Unsupervised Learning: Clustering

Zhongjian Lin
University of Georgia

November 5, 2024

Market Segmentation

We may have access to a large number of measurements (e.g. median household income, occupation, distance from nearest urban area, and so forth) for a large number of people. Our goal is to perform market segmentation by identifying subgroups of people who might be more receptive to a particular form of advertising, or more likely to purchase a particular product. The task of performing market segmentation amounts to clustering the people in the data set.

Clustering

Clustering is the task of partitioning the dataset into groups, called clusters. The goal is to split up the data in such a way that points within a single cluster are very similar and points in different clusters are different. Similarly to classification algorithms, clustering algorithms assign (or predict) a number to each data point, indicating which cluster a particular point belongs to. Data sometimes is represented directly in terms of the proximity (alikeness or affinity) between pairs of objects. These can be either similarities or dissimilarities (difference or lack of affinity).

Example

In social science experiments, participants are asked to judge by how much certain objects differ from one another. Dissimilarities can then be computed by averaging over the collection of such judgments. This type of data can be represented by an $N \times N$ matrix D, where N is the number of objects, and each element $d_{ii'}$ records the proximity between the ith and i'th objects. This matrix is then provided as input to the clustering algorithm.

Dissimilarities Based on Attributes

Most often we have measurements x_{ij} for $i = 1, 2, \cdots, N$, on variables $j = 1, 2, \cdots, p$. Since most of the popular clustering algorithms take a dissimilarity matrix as their input, we must first construct pairwise dissimilarities between the observations. In the most common case, we define a dissimilarity $d_j(x_{ij}, x_{i'j})$ between values of the jth attribute, and then define

$$D(x_i, x_i') = \sum_{j=1}^{p} d_j(x_{ij}, x_{i'j}),$$

as the dissimilarity between objects i and i'. By far the most common choice is squared distance

$$d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2.$$

However, other choices are possible, and can lead to potentially different results. For nonquantitative attributes (e.g., categorical data), squared distance may not be appropriate. In addition, it is sometimes desirable to weigh attributes differently rather than giving them equal weight

Object Dissimilarity

In general, we have dissimilarity of two observations defined as a weighted average

$$D(x_i, x_i') = \sum_{j=1}^{p} w_j d_j(x_{ij}, x_{i'j}), \text{ s.t. } \sum_{j=1}^{p} w_j = 1.$$

The goal of cluster analysis is to partition the observations into groups ("clusters") so that the pairwise dissimilarities between those assigned to the same cluster tend to be smaller than those in different clusters.

Clustering

- K-means clustering: K-means clustering is one of the simplest and most commonly used clustering algorithms. It tries to find cluster centers that are representative of certain regions of the data.
- Agglomerative Clustering: Agglomerative clustering refers to a collection of clustering algorithms that all build upon the same principles: The algorithm starts by declaring each point its own cluster, and then merges the two most similar clusters until some stopping criterion is satisfied.
- Density based spatial clustering of applications with noise (DBSCAN).

K-means Clustering

K-means clustering is a simple and elegant approach for partitioning a data set into K distinct, non-overlapping clusters. To perform K-means clustering, we must first specify the desired number of clusters K; then the K-means algorithm will assign each observation to exactly one of the K clusters.
The K-means clustering procedure results from a simple and intuitive mathematical problem. We begin by defining some notation. Let $C_1, \cdots, C_K$ denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

- $C_1 \cup C_2 \cup \cdots \cup C_K = \{1, \cdots, n\}$. In other words, each observation belongs to at least one of the K clusters.
- $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$. In other words, the clusters are nonoverlapping: no observation belongs to more than one cluster.

The idea behind K-means clustering is that a good clustering is one for which the within-cluster variation is as small as possible. We want to solve the problem

$$\min_{C_1, \cdots, C_K} \sum_{k=1}^{K} W_k(C),$$

where $W_k(C)$ is a measure of within-cluster variation for cluster C_k.

