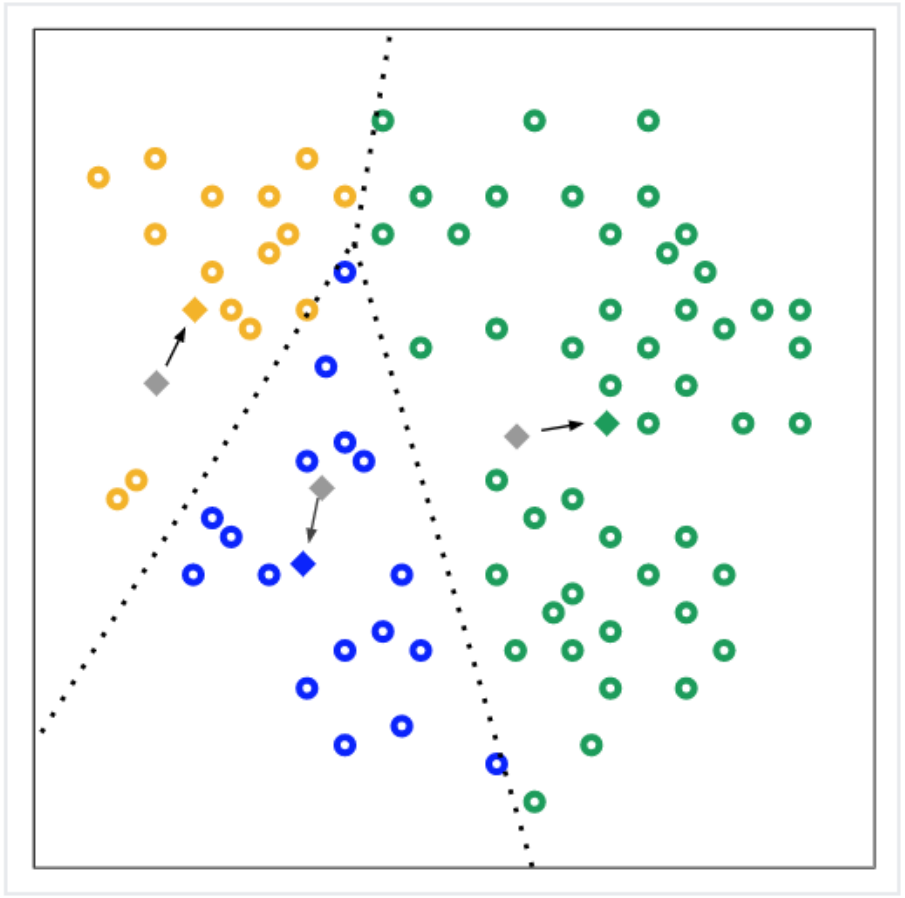
Week 10 Unsupervised learning/k-means Clustering Algorithm

University of Salford MANCHESTER

Step Three

- For every cluster, the algorithm recomputes the centroid by taking the average of all points in the cluster. The changes in centroids are shown in Figure by arrows. Since the centroids change, the algorithm then re-assigns the points to the closest centroid.

$$\mu_k \leftarrow \frac{1}{|\{i : z_i = k\}|} \sum_{i:z_i=k} \phi(x_i)$$

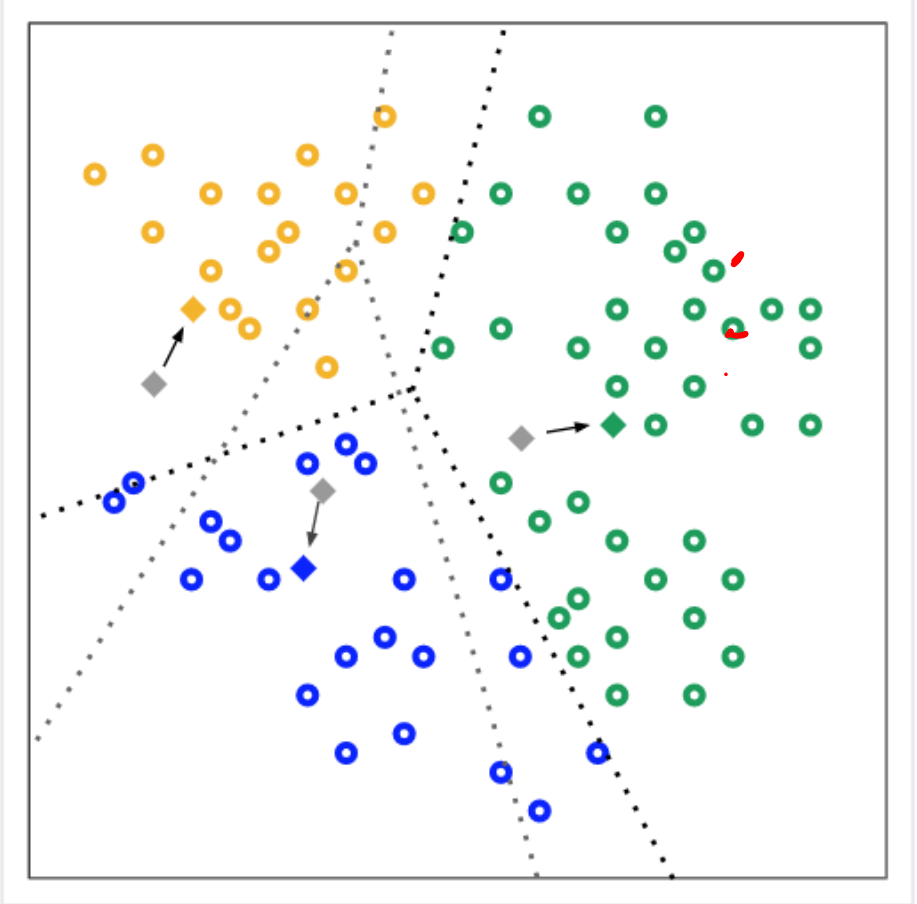


Week 10 Unsupervised learning/k-means Clustering Algorithm

University of Salford MANCHESTER

Step Four

- The algorithm repeats the calculation of centroids and assignment of points until points stop changing clusters. When clustering large datasets, you stop the algorithm before reaching convergence, using other criteria instead.



Week 10 Unsupervised learning/k-means Clustering Algorithm

University of Salford MANCHESTER

Algorithm: K-means

Initialize $\mu = [\mu_1, \ldots, \mu_K]$ randomly.

For $t = 1, \ldots, T$:

Step 1: set assignments z given μ

For each point $i = 1, \ldots, n$:

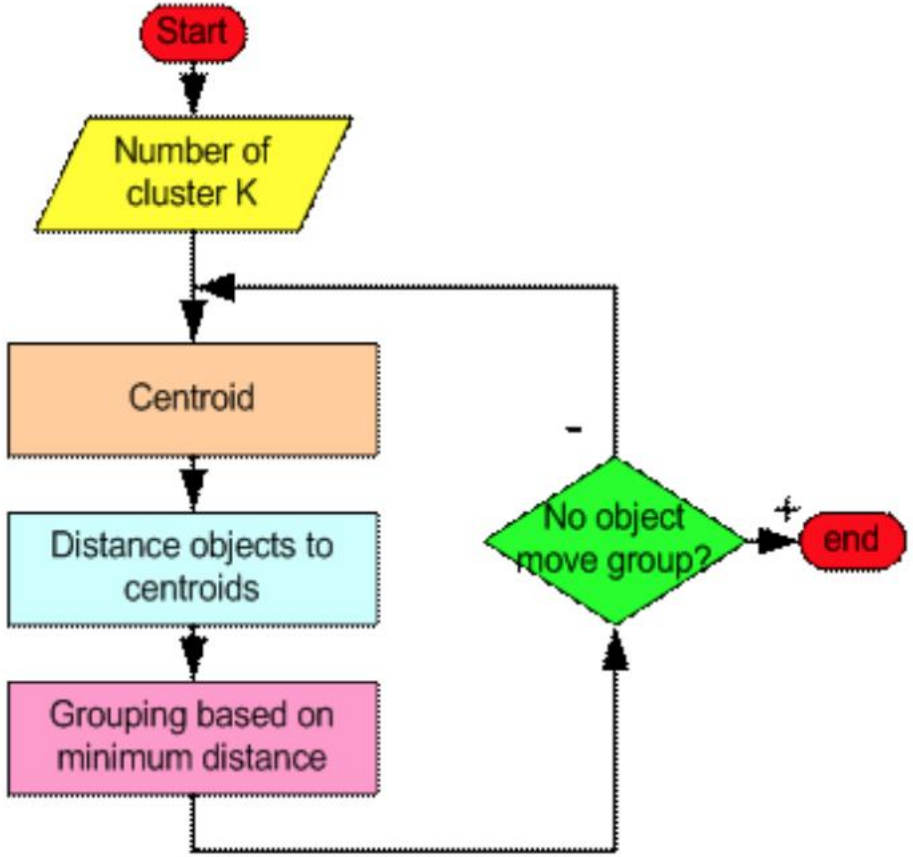
$$z_i \leftarrow \arg\min_{k=1,\ldots,K} \|\phi(x_i) - \mu_k\|^2$$

Step 2: set centroids μ given z

For each cluster $k = 1, \ldots, K$:

$$\mu_k \leftarrow \frac{1}{|\{i : z_i = k\}|} \sum_{i:z_i=k} \phi(x_i)$$

Week 10 Unsupervised learning/k-means Clustering Algorithm



Week 10 Unsupervised learning/k-means Code

University of Salford MANCHESTER

]:
```
# loads and displays our summary statistics of our data
multishapes = pd.read_csv("./data/multishapes.csv")
ms_df = multishapes[['x','y']]
ms_df.describe()
```

