Chapter 11
Clustering

11.1 Introduction to clustering

So far we have discussed many different supervised learning algorithms, from linear regression to decision trees. Remember, supervised learning is where the data inputs map to a known output. In this chapter, we will learn about our first unsupervised learning algorithm, meaning it uses an unlabeled data set to form predictions.

Definition 11.1: Clustering

Clustering is an unsupervised and automatic process of trying to identify and group similar data points within a larger data set.

Although the primary use of clustering is for unlabeled data, we can still use clustering algorithms to fit to known labels, like "world events" or "sports". This supervised learning version of clustering, as we have learned previously, is **classification**.

Example(s)

Clustering has a number of uses, varying from simple to complex:
- Classifying news article topics, for example between "world events" and "sports".
- Clustering groups of people according to their genome.
- Recommending entertainment for consumers using user preferences. However, topics and preferences are often not well defined, making this problem challenging.

Clustering uses the given inputs, for example $x_1, x_2, ..., x_n$, and attempts to learn the structure from the unlabeled data X and assign them to groups. If we decide that there are 3 groups we wish to cluster, the outputs would be $z_1, z_2, ..., z_n$ where z_i is either one of those three classes, for example 1, 2, or 3. More formally, clustering uses a given x_i and assigns $z_i \in [1, 2, , ..., k]$ where k is the number of clusters to be identified. Note that z has been used here instead of the more typical y. As a reminder, clustering is an unsupervised learning approach so to reduce confusion between some ground truth label for an observation y_i, we have used z_i to denote the assigned cluster. A cluster is usually based on the closest centroid, which is the average for some group of data.

As usual, we should define some objective which determines how good each of the assignments are. One example could be euclidean distance from the centroid, such that points far away from a center would have high error, while those very close to a center have low error. In this case a close distance would represent a strong similarity between points. However, it is important to remember that clustering is not always straight forward and distance alone might not determine specific clusters.

11.2 K-Means

11.2.1 Overview of K-Means algorithm

K-means is the first clustering algorithm we will discuss. The main objective for k-means is to assign points to a cluster based off the point's distance from a centroid. This means that a shorter distance will create

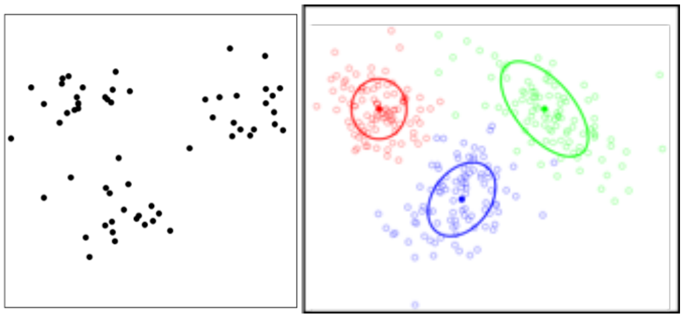

Figure 11.56: Unlabeled input data on the left with 3 defined clusters on the right.

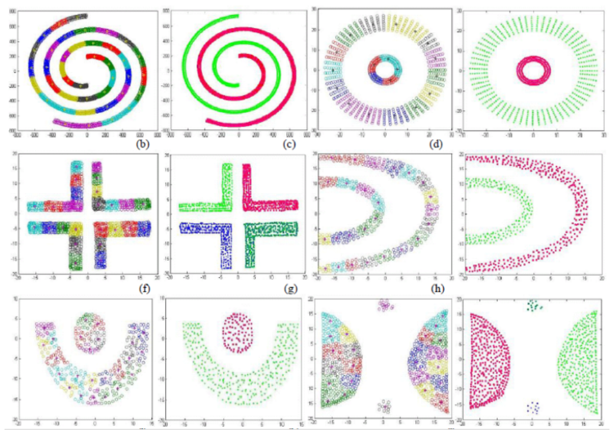

Figure 11.57: Many different examples of more complicated clusters.

better clustering when using this algorithm.

Algorithm 3 K-Means Clustering

```
Given a data set of $n$ data points and a particular choice of $k$
Step 0: Initialize $k$ centroids randomly to points in the data set

while not converged do
    Step 1: Assign each example to its closest cluster centroid
    Step 2: Update the centroids to be the average of all the points assigned to that cluster
return cluster assignments as closest centroid to each data point
```

Above is the basic outline for K-Means. However, there are many different ways to approach each of the steps, depending on the use case.