There are many possible ways to define this concept, but by far the most common choice involves *squared Euclidean distance*:

$$W_k(C) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{ij'})^2$$

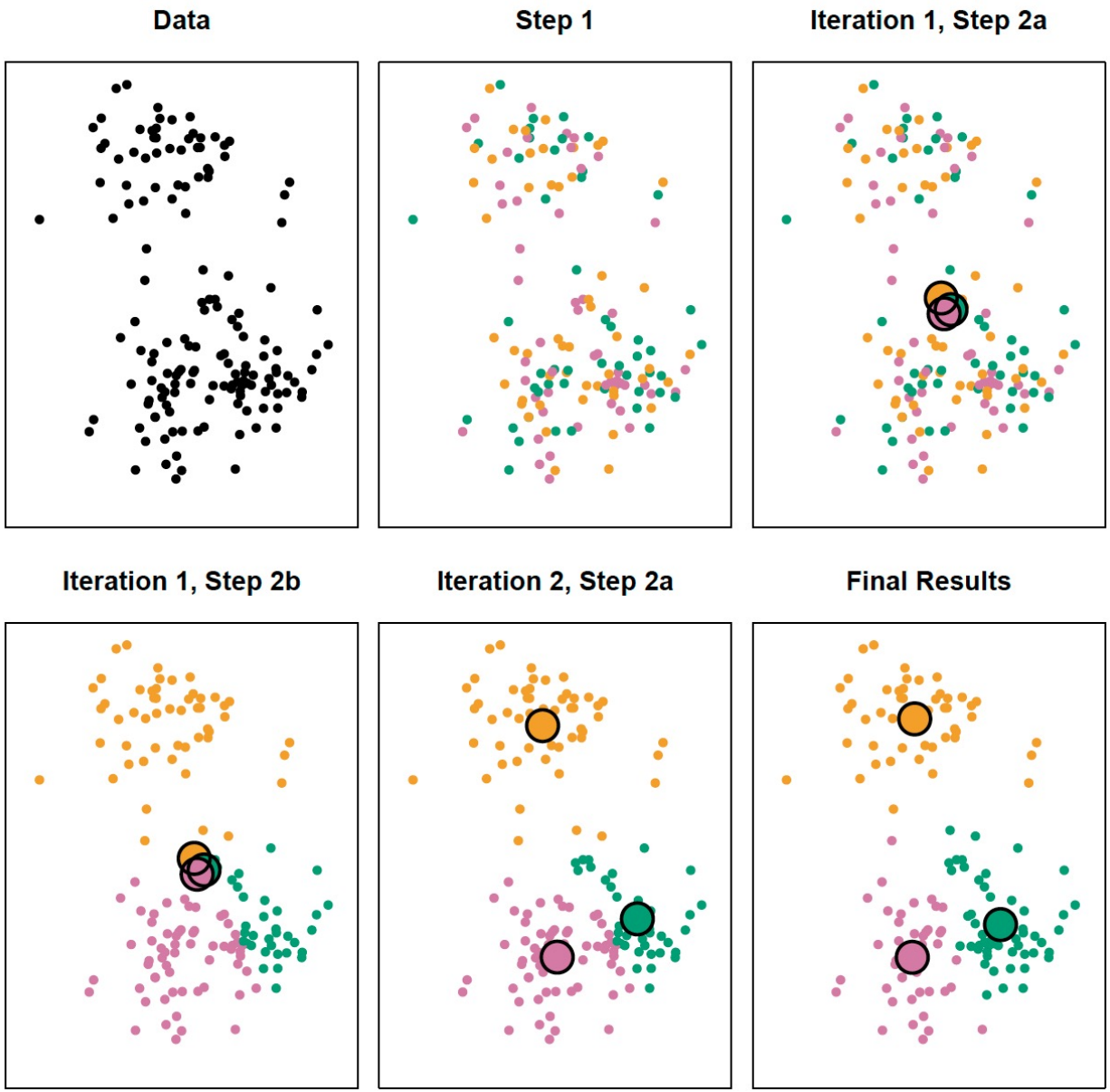
We then have optimization problem that defines K-means clustering

$$\min_{C_1,\cdots,C_K} \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{ij'})^2,$$

K-means Clustering Algorithm

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
   1. For each of the K clusters, compute the cluster *centroid*. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
   2. Assign each observation to the cluster whose centroid is closest (where closest is defined using Euclidean distance).