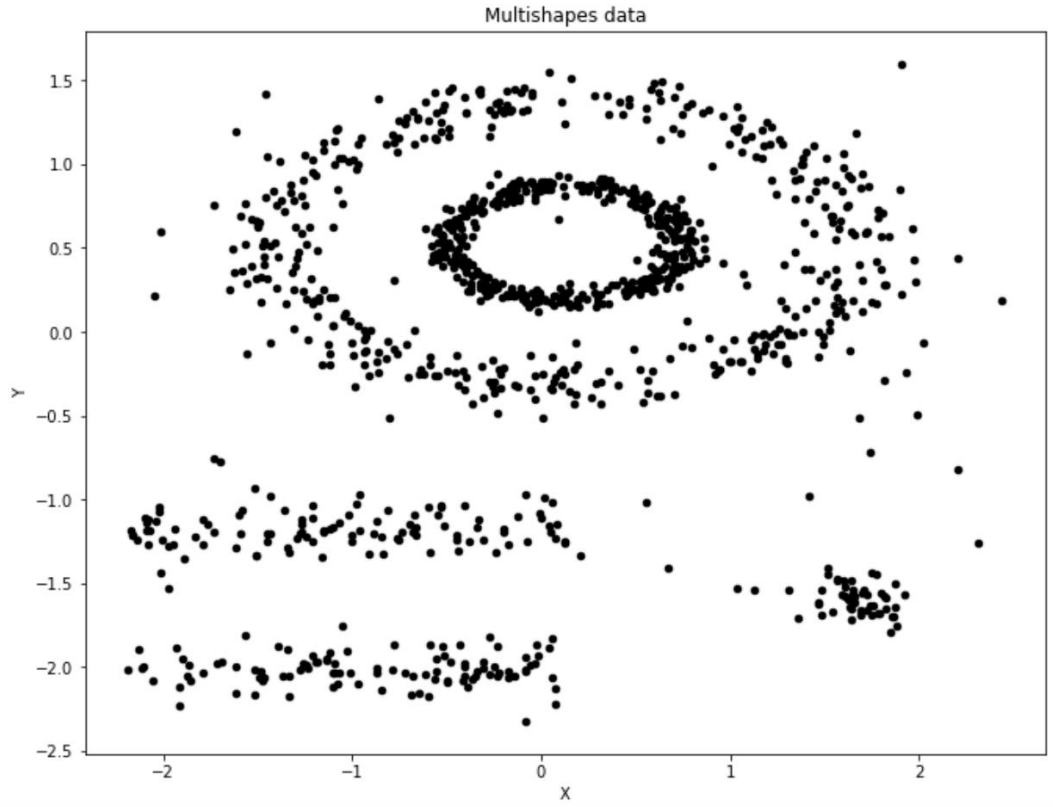
]:

	x	y
count	1100.000000	1100.000000
mean	-0.081222	-0.625431
std	0.644967	1.176170
min	-1.489180	-3.353462
25%	-0.478839	-1.126752
50%	-0.132920	-0.297040
75%	0.366072	0.250817
max	1.492208	1.253874

]:
```
# scales our data
scaled_df = pd.DataFrame(preprocessing.scale(ms_df), index=multishapes['shape'], columns = ms_df.columns)
scaled_df.describe()
```

Week 10 Unsupervised learning/k-means Code

```
# plots our data
msplot = scaled_df.plot.scatter(x='x',y='y',c='Black',title="Multishapes data",figsize=(11,8.5))
msplot.set_xlabel("X")
msplot.set_ylabel("Y")
plt.show()
```



Week 10 Unsupervised learning/k-means Code

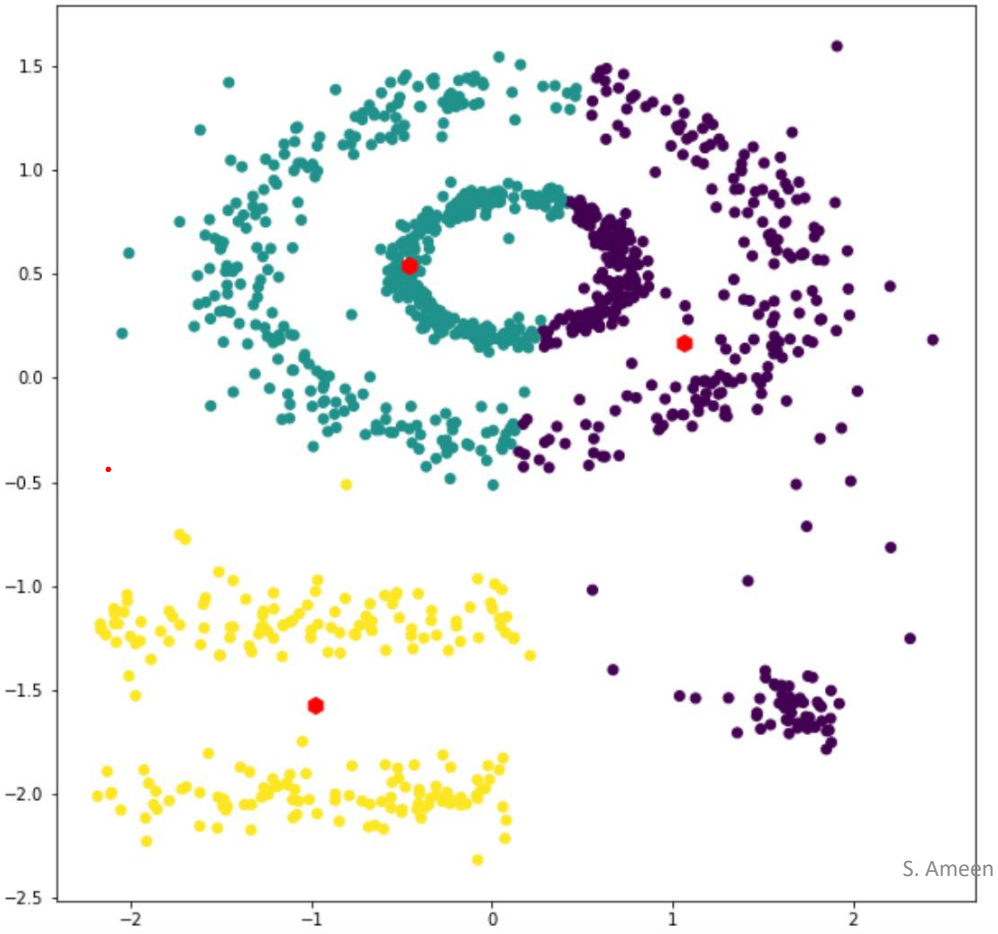
University of Salford MANCHESTER

```
from sklearn.cluster import KMeans
ms_kmeans = KMeans(n_clusters=3, init='random', n_init=3, random_state=109).fit(scaled_df)
```

Week 10 Unsupervised learning/k-means Code

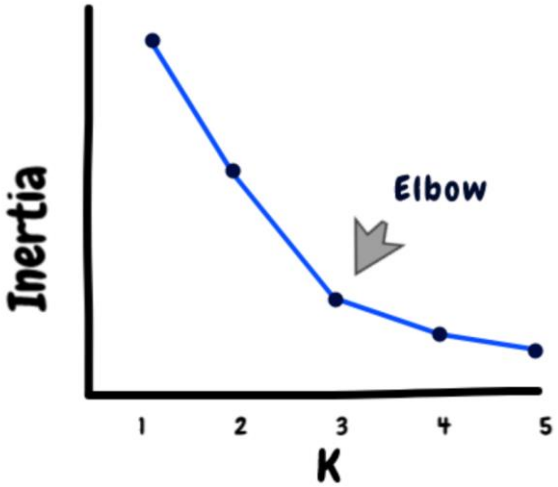
University of Salford MANCHESTER

```
plt.figure(figsize=(10,10))
plt.scatter(scaled_df['x'],scaled_df['y'], c=ms_kmeans.labels_);
plt.scatter(ms_kmeans.cluster_centers_[:,0],ms_kmeans.cluster_centers_[:,1], c='r', marker='h', s=100);
```



Week 10 Unsupervised learning/k-means Evaluating the cluster quality

1. Inertia: Intuitively, inertia tells how far away the points within a cluster are. Therefore, a small of inertia is aimed for. The range of inertia's value starts from zero and goes up.



2. Silhouette score: Silhouette score tells how far away the datapoints in one cluster are, from the datapoints in another cluster. The range of silhouette score is from -1 to 1. Score should be closer to 1 than -1.

Week 10 Unsupervised learning/k-means Evaluating the cluster quality

University of Salford MANCHESTER

1. Inertia:

```
wss = []
inti_K = 1
End_K = 13
for i in range(inti_K,End_K):
    fitx = KMeans(n_clusters=i, init='random', n_init=5, random_state=109).fit(scaled_df)
    wss.append(fitx.inertia_)
```

```
plt.figure(figsize=(11,8.5))
plt.plot(range(inti_K,End_K), wss, 'bx-')
plt.xlabel('Number of clusters $k$')
plt.ylabel('Inertia')
plt.title('The Elbow Method showing the optimal $k$')
plt.show()
```

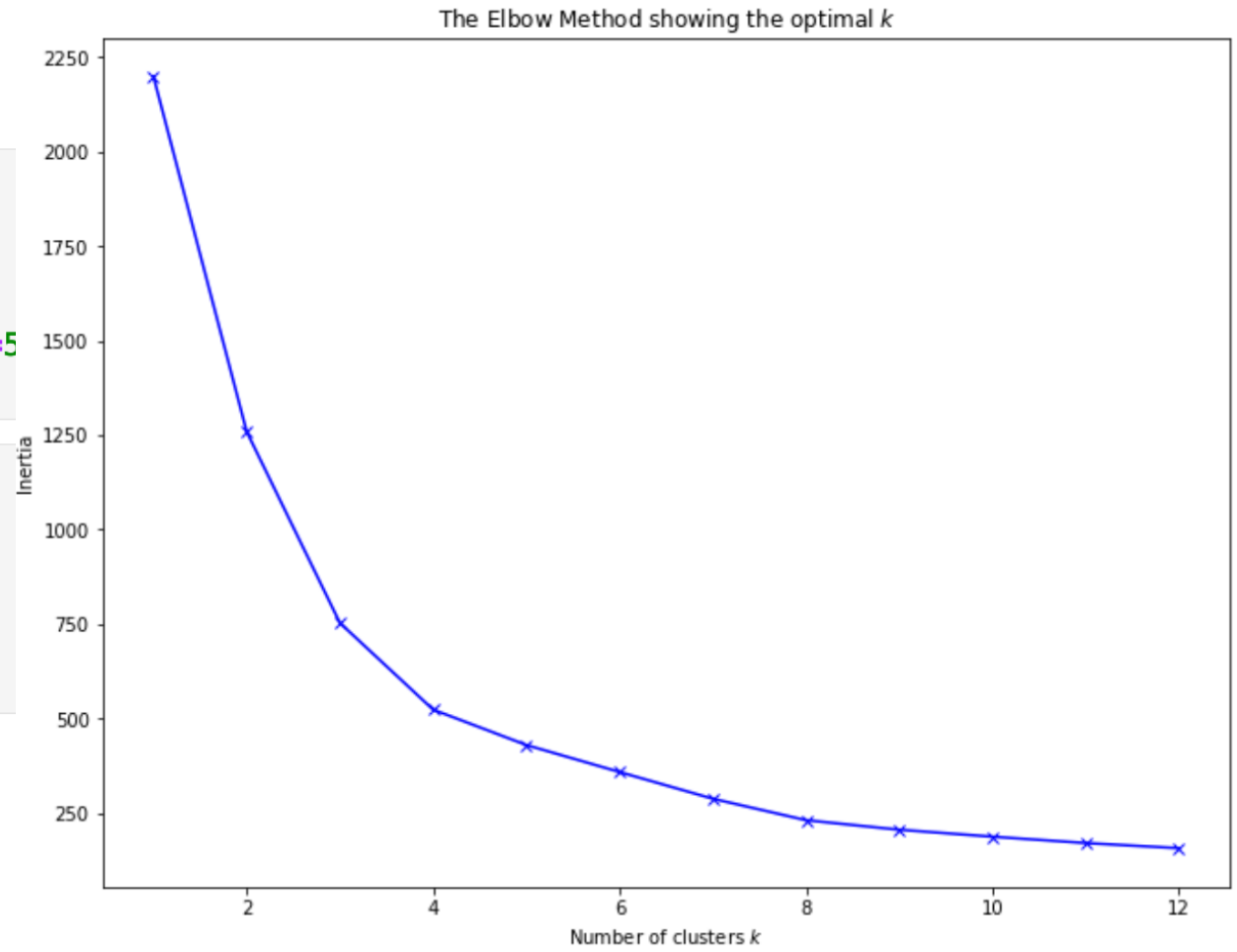
Week 10 Unsupervised learning/k-means Evaluating the cluster quality

