Lesson 1: Introduction to Clustering

Lesson Objectives

Present 3: Lesson Objectives

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

* Distinguish between supervised learning and unsupervised learning
* Explain the concept of clustering
* Implement k-means clustering algorithm using built-in Python packages

IntroductionSlide 4: Introduction

Have you ever been asked to take a look at some data and came up empty handed? Maybe you’re not familiar with the dataset, or maybe you don’t even know where to start. This may have been extremely frustrating and even embarrassing at times depending on who asked you to take care of the task.

You are not alone, and interestingly enough there are many times the data itself is simply too confusing to be made sense of. As you try and figure out what all those numbers in your spreadsheet mean, you’re most likely mimicking what many unsupervised algorithms do when they try to find meaning in data. The reality is that many datasets in the real world don’t have any rhyme or reason to them. You will be tasked with analyzing them with little background preparation. Don’t fret however – this course will prepare you to never be frustrated again when dealing with data exploration tasks.

Through this course we have developed some of the best in class content to help you understand how unsupervised algorithms work and where to use them. We cover some of the foundations of finding clusters in your data, how to reduce the size of your data so it’s easier to understand, and how each of these sides of unsupervised learning can be applied in the real world. We hope you will come away from this course with a strong real world understanding of unsupervised learning and what problems it can solve and those it cannot.

Thanks for joining us and we hope you enjoy the ride!

Discuss 5: If you were asked to understand a dataset, how would you do it? How many people in this class are familiar with supervised learning? What would you do if you got a dataset without labels, or targets, to train on?

Unsupervised Learning versus Supervised Learning

Present: 6: Unsupervised Learning versus Supervised 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 and 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 1.1: Differences between Unsupervised and Supervised Learning

Conversely, unsupervised learning encompasses the problem set of having a tremendous amount of data that is unlabeled. Labeled data in this case would be data that has a supplied “target” outcome that you are trying to find the correlation towards with supplied data (you know that you are looking for whether your food was burned in the preceding example). Unlabeled data is when you do not know what the “target” outcome is and you only have supplied input data.

Building upon the previous example, 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 10 you may finally learn how to open the fridge, on day 30 you may learn that food can go on the stove, and after 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?

Discuss 7: Name the fields where supervised or unsupervised learning is used

Clustering

Present 8: Clustering

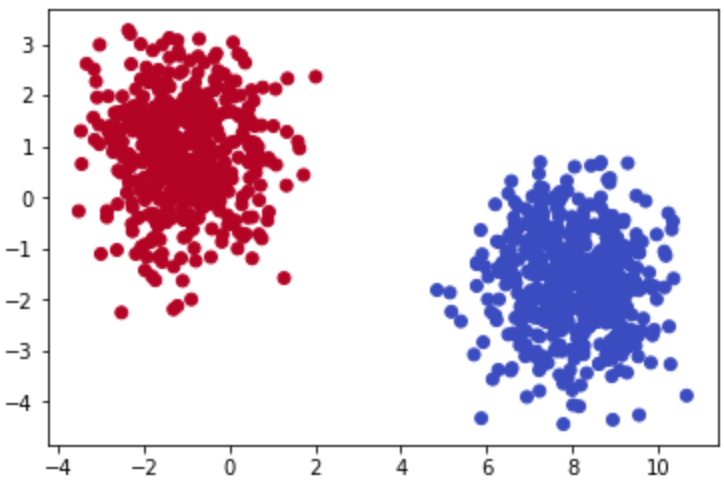
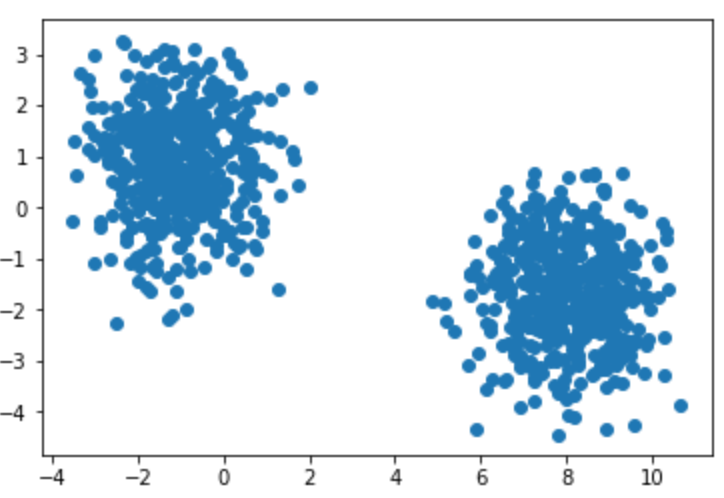
Being able to find groups of similar data that exist in your data set can be extremely valuable if you are trying to find its underlying meaning. If you were a store owner and you wanted to understand which of your customers are more valuable without a set idea of what “valuable” is, clustering is a great place to start to find patterns in your data. You may have a few high-level ideas of what denotes a valuable customer, but you aren’t entirely sure in the face of a large mountain of available data. Through clustering you can find commonalities among similar groups in your data. If you look deeper at a cluster of similar people you may learn that everyone in this group visits your website for longer periods of time than others. This can show you what value is and also provides a clean sample size for future supervised learning experiments.