K-means Clustering



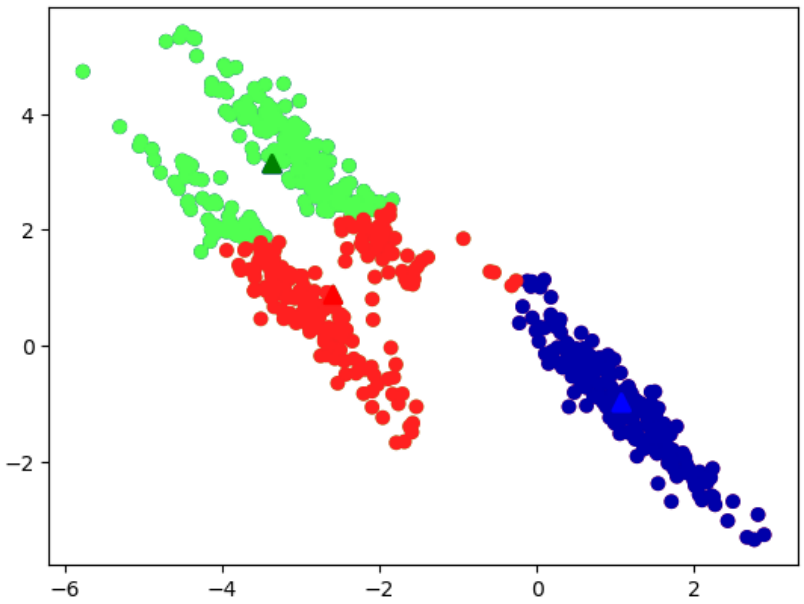
K-means Clustering

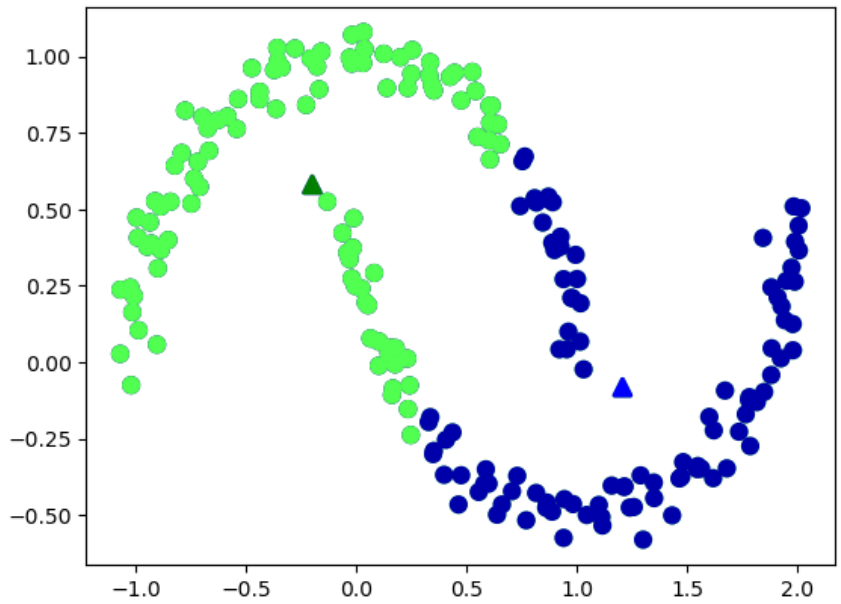
Because the K-means algorithm finds a local rather than a global optimum, the results obtained will depend on the initial (random) cluster assignment of each observation in Step 1 of Algorithm. For this reason, it is important to run the algorithm multiple times from different random initial configurations. Then one selects the best solution, i.e. that for which the objective is smallest.

Limitations of K-means Clustering

- Even if you know the "right" number of clusters for a given dataset, K-means might not always be able to recover them. Each cluster is defined solely by its center, which means that each cluster is a convex shape. As a result of this is that K-means can only capture relatively simple shapes.
- K-means also assumes that all clusters have the same "diameter" in some sense; it always draws the boundary between clusters to be exactly in the middle between the cluster centers. That can sometimes lead to surprising results.
- K-means also assumes that all directions are equally important for each cluster.
- K-means also performs poorly if the clusters have more complex shapes.