University of Salford MANCHESTER

1. Inertia:

```
wss = []
inti_K = 1
End_K = 13
for i in range(inti_K,End_K):
    fitx = KMeans(n_clusters=i, init='random', n_init=5
    wss.append(fitx.inertia_)
```

```
plt.figure(figsize=(11,8.5))
plt.plot(range(inti_K,End_K), wss, 'bx-')
plt.xlabel('Number of clusters $k$')
plt.ylabel('Inertia')
plt.title('The Elbow Method showing the optimal $k$')
plt.show()
```

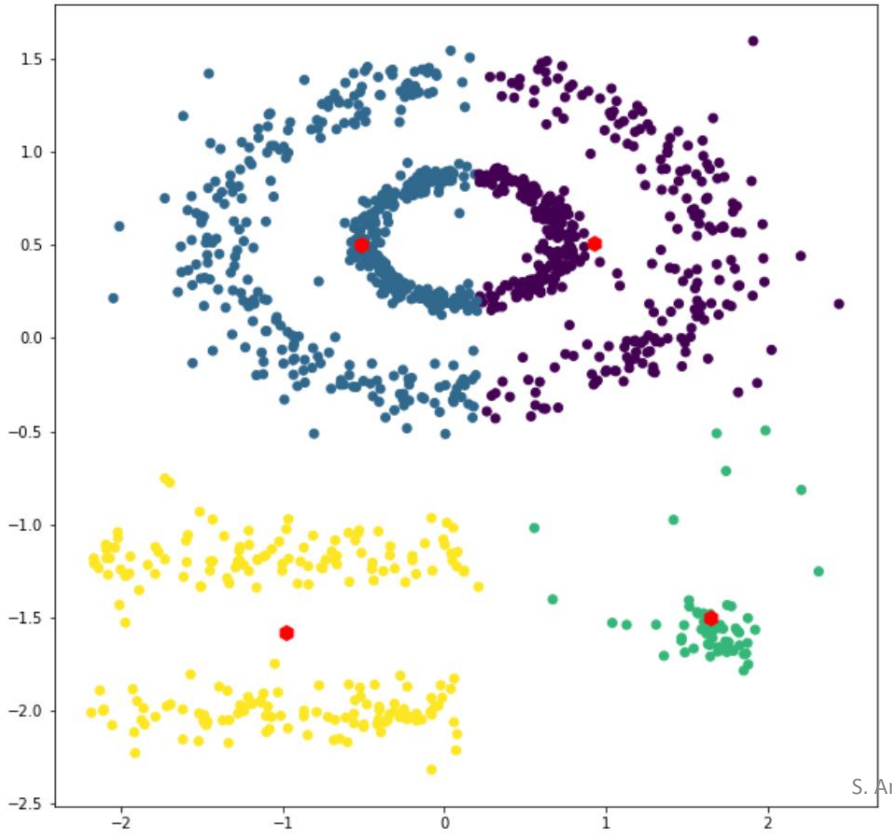


Week 10 Unsupervised learning/k-means Evaluating the cluster quality

University of Salford MANCHESTER

```
# from sklearn.cluster import KMeans
ms_kmeans = KMeans(n_clusters=4, init='random', n_init=3, random_state=109).fit(scaled_df)
```

```
plt.figure(figsize=(10,10))
plt.scatter(scaled_df['x'],scaled_df['y'], c=ms_kmeans.labels_);
plt.scatter(ms_kmeans.cluster_centers_[:,0],ms_kmeans.cluster_centers_[:,1], c='r', marker='h', s=100);
```



Week 10 Unsupervised learning/k-means Evaluating the cluster quality

University of Salford MANCHESTER

```
from kneed import KneeLocator
kl = KneeLocator(
    range(inti_K, End_K), wss, curve="convex", direction="decreasing")
kl.elbow
```

```
4
```

Week 10 Unsupervised learning/k-means Evaluating the cluster quality

University of Salford MANCHESTER

2. Silhouette

```
from sklearn.metrics import silhouette_score
fitx = KMeans(n_clusters=4, init='random', n_init=5, random_state=109).fit(scaled_df)
silhouette_score(scaled_df, fitx.labels_)
```

```
0.45880539445085916
```

-1 - indicates incorrect clustering 0 - highly overlapping clusters 1 - dense well-separated clusters

Week 10 Unsupervised learning/k-means Evaluating the cluster quality

University of Salford MANCHESTER

2. Silhouette

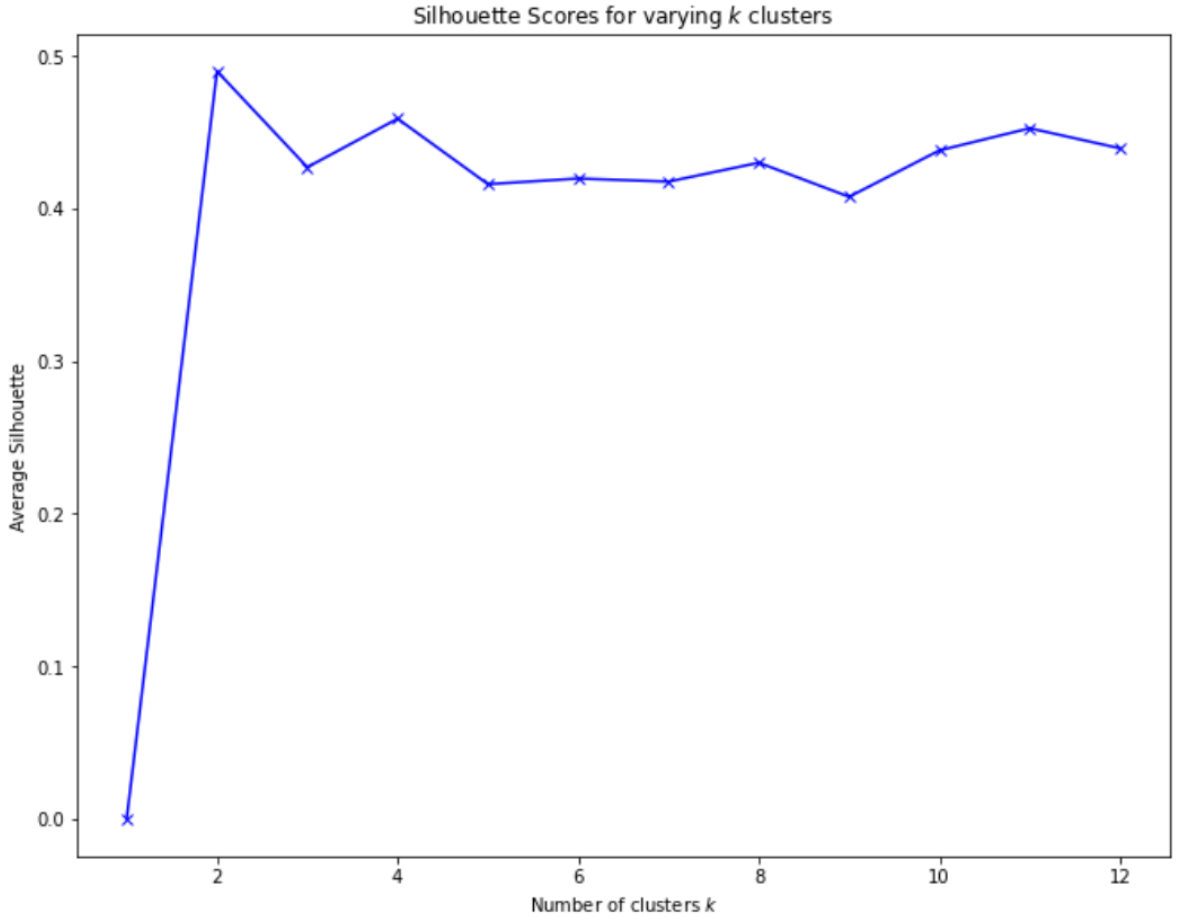
```
# Check with more k
from sklearn.metrics import silhouette_score
inti_K = 1
End_K = 13
scores = [0]
for i in range(inti_K+1,End_K):
    fitx = KMeans(n_clusters=i, init='random', n_init=5, random_state=109).fit(scaled_df)
    score = silhouette_score(scaled_df, fitx.labels_)
    scores.append(score)
```

```
plt.figure(figsize=(11,8.5))
plt.plot(range(inti_K,End_K), np.array(scores), 'bx-')
plt.xlabel('Number of clusters $k$')
plt.ylabel('Average Silhouette')
plt.title('Silhouette Scores for varying $k$ clusters')
plt.show()
```

Week 10 Unsupervised learning/k-means Evaluating the cluster quality

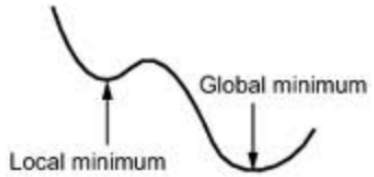
University of Salford MANCHESTER

2. Silhouette



Week 10 Unsupervised learning/k-means/Local minima

K-means is guaranteed to converge to a local minimum, but is not guaranteed to find the global minimum.



