# 7.2 Hierarchical Clustering

May 9, 2019

## 1 Chapter 7.2 Hierarchical Clustering

Another approach to clustering is **hierarchical clustering**. Unlike $k$-means, hierarchical clustering does not require specifying the number of clusters in advance. It will learn an entire *hierarchy* of models: a 1-cluster model, a 2-cluster model, and so on, up to an $n$-cluster model in which each of the $n$ observations is in its own cluster. In this hierarchy, the $(k+1)$-cluster model is obtained by splitting one of the clusters in the $k$-cluster model in two.

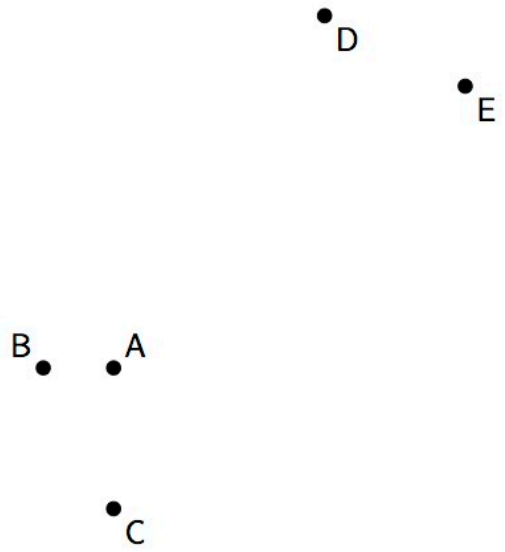
How do we learn a hierarchy of clusters? There are two general approaches:

- **Divisive** ("top-down"): Start with all of the observations in a single cluster. Recursively split clusters into smaller clusters, until every observation is in its own cluster.
- **Agglomerative** ("bottom-up"): Start with each observation in its own cluster. Recursively merge clusters into larger clusters, until every observation is in a single cluster.

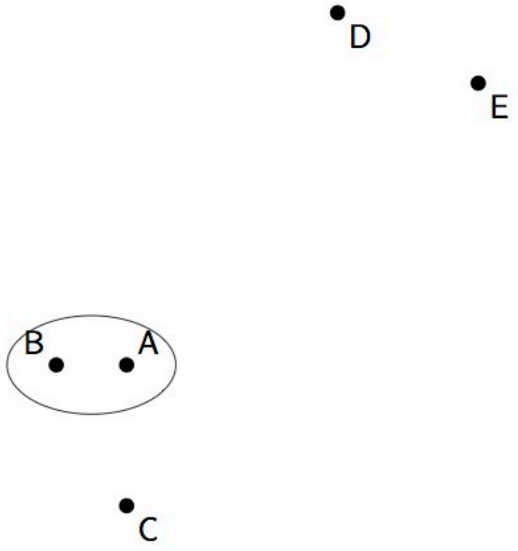
In this section, we will focus on *agglomerative* hierarchical clustering algorithms, which are more commonly used and easier to understand.

## 2 The Algorithm, in Pictures

Suppose we have a dataset consisting of the 5 points shown below.



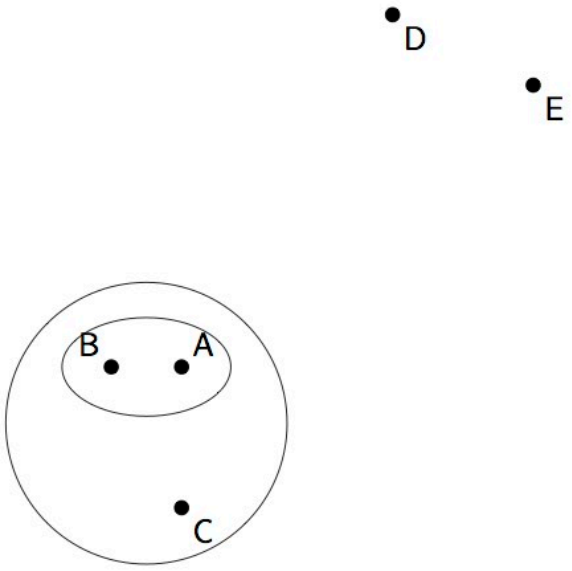

In agglomerative hierarchical clustering, each point starts out in its own cluster, and we successively merge clusters. The first step is clearly to merge A and B into one cluster, since they are the two closest points.



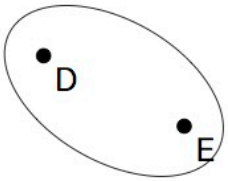

We now have 4 clusters: {A, B}, {C}, {D}, {E}. What do we merge next? Here things become somewhat tricky. So far, we have only defined the distance between two *points*. But now we need a way to measure the distance between a *cluster* and a *point*, and more generally, the distance

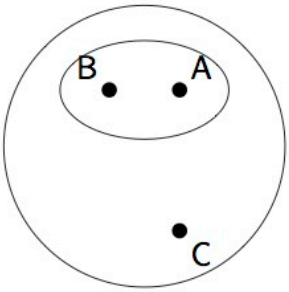
between two *clusters*. It turns out that there are several ways to extend a distance metric between points to a distance metric between clusters. These different extensions are called **linkages**. The choice of linkage can have a significant influence on the clusters that are obtained. We will return to this point later.

For now, it seems that no matter how you look at it, the "closest" clusters are clearly {C} and {A, B}, so let's merge them.