Limitations of K-means Clustering

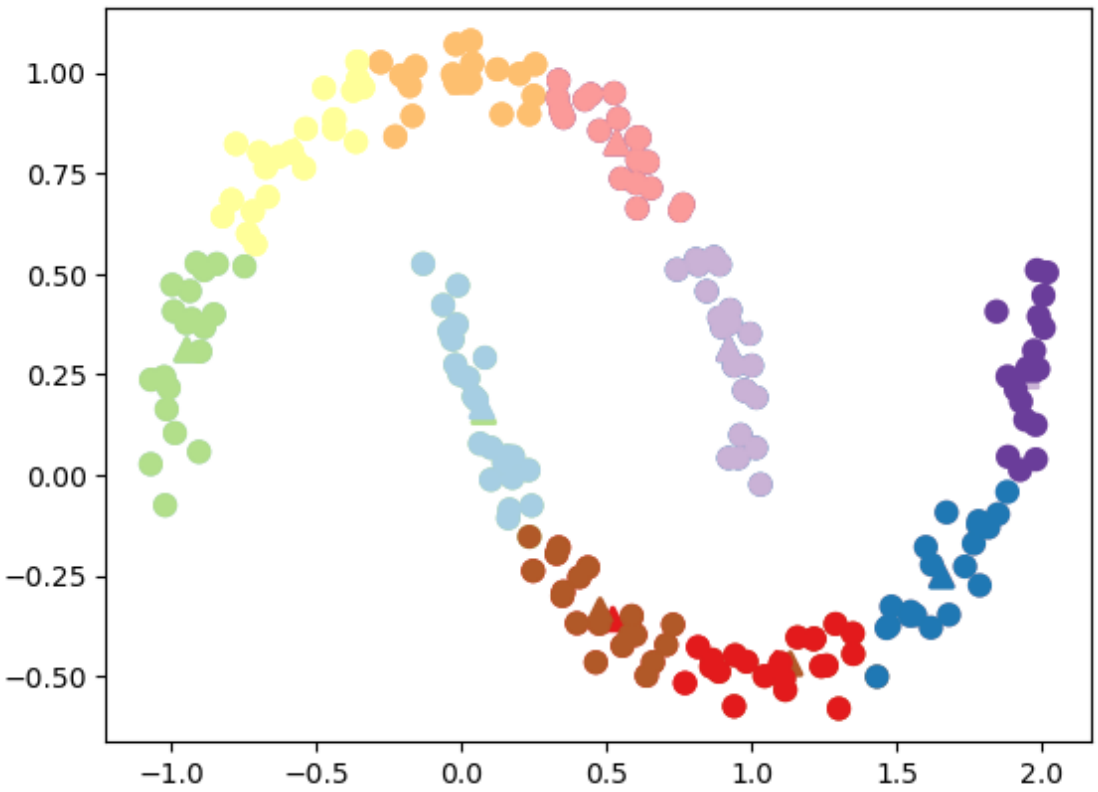




Decomposition of K-meamns Clustering

Even though k-Means is a clustering algorithm, there are interesting parallels between k-Means and decomposition methods like PCA. K-means on the other hand tries to represent each data point using a cluster center. You can think of that as each point being represented using only a single component, which is given by the cluster center. This view of K-means as a decomposition method, where each point is represented using a single component, is called *vector quantization*.

