Lesson 2: Hierarchical Clustering

Lesson Objectives

By the end of this lesson, you will be able to:

* Explain the intuition behind the different types of Hierarchical Clustering
* Implement the Hierarchical Clustering algorithm from scratch and using packages
* Understand the differences between K-Means and Hierarchical Clustering

Introduction

Clustering Refresh

Last lesson covered both the high-level intuition and in-depth details of one of the most basic clustering algorithms: K-Means. While it is indeed a simple approach, do not discredit it as a valuable addition to your toolkit as you continue your exploration of the unsupervised learning world. In many real-world use cases companies experience groundbreaking discoveries through the simplest methods such as K-Means or Linear Regression (for supervised learning). As a refresher, lets quickly walk through what clusters are and how K-Means works to find them.



Figure 2.1: Recall the attributes that separate supervised and unsupervised problems

If you were given a random collection of data without any guidance, you would likely start your exploration using basic statistics – what are the mean, median, mode of each of the features? Recall that from a high-level data simply exists and whether it is a supervised or unsupervised learning is ascribed by the data goals you have set for yourself or by your manager. If you were to determine that one of the features was actually a label and you wanted to see how the remaining features in the data set influence it, this would become a supervised learning problem. However, if after initial exploration you realize that the data you have is simply a collection of features without a target in mind (collection of health metrics, purchase invoices from a web store, etc) then you can analyze it through unsupervised methods.

The classic example of unsupervised learning is finding clusters of similar customers in a collection of invoices from a web store. Your hypothesis is that by understanding which people are most similar, you can create more granular marketing campaigns that appeal to each cluster’s interests. One way to achieve these clusters of similar users is through K-Means.

K-Means clustering works by finding “K” number clusters in your data through pairwise Euclidean distance calculations. “K” points (also called centroids) are randomly initialized in your data and the distance is calculated from each data point to each of the centroids. The minimum of these distances designates which cluster a data point belongs to. Once every point has been assigned to a cluster, the mean intra-cluster data point is calculated as the new centroid. This process is repeated until the newly calculated cluster centroid no longer changes position.

Hopefully this has served as a refresh of concepts that you remembered from Lesson 1. If not, please take the time to go back and review before moving on – aspects of K-Means are built upon in Lesson 2 to create more complex forms of clustering!

The Organization of Hierarchy

Both the natural and human-made world contain many examples of how organizing systems into hierarchies makes a lot of sense. A common representation that is developed from these hierarchies can be seen in tree-based data structures. Imagine you had a parent node with any number of child nodes, that could subsequently be parent nodes themselves. By organizing concepts into a tree structure, you can build an information-dense diagram that clearly shows how things are related amongst their peers as well as their larger abstract concepts.

An example to help illustrate this concept from the natural world can be seen in how we view the hierarchy of animals that goes from parent classes to individual species:

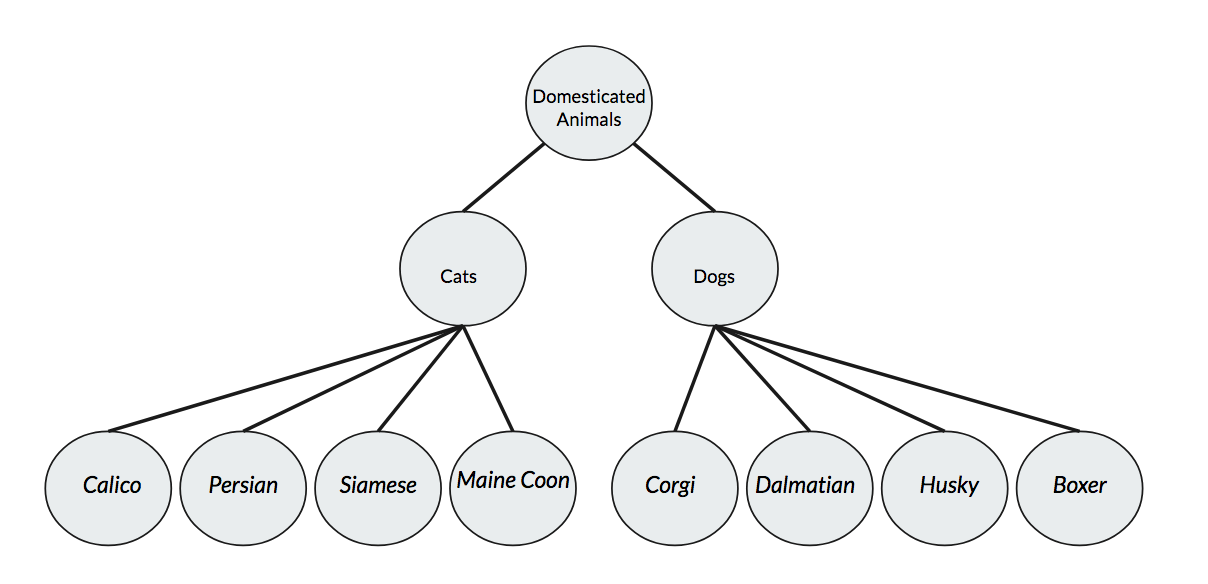


Figure 2.2: Navigating animal species relationships in a hierarchical tree structure

In Figure 2.2 you can see an example of how relational information between a variety of animals can be easily mapped out in a way that both saves space while still transmitting a large amount of information. This example can be seen as both a tree of its own showing how cats and dogs are different but both domesticated animals, as well as a potential piece of a larger tree that shows a breakdown of domesticated versus non-domesticated animals.