[demo: getting stuck in local optima, seed = 100]

Solutions:

- Run multiple times from different random initializations
- Initialize with a heuristic (K-means++)

Week 10 Unsupervised learning/k-means/Local minima

University of Salford MANCHESTER

```
fitx = KMeans(n_clusters=4, n_init=5, random_state=109).fit(scaled_df)
```

sklearn.cluster.KMeans

class sklearn.cluster.**KMeans**(*n_clusters=8*, ***, *init='k-means++'*, *n_init=10*, *max_iter=300*, *tol=0.0001*, *precompute_distances='deprecated'*, *verbose=0*, *random_state=None*, *copy_x=True*, *n_jobs='deprecated'*, *algorithm='auto'*) [source]

K-Means clustering.

Read more in the User Guide.

Parameters:

n_clusters : *int, default=8*
The number of clusters to form as well as the number of centroids to generate.

init : *{'k-means++', 'random'}, callable or array-like of shape (n_clusters, n_features), default='k-means++'*
Method for initialization:

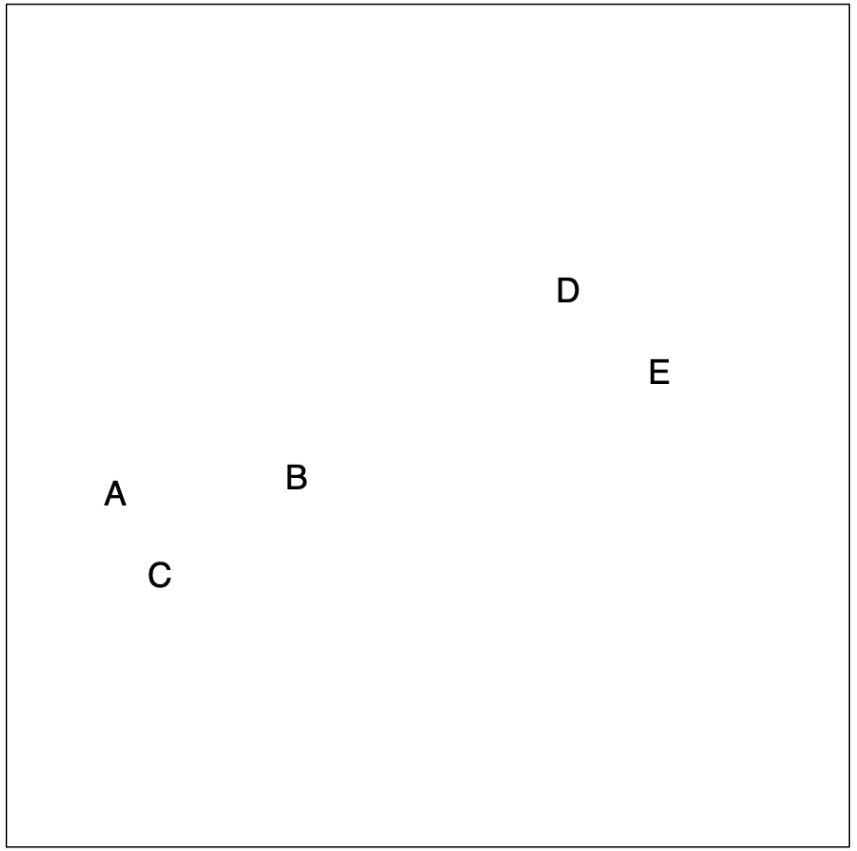
Week 10 Unsupervised learning / Types of Clustering

University of Salford MANCHESTER

Agglomerative

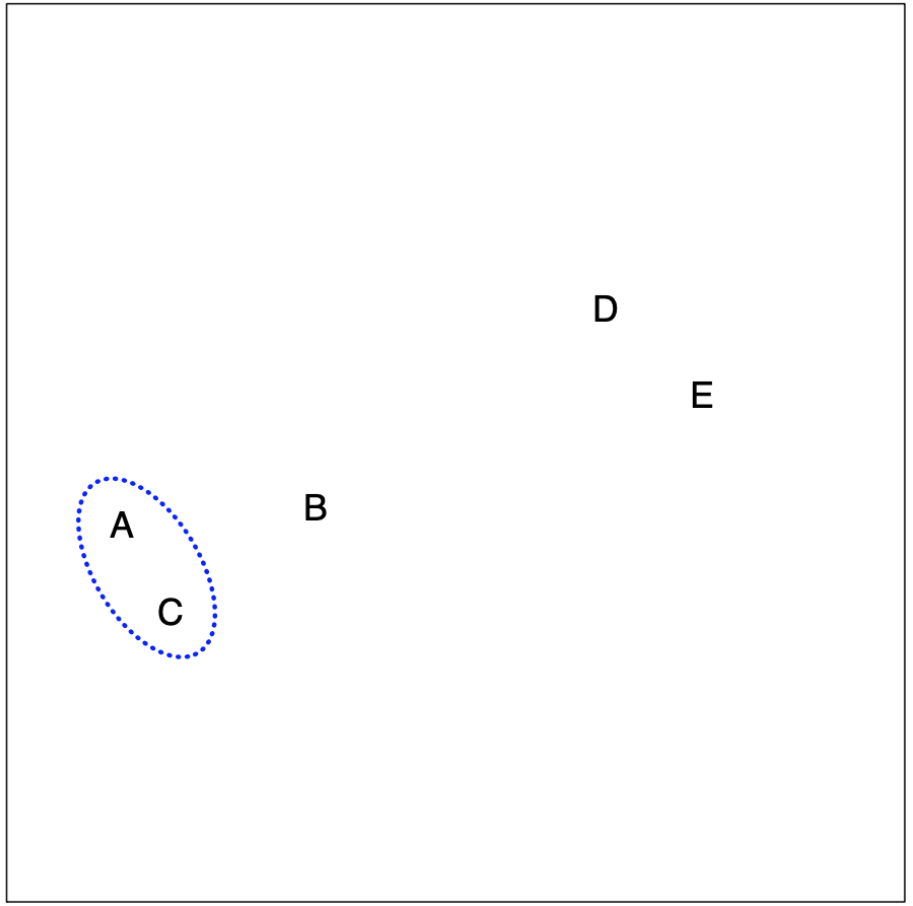
Week 10 Hierarchical clustering - agglomerative

- Each point starts as its own cluster



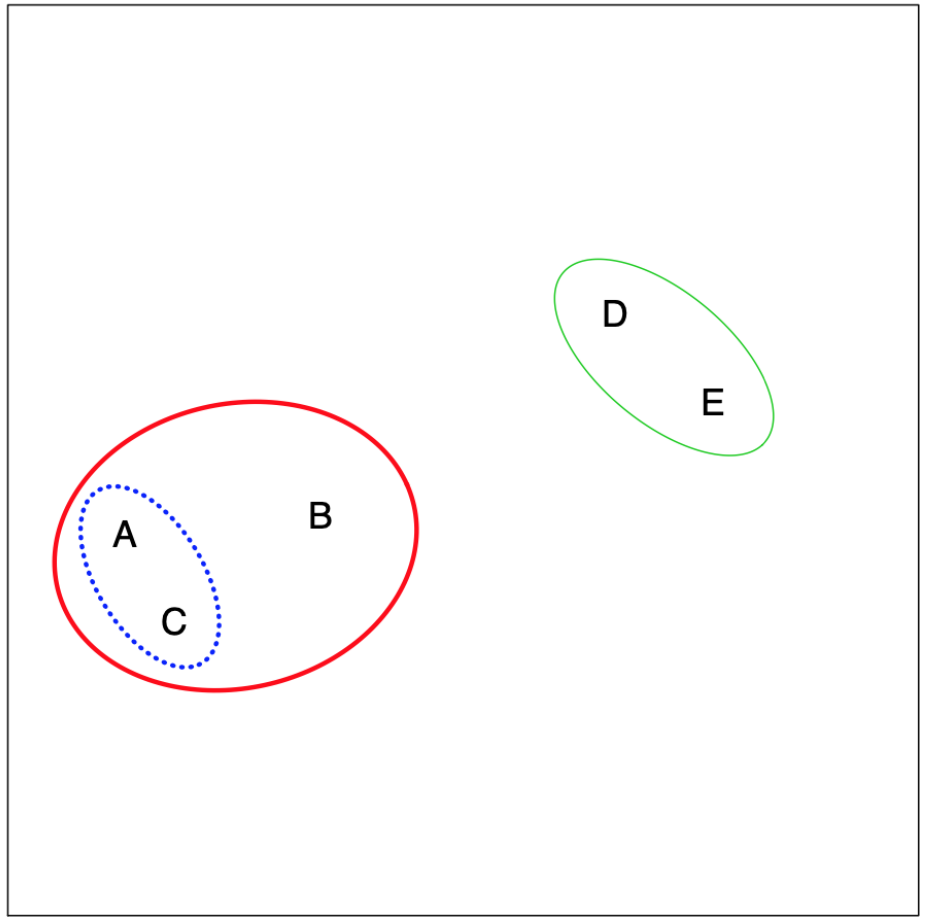
Week 10 Hierarchical clustering - agglomerative

- Each point starts as its own cluster
- We merge the two clusters (points) that are closet to each other



