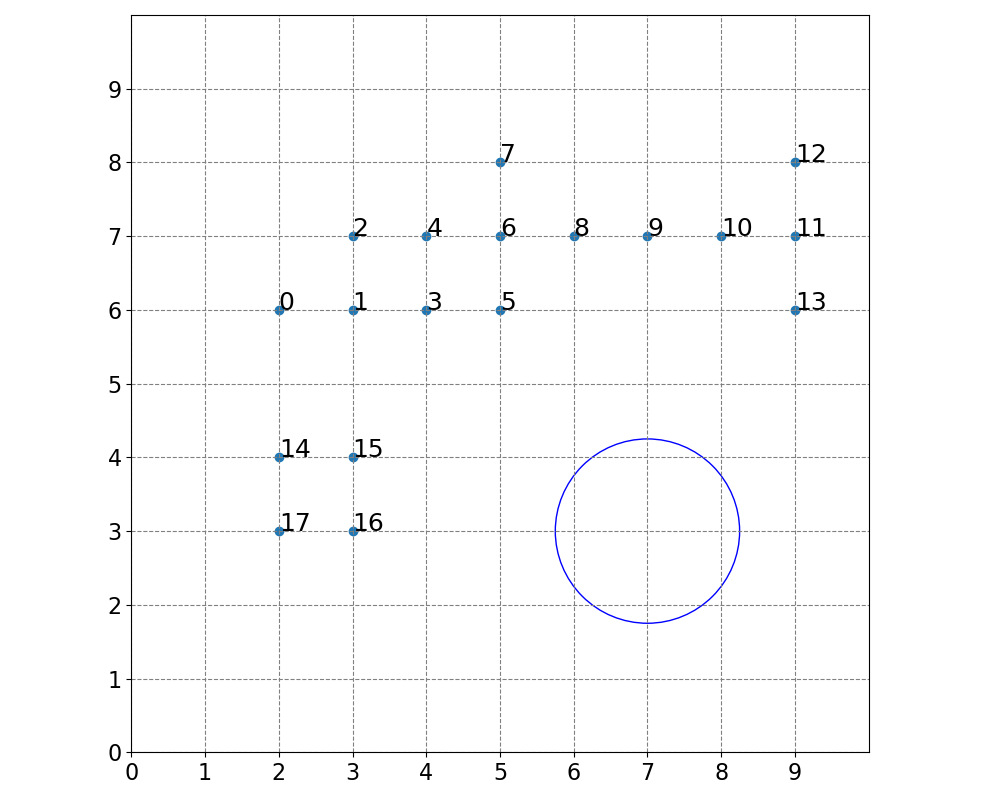
**CS5228 Tutorial 2 - Clustering**

**Q1: DBSCAN**

The following figure shows a toy dataset with 17 points.



1. Run DBSCAN by hand with and . (The blue circle shows a radius of ) List all core points, border points, and outliers (noise points).  
     
   Core points: 1, 3, 4, 6, 11  
   Border points: 0, 2, 5, 7, 8, 10, 12, 13  
   Noise points: 9, 14, 15, 16, 17
2. How many clusters are there, and what are their data points?  
     
   Cluster 1: 0 to 8  
   Cluster 2: 10 to 13
3. Can you add 2 data points such that the resulting clustering contains only 1 cluster and no noise?  
     
   Example solution: (2.5, 5), (7, 6.8)  
   (Many other solutions are possible. Adding the 2nd point as a duplicate (7, 7) is also fine)