In the event that most of the students here are not biologists, let’s move back towards the concept of a web store selling products. If you sold a large variety of products you likely want to create a hierarchical system of navigation for your customers. By withholding all of the information in your product catalog, a customer will only be exposed to the path down the tree that matches their interests. An example of hierarchical benefits to navigation can be seen in Figure 2.3:

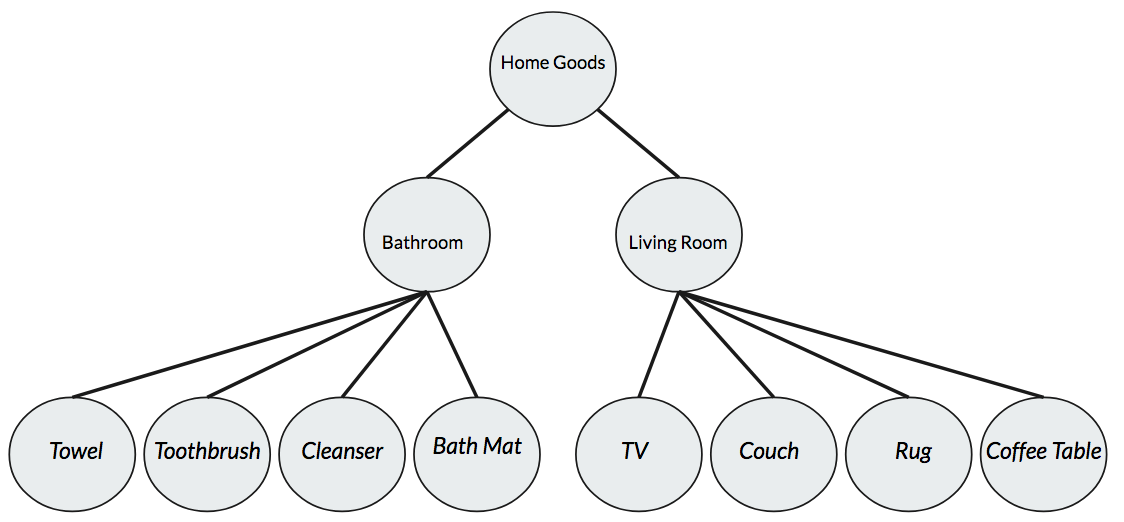


Figure 2.3: Navigating product categories in a hierarchical tree structure

Clearly the benefits of a hierarchical system of navigation cannot be overstated for improving your customer experience! By organizing information into a hierarchical structure, you can build an intuitive structure out of your data that demonstrates explicit nested relationships. If this sounds like another approach to finding clusters in your data, you’re definitely thinking on the right track! Through the use of similar distance metrics such as Euclidean Distance from K-Means, we can develop a tree that shows the many cuts of data that allow a user to subjectively create clusters at their discretion.

Introduction: Hierarchical Clustering

Up until this point we have shown that hierarchies can be excellent structures to organize information into that clearly show nested relationships among data points. While this is helpful to understand parent/child relationships between items, it can also be very handy when forming clusters. Expanding on the animal example of the prior section, imagine you were simply presented with two features of animals: their height (measured from tip of the nose to end of tail), and their weight. From this information you have to recreate the same structure of identifying which records in your dataset correspond to dogs or cats, as well as their relative sub-species.

Since you were only given animal heights and weights you will never be able to extrapolate what the specific species names are. However, by analyzing the features that you have been provided you can develop a structure within the data that serves as an approximation of what animal species exist in your data. This perfectly sets the stage for an unsupervised learning problem that is well solved with hierarchical clustering.

Figure 2. : Example of a 2-feature dataset comprised of animal height and weight

One way to approach hierarchical clustering is by starting with each data point serving as its own “cluster” and recursively joining similar points together to form clusters – this is known as agglomerative hierarchical clustering. We will go into detail on the different ways of approaching hierarchical clustering in a later section.

In the agglomerative hierarchical clustering approach, the concept of data point similarity can be thought of in the paradigm we saw during K-Means. In K-Means we used Euclidean distance to calculate the distance of individual points to centroids of expected “K” clusters. For this approach to hierarchical clustering we will re-use the same distance metric to determine similarity between records in our data set.

Eventually by grouping individual records from the data with their most similar records recursively, you end up building a hierarchy from the bottom up. What starts as many individual single-member clusters join together into one single cluster at the top of our hierarchy.