Discussion 9: Can you give examples from past work experience where clustering has proven valuable? If you haven’t used it before, can you provide some examples of why knowing what records are similar in your data set may be valuable?

Identifying Clusters

Present 10: Identifying Clusters

Below figures provide an extremely oversimplified example of clustering.



Figures 1.2 (left) and 1.3 (right): Scatter plots clearly showing clusters that exist among a provided dataset.

Both figures display randomly generated number pairs (x,y coordinates) pulled from a Gaussian distribution. Just by simply glancing at figure 1.2 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.

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. As an example, imagine you had a room with 10 people in it and each person had a job either in finance or as a scientist. If you told all of the financial workers to stand together and all the scientists to do the same, you have effectively formed two clusters based on job types. Finding clusters can be immensely valuable in identifying items that are more similar to, and on the other end, quite different from each other.

Two-Dimensional Data

To understand this, imagine that you were given a simple 1,000-row dataset by your employer that had 2 columns of numerical data as follows:

Figures 1.4: 2-dimensional raw data in a NumPy array

At first glance this dataset provides no real structure or understanding – confusing to say the least!

A dimension in a dataset is another way of simply counting the number of features available. In most organized data tables you can view the number of features as the number of columns. So, using the 1,000-row dataset example of size (1,000 x 2), you will have 1,000 observations across 2 dimensions:

You begin by plotting the first column against the second column to get a better idea of what the data structure looks like. There will be plenty of times where the cause of differences between groups will prove to be underwhelming, however the cases that have differences that you can take action on are extremely rewarding!

Activity 1: Identifying the Clusters in Data

Present 12: Activity 1: Identifying the Clusters in Data. Continued till slide 15

Scenario: You are given two-dimensional plots. Please look at the provided two-dimensional graphs and identify the clusters, to drive the point home that machine learning is important

Aim: Without using any algorithmic approaches, identify where the clusters exist in the data.

Outcome: This exercise will help start to build the intuition around how we identify clusters using our own eyes and thought processes. As you complete the exercises think of the rationale why a group of data points should be considered a cluster versus not.