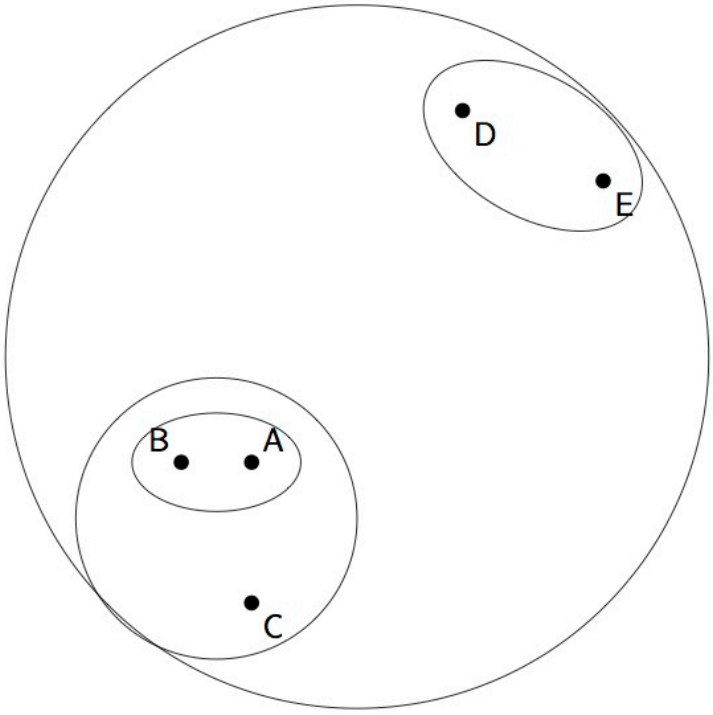



Next, we merge D and E into one cluster.



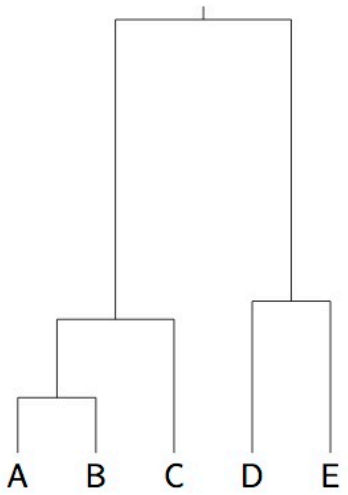




And finally, we merge the two clusters into one large cluster containing all 5 observations.

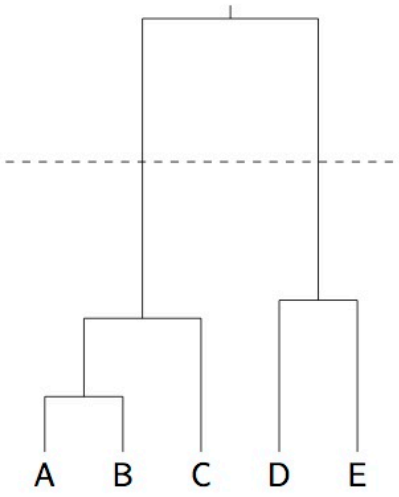


## 3 Dendrograms: A Way to Visualize Hierarchical Clustering

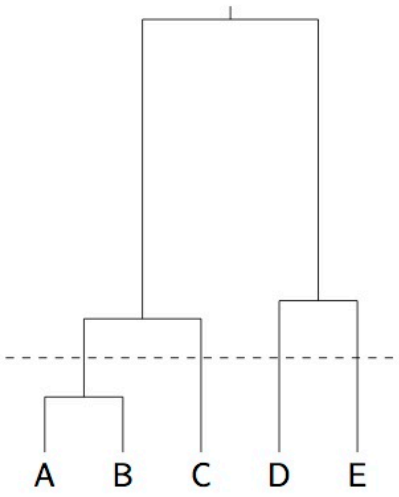
A **dendrogram** (from the Greek word *dendron*, meaning "tree") is a visualization of a hierarchy of clusters. The individual observations are listed at the tips of the branches. Each U-shaped merger of two branches on this diagram corresponds to a merger of two clusters. The height at which the branches merge represents the distance between the clusters before they merged.




Suppose we want two clusters. We would cut the dendrogram at a height where there are exactly two branches remaining. We can see easily from the dendrogram that the resulting clusters are {A, B, C} and {D, E}.




Similarly, if we wanted four clusters, we would cut the dendrogram at a height where there are exactly four branches remaining. We can see from the dendrogram that the resulting clusters would be {A, B}, {C}, {D}, {E}.



### 3.1 More about Linkages

Let's return to the question of how to measure the distance between two clusters. A distance metric between clusters, which we will denote by $D$, is derived from a distance metric between points, which we will denote by $d$. (The distance metric $d$ might be, for example, Euclidean distance, Manhattan distance, etc.). There are many ways to extend a distance metric for points to a distance metric for clusters. These different extensions are called **linkages**. Some common ones are:

- **single linkage:** the distance between the two *closest* points in the clusters

$$D(A, B) = \min\{d(a, b) : a \in A, b \in B\}$$

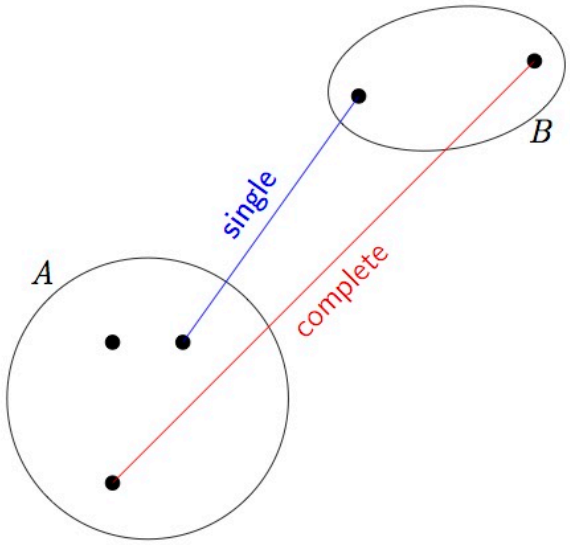
- **complete linkage:** the distance between the two *furthest* points in the clusters