In-Depth: Hierarchical Clustering

To see how agglomerative hierarchical clustering works, we can trace the path of a simple toy program as it merges together to form a hierarchy:

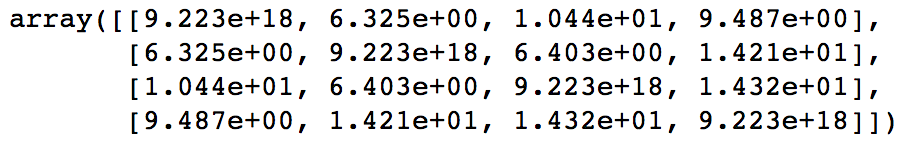
1. Given **n** sample data points, view each point as an individual “cluster” with just that one point as a member.
2. Calculate pairwise Euclidean distance between the centroids of all the clusters in your data.
3. Group the closest point pairs together.
4. Repeat steps 2 and 3 until you reach a single cluster containing all data in your set.
5. Plot a dendrogram to show how your data has come together in a hierarchical structure.
6. Decide at which level you want to create clusters at.

While slightly more complex than K-Means, it does not change too much from a logistical perspective! Here is a simple example walking through the above steps in slightly more detail:

1. Given **4** sample data points, view each point as its own cluster.

* [ (1,7) ], [ (-5,9) ], [ (-9,4) ] , [ (4, -2) ]

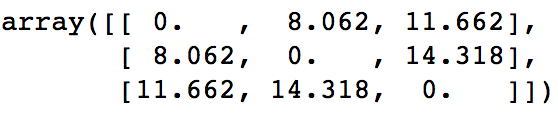
1. Calculate pairwise Euclidean distance between the centroids of all clusters.

* 

1. Group the closest point pairs together.

* In this case, Points [1,7] and [-5,9] join into a cluster with the remaining 2 points left as single-member clusters.

1. Repeat steps 2 and 3 until you reach a single cluster containing all data in your set.

* Clusters (3 total now): [ [ [1,7], [-5,9] ], [-9,4], [4,-2] ]
* Calculate the centroid of the 2-member cluster:
  + *mean*([ [1,7], [-5,9]) = [-2,8]
* Add centroid to 2 single-member clusters and re-calculate distances:
  + [ [-2,8], [-9,4], [4,-2] ]
  + 
* Point [-9,4] is added to Cluster 1:
  + [ [ [1,7], [-5,9], [-9,4] ], [4,-2] ]
* Repeat centroid calculation and group closest pairs together. Since there is only one point left, you can just add it to Cluster 1.

1. Plot a dendrogram

At the end of this process you can visualize the hierarchical structure you created through a dendrogram. This plot shows how data points are similar and will look familiar to the hierarchical tree structures we discussed before. Once you have this dendrogram structure you can interpret how the data points relate together and subjectively decide at which “level” the clusters should exist.

Revisiting the animal taxonomy example from above that involved dog and cat species, imagine you were presented with the following dendrogram:

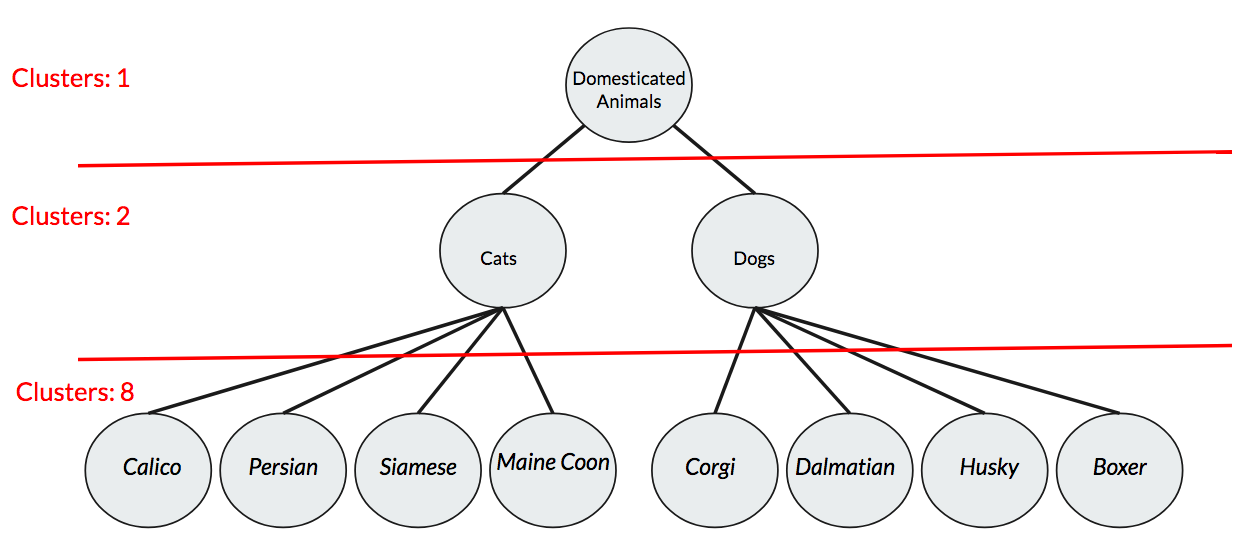


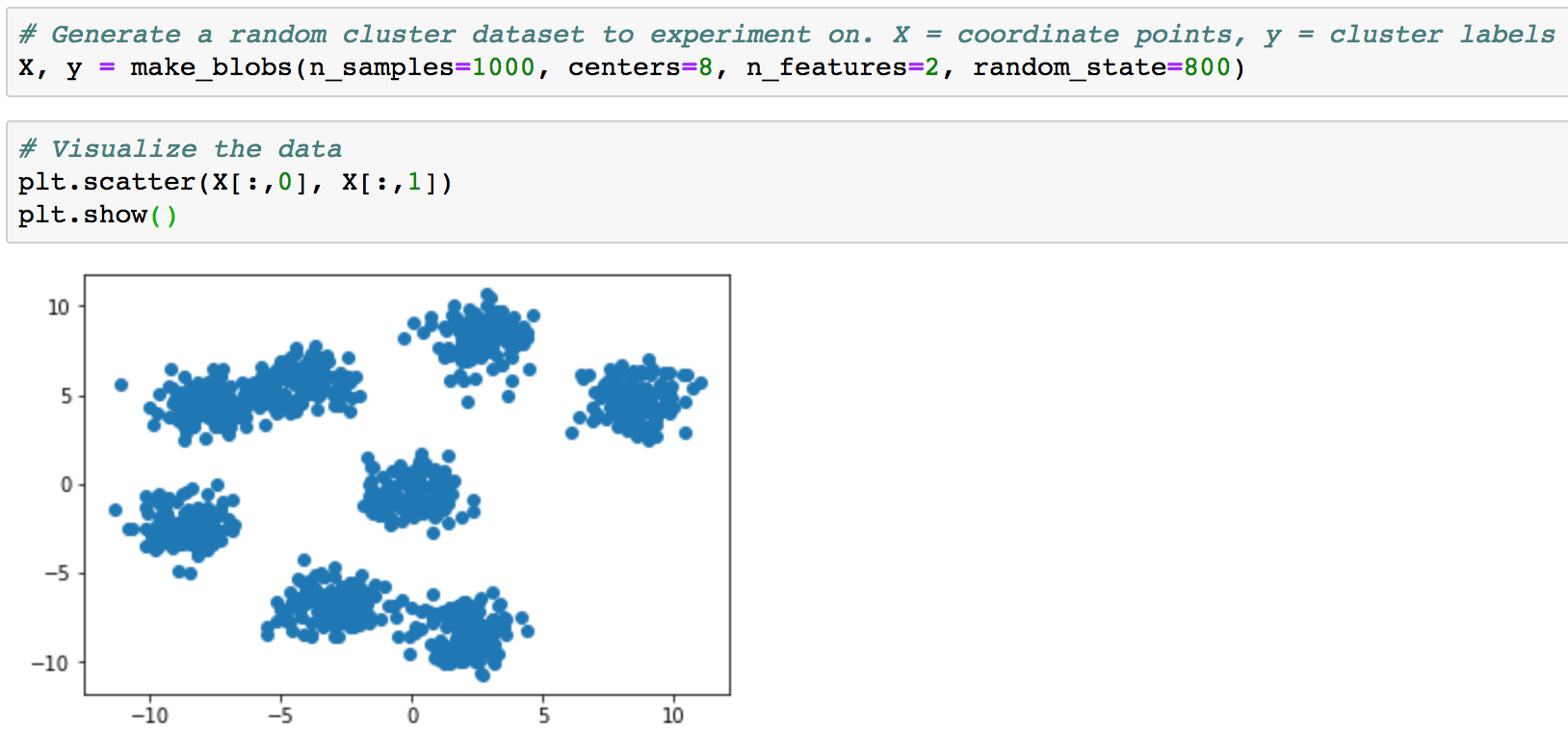
Figure 2. : Animal taxonomy dendrogram

The great thing about hierarchical clustering and dendrograms is that you can see the entire breakdown of potential clusters to choose from. If you were just interested in grouping your species dataset into dogs and cats, you could stop clustering at the first level of grouping. However, if you wanted to group all species into domesticated or non-domesticated animals, you could stop clustering at level two.

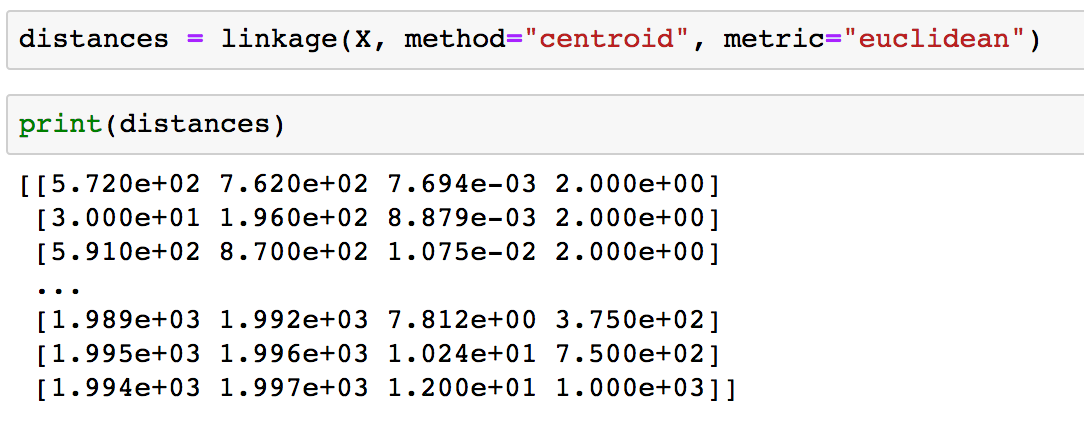
Exercise 1: Building a Hierarchy

Let’s try implementing the above hierarchical clustering approach in Python. With the framework for intuition laid out, we can now explore building hierarchical cluster with some helper functions provided in SciPy.

