Member-only story

How Does k-Means Clustering in Machine Learning Work?

Anas Al-Masri · Follow
Published in Towards Data Science
9 min read · May 14, 2019

One of the most famous topics under the realm of Unsupervised Learning in Machine Learning is k-Means Clustering. Even though this clustering algorithm is fairly simple, it can look challenging to newcomers into the field. In this post, I try to tackle the process of k-Means Clustering with two different examples. The first example will focus more on the big picture and visualizing the process, while the second example focuses on the underlying calculation involved.

Unsupervised Learning and Clustering

The main difference between Supervised and Unsupervised learning algorithms is the absence of data labels in the latter. With unsupervised learning, data features are fed into the learning algorithm, which determines how to label them (usually with numbers 0,1,2..) and based on what. This "based on what" part dictates which unsupervised learning algorithm to follow.

Most unsupervised learning-based applications utilize the sub-field called *Clustering*. **Clustering is the process of grouping data samples together into *clusters* based on a**

certain feature that they share — exactly the purpose of unsupervised learning in the first place.

So, what is k-Means, and why would we use it?

Being a clustering algorithm, k-Means takes data points as input and groups them into k clusters. This process of grouping is the training phase of the learning algorithm. **The result would be a model that takes a data sample as input and returns the cluster that the new data point belongs to, according the training that the model went through.** How can this be useful? Well, that's how content promotion and recommendation usually works — in a very simplistic manner. Websites may choose to put people in bubbles (i.e. clusters) with other people who share similar activities (i.e. features) on the website. This way, the recommended content will be somewhat on-point, as existing users with similar activities are likely to be interested in similar content. Moreover, as a new person goes into the ecosystem of the website, that person will be placed within a particular cluster, and the content recommendation system takes care of the rest.

Building on that idea, k-Means is just a clustering algorithm. It uses the distance between points as a measure of similarity, based on k averages (i.e. means). This is a very interesting algorithm, so let's get down to business.

Putting the k in k-Means

> The idea behind k-Means is that, we want to add k new points to the data we have. Each one of those points — called a Centroid — will be going around trying to center itself in the middle of one of the k clusters we have. Once those points stop moving, our clustering algorithm stops.

As you might've suspected, the value of k is of great importance. This k is called a hyper-parameter; a variable whose value we set before training. This k specifies the number of clusters we want the algorithm to yield. This number of clusters is actually the number of centroids going around in the data.

Before we go further, let's take a look at how everything so far fits within the big picture:

- We know that the core of Machine Learning lies on the idea of generalization — making a reliable prediction of outputs for inputs that the model has never seen before.
- Unsupervised Learning is all about grouping data samples together, regardless of their labels (if they have any).
- Clustering is an Unsupervised Learning algorithm that groups data samples into k clusters.
- The algorithm yields the k clusters based on k averages of points (i.e. centroids) that roam around the data set trying to center themselves — one in the middle of each cluster.

k-Means: in brief

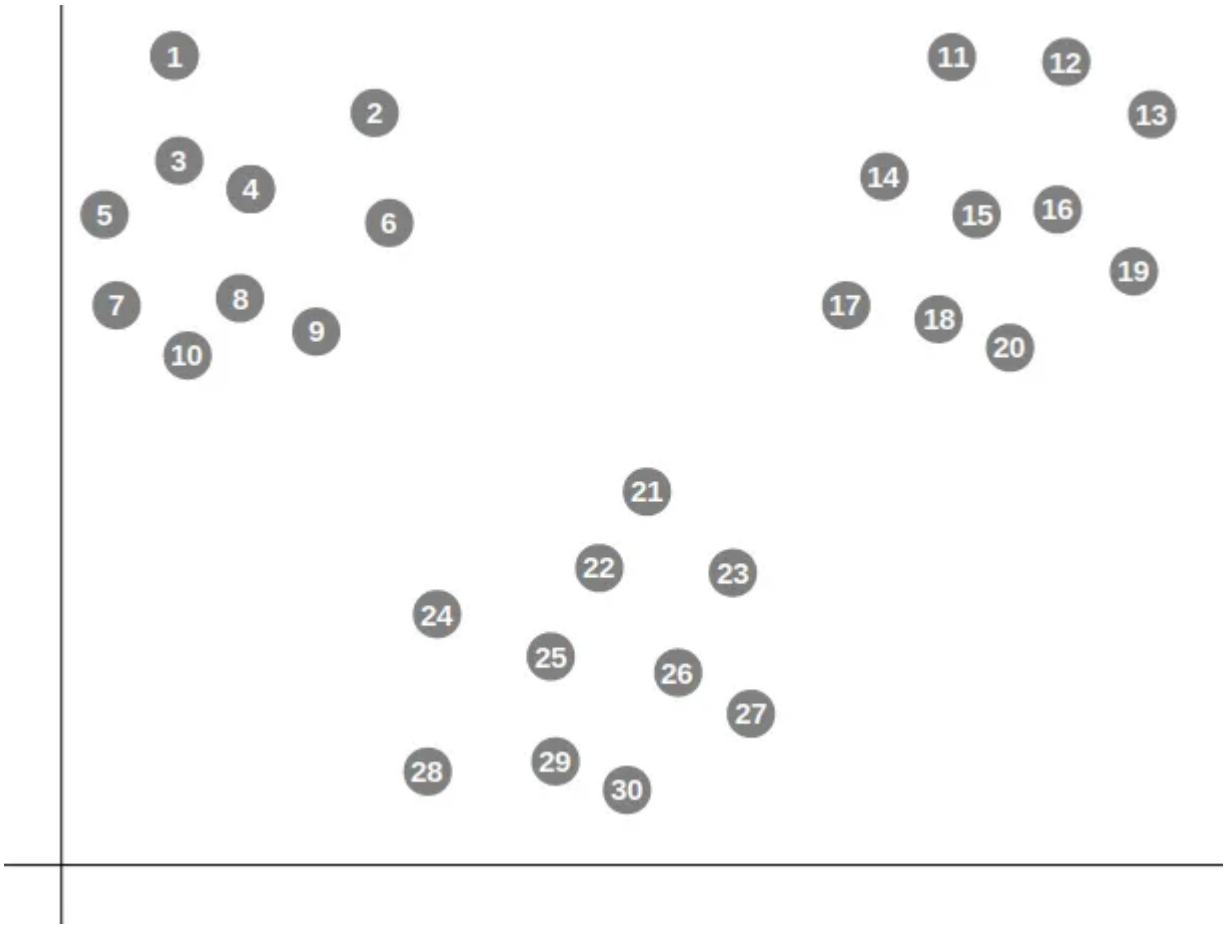
What is a better summary of an algorithm than pseudo-code?

```
Assign initial values for each u (from u=1 till u=k);

Repeat {
        Assign each point in the input data to the u that is closest
        to it in value;

Calculate the new mean for each u;

if all u values are unchanged { break out of loop; }
}
```