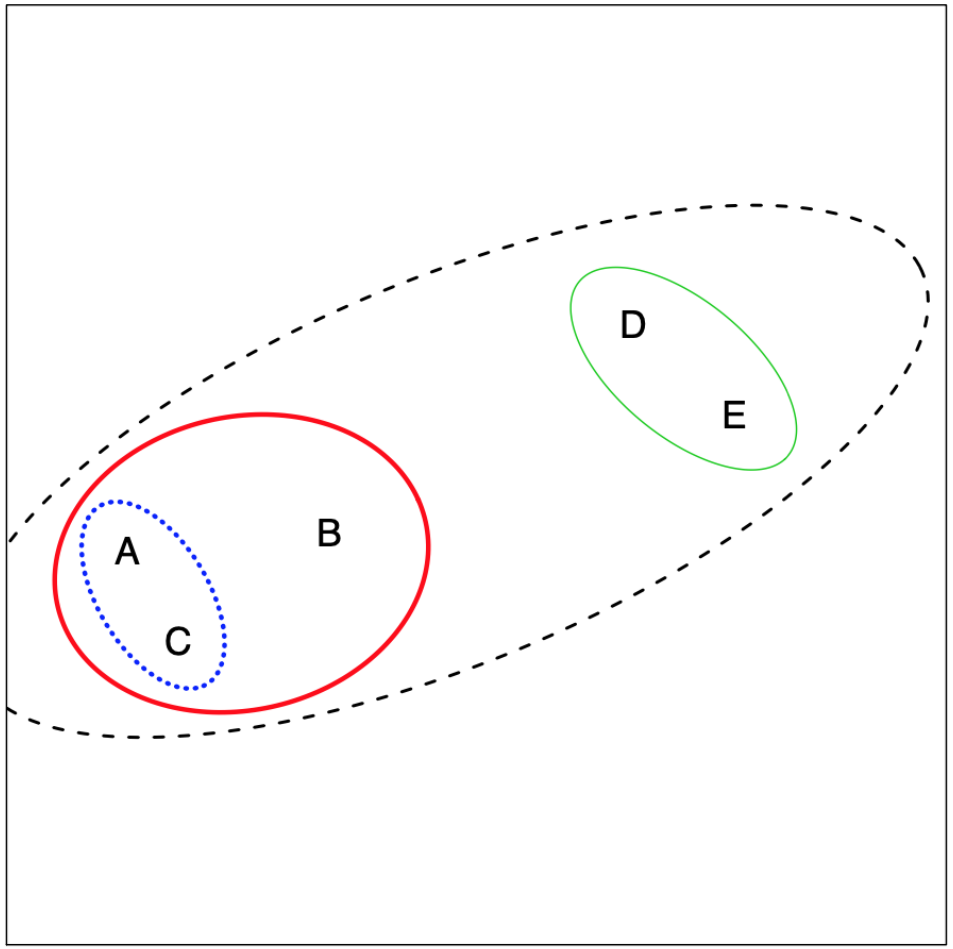
Week 10 Hierarchical clustering - agglomerative

- Each point starts as its own cluster
- We merge the two clusters (points) that are closet to each other
- Then we merge the next two closest clusters
- Then the next two closest clusters...



Week 10 Hierarchical clustering - agglomerative

- Each point starts as its own cluster
- We merge the two clusters (points) that are closet to each other
- Then we merge the next two closest clusters
- Then the next two closest clusters...
- Until at last all of the points are all in a single cluster



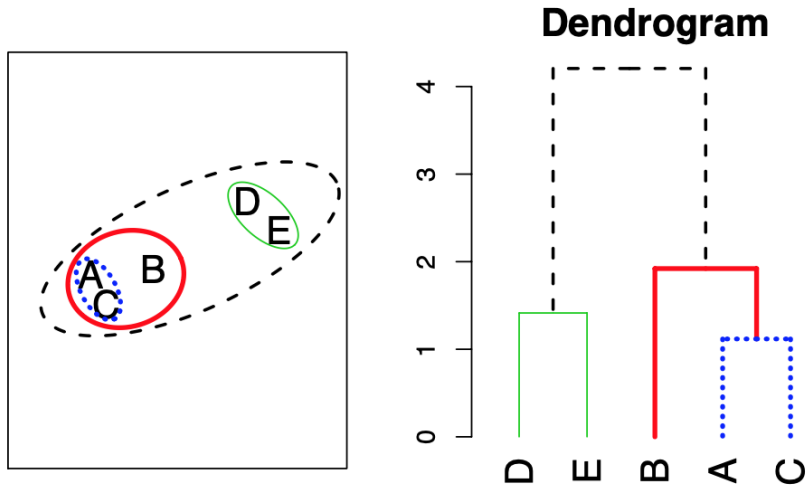
Week 10 Hierarchical clustering - agglomerative

University of Salford MANCHESTER

Agglomerative Hierarchical Clustering

- Start with each point in its own cluster.
- Identify the two closest clusters. Merge them.
- Repeat until all points are in a single cluster

To visualize the results, we can look at the resulting dendrogram



y-axis on dendrogram is (proportional to) the distance between the clusters that got merged at that step

Week 10 Hierarchical clustering – agglomerative - Code

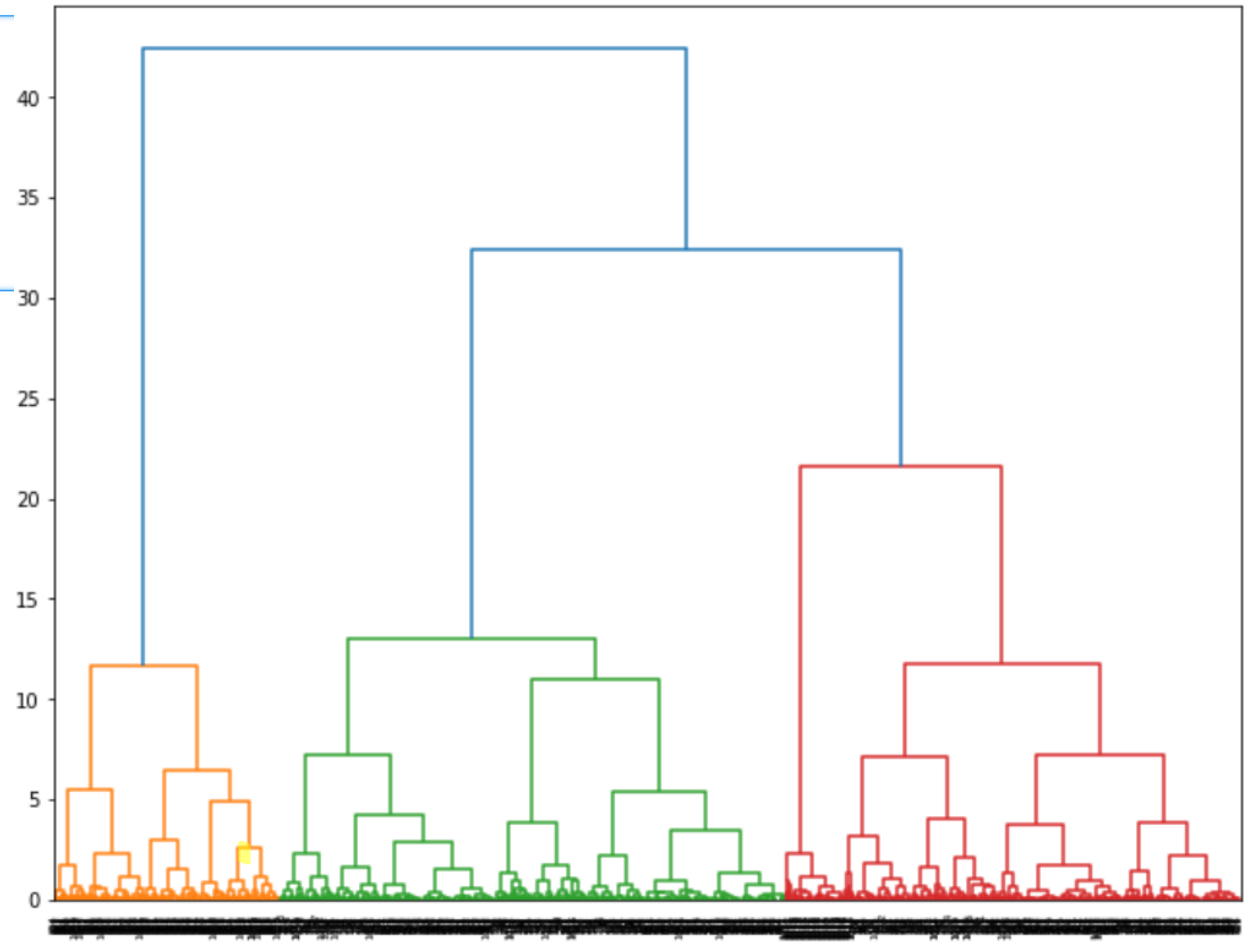
University of Salford MANCHESTER

```
import scipy.cluster.hierarchy as hac
from scipy.spatial.distance import pdist

plt.figure(figsize=(11,8.5))
dist_mat = pdist(scaled_df, metric="euclidean")
ward_data = hac.ward(dist_mat)
hac.dendrogram(ward_data);
```

Week 10 Hierarchical clustering – agglomerative - Code


```
import scipy.cluster.hierarchy as hac
from scipy.spatial.distance import pdist

plt.figure(figsize=(11,8.5))
dist_mat = pdist(scaled_df, metric="euclidean")
ward_data = hac.ward(dist_mat)
hac.dendrogram(ward_data);
```

Ward's minimum variance method



Week 10 Hierarchical clustering – agglomerative - Code

University of Salford MANCHESTER

3b.2 Quality of Clusters: Silhouette

```
labellings = hac.fcluster(ward_data, t=25, criterion='distance')
silhouette_score(scaled_df, labellings)
```

```
0.4182759392486782
```

Week 10 Hierarchical clustering – agglomerative - Code

University of Salford MANCHESTER

3n.3 Code (via `sklearn`):

```
from sklearn.cluster import AgglomerativeClustering
clustering = AgglomerativeClustering().fit(scaled_df)
clustering.labels_
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
silhouette_score(scaled_df, clustering.labels_)
```

```
0.47265149302914106
```

Week 10 Hierarchical clustering – agglomerative - Linkages

University of Salford MANCHESTER

Common linkage types

Linkage	*Description*
Complete	Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities.
Average	Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.