First let’s generate some dummy data:

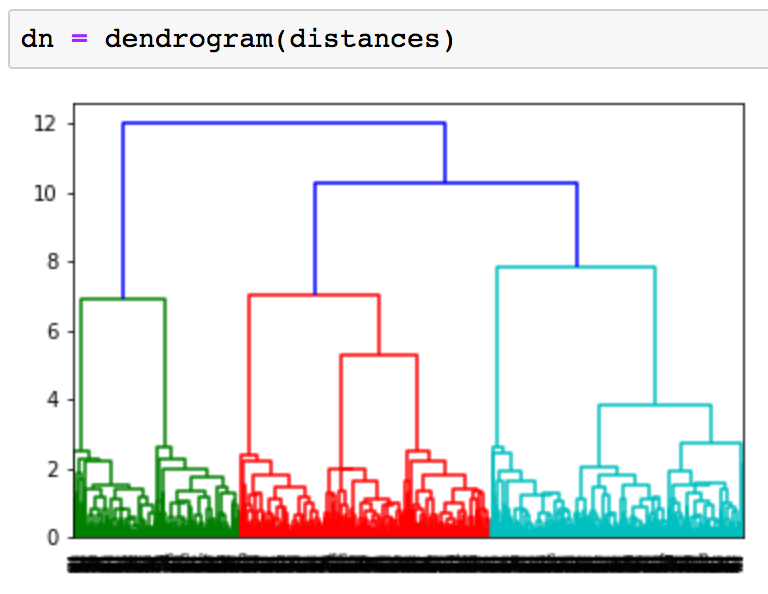


After plotting this simple toy example, it should be pretty clear that our dummy data is comprised of 8 clusters. We can easily generate the distance matrix using the built-in scipy package ‘linkage’:

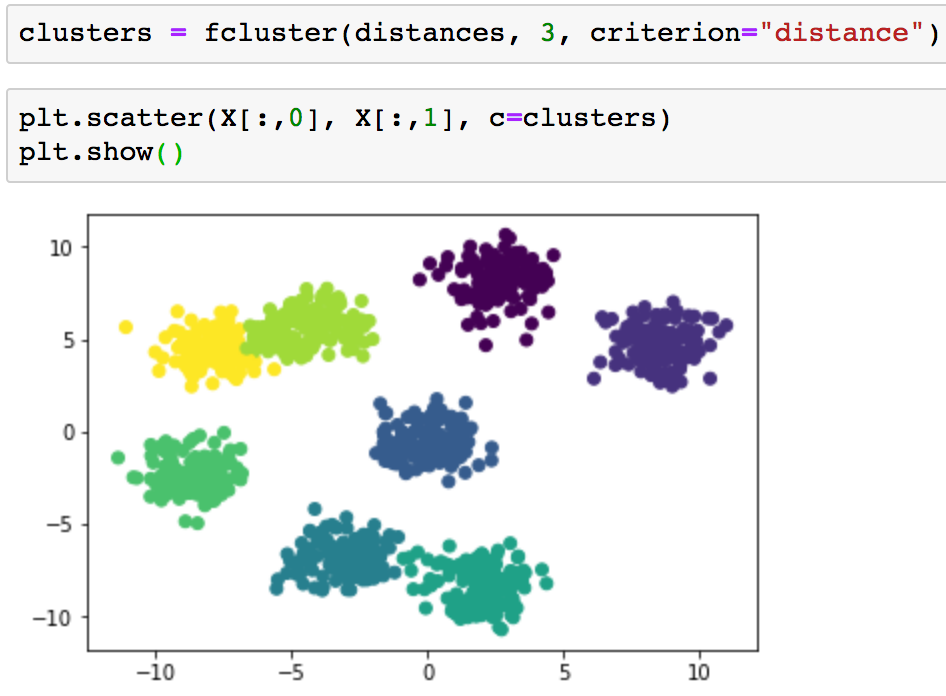


You can see the first situation where customizing the parameters really drives performance in finding the linkage matrix. If you recall our steps above, linkage works by simply calculating the distances between each of the data points. In the linkage function we have the option to select both metric as well as method (more on this later).

After we determine the linkage matrix we can easily pass that through to the dendrogram function provided by scipy:



This plot will give us some perspective on the potential breakouts of our data. Using this information we can wrap up our exercise of hierarchical clustering by using the fcluster function from scipy:



Through just calling a few helper functions provided by scipy, you can easily implement agglomerative clustering in just a few lines of code! While scipy does help with many of the intermediate steps, this is still an example that is a bit more verbose than you will probably see in your regular work. More streamlined implementations will be covered later.