- **Step 0:** Usually we initialize the centroids $\mu_1, \mu_2, ..., \mu_k$ randomly, however there are smarter initialization techniques which will be discussed below.
- **Step 1:** We assign each example to its closest centroid such that:

$$z_i = \underset{j}{\text{argmin}} ||\mu_i - x_i||_2^2$$

- **Step 2:** Now we update the centroids to be the mean of all the point assignments from the previous step (i.e. the center of mass for the cluster).

$$\mu_j = \frac{\sum_{i=1}^{n} 1(z_i = j)x_i}{\sum_{i=1}^{n} 1(z_i = j)}$$

- **Convergence:** There are multiple different stopping conditions. Some include:
  - When the cluster assignments haven't changed.
  - Centroids haven't changed locations.
  - Some number of max iterations have passed.

  It is also important to remember that K-Means is heavily dependent on its starting assignments so the algorithm may only converge to a local optima rather than a global one. To get around this, we can either use a smarter initialization compared to random and/or by running K-Means multiple times to ensure we arrive at similar results across trials.

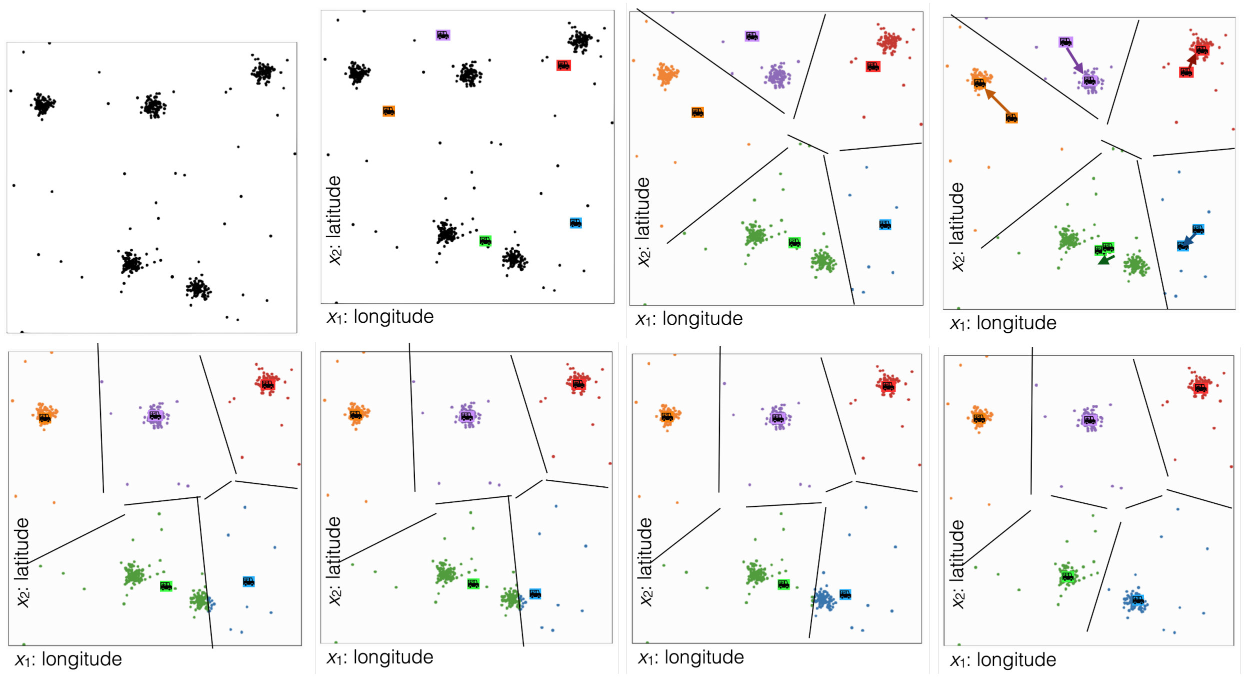

Figure 11.58: K-Means algorithm visualized.

11.2.2 Choosing k in K-Means

It is also important to remember that k in K-Means is a hyperparameter which defines the number of clusters to find so that results will be very different for different values of k. Choosing the correct k is very important as a k that is too low, for example, could be grouping things together when they really are distinct groups. Illustrating this, the green and blue groups from the example above would likely get grouped together if $k = 4$, instead of the correct choice of $k = 5$. Following that same logic, a k that is too high will put 2 (or more) groups together when they should be one.

