Chapter 1: Introduction to Unsupervised Learning

Unsupervised learning is one of the most exciting areas of development in machine learning today. If you have explored machine learning coursework before you are probably familiar with the common breakout of problems into either supervised or unsupervised learning. Supervised learning encompasses the problem set of having a labeled dataset that can be used to either classify (for example, predicting smoker or non-smoker if you’re looking at a lung health data set) or fit a regression line on (for example, predicting the sale price of a home based on how many bedrooms it has). This model most closely mirrors an intuitive human approach to learning. If you wanted to learn how to not burn your food with a basic understanding of cooking, you can build a dataset by putting your food on the burner and seeing how long it takes (input) for your food to burn (output). Eventually as you continue to burn your food you will build a mental model of when burning will occur and avoid it in the future. Development in supervised learning has been fast-paced in valuable, but it has since simmered down in recent years – many of the obstacles in the world of knowing your data have already been tackled.

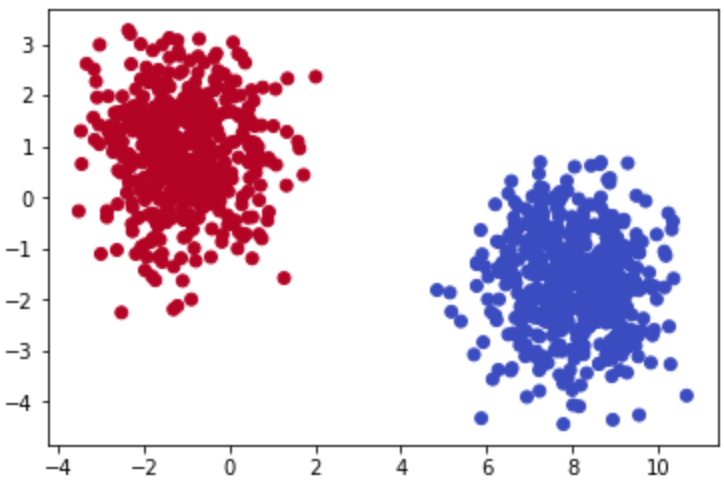
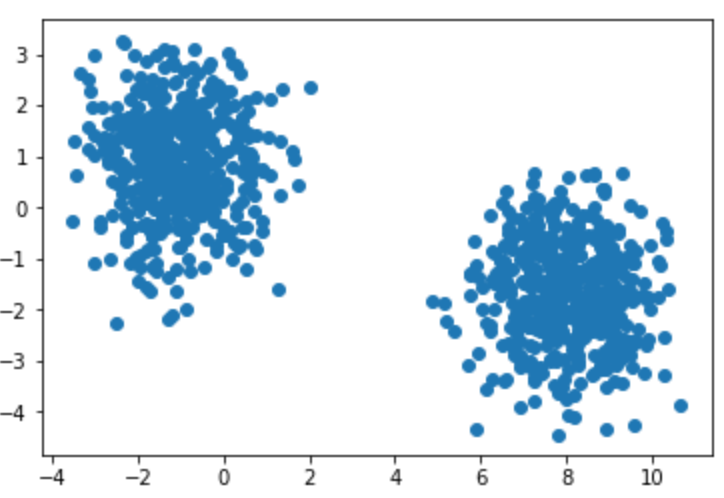


Figure 1: Differences between Unsupervised and Supervised Learning

Conversely, unsupervised learning encompasses the problem set of having a tremendous amount of data that is unlabeled. Building upon the previous examples, imagine you were just dropped on planet Earth with zero knowledge of how cooking works. You are given 100 days, a stove, and a fridge full of food without any instructions on what to do. Your initial exploration of a kitchen can go in infinite directions – on day 1 you may finally learn how to open the fridge, on day 30 you may learn that food can go on the stove, and on the many days in between you may have unwittingly made an edible meal. As you can see, trying to find meaning in a kitchen devoid of adequate informational structure leads to very noisy data that is completely irrelevant to actually preparing a meal. Unsupervised learning can be an answer to this problem. By looking back at your 100 days of data, clustering can be used to find patterns of similar days where a meal was produced, and you can easily review what you did those days. However, unsupervised learning isn’t a magical answer – by simply finding clusters it can just as likely help you find pockets of similar yet ultimately useless data.