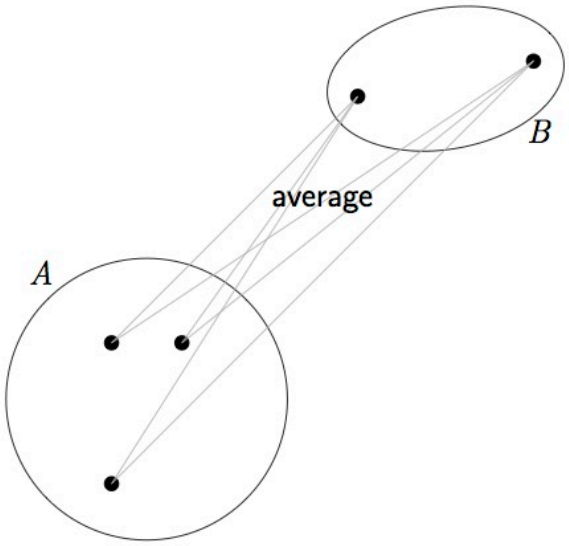
$$D(A, B) = \max\{d(a, b) : a \in A, b \in B\}$$

- **average linkage:** the average of all pairwise distances between points in the clusters

$$D(A, B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$$

These three linkages are illustrated in the diagrams below.






## 4 Hierarchical Clustering in *scikit-learn* and *scipy*

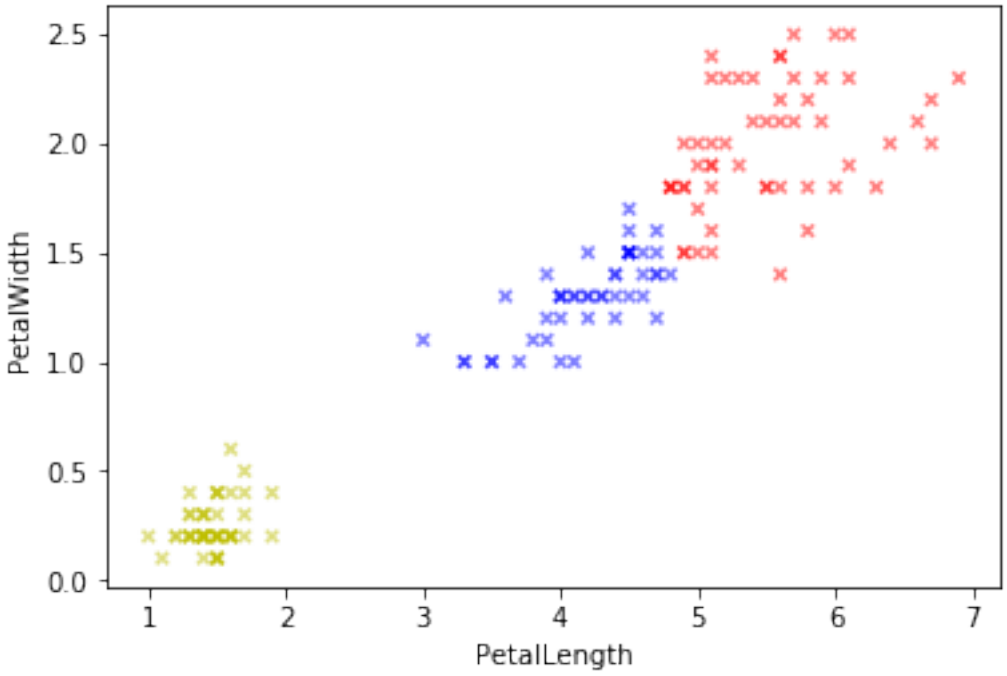
Although hierarchical clustering is intuitive, it is not straightforward to implement, so we will not pursue that here. Instead, we will use two professional implementations of hierarchical clustering: *scikit-learn*'s and *scipy*'s.

We start with *scikit-learn* because its API is probably more familiar to you by now. In *scikit-learn*, you have to specify the number of clusters upfront. Let's fit a hierarchical clustering model to the iris data set.

```
In [1]: %matplotlib inline
        import pandas as pd
        iris_df = pd.read_csv("https://raw.githubusercontent.com/dlsun/data-science-book/master
        X_train = iris_df[["PetalLength", "PetalWidth"]]

In [2]: from sklearn.cluster import AgglomerativeClustering

        model = AgglomerativeClustering(n_clusters=3,
                                        affinity="euclidean",
                                        linkage="complete")
        model.fit(X_train)
        clusters = model.labels_

        clusters = pd.Series(clusters).map({
            0: "r",
            1: "b",
            2: "y"
```

```
})

X_train.plot.scatter(x="PetalLength", y="PetalWidth",
                     c=clusters, marker="x", alpha=.5)
```

```
Out[2]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc7509d70f0>
```



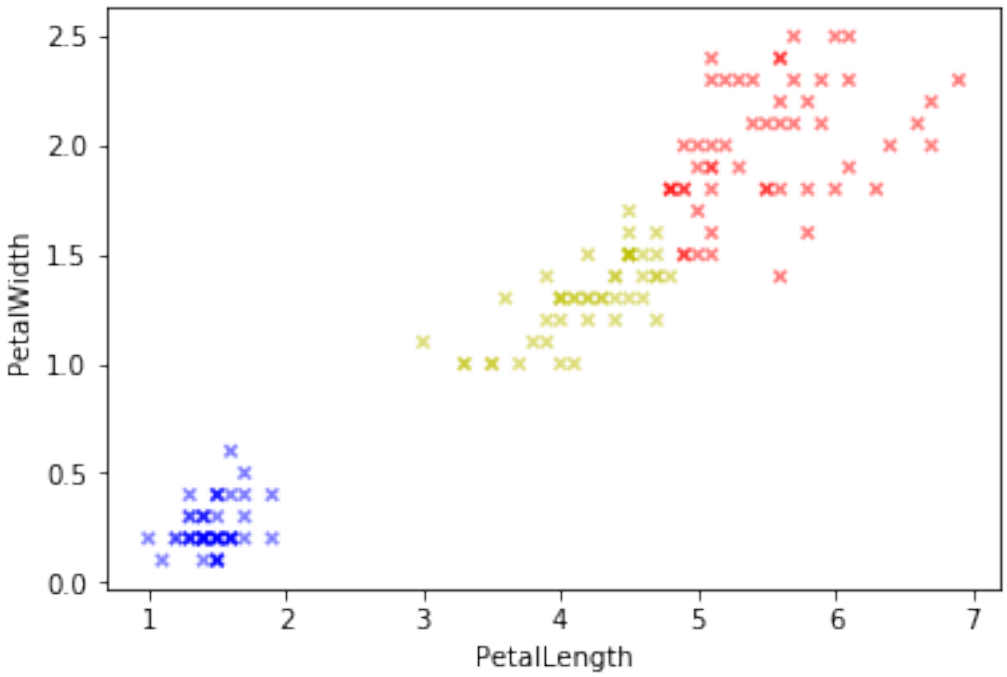

The disadvantage of *scikit-learn* is that it does not draw dendrograms. For dendrograms, we turn to `scipy`. The hierarchical clustering API in `scipy` works as follows:

- First, you create a *linkage matrix* that encodes the clustering.
- Then, you call `fcluster()` on this linkage matrix to get the cluster assignments or `dendrogram()` to get a plot of the dendrogram. (This function also returns a bunch of other output which you probably do not need; you can suppress the output by adding a semicolon to the end of the line.)

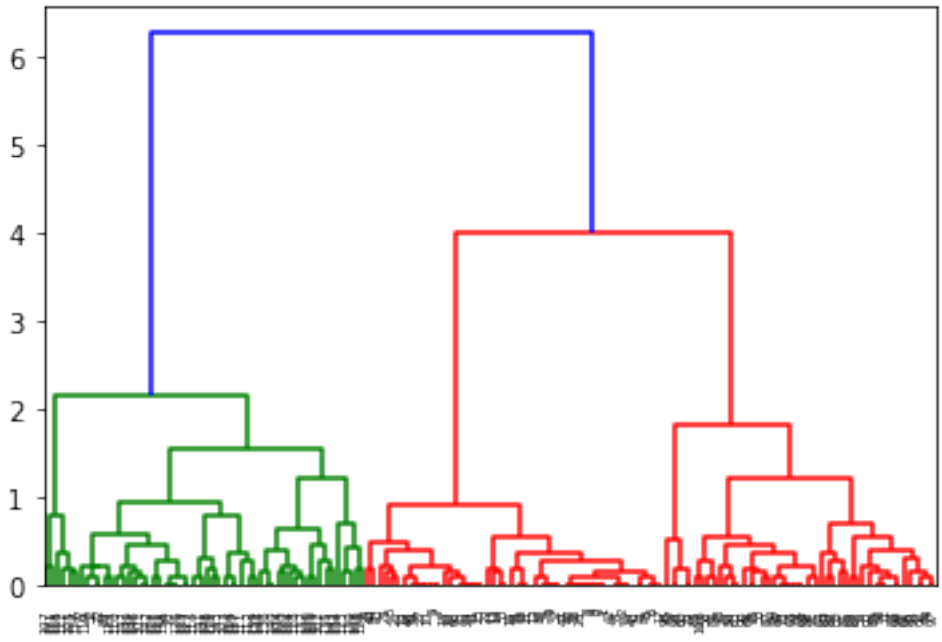
```
In [3]: from scipy.cluster.hierarchy import fcluster, linkage, dendrogram

        Z = linkage(X_train, method="complete", metric="euclidean")
        clusters = fcluster(Z, 3, criterion="maxclust")

        clusters = pd.Series(clusters).map({
            1: "r",
            2: "b",
            3: "y"
        })
```

```
X_train.plot.scatter(x="PetalLength", y="PetalWidth",
                     c=clusters, marker="x", alpha=.5)
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc74f158ba8>




In [5]: `dendrogram(Z);`



## 5 Exercises

**Exercise 1.** Fit a hierarchical clustering model to the Titanic passengers dataset (`https://raw.githubusercontent.com/dlsun/data-science-book/master/data/titanic.csv`). You are free to choose which features to include (but include both categorical and quantitative features) and the linkage function. Then, choose a number of clusters that seems appropriate. Look at the profiles of the passengers in each cluster. Can you come up with an "interpretation" of each cluster based on the passengers in it?

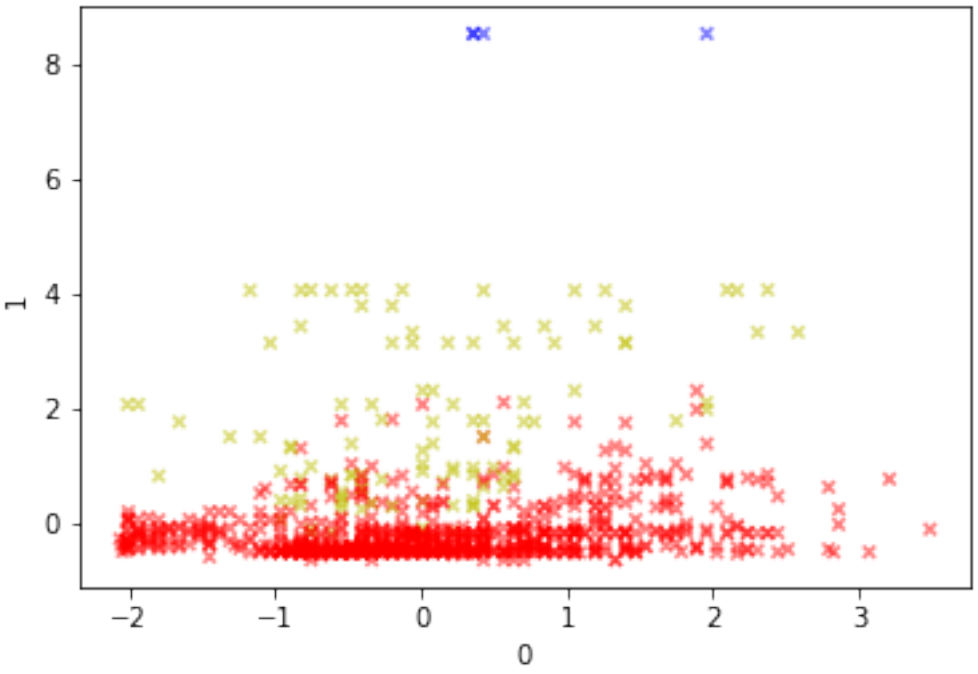
```
In [7]: titanic = pd.read_csv("https://raw.githubusercontent.com/dlsun/data-science-book/master
        titanic.head()
```