Generating new features using K-means

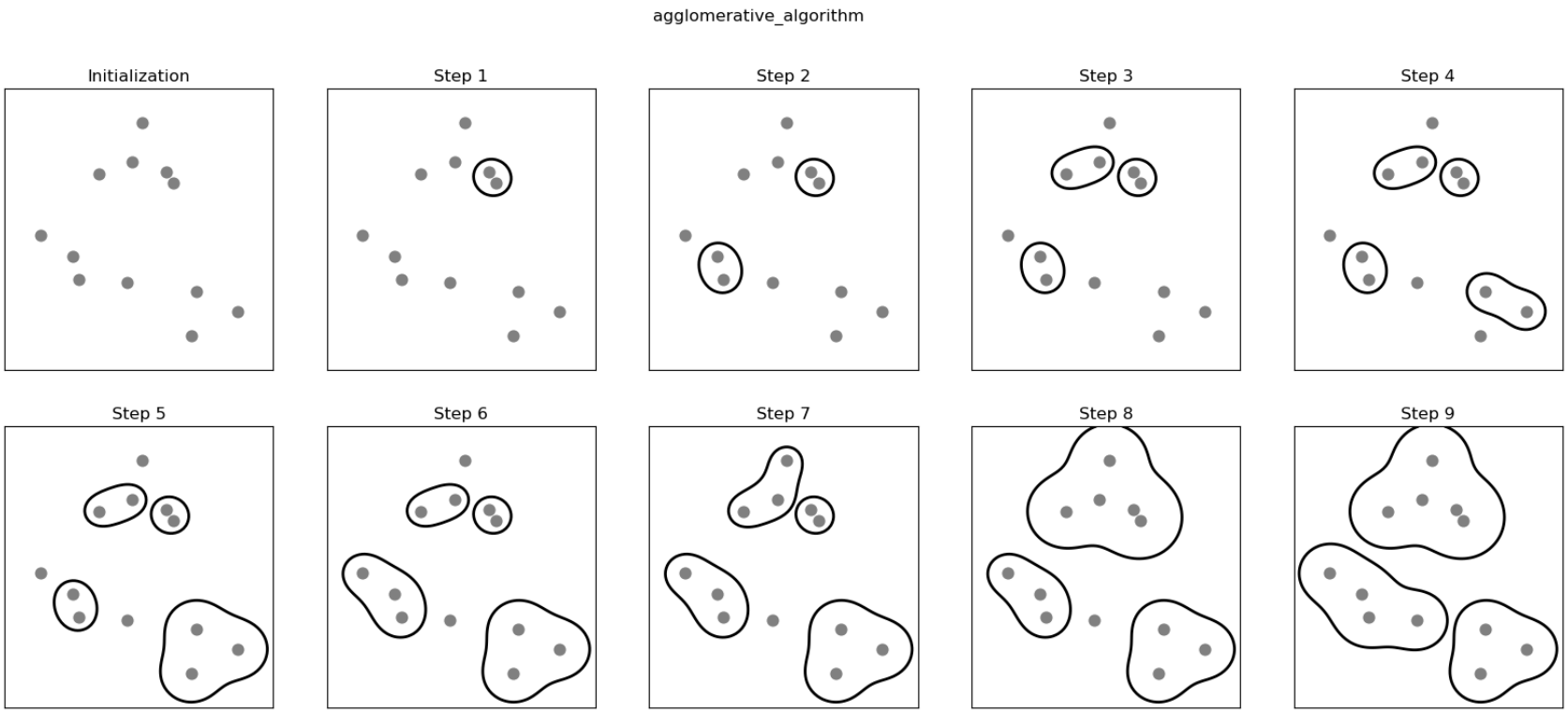


We can see this as the data being represented using 10 components (that is, we have ten new features), with all features being zero, apart from the one that represents the cluster center the point is assigned to. Using this 10-dimensional representation, it would now be possible to separate the two half-moon shapes using a linear model, which would not have been possible using the original two features.

Agglomerative Clustering

- Agglomerative clustering refers to a collection of clustering algorithms that all build upon the same principles: The algorithm starts by declaring each point its own cluster, and then merges the two most similar clusters until some stopping criterion is satisfied.
- The stopping criterion implemented in scikit-learn is the number of clusters, so similar cluster are merged until only the specified number of clusters is left.
- There are several linkage criteria that specify how exactly "most similar cluster" is measured.
  - "ward", which is the default choice. Ward picks the two clusters to merge such that the variance within all clusters increases the least. This often leads to clusters that are relatively equally sized.
  - "average" linkage merges the two clusters that have the smallest average distance between all their points.
  - "complete" linkage (also known as maximum linkage) merges the two clusters that have the smallest maximum distance between their points.

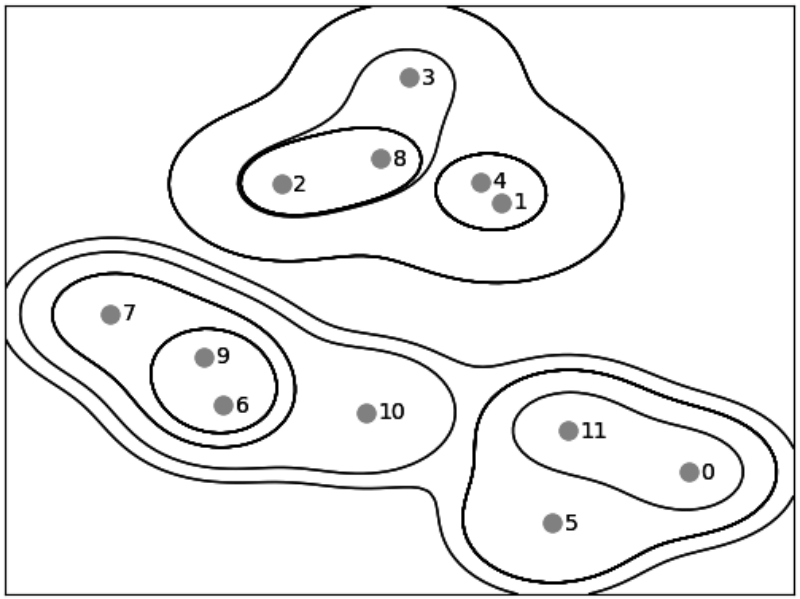
Agglomerative Clustering



In the beginning, each point is its own cluster. Then, in each step, the two clusters that are closest are merged. In the first four steps, two single point clusters are picked and these are joined into two-point clusters. In step four, one of the two-point clusters is extended to a third point, and so on. In step 9, there are only three clusters remaining. As we specified that we are looking for three clusters, the algorithm then stops.