This challenge is what makes unsupervised learning so exciting. How can we find smarter techniques to speed the process to finding clusters of information that are beneficial to our end goals?

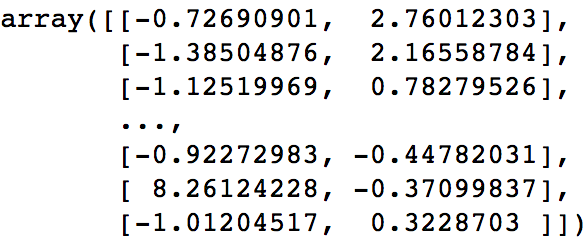
What is a cluster?



Figures 2 (left) and 3 (right): Scatter plots clearly showing clusters that exist among a provided dataset.

Rewinding a bit from where unsupervised learning fits into the larger machine learning environment, let’s begin by understanding the building blocks of clustering. The most basic definition finds clusters simply as groupings of similar data as subsets of a larger data set. Finding clusters can be immensely valuable in identifying items that are more similar to, and on the other end, quite different from each other.

To understand this idea further imagine that you were given a simple 1,000-row dataset by your employer that had 2 columns of numerical data (Figure 4). At first glance this dataset provides no real structure or understanding – confusing to say the least!



Figures 4: 2-dimensional raw data in a NumPy array

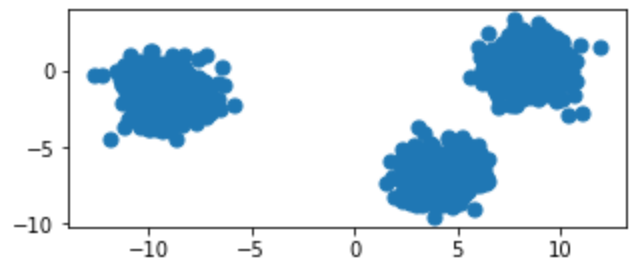
You begin by plotting the first column against the second column to get a better idea of what the data structure looks like. Figures 2 and 3 provide an extremely oversimplified example of what clustering is. Both figures display randomly generated number pairs (x,y coordinates) pulled from a gaussian distribution. Just by simply glancing at Figure 2 it should be plainly obvious where the clusters exist in your data - in real life it will never be this easy. Now that you know that the data can be clearly separated into 2 clusters, you can start to understand what differences exist between the two groups. There will be plenty of times where the cause of these differences will prove to be underwhelming, however the cases that have differences that you can take action on are extremely rewarding!

K-Means Clustering: Theory

Hopefully by now you can see that finding clusters is extremely valuable in a machine learning workflow. However, how can you actually find these clusters? One of the most basic yet popular approaches is by using a cluster analysis called K-Means Clustering. The workflow behind K-Means is actually quite intuitive – here is the no-math introduction to the clustering analysis:

1. Pick K centroids (K = expected distinct # of clusters)
2. Randomly place K centroids anywhere amongst your existing training data
3. Calculate the Euclidean distance from each centroid to all the points in your training data
4. Training data points get grouped in with their nearest centroid
5. Amongst data points grouped into each centroid, calculate the mean data point and move your centroid to that location.
6. Repeat the above process until convergence, or when membership in each group no longer changes.