```
Out[7]:    pclass  survived                                             name     sex  \
        0       1         1                    Allen, Miss. Elisabeth Walton  female
        1       1         1                   Allison, Master. Hudson Trevor    male
        2       1         0                     Allison, Miss. Helen Loraine  female
        3       1         0             Allison, Mr. Hudson Joshua Creighton    male
        4       1         0  Allison, Mrs. Hudson J C (Bessie Waldo Daniels)  female

              age  sibsp  parch  ticket      fare    cabin embarked boat   body  \
        0  29.0000      0      0   24160  211.3375       B5        S    2    NaN
        1   0.9167      1      2  113781  151.5500  C22 C26        S   11    NaN
        2   2.0000      1      2  113781  151.5500  C22 C26        S  NaN    NaN
        3  30.0000      1      2  113781  151.5500  C22 C26        S  NaN  135.0
        4  25.0000      1      2  113781  151.5500  C22 C26        S  NaN    NaN

                             home.dest
        0                   St Louis, MO
        1  Montreal, PQ / Chesterville, ON
        2  Montreal, PQ / Chesterville, ON
        3  Montreal, PQ / Chesterville, ON
        4  Montreal, PQ / Chesterville, ON
```

```
In [25]: from sklearn.feature_extraction import DictVectorizer
         from sklearn.preprocessing import StandardScaler

         X_train_dict = titanic[["pclass", "survived", "sex", "age", "fare"]].dropna().to_dict

         vec = DictVectorizer(sparse=False)
         vec.fit(X_train_dict)
         X_train = vec.transform(X_train_dict)

         scaler = StandardScaler()
         scaler.fit(X_train)
         X_train_sc = scaler.transform(X_train)
```

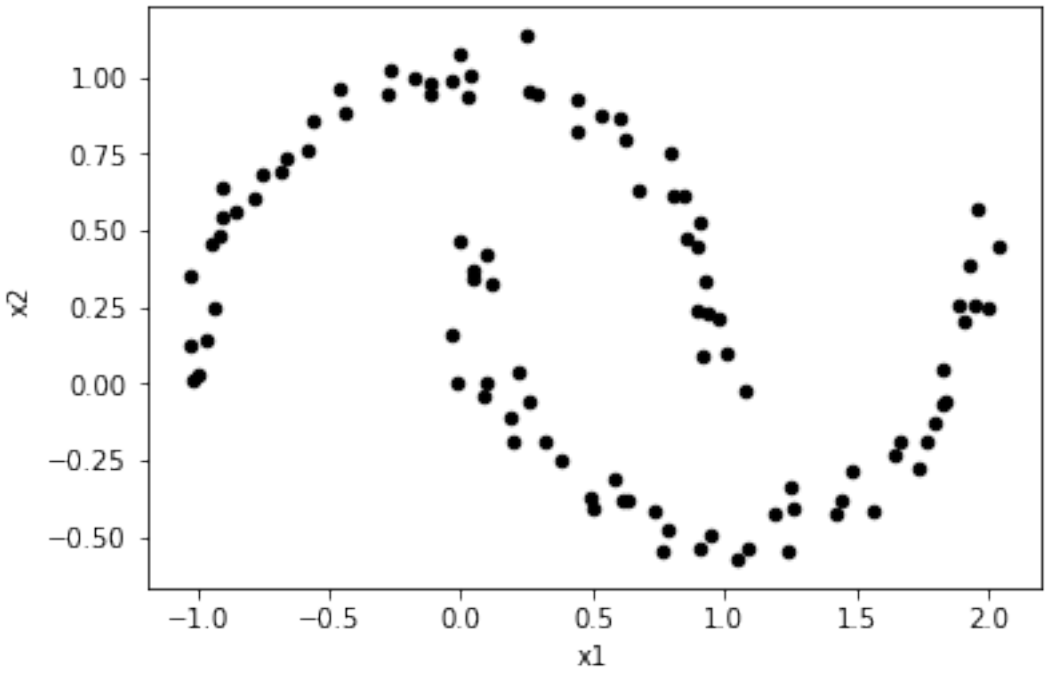
```
model = AgglomerativeClustering(n_clusters=3,
                                affinity="euclidean",
                                linkage="complete")
model.fit(X_train_sc)
clusters = model.labels_

clusters = pd.Series(clusters).map({
    0: "r",
    1: "b",
    2: "y"
})

df = pd.DataFrame(X_train_sc)