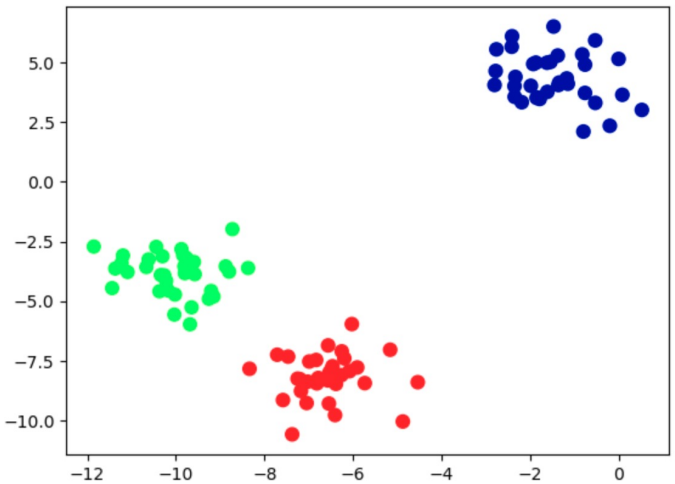
Hierarchical Clustering

Agglomerative clustering produces what is known as a hierarchical clustering. The clustering proceeds iteratively, and every point makes a journey from being a single point cluster to belonging to some final cluster. Each intermediate step provides a clustering of the data (with a different number of clusters). It is sometimes helpful to look at all possible clusterings jointly. The figure below shows an overlay of all possible clusterings shown in Figure agglomerative_ algorithm, providing some insight into how each cluster breaks up into smaller clusters.



Agglomerative Clustering

```
from sklearn.cluster import AgglomerativeClustering
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
import mglearn
X, y = make_blobs(random_state=1)
agg = AgglomerativeClustering(n_clusters=3)
assignment = agg.fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=assignment, cmap=mglearn.cm3, s=60)
```



Because of the way the algorithm works, agglomerative clustering can not make predictions for new data points. Therefore, agglomerative clustering has no predict method. Furthermore, agglomerative clustering still fails at separating complex shapes like the two_moons dataset.

DBSCAN

Another very useful clustering algorithm is DBSCAN (which stands for "Density based spatial clustering of applications with noise"). The main benefits of DBSCAN are

- it does not require the user to set the number of clusters a priori
- it can capture clusters of complex shapes
- it can identify point that are not part of any cluster