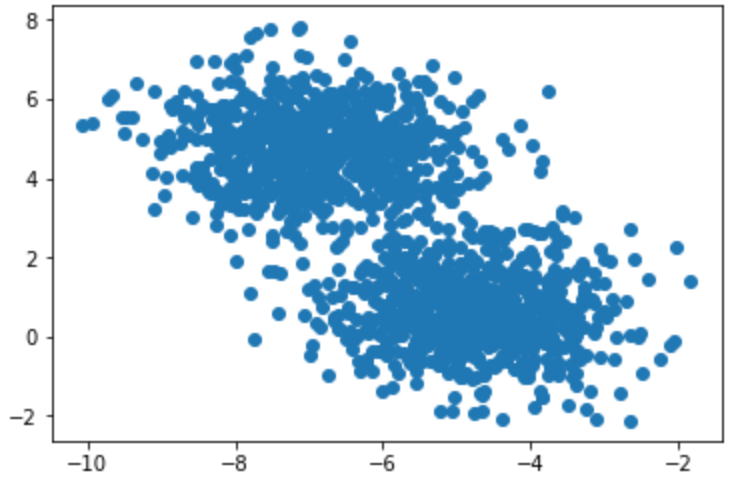


Figure1.5 Two-dimensional scatter-plot

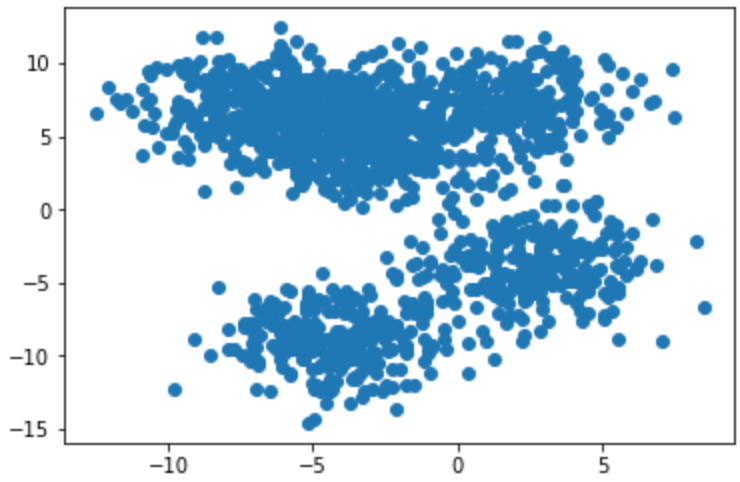


Figure1.6 Two-dimensional scatter-plot

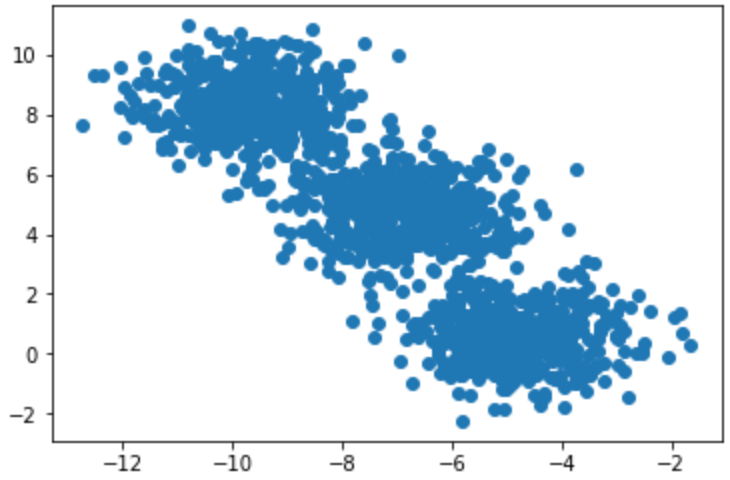


Figure1.7 Two-dimensional scatter-plot

Note:

The solution for this activity can be found on page XXX

Solution for Activity 1: Identifying the Clusters in Data

Scenario: You are given two-dimensional plots. Please look at the provided two-dimensional graphs and identify the clusters, to drive the point home that machine learning is important

Aim: Without using any algorithmic approaches, identify where the clusters exist in the data.