df.plot.scatter(x=0, y=1,
                c=clusters, marker="x", alpha=.5)
```

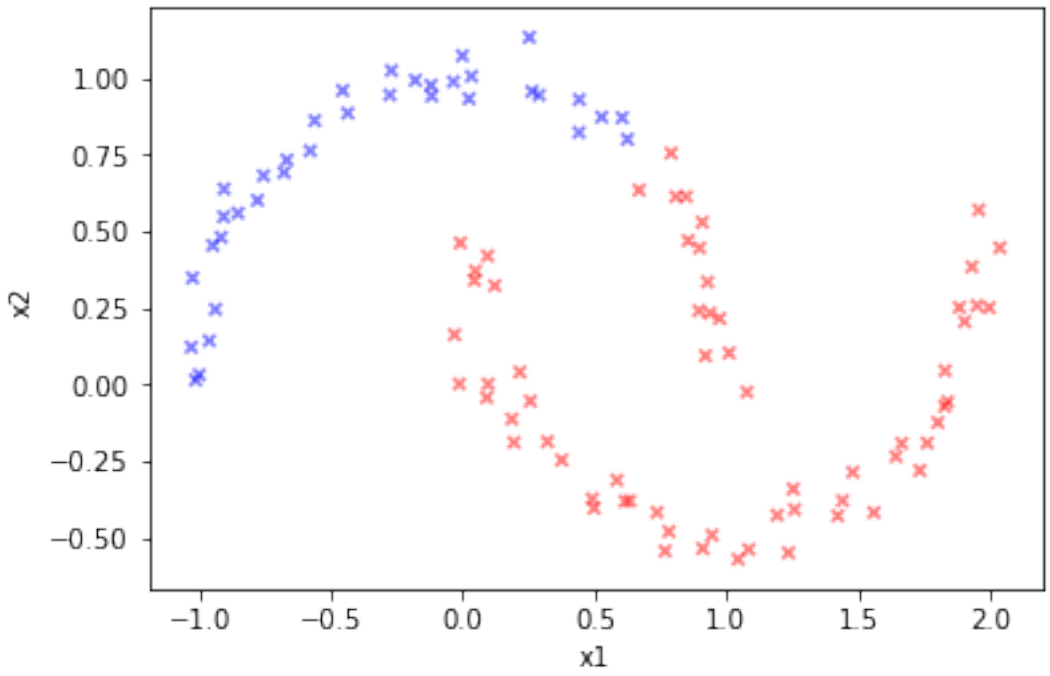
```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc74b5c4358>
```

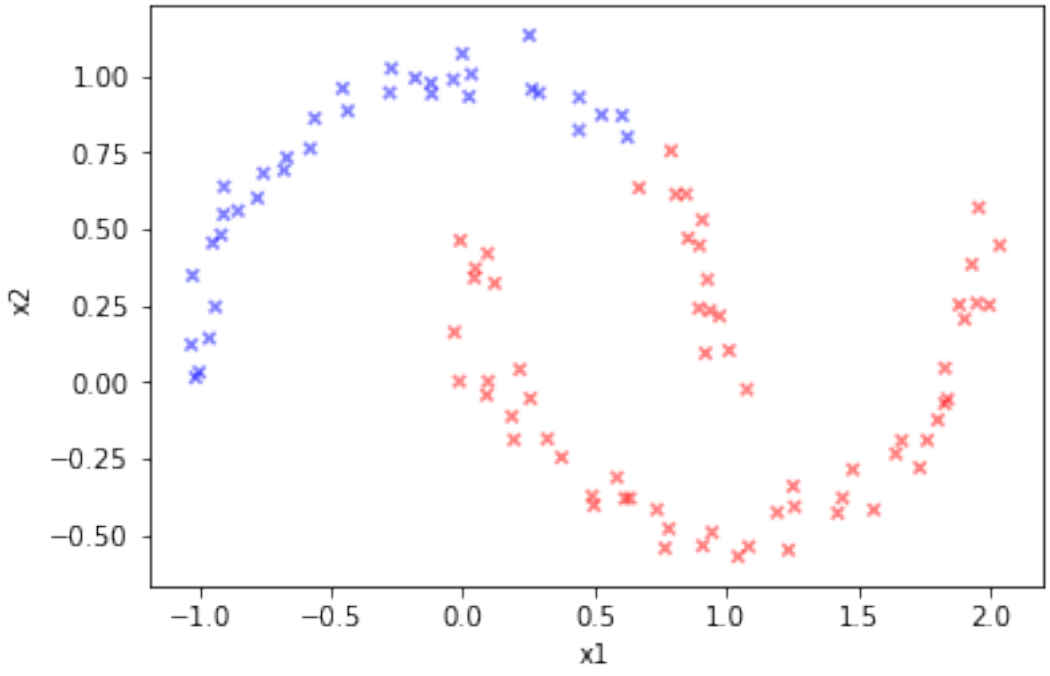


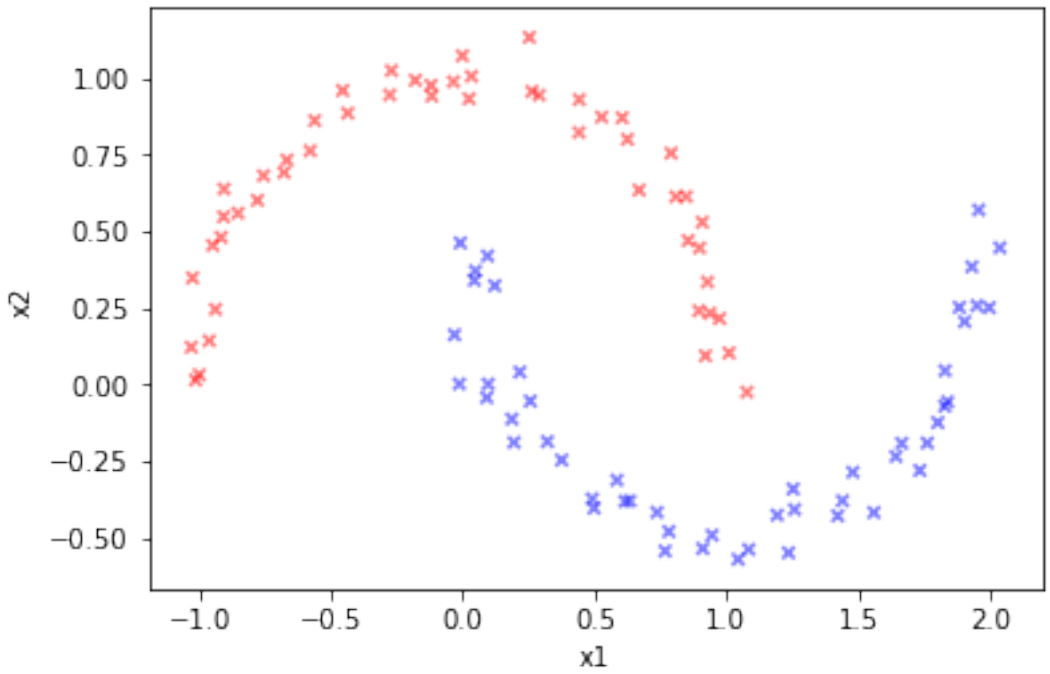

**Exercise 2.** The code below reads in the "two moons" dataset, a synthetic dataset that is used to evaluate clustering algorithms. What clusters do you think hierarchical clustering will find if you use single linkage? What if you use average linkage? Once you have a hypothesis for each type of linkage, test out your hypothesis by fitting the model to this dataset and plotting the resulting clusters.

In [30]:
```
# TYPE YOUR CODE HERE
moons = pd.read_csv("https://raw.githubusercontent.com/dlsun/data-science-book/master/
moons.plot.scatter(x="x1", y="x2", color="k")