Week 10 Hierarchical clustering – agglomerative - Linkages

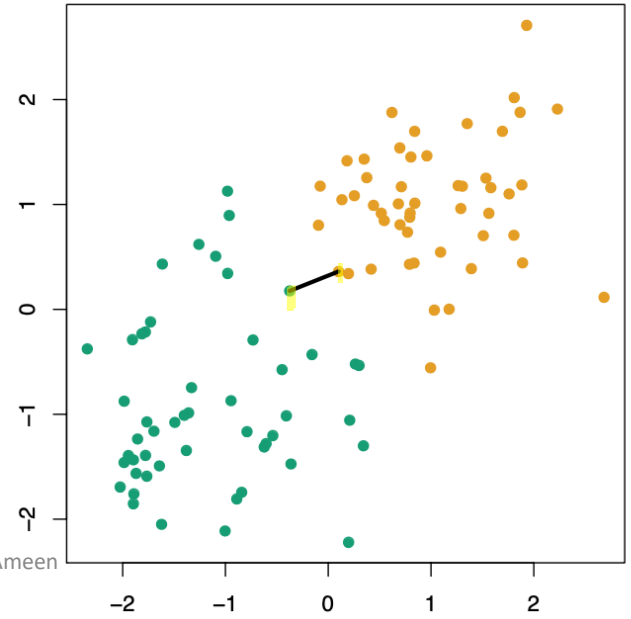
University of Salford MANCHESTER

Single linkage

In single linkage (i.e., nearest-neighbor linkage), the dissimilarity between G, H is the smallest dissimilarity between two points in different groups:

$$d_{\text{single}}(G, H) = \min_{i \in G,\, j \in H} d(x_i, x_j)$$

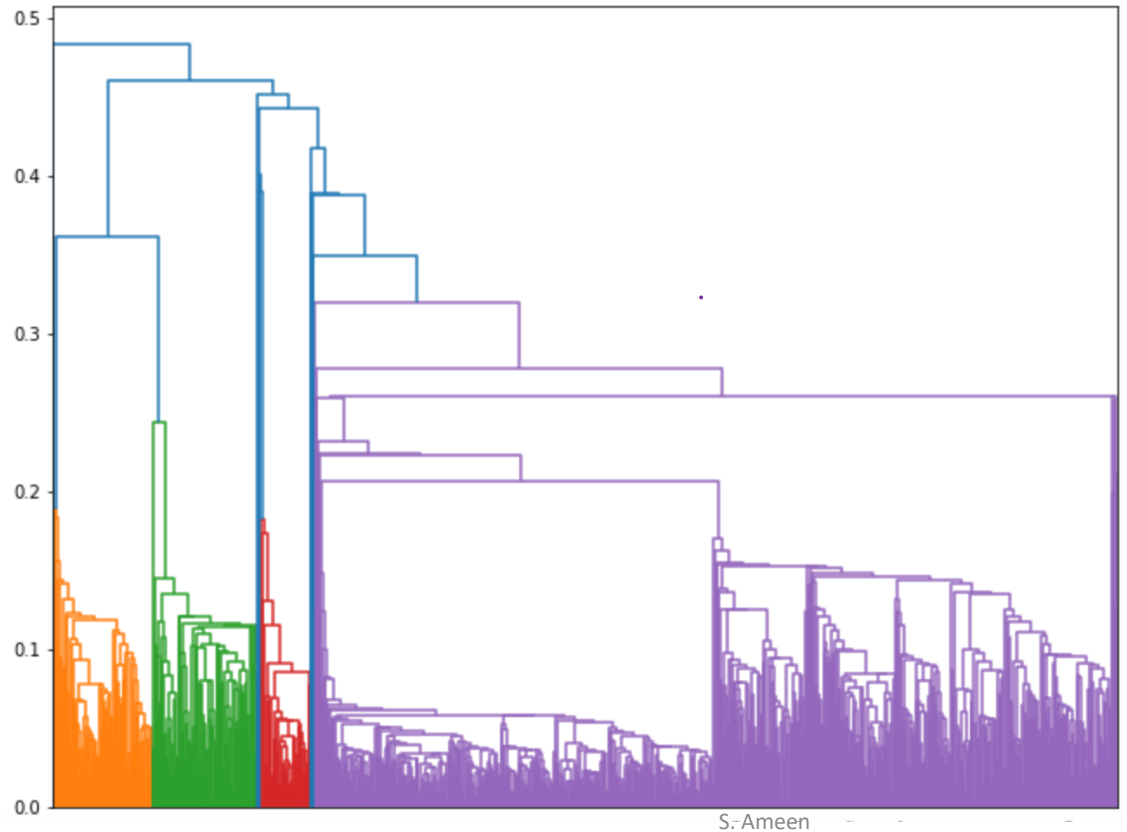
Example (dissimilarities d_{ij} are distances, groups are marked by colors): single linkage score $d_{\text{single}}(G, H)$ is the distance of the closest pair



Week 10 Hierarchical clustering – agglomerative - Linkages

University of Salford MANCHESTER

```
# Single
plt.figure(figsize=(11,8.5))
dist_mat = pdist(scaled_df, metric="euclidean")
ward_data = hac.single(dist_mat)
hac.dendrogram(ward_data);
```



Week 10 Hierarchical clustering – agglomerative - Linkages

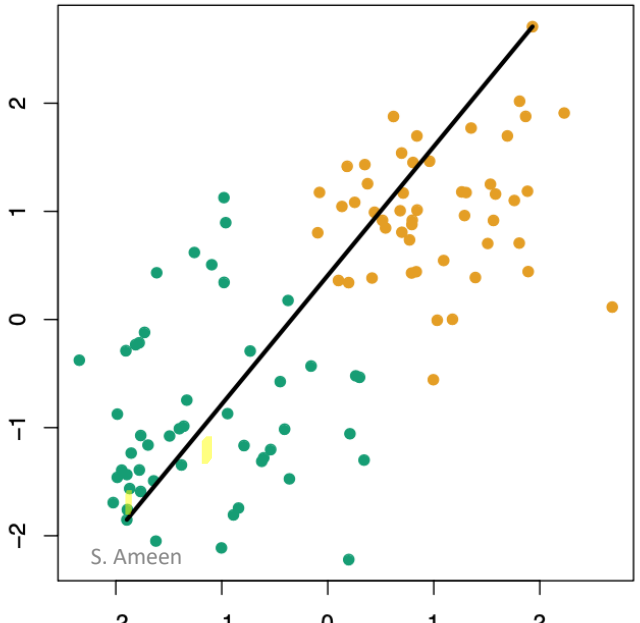
University of Salford MANCHESTER

Complete linkage

In complete linkage (i.e., furthest-neighbor linkage), dissimilarity between G, H is the largest dissimilarity between two points in different groups:

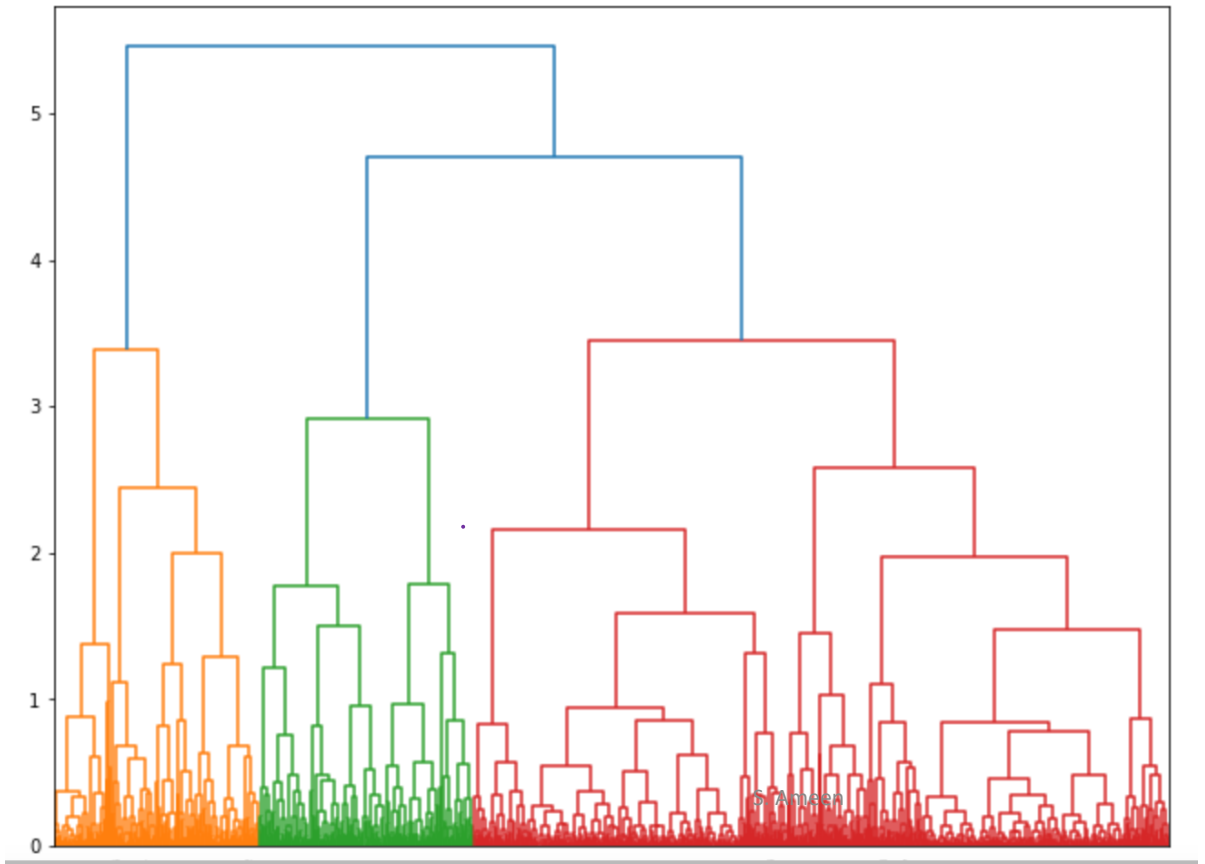
$$d_{\text{complete}}(G, H) = \max_{i \in G,\, j \in H} d(x_i, x_j)$$

Example (dissimilarities d_{ij} are distances, groups are marked by colors): complete linkage score $d_{\text{complete}}(G, H)$ is the distance of the furthest pair



Week 10 Hierarchical clustering – agglomerative - Linkages

```
# complete
plt.figure(figsize=(11,8.5))
dist_mat = pdist(scaled_df, metric="euclidean")
ward_data = hac.complete(dist_mat)
hac.dendrogram(ward_data);
```



Week 10 Hierarchical clustering – agglomerative - Linkages

University of Salford MANCHESTER

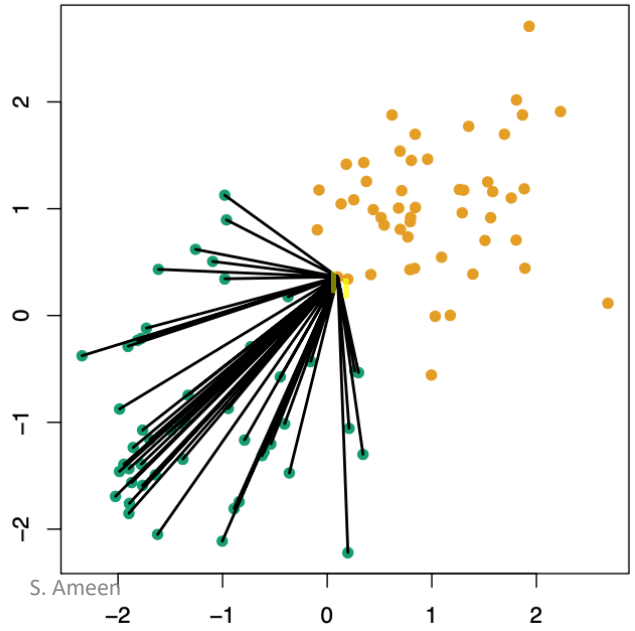
Average linkage

In average linkage, the dissimilarity between G, H is the average dissimilarity over all points in opposite groups:

$$d_{\text{average}}(G, H) = \frac{1}{|G| \cdot |H|} \sum_{i \in G,\, j \in H} d(x_i, x_j)$$