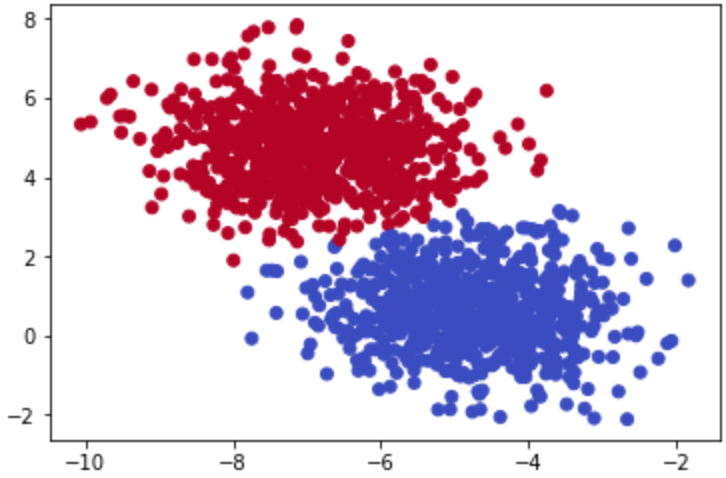


Figure 1.12: Clusters in the scatterplot for figure 1.5

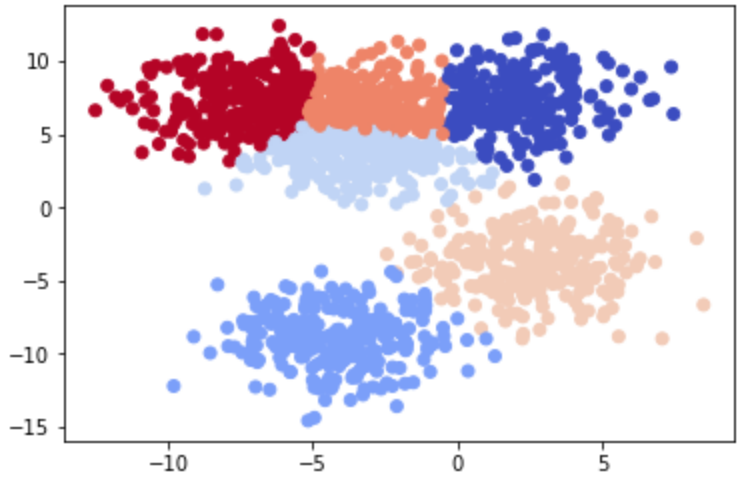


Figure 1.13: Clusters in the scatterplot for figure 1.6

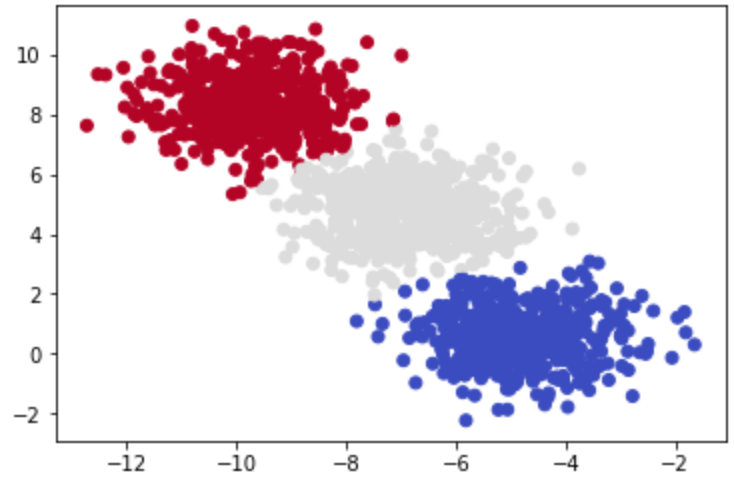


Figure 1.14: Clusters in the scatterplot for figure 1.7

Discuss 13: How did you derive clusters in Figure 1.6?

Introduction to K-Means Clustering

Present 14: Introduction to K-Means Clustering

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. K-Means works by searching for K clusters in your data and the workflow is actually quite intuitive – we will start with the no-math introduction to K-Means followed with an implementation in Python:

No-Math K-Means Walkthrough

Present 15: K-Means Clustering: No Math Intro. Continued till slide 21

1. Pick K centroids (K = expected distinct # of clusters)

Discussion 16: What is a centroid?

1. Randomly place K centroids anywhere amongst your existing training data
2. Calculate the Euclidean distance from each centroid to all the points in your training data
3. Training data points get grouped in with their nearest centroid
4. Amongst data points grouped into each centroid, calculate the mean data point and move your centroid to that location.
5. 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 a simple cluster example:

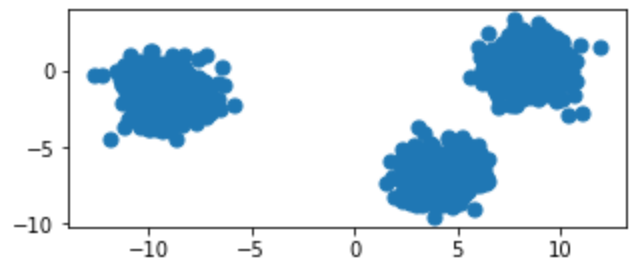


Figure 1.8 Original raw data charted on x,y coordinates

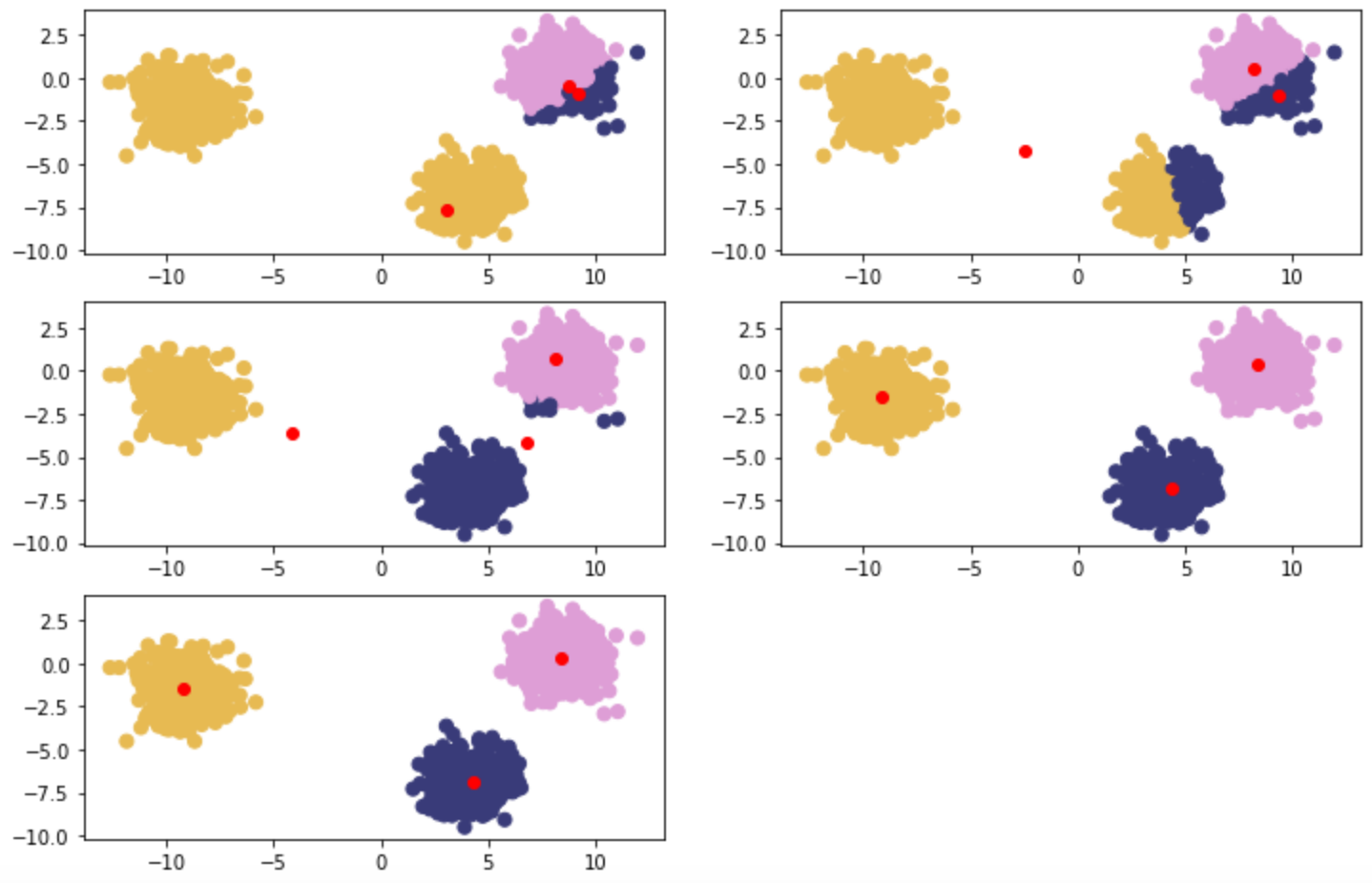


Figure 1.9: Clockwise from top left – Red points are randomly initialized centroids, and closest data points are assigned to groupings of each centroid (denoted by distinct color).

K-Means Clustering In-Depth Walkthrough

Present 17: K-Means Clustering In-Depth Walkthrough

To understand K-Means at a deeper level, let’s walk through the example given in the introductory section again with some of the math that supports K-Means. The key component at play is the Euclidean distance formula:



Figure 1.11: 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.