And that’s it! Here is the above process laid out step by step with the previous simple cluster examples:



Original raw data charted on x,y coordinates

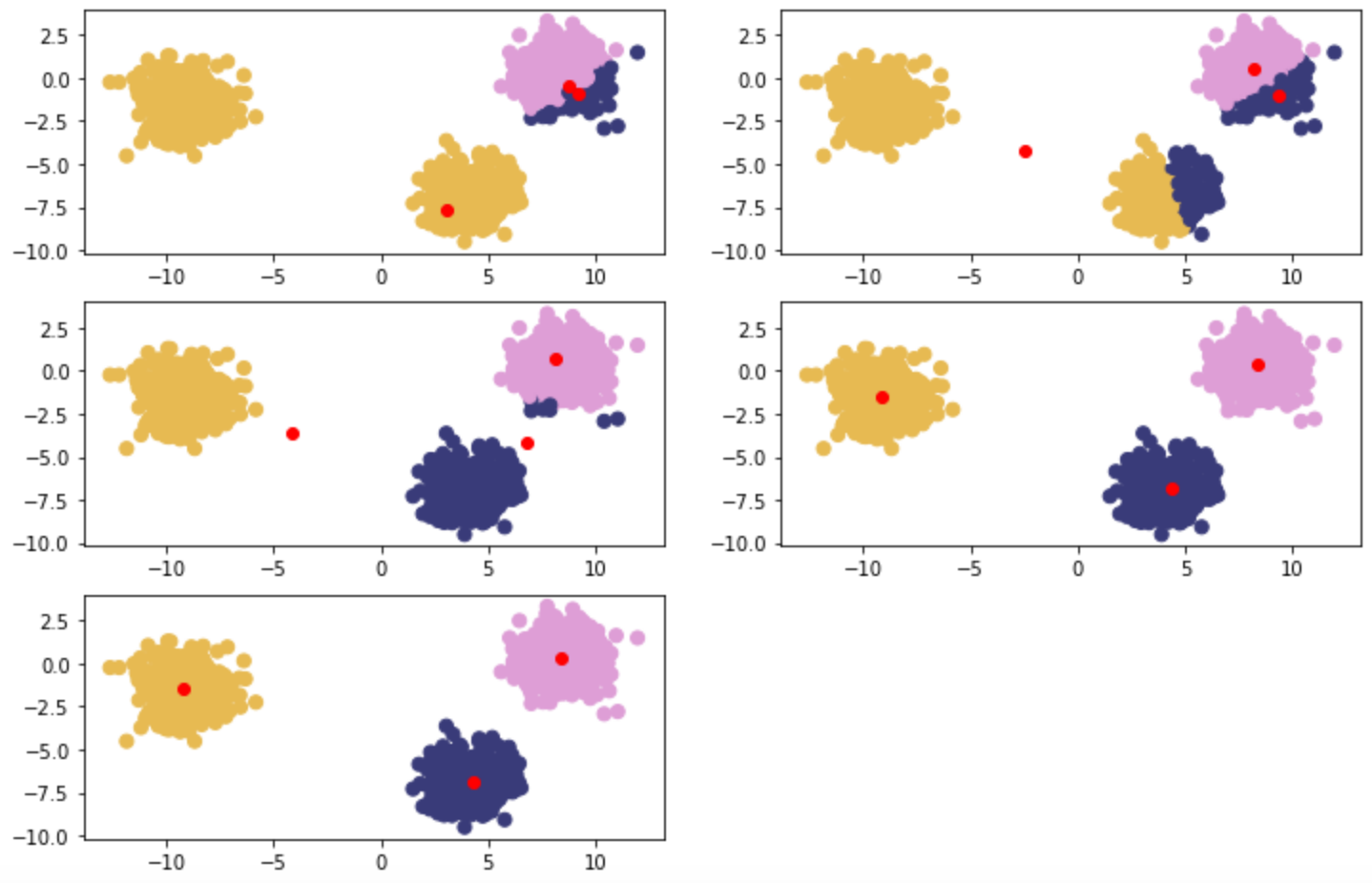


Figure 5: Clockwise from top left – Red points are randomly initialized centroids, and closest data points are assigned to groupings of each centroid (denoted by distinct color).

K-Means Clustering: In Depth

The above examples are clear to visualize when your data is only two-dimensional. This is for convenience to help drive the point home of how K-Means works and could lead you into a false understanding regarding how easy clustering is. In many of your own applications, your data will likely be orders of magnitude larger to the point that it cannot be perceived by visualization (anything beyond 3 dimensions will be imperceptible to human understanding). In the above examples you could mentally work out a few 2-dimensional lines to separate the data into their own groups. At higher dimensions you will need to be aided by a computer to find an n-dimensional hyperplane that adequately separates the dataset. In practice, this is where clustering methods such as K-Means provide significant value.