k-Means: in detail

If you've read any of my posts before, you probably know that I like to explain with an example first then talk about the technical aspect of our topic. Moreover, I don't like to walk the reader through the overwhelming mathematics behind topics, as I believe those are more important to researchers than to people who have a self-interest in the matter.

Back to k-Means and our first example. Let's say we have a data set that looks like the following figure when plotted out:

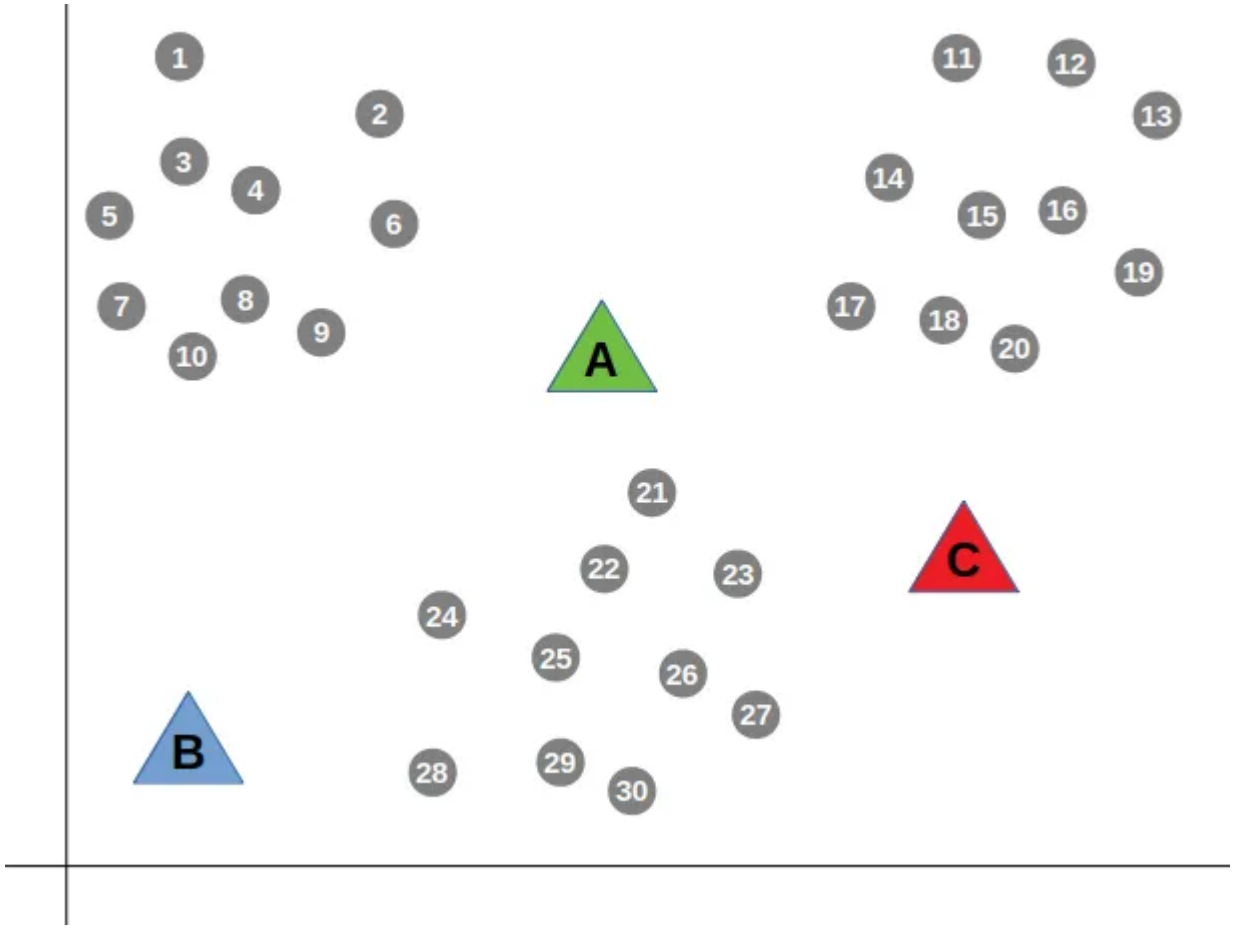


To us humans, this data looks like it perfectly fits within three groups (i.e. clusters). However, machines can't see that, as those points are actual data "points" whose values are just numbers that cannot be sensible to the machine.

Referring to the objective of clustering, we have a set of unlabeled data points that we want to put in groups. Those groups are usually labeled with numbers (0,1,2..) by the algorithm itself. Implicitly, what we really need is actually a decision boundary that separates the groups. Why? In practice, inference works by associating a data point with a respective cluster. That's where the decision boundary appears to be important.

> *k-Means clustering is all about putting the training points we have into clusters. But the purpose of it follows the same idea. We want to know which data points belong together without having any labels for any of them.*

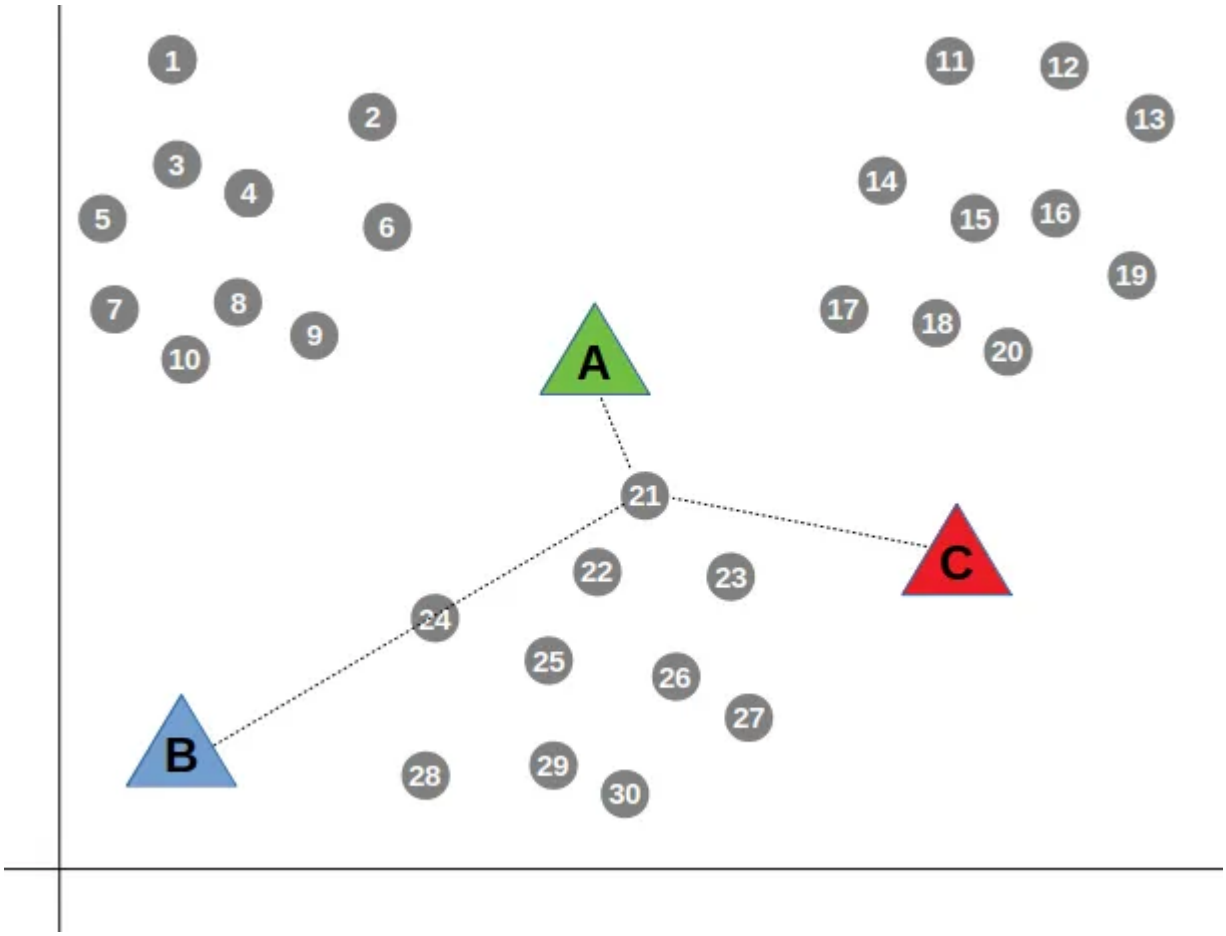
We start the algorithm by **placing k different averages** (i.e. means) whose values are either initialized randomly or set to real data points on the plane. Let's start with **k=3**, as the data "seems" to be falling into three groups (we will get back to this "seems" word later in the post). For explanation-related purposes, lets randomly initialize the values (i.e. positions) of the averages:



Now, the algorithm goes through the data points one-by-one, measuring the distance between each point and the three centroids (A, B and C). The algorithm then groups

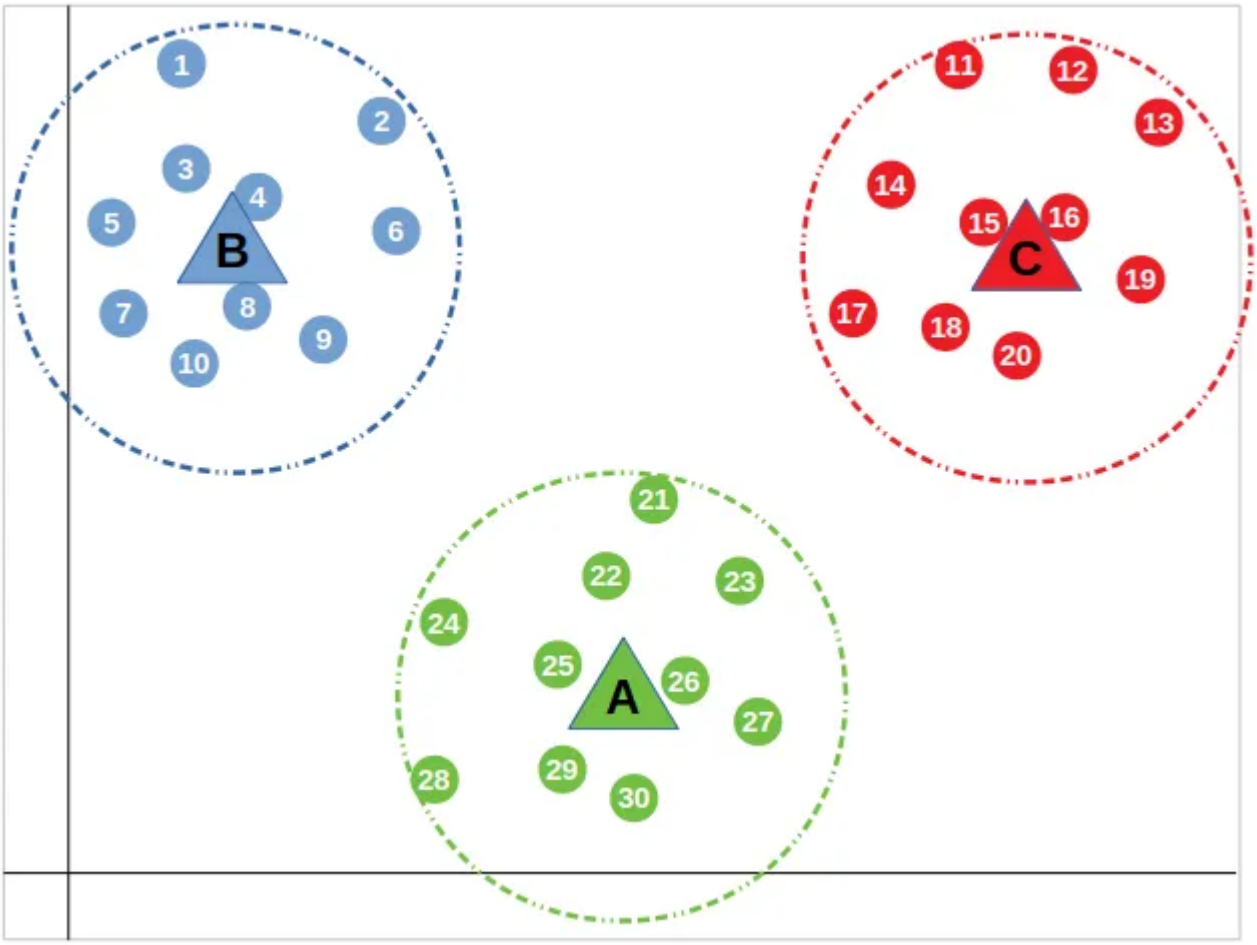
the the data point with the closest centroid (i.e. closest in distance).

For instance, data point number 21 will belong to group A in the green color, merely because it's closer in distance to centroid A:



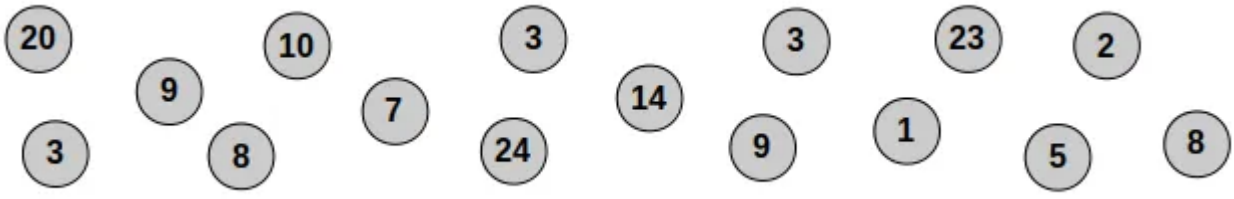
As soon as we're done associating each data point with its closest centroid, we re-calculate the means — the values of the centroids; the new value of a centroid is the sum of all the points belonging to that centroid divided by the number of points in the group.