The process is as follows:

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

Deeper Dimensions

Preceding 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 previous 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.

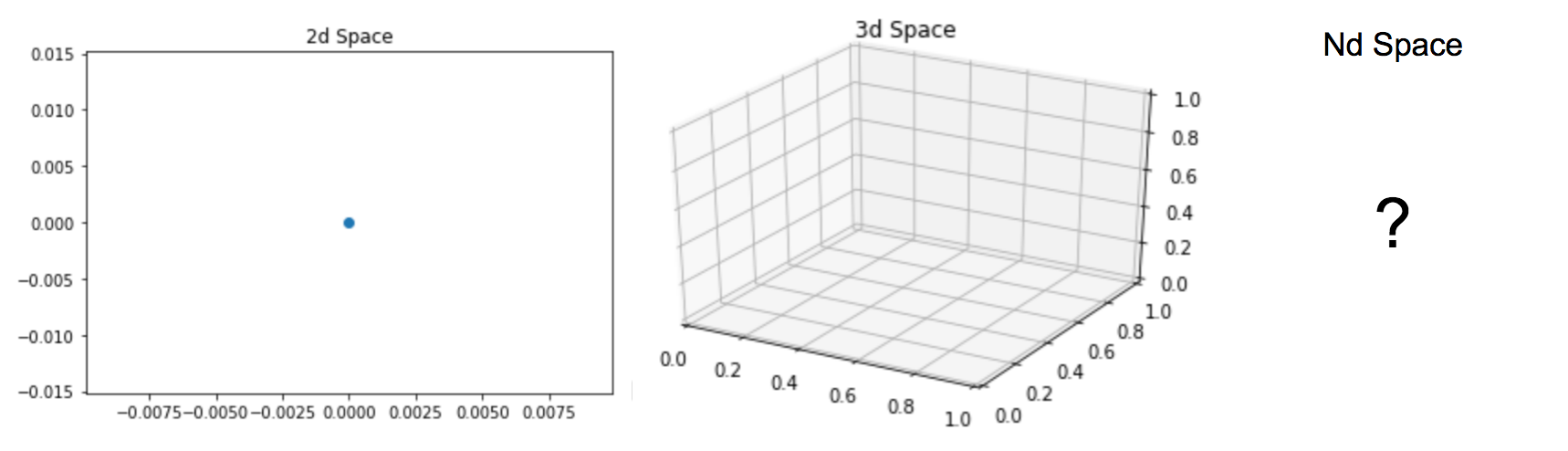


Figure 1.10: 2d plot, 3d plot, nd plot

Exercise 1: Calculating Euclidean Distance in Python

Slide 18: Exercise 1: Calculating Euclidean Distance in Python

**Scenario:** In this exercise we will create an example point along with 3 sample centroids to help illustrate how Euclidean distance works. Understanding this distance formula is foundational to the rest of our work in clustering.

**Outcome:** Be able to implement Euclidean distance from scratch and fully understand what it does to points in a feature space.

In this exercise we will be using the standard Python built-in math package. There are no prerequisites for using the math package and it is included in all standard installations of Python. As the name hints at, this package is very useful allowing to use a variety of basic math building blocks off the shelf such as exponentials, square roots, and more.

1. Open Jupyter notebook and create a naïve formula that captures the direct math of Euclidean distance as follows:

def dist(a, b):

return math.sqrt(math.pow(a[0]-b[0],2) + math.pow(a[1]-b[1],2))

This approach is considered naïve because it performs element-wise calculations on your data points (slow) compared to a more real world implementation using vectors and matrix math to achieve significant performance increases.

1. Create the data points in Python as follows:

centroids = [ (2, 5), (8, 3), (4,5) ]

x = (0, 8)

1. Use the formula you created to calculate the Euclidean distance between the example point and each of the three centroids you were provided:

centroid\_distances =[]

for centroid in centroids:

centroid\_distances.append(dist(x,centroid))

print(centroid\_distances)

print(np.argmin(centroid\_distances))

## [3.61, 9.43, 5.00]

## 0

Since Python is zero-indexed, a position of 0 as the minimum in our list of centroid distances signals to us that the example point x will be assigned to the #1 centroid of three.