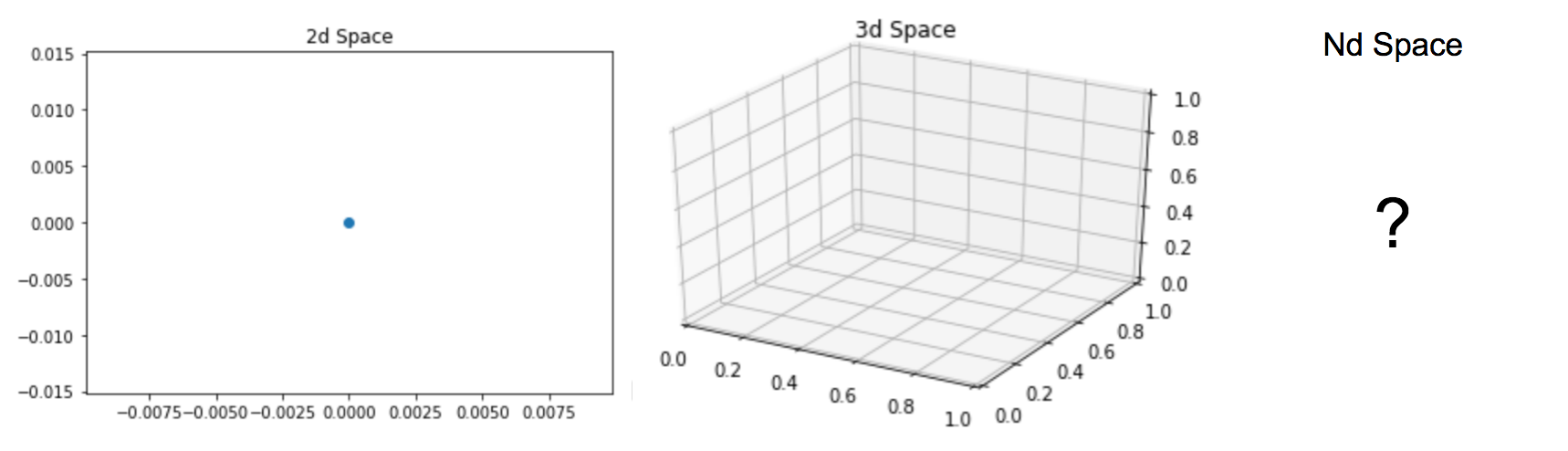


Figure 6: 2d plot, 3d plot, nd plot

To add more detail let’s walk through the example given in the Theory section again with some of the math that supports K-Means. The key component at play is the Euclidean distance formula:

Centroids are randomly set at the beginning as points in your n-dimensional space. Each of these centers is fed into the above formula as (a,b), and a point in your space is fed in as (x,y). Distances are calculated between each point and the coordinates of every centroid, with the centroid of smallest distance away chosen as the point’s group. This process can be seen below:

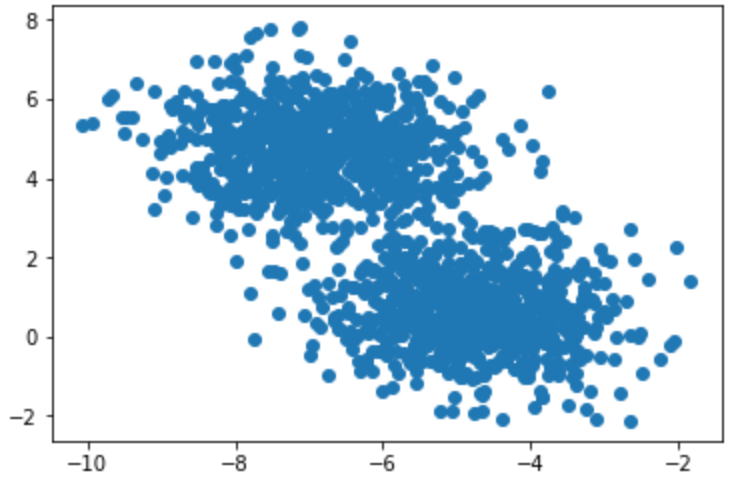
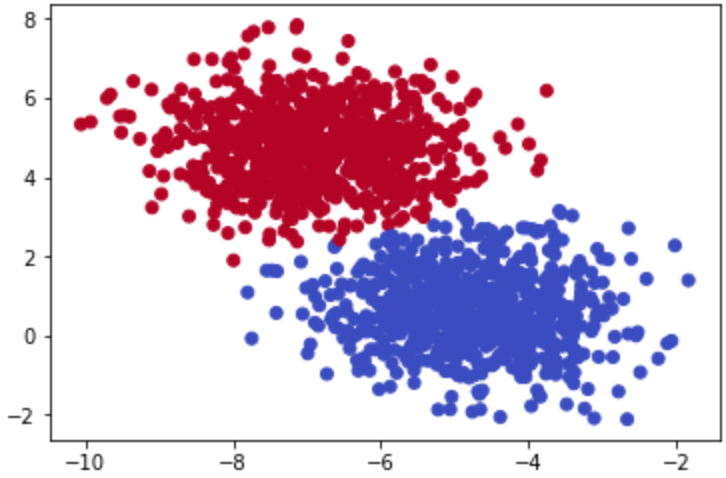
* Random Centroids: [ (2,5) , (8,3) , (4, 5) ]
* Arbitrary point x: (0, 8)
* Distance from point to each centroid: [ 3.61, 9.43, 5.00 ]
* Point x is assigned to Centroid 1.

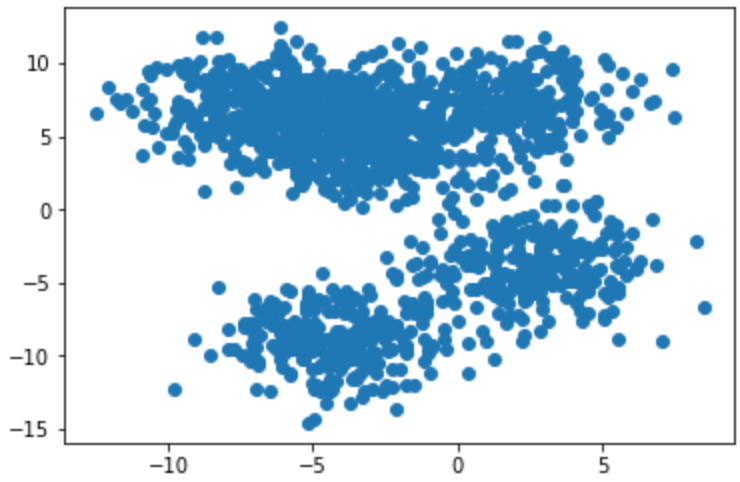
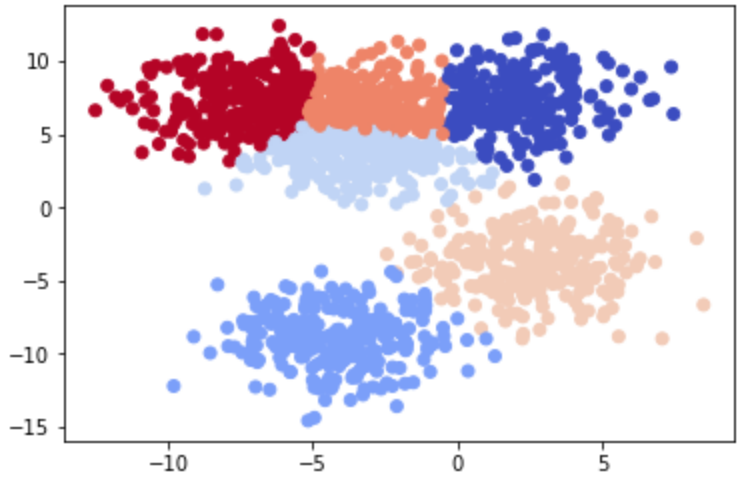
This process is repeated for every point in the dataset until each point is assigned to a cluster. After each point is assigned, the mean point is calculated among all of the points **within** each cluster. Calculation of mean among these points is the same as calculating a mean between single integers. An example of this can be seen here:

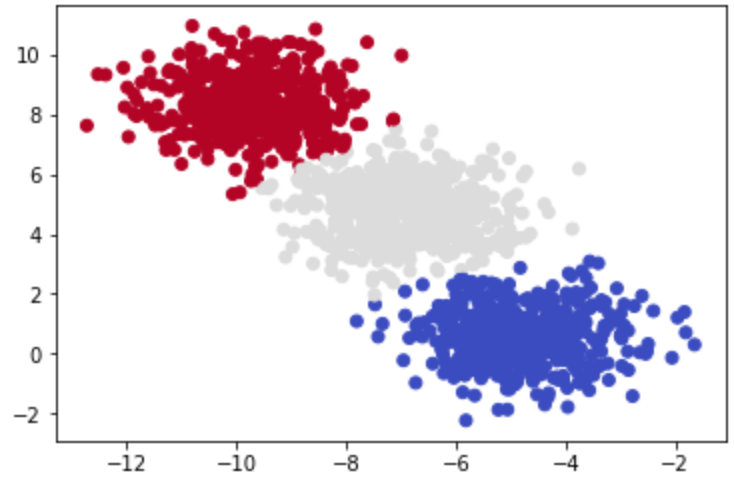
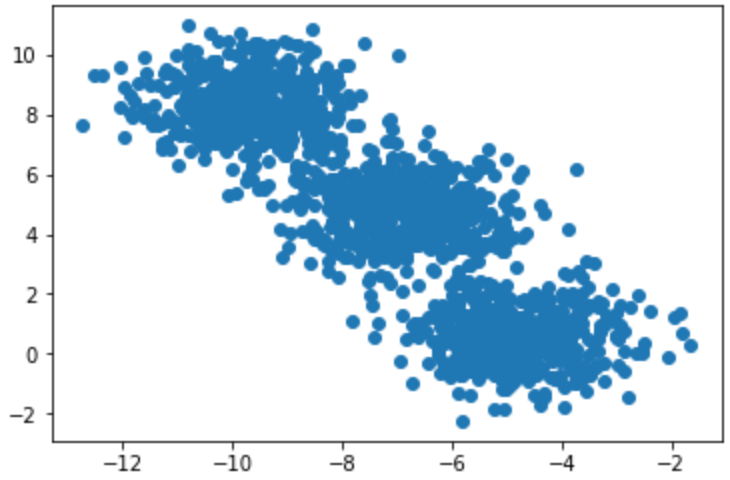
* Cluster 1 Membership: [ (0,8), (3,8), (3,4) ]
* Mean point calculation: [ (0+3+3)/3 , (8+8+4)/3 ] = (2, 6.67)
* Centroid for Cluster one becomes (2, 6.67)
* Repeat process until distance between new and old centroids = 0

Activity: Clustering

For this activity please look at the provided 2-dimensional graphs and see if you can identify where the clusters exist in the data:



Activity: K-Means Clustering

In this activity you will take what you have learned in the above sections and implement K-Means from scratch in Python. Please open your favorite editing platform and try the following:

* Using NumPy and the above Euclidean distance formula, write a function that calculates the distance between two coordinates.
* Write a function that calculates the distance from centroids to each of the points in your dataset and returns the cluster membership.
* Write a K-Means function that takes in a dataset and number of clusters (K) and returns final cluster centroids, as well as the data points that make up that cluster’s membership.

After implementing K-Means from scratch, apply your custom algorithm to the Iris dataset located here: <https://archive.ics.uci.edu/ml/datasets/iris>

Remove the classes supplied in this dataset and see if your K-Means algorithm can group the different Iris species into their proper groups just based on plant characteristics!