X_train = moons[["x1", "x2"]]

for link in ["complete", "average", "single"]:
    model = AgglomerativeClustering(n_clusters=2,
                                    affinity="euclidean",
                                    linkage=link)
    model.fit(X_train)
    clusters = model.labels_

    clusters = pd.Series(clusters).map({
        0: "r",
        1: "b"
    })

    X_train.plot.scatter(x="x1", y="x2",
                         c=clusters, marker="x", alpha=.5)
```




1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
x2
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x1


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
x2
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x1




```
In [27]: moons.head()
```

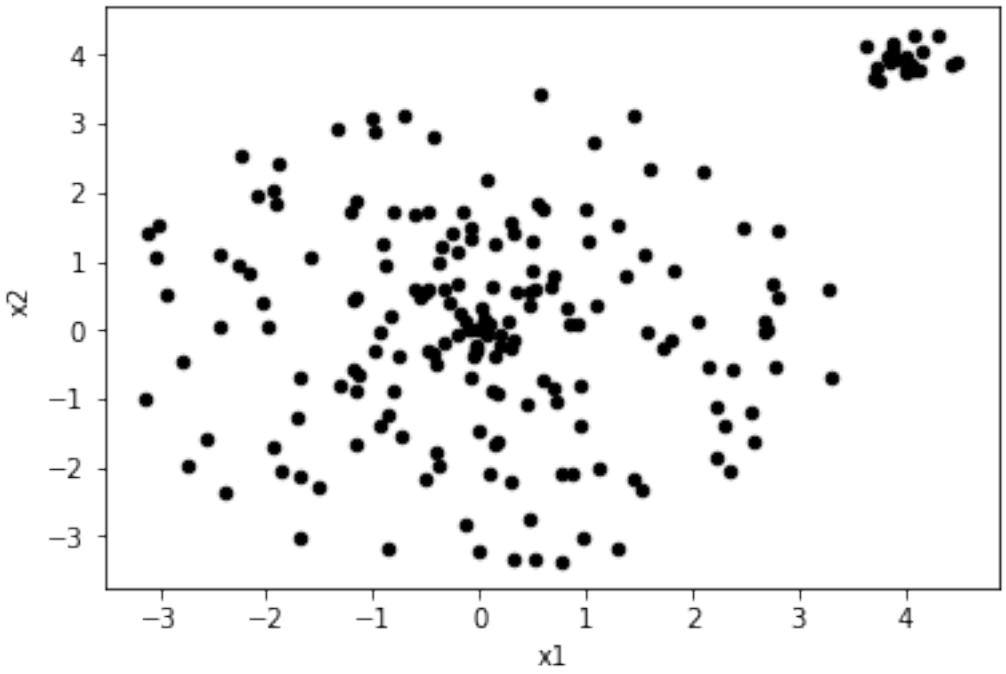
```
Out[27]:          x1        x2
         0  0.615605 -0.380346
         1  1.254054 -0.339894
         2 -0.853624  0.560419
         3  0.910649 -0.534494
         4  1.905174  0.206167
```

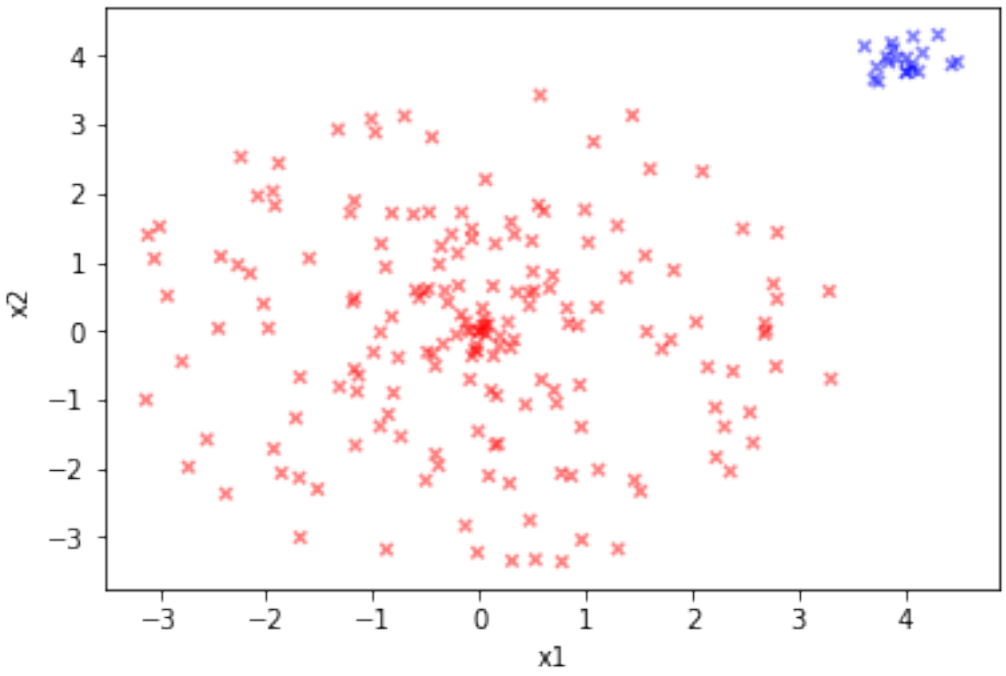
**Exercise 3.** The code below reads in the "satellite" dataset, a synthetic dataset that is used to evaluate clustering algorithms. What clusters do you think hierarchical clustering will find if you use single linkage? What if you use average linkage? Once you have a hypothesis for each type of linkage, test out your hypothesis by fitting the model to this dataset and plotting the resulting clusters.

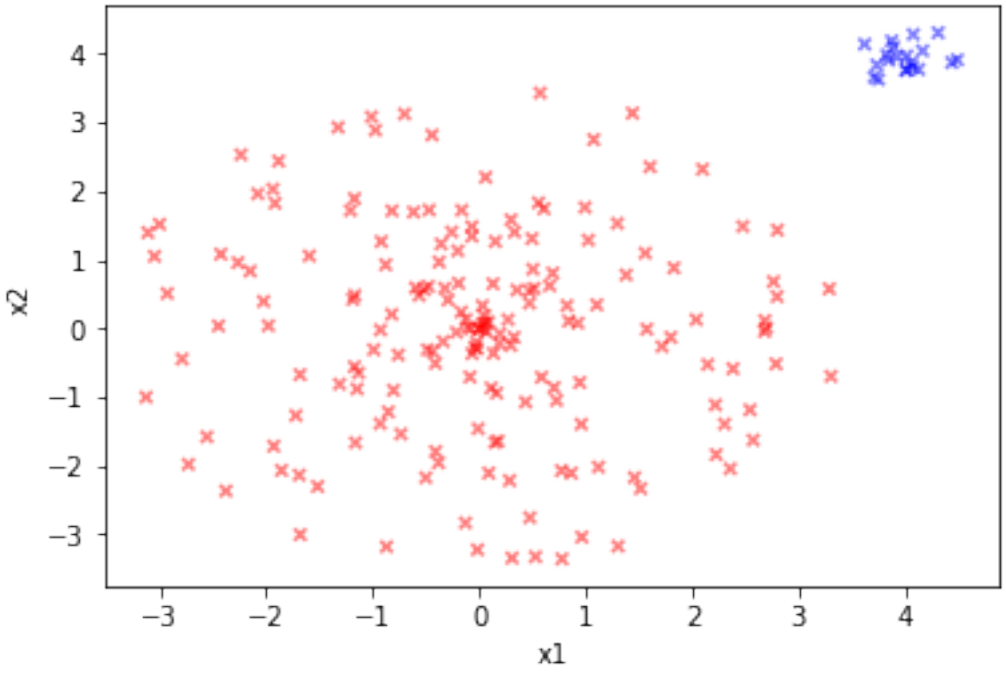