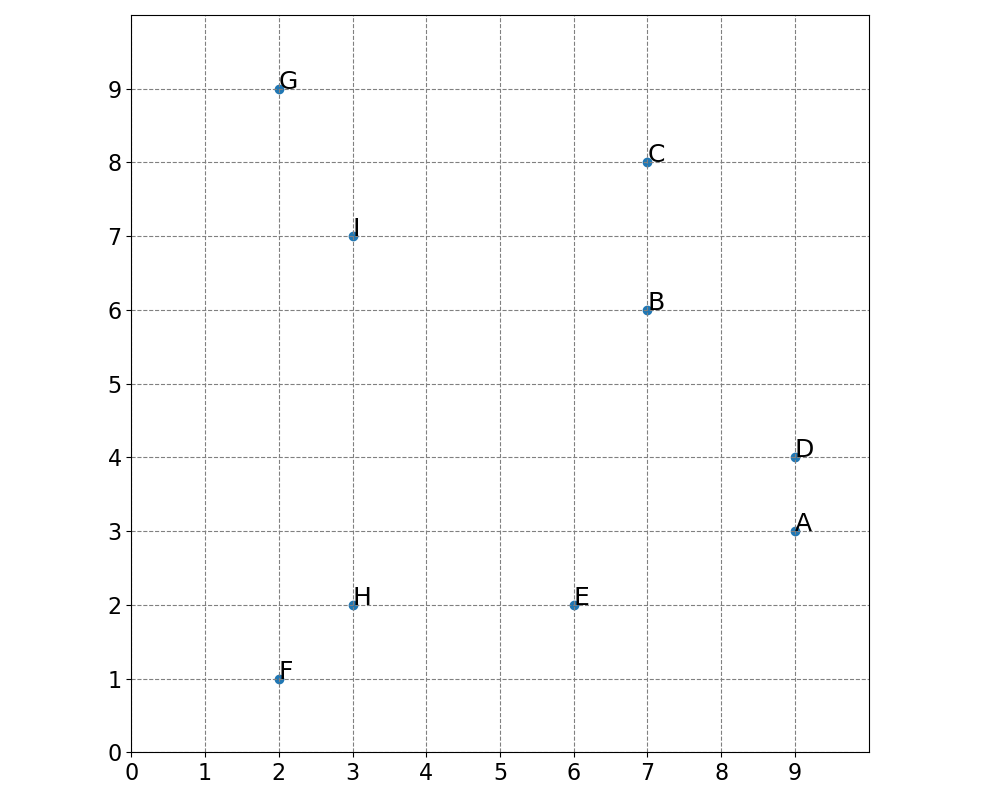
**Q2: DBSCAN vs K-Means**

1. Name a few fundamental differences between K-Means and DBSCAN.  
     
   K-Means is center-based (it characterizes each cluster by a center) while DBSCAN does not.  
   K-Means is defined as an optimization problem (hence there is the notion of local and global optima) while DBSCAN is not an optimization problem.  
   K-Means considers relative similarities / distances (e.g. scaling all distances does not change K-Means), while DBSCAN considers absolute similarities / distances (assuming is kept fixed)
2. Name a few meaningful criteria to decide whether to use K-Means or DBSCAN on a task?  
     
   DBSCAN can treat some points as noise, which can be useful if we have outliers  
   DBSCAN is better when clusters are very non-blob-shaped  
   K-Means may be more suitable when the number of clusters k is known / predefined based on the application context
3. Name a few example tasks and discuss whether K-Means or DBSCAN would be your method of choice.  
     
   

* Comments: generally, the ChatGPT answer is mostly quite reasonable (and some points are even quite interesting, like 3). The most questionable point is 4, on DBSCAN and high dimensional text data; DBSCAN is distance-based and can easily suffer from the curse of dimensionality (where distance-based methods perform poorly in high dimensions). This is dataset specific – in some cases, high dimensional data may actually lie on a lower dimensional space (e.g. if the features are highly correlated) and thus DBSCAN \*could\* work well, but it is not guaranteed. Generally, dimensionality reduction followed by clustering is a reasonable choice for clustering high dimensional data. For clustering text data, utilizing word embeddings or deep learning-based text embedding approaches (both out of syllabus), can help to transform the data into a more suitable space for subsequent clustering steps.  
  Also for point 1, it is also dataset-specific whether customer behaviour should be expected to have spherical clusters. So this requires visualizing the data and obtained clusters.   
  To sum up, the things that ChatGPT says shouldn’t be treated as totally reliable ☺ please be careful to verify them before using them.

**Q3: Hierarchical Clustering**

Compared to K-Means and DBSCAN, hierarchical clustering (specifically Agglomerative Nesting or AGNES) is a hierarchical clustering method. As such, each data point may belong to different clusters depending on the hierarchy level (e.g. higher level clusters can contain lower level clusters). AGNES yields complete clusterings, where each point belongs to at least one cluster.