Hierarchical Clustering Attributes: Linkage

In Exercise 1 you implemented hierarchical clustering using what is known as Centroid Linkage. Linkage is the concept of determining how you calculate distances between clusters and is dependent on the type of problem you are facing. Centroid Linkage was chosen for the first activity as it essentially mirrors the new centroid search we used in K-Means. However, this is not the only option when it comes to clustering data points together. Two other popular choices for determining distances between clusters are Single Linkage and Complete Linkage.

Single Linkage works by finding the minimal distance between a pair of points between two clusters as its criteria for linkage. Said another way, it essentially works by combining clusters based on the closest points between the two clusters. Mathematically this is expressed as:

***dist(a,b) =* min( dist( a[i]), b[j] ) )**

Complete Linkage can be seen as opposite of Single Linkage and works by finding the maximal distance between a pair of points between two clusters as its criteria for linkage. Said another way, it works by combining clusters based on the furthest points between the two clusters. Mathematically this is expressed as:

***dist(a,b) =* max( dist( a[i]), b[j] ) )**

Determining what linkage criteria is best for your problem is as much art as science and is heavily dependent on your particular data set. One reason to choose Single Linkage is that your data are closely similar in a nearest neighbor sense, and when there are differences the data is extremely dissimilar. Since Single Linkage works by finding closest points it will not be affected by these distant outliers. Conversely, Complete Linkage may be a better option if your data is distant from each other inter-cluster, however it is quite dense intra-cluster. Centroid Linkage has similar benefits but falls apart if the data is very noisy and there are less clearly defined “centers” of clusters. Typically, the best approach is to try a few different linkage criteria options and see which fits your data in a way that’s most relevant to your goals.

Activity 1: Linkage Criteria

**Scenario:** You are given a data set without prior background information and are requested to find the hierarchical clustering linkage that fits it the best.

**Aim:** Given what you’ve learned about Agglomerative Clustering with Centroid Linkage, evaluate the remaining Linkage types and consider when you would use each in practice.

Hierarchical Clustering Attributes: Agglomerative vs Divisive

Our instances of hierarchical clustering this far have all been agglomerative – built from the bottom up. While this is typically the most common approach to this type of clustering, it is important to know that it is not the only way a hierarchy can be created. The opposite hierarchical approach, built form the top up, can also be used to create your taxonomy. This approach is called Divisive Hierarchical Clustering and works by starting all data points in your set in one massive cluster. Many of the internal mechanics of the divisive approach prove to be quite similar to agglomerative.