*Food for thought:* How do the results here compare to *k*-means?

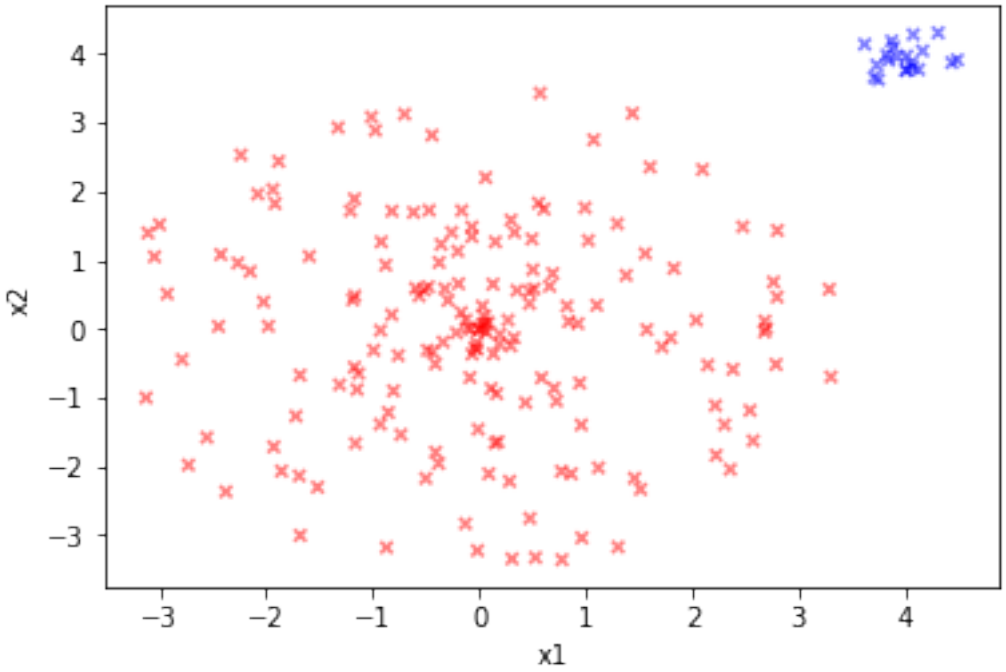
```
In [33]: # TYPE YOUR CODE HERE
         satellite = pd.read_csv("https://raw.githubusercontent.com/dlsun/data-science-book/mas
         satellite.plot.scatter(x="x1", y="x2", color="k")

         X_train = satellite[["x1", "x2"]]

         for link in ["complete", "average", "single"]:
             model = AgglomerativeClustering(n_clusters=2,
                                             affinity="euclidean",
```

```
                                    linkage=link)
model.fit(X_train)
clusters = model.labels_

clusters = pd.Series(clusters).map({
    0: "r",
    1: "b"
})

X_train.plot.scatter(x="x1", y="x2",
                     c=clusters, marker="x", alpha=.5)
```




x2
x1
4
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
4


x2
x1
4
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
4




```
In [32]: satellite.head()
```

Out[32]:

| | x1 | x2 |
|---|---|---|
| 0 | 0.516149 | 0.580760 |
| 1 | -0.976030 | 2.887041 |
| 2 | 0.473403 | 0.547144 |
| 3 | -1.326761 | 2.928621 |
| 4 | -0.735777 | -1.542989 |