1. Perform AGNES by hand with single linkage. Write down the steps involved (in any format).

Merge AD (dist = 1)

Merge FH (dist = 1.41)

Merge BC (dist = 2)

Merge GI (dist = 2.24)

Merge ADBC (dist = 2.83)

Merge EFH (dist = 3)­­

Merge ABCDEFH (dist = 3.16)

Merge all (dist = 4.12)

­­­ **Q4: Clustering Evaluation**

There is no perfect way to evaluate clusters – though using “ground truth” or external cluster evaluation is generally preferred if our ground truth label is indeed meaningful and of interest. Overall, it is important to be aware of the limitations and interpretation of different evaluation methods.

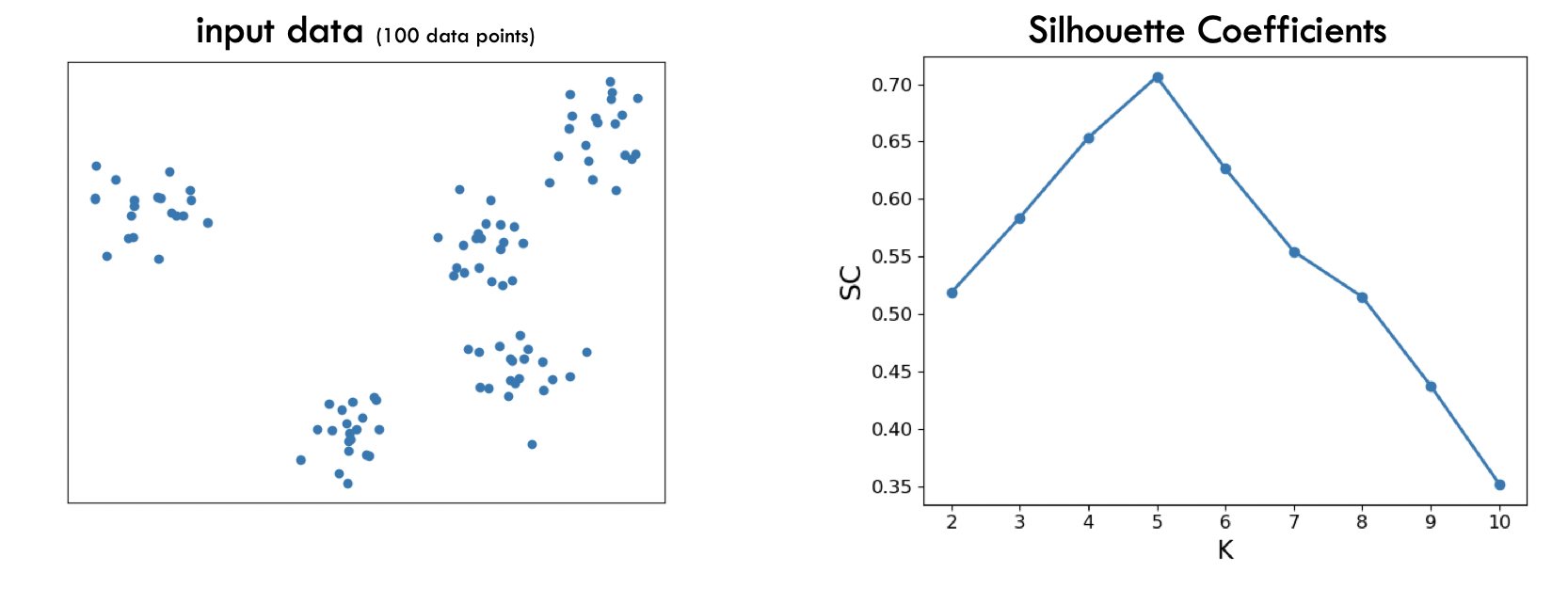
1. What are some limitations of using With Cluster Sum of Squares (WCSS) as a cluster evaluation metric?

SSE favors blob-like clusters

SSE is always decreasing as more clusters are added (so it generally prefers more clusters)

1. Which limitations of WCSS does the Silhouette Score successfully mitigate?

It mitigates the limitations of SSE always preferring more clusters (as we see in the lecture example – Silhouette Score does often decrease as “unnecessary” clusters are added:



Intuitively, adding unnecessary clusters (e.g. splitting a single cluster into 2) can cause the gap between “within cluster” and “across cluster” distances to shrink, bringing the silhouette score s(x) closer to 0 and thus lowering the overall silhouette coefficient as well.