We keep doing the above until no centroid changes its value in re-calculation. This means that each centroid has centered itself in the middle of its cluster, which is surrounded by its own circular decision boundary:



Another Example

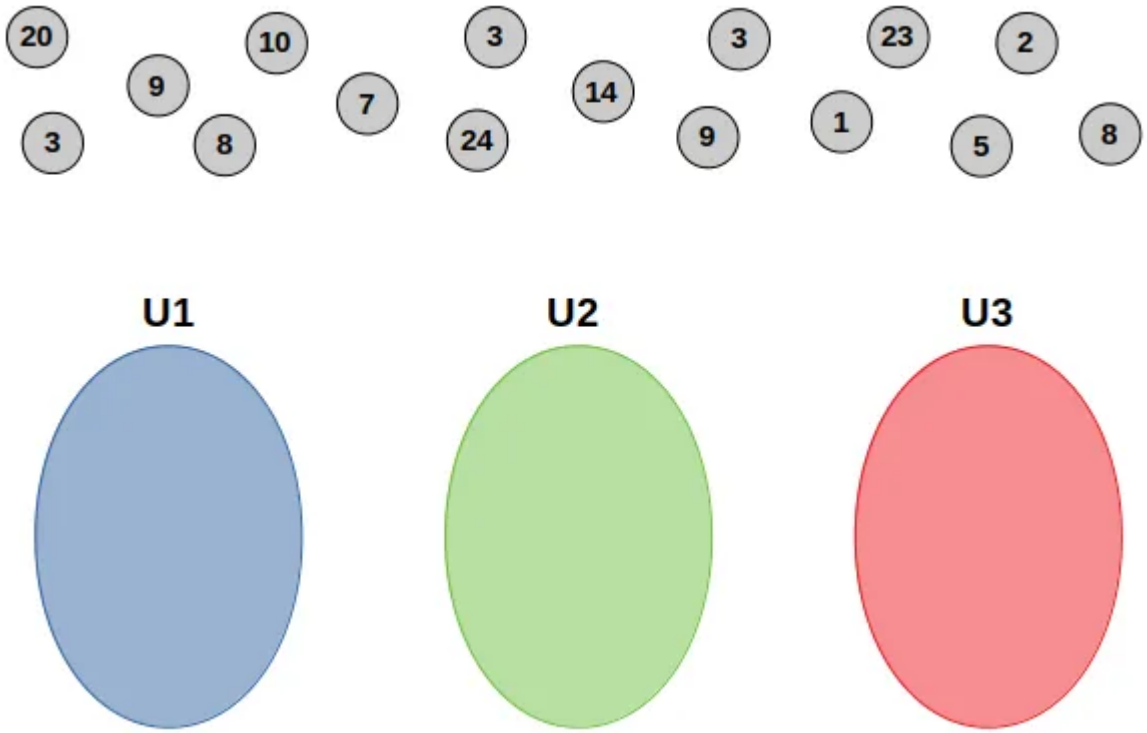
Let's take another example, but from a different perspective this time. Let's say we have the following set of points that we want to cluster into 3 groups:



This data is not put in a visually-appealing manner. We just have a set of points that we want to cluster. Another important note is that the values in the circles symbolizing those data points are actual values of the points. They're not put a way of showing the sequence of data like our previous example. Rather, these values are a quantification of

some feature value F. I put them this way to make it easier on you to understand how the calculation of means works.

Let's prepare our empty clusters:



Think of these clusters as just bags that will contain points from our data set.

Let's initialize the U values (i.e. means/centroids) to:

```
U1 = 6
U2 = 12
U3 = 18
```

These values can be random, but for simplicity reasons, they were chosen to be uniformly scattered along the data space (1 through 24).