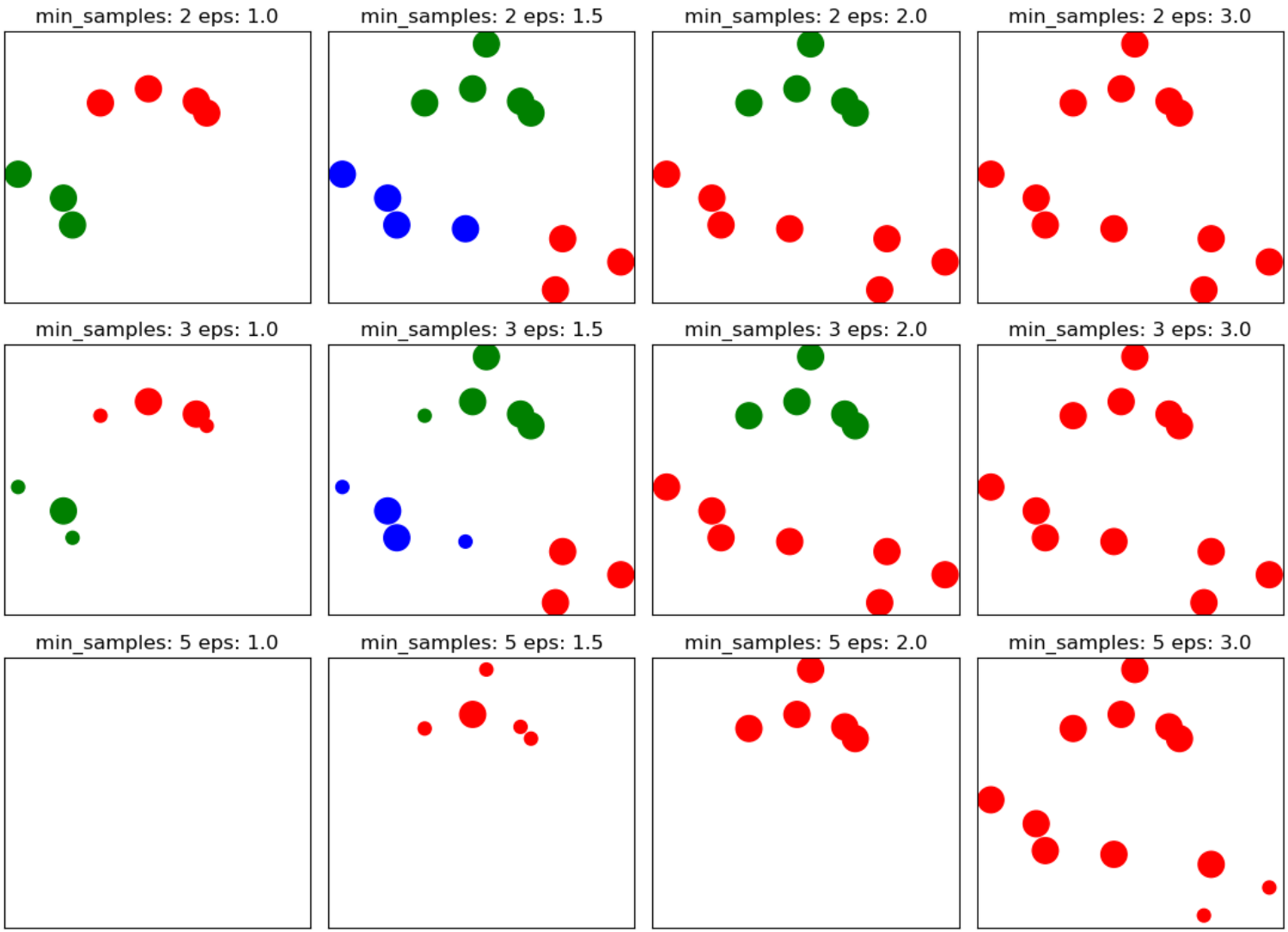
The way DBSCAN works is by identifying points that are in "crowded" regions of the feature space, where many data points are close together. These regions are referred to as dense regions in feature space. The idea behind DBSCAN is that clusters form dense regions of data, separated by regions that are relatively empty. There are two parameters in DBSCAN, *min_samples* and *eps*. If there are at least *min_samples* many data points within a distance of *eps* to a given data point, it's called a core sample. Core samples that are closer than the distance eps are put into the same cluster by DBSCAN.

DBSCAN

The procedure of DBSCAN is

- The algorithm works by picking a point to start with.
- It then finds all points with distance eps or less. If there are less than *min_samples* points within distance *eps* or less, this point is labeled as noise, meaning that this point doesn't belong to any cluster.
- If there are more than *min_samples* points within a distance of *eps*, the point is labeled a core sample and assigned a new cluster label. Then, all neighbors (withing eps) of the point are visited. If they have not been assigned a cluster yet, they are assigned the new cluster label we just created. If they are core samples, their neighbors are visited in turn, and so on.
- The cluster grows, until there are no more core-samples within distance *eps* of the cluster.
- Then another point, which hasn't yet been visited, is picked, and the same procedure is repeated.
- In the end, there are three kinds of points: *core points*, points that are within distance *eps* of core points (called boundary points), and noise.

Effect of changing *eps* and *min_samples*

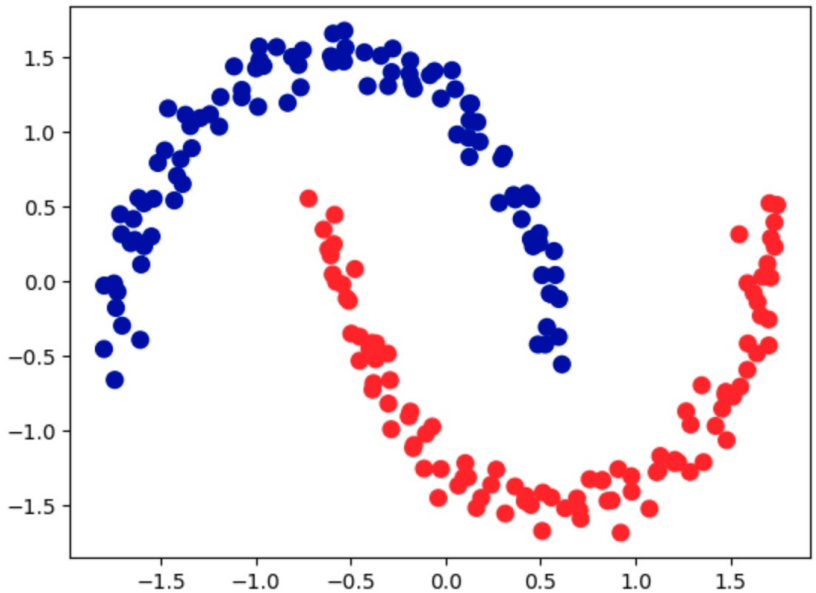


In this plot, points that belong to clusters are colored, while the noise points are shown in white. Core samples are shown as large points, while border points are displayed as smaller points.