Definition 11.2: Heterogeneity Objective

$$\underset{z,\mu}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{i=1}^{n} 1\{z_i = j\} ||\mu_j - x_i||$$

This is the objective that K-Means uses, which is equivalent to minimizing the distance from each centroid to its group of examples.

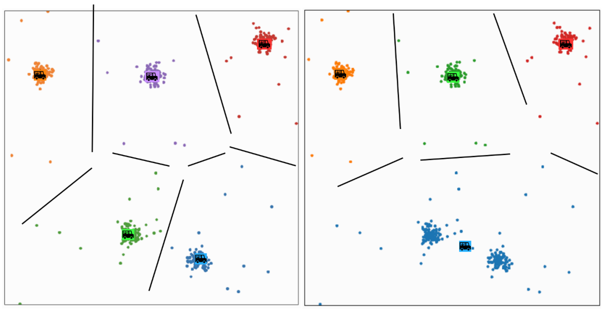

Figure 11.59: Example of choosing different values of k. ($k = 5$ on right and $k = 4$ on left)

However, the choice of k is often not easy to see, either because the dimension of the data is high or it is not easy to visualize the data. One way to approach this problem is to use a validation set / k-fold approach like discussed in previous chapters to see which k is the best choice for clustering. This is a good idea but slightly different when put into practice. Clustering is different than trying to solve for the quality metric's minimum. If that was our goal, the optimum choice of k would be n, the number of data points, such that the total distance from each point to its centroid is 0. Clearly, this is not helpful in terms of clustering.

Classification quality metrics do not apply to clustering. Classification learns from minimizing the error between a prediction and an actual label while in clustering the "labels" don't have meaning, so it is up to human input to determine what good clustering is and isn't.

Our goal of minimizing error, similar to the idea of overfitting from previous chapters, has caveats. One way we can approach using a validation set with hard-to-visualize data is the elbow method. We plot the quality metric across different values of k (multiple times for each k due to reasons stated above) and see where there is an "elbow" in the graph. This corresponds to reaching the optimal k value, as the error will drop sharply, but after adding more points the error should only reduce slightly run after run.

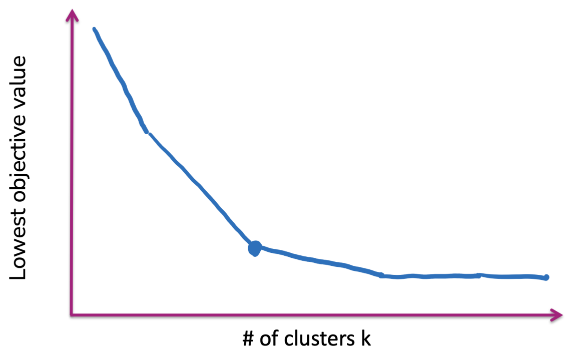

Figure 11.60: Choosing the best k using the elbow method.

Example(s)

Movies: choosing k when clustering movies should most likely be used with a pre-determined number categories as a human choice, rather than chasing the k which has the lowest validation set loss.

There are a number of other methods to choose k also: Bayesian Information Criterion, Silhouette, etc.

11.2.3 Effects of Initialization in K-Means

As previously mentioned, the initialization of the centroids in K-Means can have a large impact on the outcome of the clusters. First, it is important to remember that the label assigned in K-Means does not have any meaning. We know that runs of K-Means work well if the same groups of points are assigned together, but not necessarily if a certain centroid is always assigned to the same group of points.

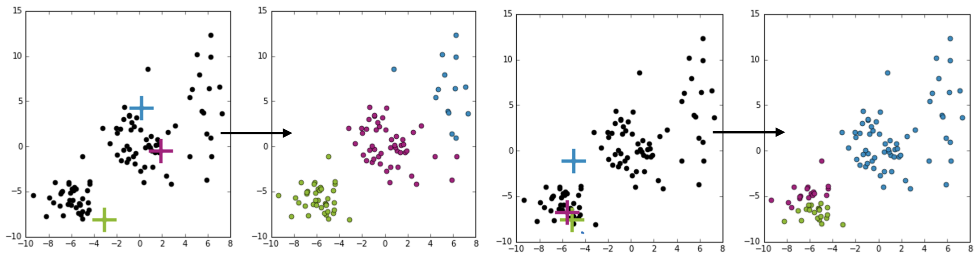

Figure 11.61: Initialization can affect cluster outcomes.