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.

Now that you have found clusters in your data using Euclidean distance as the primary metric, think back to how you did this easily in Activity 1. It is very intuitive for our human mind to see groups of dots on a plot and determine which dots belong to discrete clusters. However, how do we ask a naïve computer to repeat this same task? By understanding this Exercise, you help teach a computer an approach to forming clusters of its own with a notion of distance. We will build upon how we use these distance metrics in the next exercise.

A quick example of this Exercise can be seen here:

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

Given the points that have been assigned to Cluster 1, find the mean point between all of the points to find the new centroid:

cluster\_1\_points =[ (0,8), (3,8), (3,4) ]

print([ (0+3+3)/3, (8+8+4)/3 ])

## [2.00,6.67]

Once you have moved the centroid location to the new mean point of (2, 6.67), you can compare it to the initial list of centroids you entered the problem with. If the new mean point is different than the centroid that is currently in your list, that means you have to go through another iteration of the above two exercises. Once the new mean point you calculate is the same as the centroid you started the problem, you have completed a run of K-Means and reached a point called convergence.

Activity 2: Implementing K-Means Clustering

Present 20: Activity 2: Implementing K-Means.Continued till slide 27

Scenario: You are asked in an interview to implement a K-Means clustering algorithm from scratch to prove you understand how it works. We will be using the Iris dataset provided by the UCI ML repository. The Iris data set is a classic in the data science world, and has features that are used to predict Iris species. See below for download location.

Aim: To truly understand how something works you need to build it from scratch. 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:

1. Using NumPy or the Math package and the above Euclidean distance formula, write a function that calculates the distance between two coordinates. NumPy is a scientific computing package for Python that pre-packages common mathematical functions in highly optimized formats. By using a package like NumPy or Math, we help cut down the time spent creating custom math functions from scratch and instead focus on developing our solutions.
2. Write a function that calculates the distance from centroids to each of the points in your dataset and returns the cluster membership.
3. 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!

Outcome: By completing this exercise you will gain hands-on experience tuning a K-Means clustering algorithm for a real-world data set. The Iris data set is seen as a classic “hello world” type problem in the data science space and is helpful to test foundational techniques on. Your final clustering algorithm should do a decent job at finding the 3 clusters of Iris species types that exist in the data.

Note:

The solution for this activity can be found on page XXX

Discuss 21: Why does the barebones Euclidean distance method perform poorly compared to pre-packaged ones?

Summary

Present 22: Summary

In this lesson we have explored what clustering is and why it could be important in a variety of data challenges. Building upon this foundation of clustering knowledge you implemented K-Means, which is one of the simplest yet most popular methods of unsupervised learning. If you have reached this summary and can repeat what K-Means does step-by-step to your fellow classmate, good job! If not please go back and review the material above – the content only grows in complexity from here. From here we will be moving on to hierarchical clustering, which in one configuration re-uses the centroid learning approach that we used in K-Means. We will build upon this approach by outlining additional clustering methodologies and approaches.

Practice Questions

1. How are supervised and unsupervised learning different?

Supervised has labels, Unsupervised does not

Supervised means you have to monitor the process the whole time, Unsupervised does not

Supervised only works on smaller datasets, and you need to use Unsupervised with larger datasets

1. What is a cluster?

A group of models used to train on data

A collection of features that are important to your model

A grouping of similar data

1. Finding clusters in your data is always valuable

True

False

1. How are dimensions expressed in a dataset?

As features, typically the number of columns in your data

As features, typically the number of rows in your data

As features, typically the number of tables in your database

1. How many dimensions can be calculated by computers?

Less than 3

3

Greater than 3

1. What does the “K” represent in K-Means clustering?

How many models will be fit to your data, with performance results averaged

How many clusters you expect to be in your data

How many dimensions your data set has

1. The starting points for K-Means are determined by taking the mean of all the points in the space.

True

False

1. What is the formula that underpins K-Means clustering?

Euclidean Distance

Manhattan Distance

Cosine Distance

1. When does a K-Means clustering algorithm finish running?

After specified number of iterations

After K/2 iterations

After convergence, when there is no longer a difference in calculated cluster centers

1. K-Means will still find clusters in a dataset even if all of the data is fairly similar.

True

False