DBSCAN

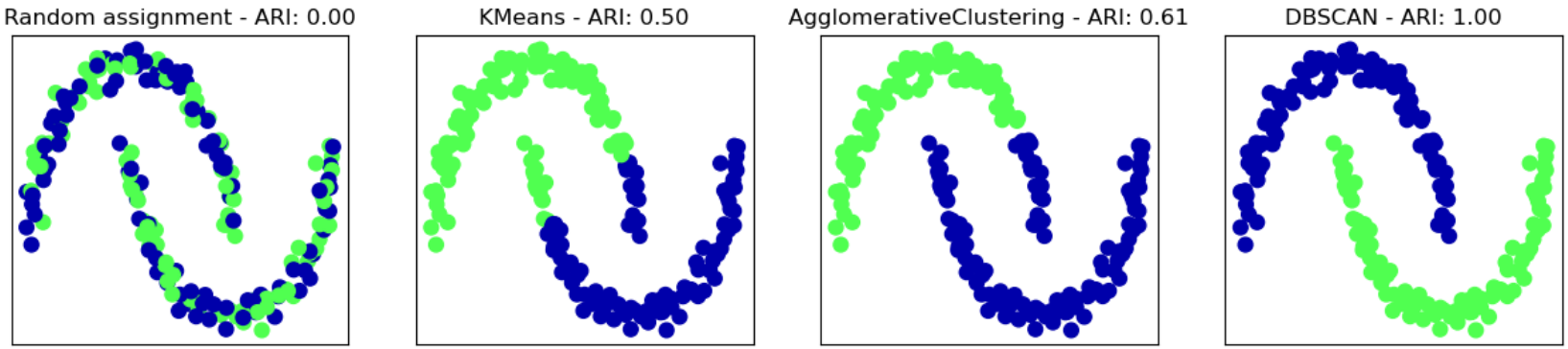
DBSCAN is somewhat slower than agglomerative clustering and k-Means, but still scales to relatively large datasets. Below is the code to use DBSCAN for the two-moon data.

```
from sklearn.preprocessing import StandardScaler
X, y = make_moons(n_samples=200, noise=0.05, random_state=0)
scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)
dbscan = DBSCAN()
clusters = dbscan.fit_predict(X_scaled)
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=clusters, cmap=mglearn.cm2, s=60)
plt.show()
```



Comparing and evaluating clustering algorithms

After talking about the algorithms behind K-means, agglomerative clustering and DBSCAN, we will now compare them on some real world datasets. One of the challenges in applying clustering algorithms is that it is very hard to access how well a clustering algorithm worked, and to compare outcomes between different algorithms. There are metrics that can be used to assess the outcome of a clustering algorithm relative to a ground truth clustering, the most important ones being the *adjusted rand index (ARI)* and *normalized mutual information (NMI)*, which both provide a quantitative measure between 0 and 1.



The adjusted rand index provides intuitive results, with a random cluster assignment having a score of 0, and DBSCAN (which recovers the desired clustering perfectly) having a score of 1.

Clustering

- We saw above that applying and evaluating clustering is a highly qualitative procedure, and often most helpful in the exploratory phase of data analysis.
- We looked at three clustering algorithms, K-means, DBSCAN and Agglomerative Clustering. All three have a way of controlling the granularity of clustering.
- K-means and Agglomerative Clustering allow you to directly specify the number of desired clusters, while DBSCAN lets you define proximity using the eps parameter, which indirectly influences cluster size.
- All three methods can be used on large, real-world datasets, are relatively easy to understand, and allow for clustering into many clusters.
- Each of the algorithms has somewhat different strengths. K-means allows for a characterization of the clusters using the cluster means. It can also be viewed as a decomposition method, where each data point is represented by its cluster center.
- DBSCAN allows for the detection of "noise points" that are not assigned any cluster, and it can help automatically determine the number of clusters. In contrast to the other two methods, it allow for complex cluster shapes, as we saw in the two_moons example. DBSCAN sometimes produces clusters of very differing size, which can be a strength or a weakness.