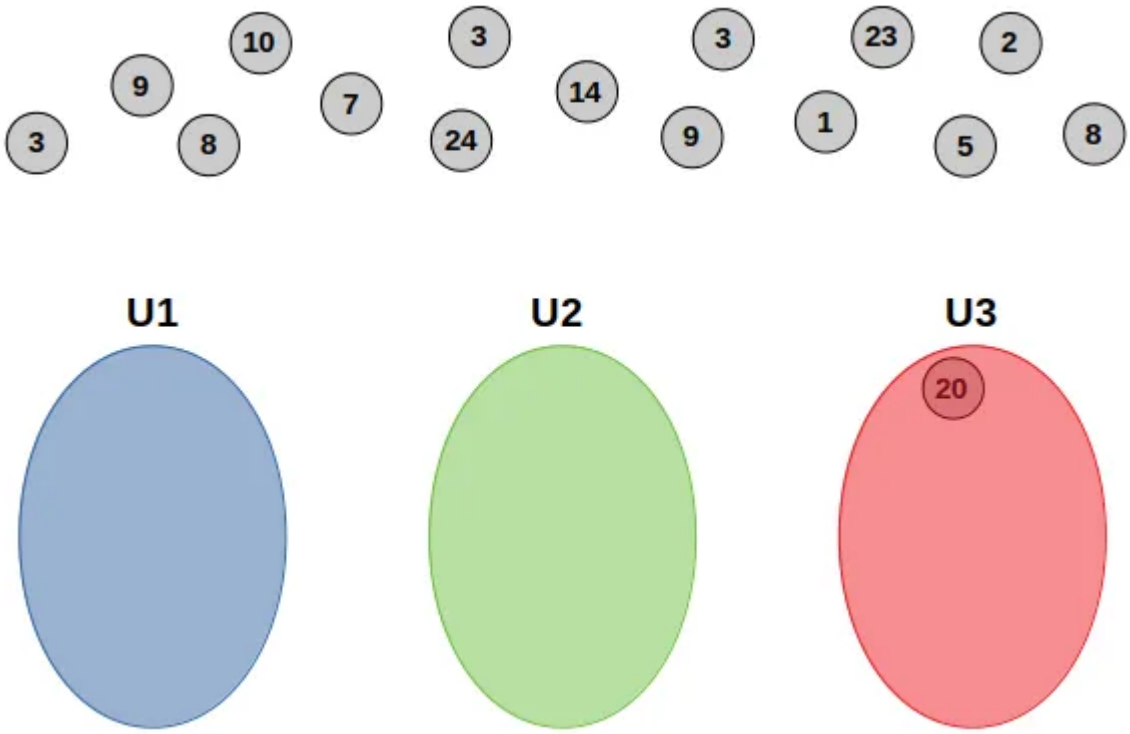
Since we already have our means, we can start calculating distances between any point whose feature F value is F and the three means (U1, U2 & U3), where the absolute distance follows:

```
distance = | F - U |
```

For starters, let's take the data point whose feature F value is 20:

```
|20 - U1| = |20 - 6| = 14
|20 - U2| = |20 - 12| = 8
|20 - U3| = |20 - 18| = 2
```

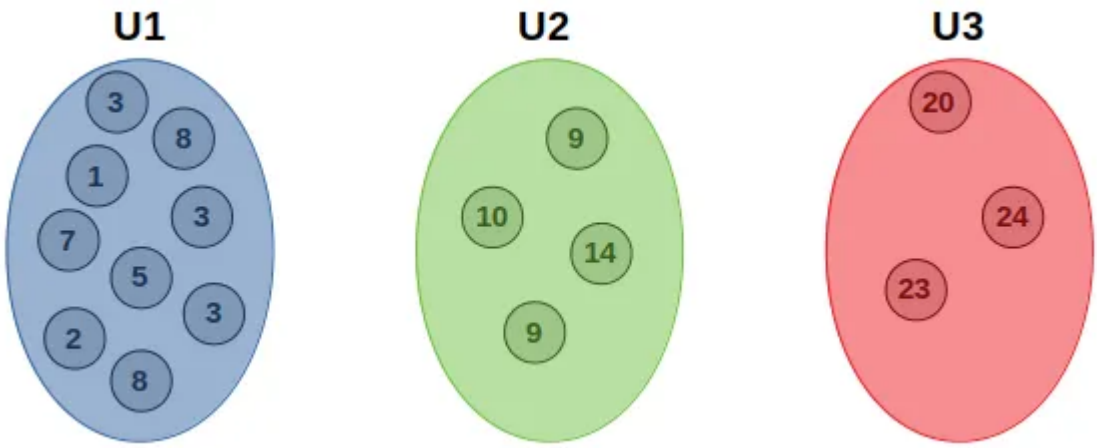
According to the above calculations, the **data point whose value is 20 is closer in distance to mean U3.** Therefore, we "label" the point as U3 by putting it in its respective cluster/bag:



And the same goes for all the other points:

```
|3 - U1| = |20 - 6| = 3
|3 - U2| = |20 - 12| = 9
|3 - U3| = |20 - 18| = 15
```

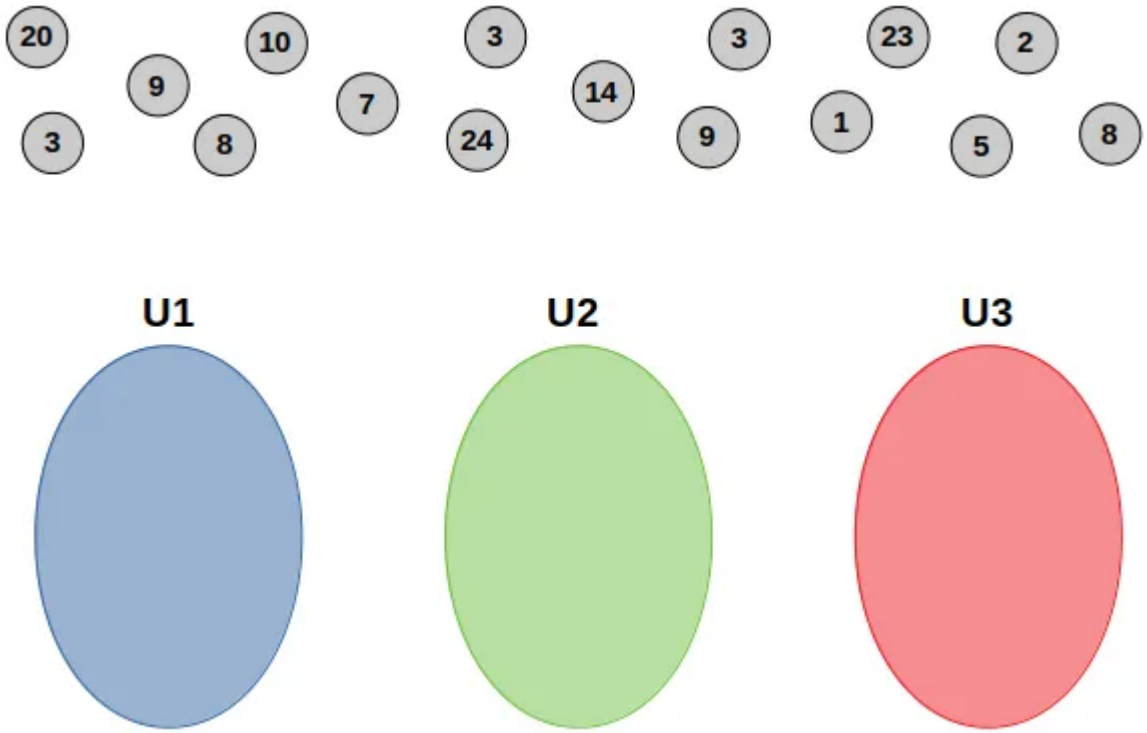
And so on, until we have all our data points in their appropriate clusters:



Following the algorithm, we now need to re-calculate the means (U1, U2 as well as U3):

```
U1 = (3+8+1+3+7+5+2+3+8) / 9 = 4.44
U2 = (9+10+14+9) / 4 = 10.5
U3 = (20+24+23) / 3 = 22.33
```

Upon the beginning of our execution, our U values were 6, 12, and 18, respectively. Now, after the first iteration, the values became 4.44, 10.5 and 22.33, respectively. We now have to go through the distance calculation step again, but with the new mean values. We empty out our bags/clusters, and start again:

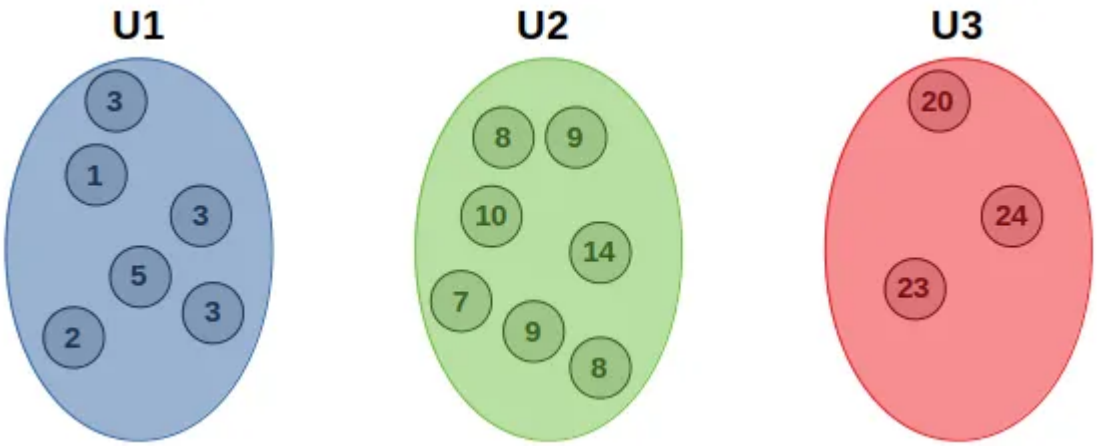


Let's start with a random point this time, say, the point whose feature F value is 8. The distance calculations would follow:

```
|8 - U1| = |8 - 4.44| = 3.56
|8 - U2| = |8 - 10.5| = 2.5
|8 - U3| = |8 - 22.33| = 14.33
```

So, the data point with the value 8 belongs to the cluster of mean value U2, as it's is closest to it in distance. If you still recall, the same data point (8) belonged to U1 in the first run. This makes it evident how important the mean values are.

Running the algorithm over and over again until the mean values don't change upon calculation will reach the following formation, with mean values 2.83, 9.29 and 22.33, respectively:



Final Notes

As you've seen in our second example, the mean values we operate with at any given point are of great importance to the reliability of the model. This does not exclude the initial values. In fact, with really bad starting positions, the algorithm could get the clustering completely wrong! Therefore, another initialization approach would be to choose our initial k positions from the data set itself, setting one of the points as a mean value in its own surroundings. Either way, it is common practice to run the algorithm on the same data set repeatedly until we find a most common solution and clustering formation — and we stick to it.

Remember the stage at which we said that data "seems" to have been partitioned into three? Well, that was only to show the importance of the human eye in such applications. Since we don't have a ground truth error estimation (as it usually requires labels), we need another measure for the hyper-parameters. For a lot of applications, a person looking at the plot and determining the k value is sufficient. However, this does not overestimate the importance of the k value in any way.

Finally, one of the apparent strengths of k-Means is the fact that it gives an average value of the feature over the cluster, not just the cluster itself. This could be very useful in segmentation-related image processing tasks — when dealing with segmentation as a clustering problem.