Given that K-Means is quite simple and only considers centralizing the points which are closest to a given centroid, K-Means might get stuck in a local optima rather than a global. Ideally we would like to initialize such that centroids start far away from one another, rather than simply at random, so that two centroids don't converge to one group, leaving 2 "real" groups potentially clustered as 1.

11.3 K-Means++

To initialize points with a wide spread we could factor in whether we choose a point to be an initialization based on how far away that point is from other centroids we have already chosen during the initialization. Using this we can initialize to a point we were are more likely to be closer to the optimal solution than purely at random. This can lead to faster and more accurate/reliable convergence of centroids. This type of initialization will also likely reach a global minima rather than only a local. However, note that this initialization is more computationally expensive at the beginning compared to random initialization as we have to keep track of each points distance to its nearest centroid and update those values each time we select of new one.

Algorithm 4 K-Means++ Initialization

```
Step 1: Choose first centroid $\mu_1$ uniformly at random from dataset

for j from 1 to $k$ do Step 2: For current set of centroids compute the distance between each datapoint and
its closest centroid
Step 3: Choose new centroid $\mu_j$ from the remaining data points with probability of $x_i$ being chosen pro-
portional to the squared distance between $x_i$ and its closest centroid (it's current assigned center) Step 4:
run K-Means from intialized centroids as normal
```

11.4 Caveats of K-Means

K-Means is a powerful tool for clustering however, like most ML algorithms, it is not perfect for every scenario. K-Means works well for **hyper-spherical** clusters of the same size but when the true clusters don't look like that, it can produce less than ideal results.

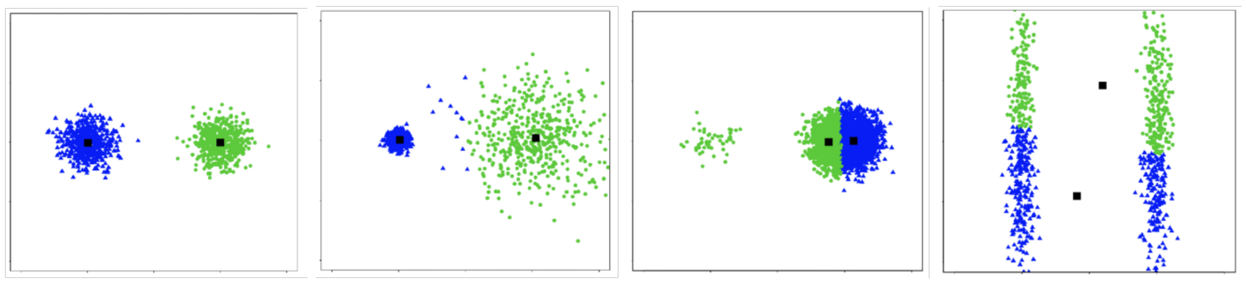

Figure 11.62: Some examples where K-Means works well and some where it fails.

To wrap up our introduction to clustering, there are a couple important points to remember:

- Usually, we don't split train / test set for clustering problems, because there are no labels to measure how good a set of clusters are.
- Similarly, definitions like bias, variance, complexity bias-variance trade off might not work for unsupervised learning problems.
- The heterogeneity objective in clustering does not have the same meaning as training error in supervised learning tasks. There is no definition of accuracy for clustering either.

11.5 Text Embedding: TF-IDF

TF-IDF (term frequency-inverse document frequency) is used to encode the relevance of a word in a document in relation to other text documents in a corpus of text. This is relevant to k-means, as it allows us to in essence, "translate" text documents into something that the algorithm can interpret, and thus perform clustering on. TF-IDF specifically places more emphasis on important words within a corpus, as can be seen in the following section.

The following outlines the process by which TF-IDF is calculated:

- let each document within a corpus be represented by an array, the length of which represent the number of unique words (v) in the entire corpus
- calculating TF: record the number of times a word appears in a document divided by the total number of words in said document, and store the results in an array of length v
- calcualting IDF: record the logarithm of the fraction of the total number of documents divided by (1 + the number of documents using the word)
- fill the original array representing the document with the element-wise (Hadamard) product of the TF and IDF arrays

`Chapter 12/Images/tf-idf.png`

As shown in the image, words that appear very frequently in the corpus will have smaller IDF, and thus, smaller TF-IDF overall, thus placing less emphasis on them. Words that do not appear as frequently will have a larger IDF, and thus, larger TF-IDF. This is a countermeasure to overemphasizing common words like ”the” (at least for the English language) in other embedding methods like bag of words.