Example (dissimilarities d_{ij} are distances, groups are marked by colors): average linkage score $d_{\text{average}}(G, H)$ is the average distance across all pairs

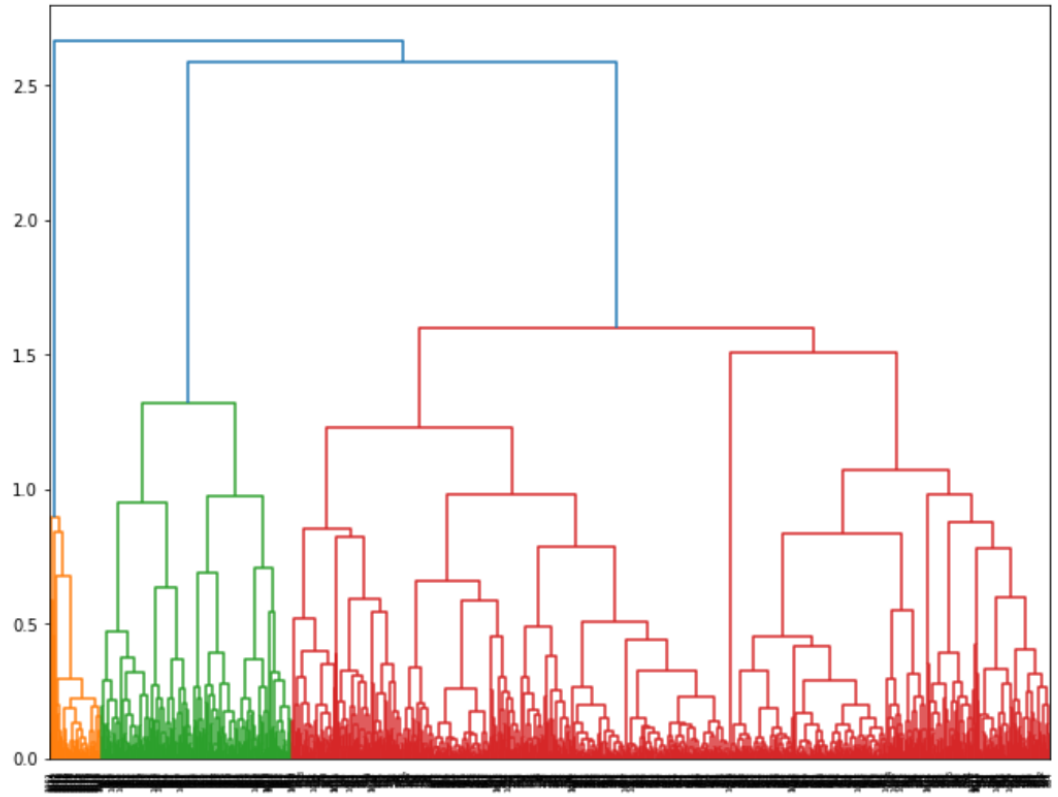
(Plot here only shows distances between the green points and one orange point)



Week 10 Hierarchical clustering – agglomerative - Linkages

University of Salford MANCHESTER

```
# average
plt.figure(figsize=(11,8.5))
dist_mat = pdist(scaled_df, metric="euclidean")
ward_data = hac.average(dist_mat)
hac.dendrogram(ward_data);
```



Week 10 Hierarchical clustering – agglomerative - Linkages

University of Salford MANCHESTER

Hierarchical clustering (scipy.cluster.hierarchy)

linkage(y[, method, metric, optimal_ordering])	Perform hierarchical/agglomerative clustering.
single(y)	Perform single/min/nearest linkage on the condensed distance matrix `y`.
complete(y)	Perform complete/max/farthest point linkage on a condensed distance matrix.
average(y)	Perform average/UPGMA linkage on a condensed distance matrix.
weighted(y)	Perform weighted/WPGMA linkage on the condensed distance matrix.
centroid(y)	Perform centroid/UPGMC linkage.
median(y)	Perform median/WPGMC linkage.
ward(y)	Perform Ward's linkage on a condensed distance matrix.

Week 10 Hierarchical clustering – agglomerative - Linkages

University of Salford MANCHESTER

sklearn.cluster.AgglomerativeClustering

class sklearn.cluster.**AgglomerativeClustering**(*n_clusters=2*, ***, *affinity='euclidean'*, *memory=None*, *connectivity=None*, *compute_full_tree='auto'*, *linkage='ward'*, *distance_threshold=None*, *compute_distances=False*) [source]

linkage : *{'ward', 'complete', 'average', 'single'}, default='ward'*

Which linkage criterion to use. The linkage criterion determines which distance to use between sets of observation. The algorithm will merge the pairs of cluster that minimize this criterion.

- 'ward' minimizes the variance of the clusters being merged.
- 'average' uses the average of the distances of each observation of the two sets.
- 'complete' or 'maximum' linkage uses the maximum distances between all observations of the two sets.
- 'single' uses the minimum of the distances between all observations of the two sets.

Week 10 Unsupervised learning / Types of Clustering

University of Salford MANCHESTER

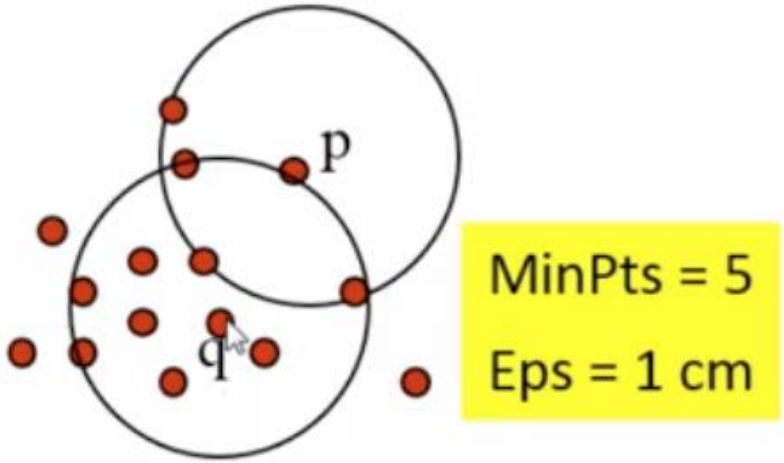
DBscan

Week 10 Hierarchical clustering - DBscan Clustering

University of Salford MANCHESTER

DBSCAN: A Density-Based Clustering Algorithm

- **Eps**: Maximum radius of the neighborhood
- **MinPts**: Minimum number of points in the **Eps** – neighborhood of a point.



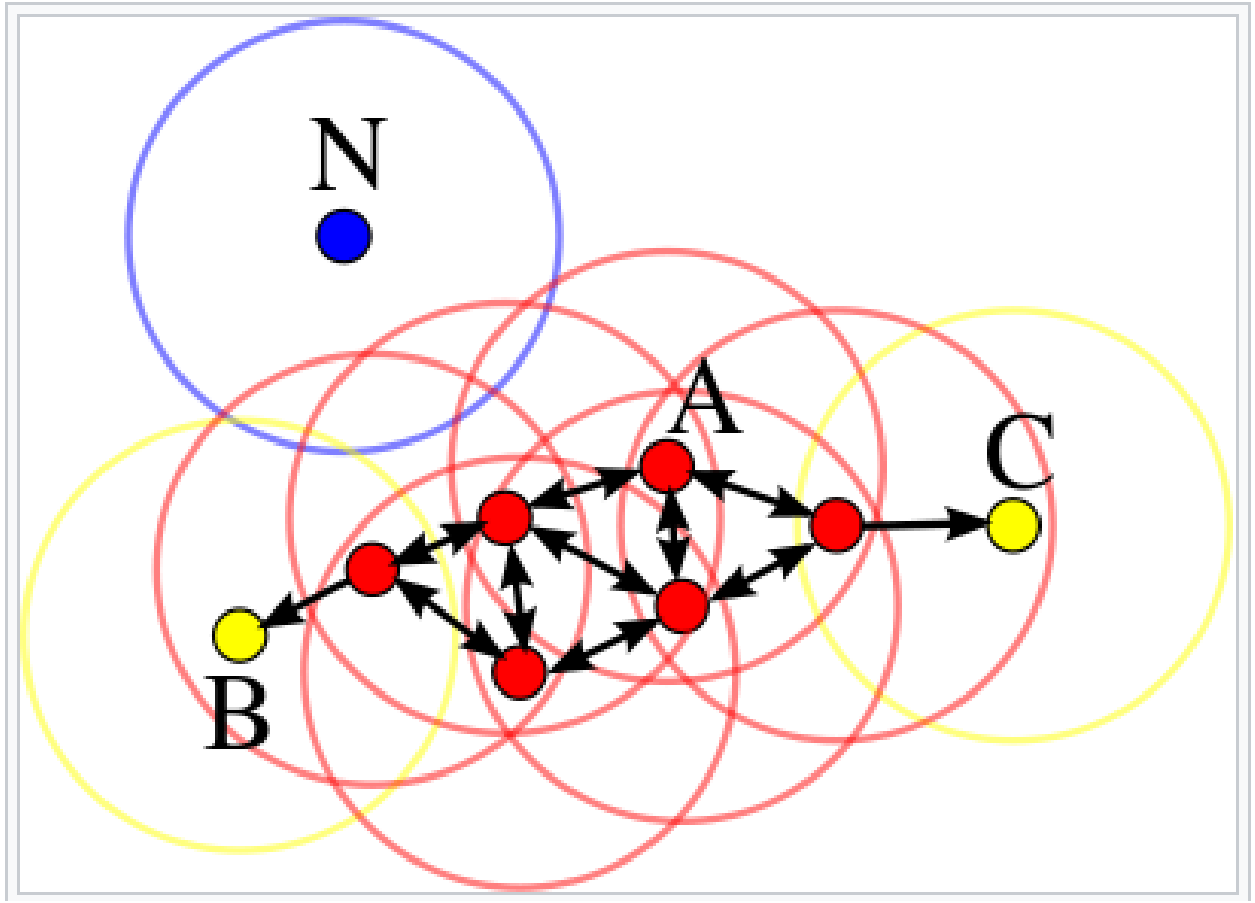
Week 10 Hierarchical clustering - DBscan Clustering

University of Salford MANCHESTER

DBSCAN: A Density-Based Clustering Algorithm

- **Core point:** A point is a core point if there are at least **minPts** number of points (including the point itself) in its surrounding area with radius **eps**.
- **Border point:** A point is a border point if it is reachable from a core point and there are less than **minPts** number of points within its surrounding area.
- **Outlier:** A point is an outlier if it is not a core point and not reachable from any core points.