Machine Learning Clustering K Means Segmentation

Follow

Written by Anas Al-Masri

433 Followers · Writer for Towards Data Science

Tech Lead. Portfolio: https://www.anasalmasri.com/

More from Anas Al-Masri and Towards Data Science

Anas Al-Masri in Towards Data Science

How Does Linear Regression Actually Work?

Linear Regression is inarguably one of the most famous topics in both statistics and Machine Learning. It is so essential to the point...

· 8 min read · Mar 18, 2019

281

Jacob Marks, Ph.D. in Towards Data Science

How I Turned My Company's Docs into a Searchable Database with OpenAI

And how you can do the same with your docs

15 min read · Apr 25

2.3K 30

Leonie Monigatti in Towards Data Science

Getting Started with LangChain: A Beginner's Guide to Building LLM-Powered Applications

A LangChain tutorial to build anything with large language models in Python

· 12 min read · Apr 25

Anas Al-Masri in Towards Data Science

Creating The Twitter Sentiment Analysis Program in Python with Naive Bayes Classification

Sentiment Analysis is a term that you must have heard if you have been in the Tech field long enough. It is the process of predicting...

· 16 min read · Feb 13, 2019

Abhinaba Banerjee in DataDrivenInvestor

Creating Features and K-Means Clustering in Feature Engineering

In the previous part, the basics of Feature Engineering were discussed along with identifying the most important features from a lot of...

8 min read · Nov 29, 2022

86

0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

Patrizia Castagno

k-Means Clustering (Python)

This section is a simple example of the section: Unsupervised Learning, I recommend reading the theory first before moving on to this...

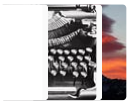
✦ · 4 min read · Dec 27, 2022

 471  1 

Lists

What is ChatGPT?

9 stories · 28 saves



Staff Picks

303 stories · 65 saves

Alexander Nguyen in Level Up Coding

Why I Keep Failing Candidates During Google Interviews...

They don't meet the bar.

· 4 min read · Apr 13

3.7K 113

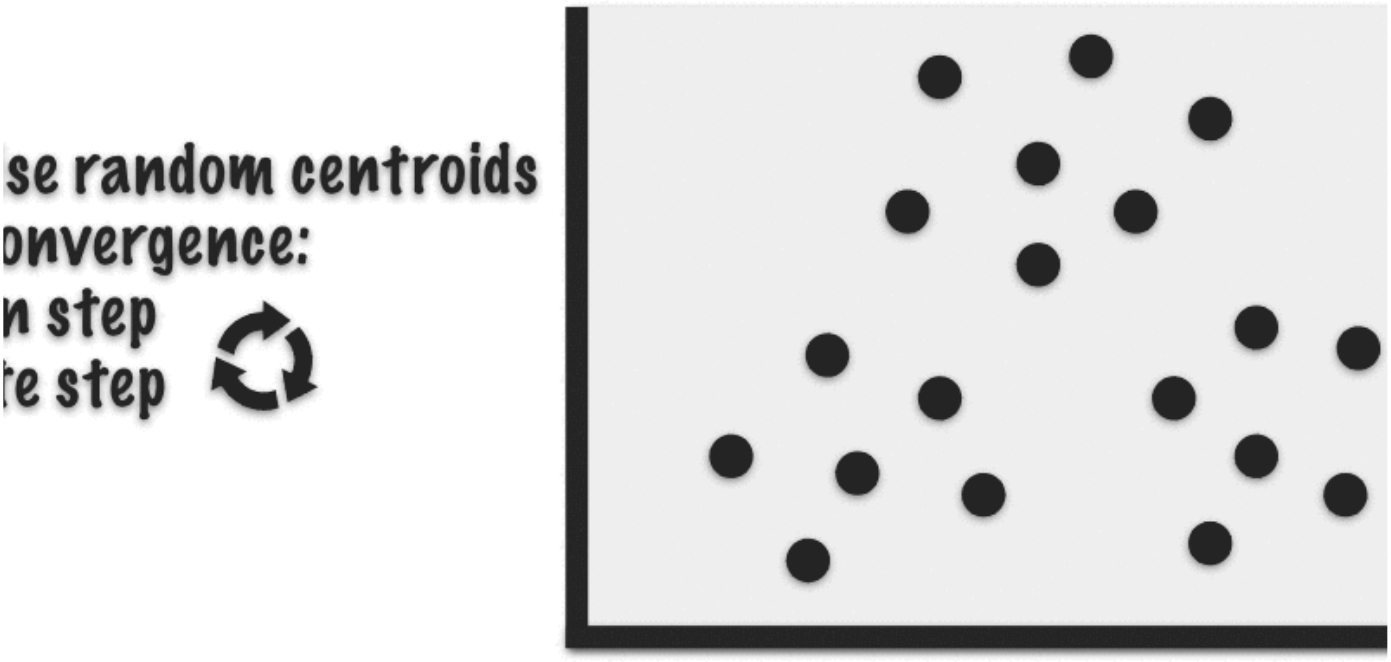
Carla Martins

How to Compare and Evaluate Unsupervised Clustering Methods?

Using Python, Scikit-Learn, and Google Colab

✨ · 20 min read · Feb 23

104



Zaheer

K-Means Clustering

Unsupervised algorithm

5 min read · Mar 23

51

Christos Panourgias

Clustering with DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups together points that are close...

4 min read · Mar 27



See more recommendations