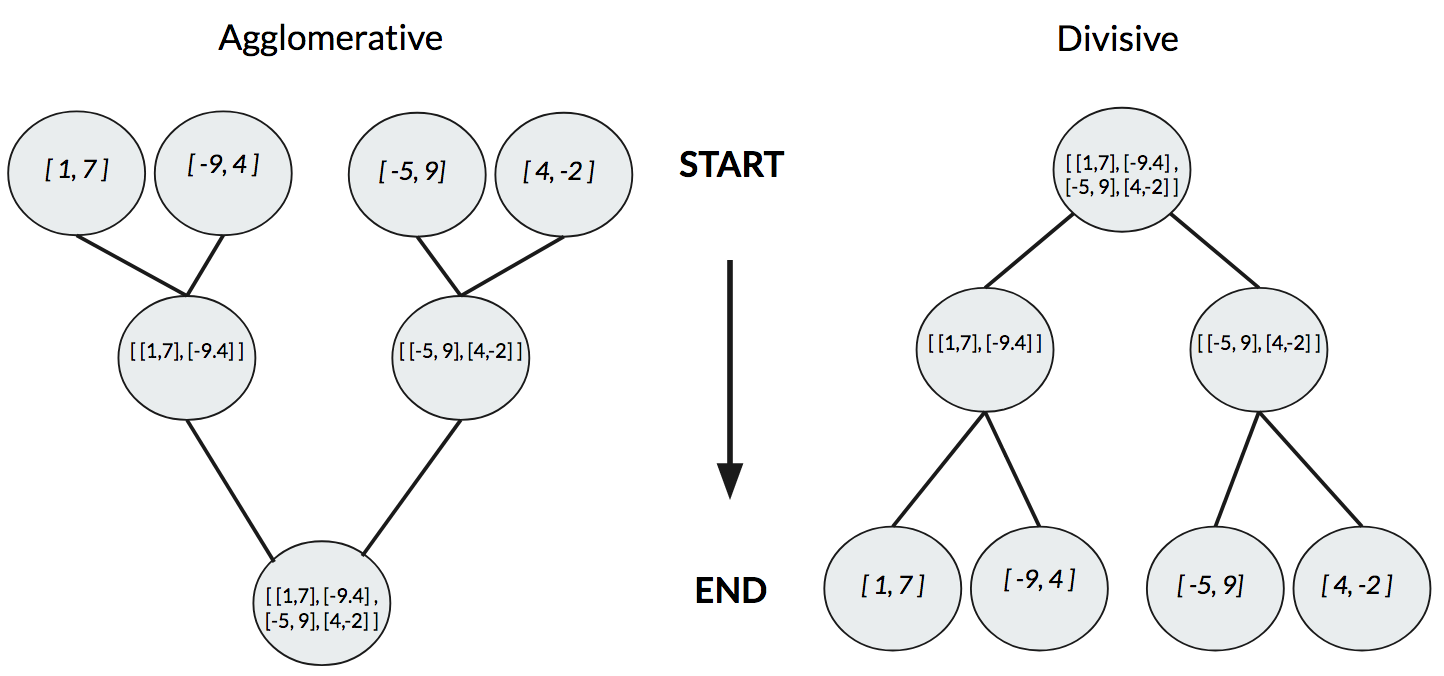


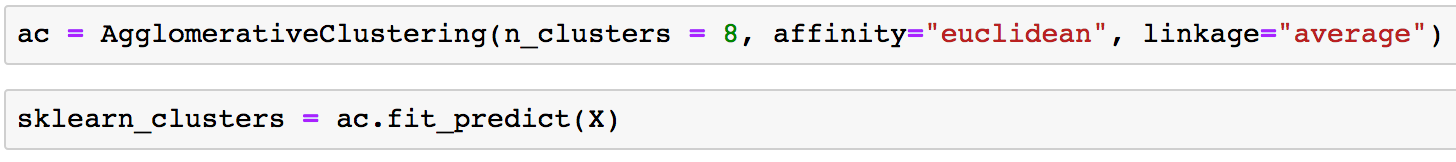
Figure 2. : Agglomerative vs Divisive Hierarchical Clustering

As with most problems in unsupervised learning, deciding the best approach is often highly dependent on the problem you are faced with solving. In general, it helps to think of Agglomerative as the bottoms-up approach and Divisive as the top-down approach – but how do they trade off in performance? Due to greedy nature seen in Agglomerative, it has the potential to be fooled by local neighbors and not see the larger implications of clusters it forms at any given time. On the flip side, Divisive has the benefit of seeing the entire data distribution as one from the beginning and choosing the best way to break down clusters. This insight into what the entire data set looks like is helpful for potentially creating more accurate clusters and should not be overlooked. Unfortunately, a top-down approach typically trades off greater accuracy with deeper complexity. In practice Agglomerative works most of the time and should be the preferred starting point when it comes to hierarchical clustering. If after reviewing the hierarchies, you are unhappy with the results it may help to take the Divisive approach.

Exercise 2: Implementing Agglomerative Clustering with Sci-Kit Learn

In most real-world use cases you will likely find yourself implementing hierarchical clustering with a package that abstracts everything away, like Sci-Kit Learn. This approach makes the work much easier and should only be used when you fully understand how hierarchical clustering works from the prior sections!

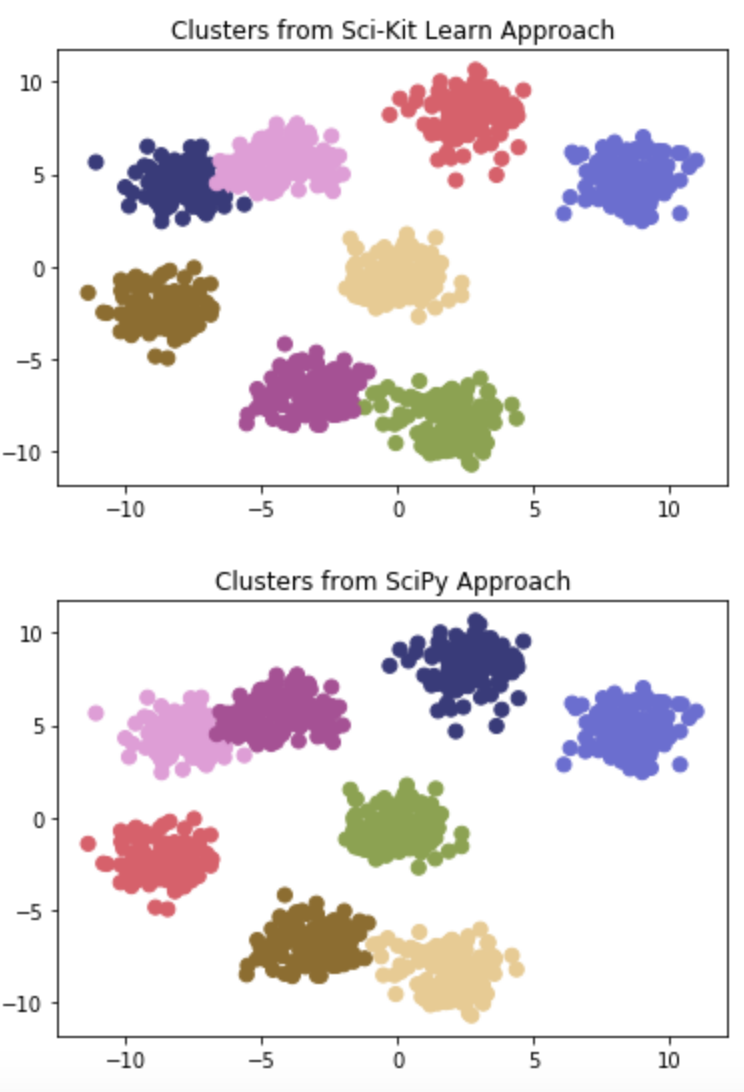
As you saw in the K-Means chapter, Sci-Kit Learn makes implementation as easy as just a few lines of code:



First we assign the model to variable ‘ac’ passing in parameters that we are familiar with such as affinity (distance function) as well as linkage (explore options as we have done in Activity 1).