Week 10 Hierarchical clustering - DBscan Clustering



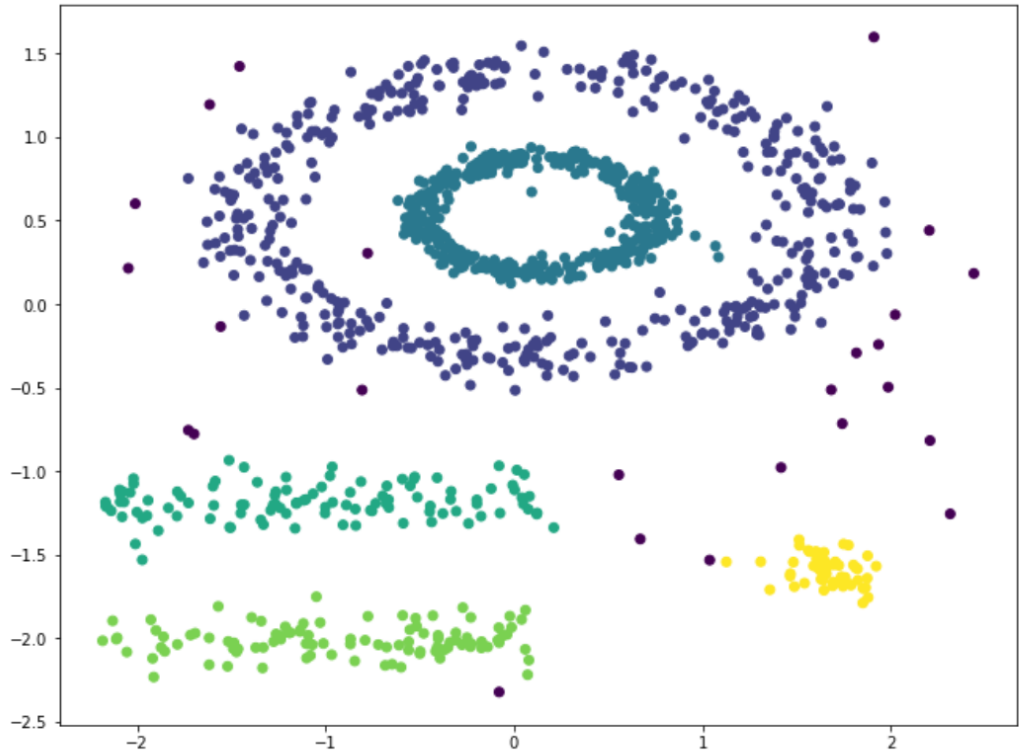
DBSCAN Clustering — Explained. Detailed theorotical explanation and… | by Soner Yıldırım | Towards Data Science

Week 10 Hierarchical clustering - DBscan - Code

University of Salford MANCHESTER

```
from sklearn.cluster import DBSCAN
fitted_dbscan = DBSCAN(eps=0.2).fit(scaled_df)
```

```
plt.figure(figsize=(11,8.5))
plt.scatter(scaled_df['x'],scaled_df['y'], c=fitted_dbscan.labels_);
```




```
silhouette_score(scaled_df, clustering.labels_)
```

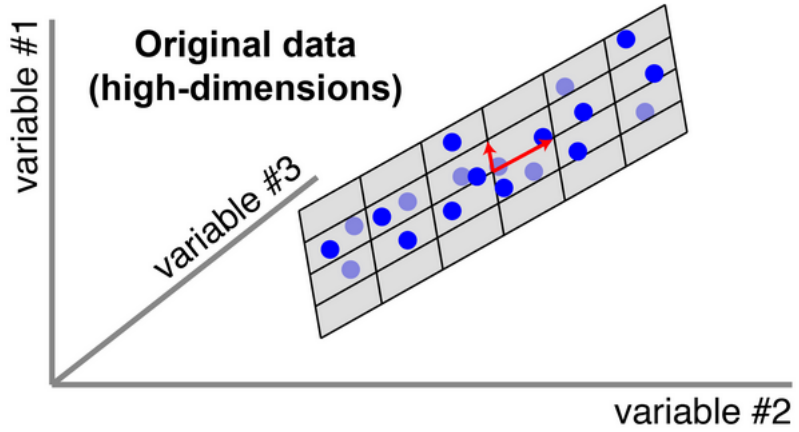
```
0.47265149302914106
```

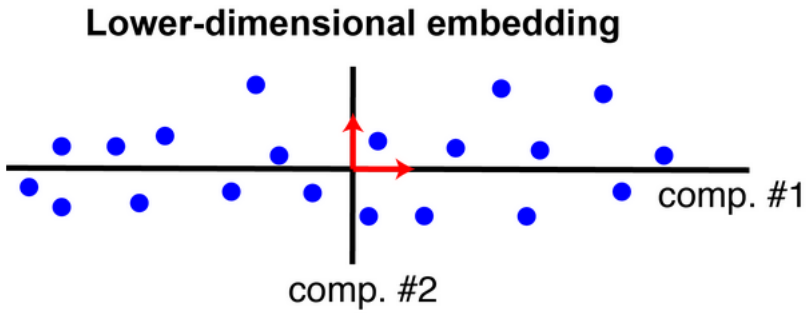
Week 10 – Revision
This Lecture:

University of Salford MANCHESTER

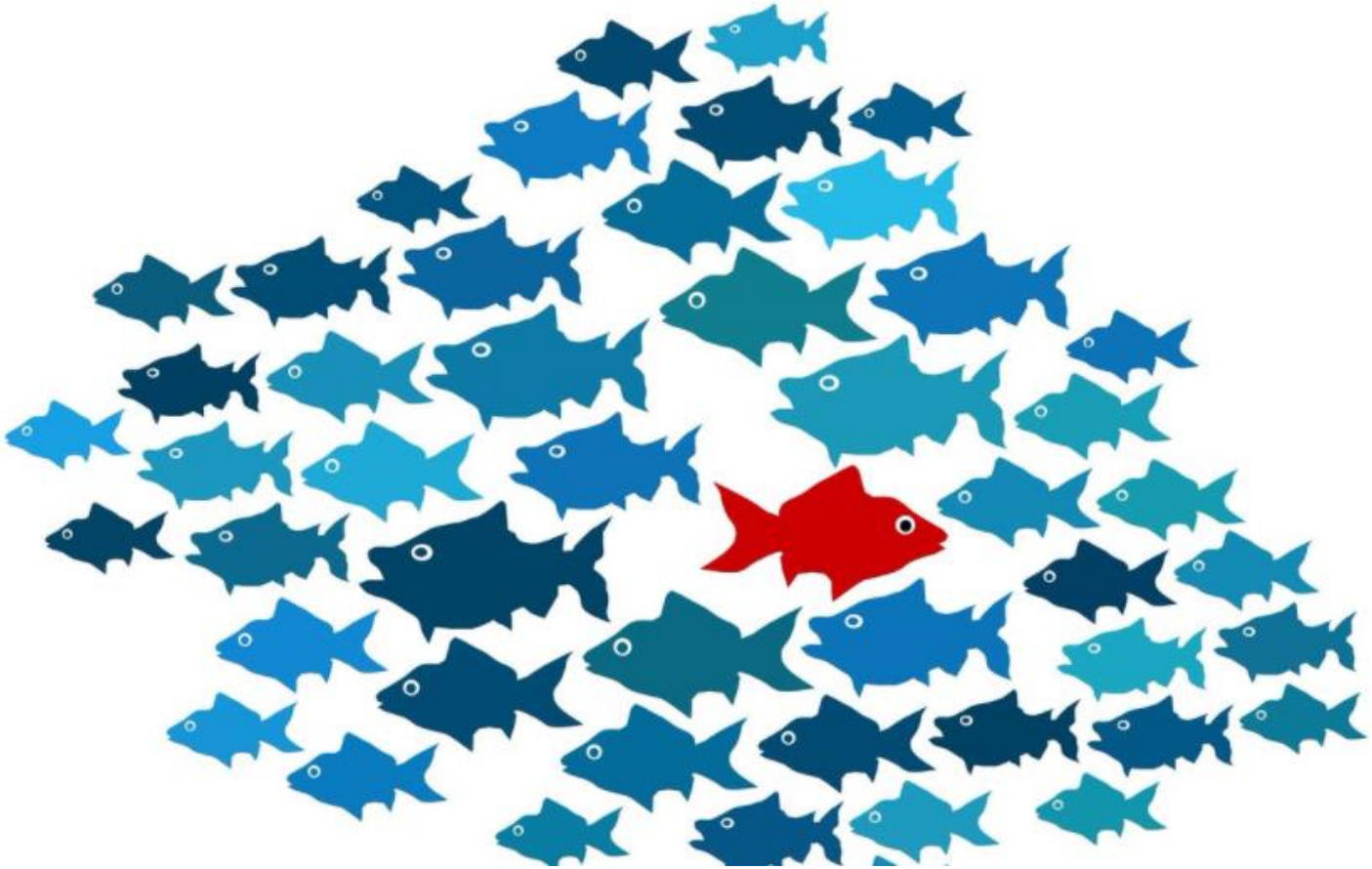
	Supervised vs **Unsupervised**	**Regression** vs **Classification**	**Parametric** vs **Non-Parametric**	**Generative** vs **Discriminative**
Linear Regression				
Logistic Regression				
k-NN				
Decision Tree				
PCA				
Clustering	**Unsupervised**	**neither**	**Non-Parametric**	**Generative**

Week 10 Unsupervised Part II PCA





Week 10 Unsupervised Part III Anomaly Detection



Week 10:
Next Lecture

University of Salford MANCHESTER

Reinforcement learning