After instantiating our model into a variable we can simply pass through the data set we are interested in to determine where cluster memberships lie using ‘.fit\_predict()’ and assigning it to an additional variable.

We can compare how each of the approaches work by comparing the final cluster results via plotting:



As you can see in our example problem the two converge to basically the same clusters. While this is great from a toy problem perspective, you will soon learn in Activity 2 that small changes to input parameters can lead to wildly different results!

Activity 2: Compare K-Means with Hierarchical Clustering

**Scenario:** Your manager is asking you to evaluate different clustering approaches to see which works best for your use case. You need to be able to report back on which clustering method is most relevant based off the data you have available.

**Aim:** Implement K-Means and Hierarchical Clustering to the same dataset, and see which approach ends up being more accurate or easier for you to use. Try different combinations of Sci-Kit Learn implementations as well as through using helper functions in SciPy/NumPy.

K-Means vs Hierarchical Clustering

Now that we have expanded upon our base of understanding how K-Means works, it is important to explore where Hierarchical Clustering fits into the picture. As mentioned in the Linkage Criteria section there is some potential direct overlap when it comes to grouping data points together through centroids. Universal to all of the approaches mentioned so far is also the use of a distance function to determine similarity. Due to our in-depth exploration in the past chapter we have kept using Euclidean distance, but understand that any distance function can be used to determine similarity.

In practice here are some quick highlights where you may choose one clustering method over another:

* Hierarchical clustering benefits from not needing to pass in an explicit “K” number of clusters a priori. You can find all potential clusters and decide where clusters are after the fact.
* K-Means clustering benefits from a simplicity perspective – often times in business use cases there is a challenge to find methods that can be explained to non-technical audiences but still be accurate enough to generate quality results. K-Means can easily fill this niche.
* Hierarchical clustering has more parameters to tweak than K-Means when it comes to dealing with abnormally shaped data. While K-Means is great at finding discrete clusters, it can falter when it comes to mixed clusters. By tweaking the parameters in Hierarchical clustering, you may find better results.
* Vanilla K-Means works by instantiating random centroids and finding closest points to those centroids. If they are randomly instantiated in areas of the feature space that are quite far away from your data then it can end up taking quite some time to converge, or it may never even get to that point. Hierarchical clustering is less prone to fall prey to this weakness.

Summary

In this chapter we discussed how hierarchical clustering works and where it may be best employed. In particular we discussed aspects of how clusters can be subjectively chosen after the fact through evaluation of a dendrogram plot. This is a huge leg up compared to K-Means if you have absolutely no idea of what you’re looking for in the data. Two key parameters that drive the success of hierarchical clustering were also discussed: agglomerative vs divisive, and linkage criteria. Agglomerative clustering takes a bottoms-up approach by recursively grouping nearby data together until one large cluster results. Divisive clustering takes a top-down approach by starting with the one large cluster and recursively breaking it down until each data point falls into its own “cluster.” Divisive clustering has the potential to be more accurate since it has a complete view of the data from the start, however it adds a layer of complexity that can decrease stability and increase runtime.

Linkage criteria grapple with the concept of how distance is calculated between candidate clusters. We have explored how centroids can make an appearance again beyond K-Means, as well as Single and Complete Linkage criteria. Single Linkage finds cluster distances by comparing the closest points in each cluster, while Complete Linkage finds cluster distances by comparing the further points in each cluster.

With the intuition built in this chapter you are now able to evaluate how both K-Means and Hierarchical Clustering can best fit the challenge you are working on.

Practice Questions

1. The two approaches to Hierarchical Clustering are Agglomerative and Degenerative
   1. True
   2. False
2. Which of these linkage criteria can be used to determine distances between clusters?
   1. Complete
   2. Single
   3. Centroid
   4. All of the above
3. Hierarchies are the most efficient way to convey information about a data set
   1. True
   2. False
4. How is Hierarchical Clustering a better approach to clustering than K-Means?
   1. It is faster than K-Means
   2. You don’t need to determine number of clusters a priori
   3. It handles simple data sets in a more efficient manner
   4. All of the above
5. How is K-Means a better approach to clustering than Hierarchical Clustering?
   1. It is faster than Hierarchical Clustering
   2. It is slower but more accurate
   3. You don’t need to determine what K is a priori
   4. K-Means is much more complex but more accurate
6. A hierarchy shows the relationships from parent nodes to their children nodes
   1. True
   2. False
7. In a hierarchy, children nodes can also themselves be parents to other children nodes
   1. True
   2. False
8. Why are dendrograms helpful in determining clusters?
   1. It shows how correlated features in the data set are
   2. They give an idea how clusters could be formed with which members
   3. If they fail, it’s a sign that the data is too complex for hierarchical clustering
   4. They aren’t helpful
9. Agglomerative clustering adopts the tops-down approach, while Divisive adopts the bottoms-up approach.
   1. True
   2. False
10. Hierarchical Clustering in sci-kit learn is more modular and should always be preferred over scipy implementations.
    1. True
    2. False