Clustering and PCA Assignment Part – 2

By

Rachit Dev

Question 1: Assignment Summary

Briefly describe the "Clustering of Countries" assignment that you just completed within 200-300 words. Mention the problem statement and the solution methodology that you followed to arrive at the final list of countries. Explain your main choices briefly (why you took that many numbers of principal components, which type of Clustering produced a better result and so on)

Note: You don't have to include any images, equations or graphs for this question. Just text should be enough.

Answer 1: We performed the reading and understanding the data. Then we read the data dictionary to understand the data. We checked on the missing values there were none. Then we prepared the data for modelling by removing the column ‘country’ as it is a string value and others were numeric. Then we checked for the outliers and found that there is no reason to remove any outlier, outliers can be handled through scaling and I used PowerTranformer scaler which reduced the outliers. We used the outlier treatment just to show what was asked in the assignment. Then we used PowerTransformer to perform a Yeo-Johnson power transformation on our data for modelling in order to ensure that our features are normally distributed. Then we applied PCA on the dataset and the value of k was found to be 5 where over 95% information of all the columns was contained. Then we created the dataframe of the components. Then we performed the silhoutte analysis and choose clusters = 3. Assigned the cluster values in the dataframe as cluster_id. Then we performed hierarchical clustering and choose clusters = 3. Associated the value of cluster in the dataframe as cluster_labels. Then we created a full dataset of components, cluster_labels, cluster_ids and the original columns. We did some exploratory data analysis which can be found in the jupyter notebook. Then we created a dataframe with the suggested columns as mentioned in the assignment. We did data analysis in the last dataframe to get the list of coutries which are really in the need of aid.

Question 2: Clustering

Answer 2:

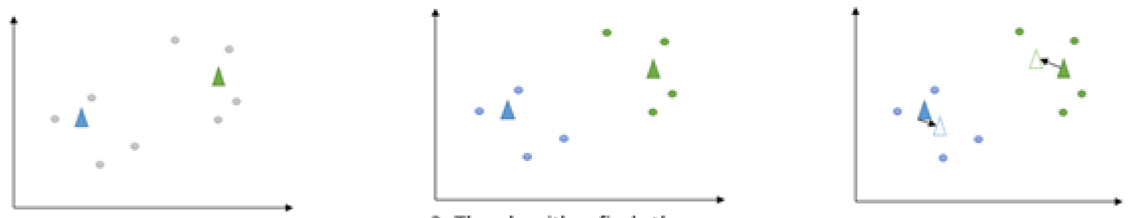
a) Compare and contrast K-means Clustering and Hierarchical Clustering.

K-means

The first step of this algorithm is creating, among our unlabeled observations, *c* new observations, randomly located, called ‘centroids’. The number of centroids will be representative of the number of output classes (which, remember, we do not know). Now, an iterative process will start, made of two steps:

- First, for each centroid, the algorithm finds the nearest points (in terms of distance that is usually computed as Euclidean distance) to that centroid, and assigns them to its category;
- Second, for each category (represented by one centroid), the algorithm computes the average of all the points which has been attributed to that class. The output of this computation will be the new centroid for that class.

Every time the process is reiterated, some observations, initially classified together with one centroid, might be redirected to another one. Furthermore, after several reiterations, the change in centroids’ location should be less and less important since the initial random centroids are converging to the real ones. This process ends when there is no more change in centroids’ position.



1. Two centroids are randomly located among data

2. The algorithm finds the nearest observations for each centroid, and then attributes them to each centroid’s class

3. New centroids, resulting from the average of each class observations, are computed, and then the process is repeated

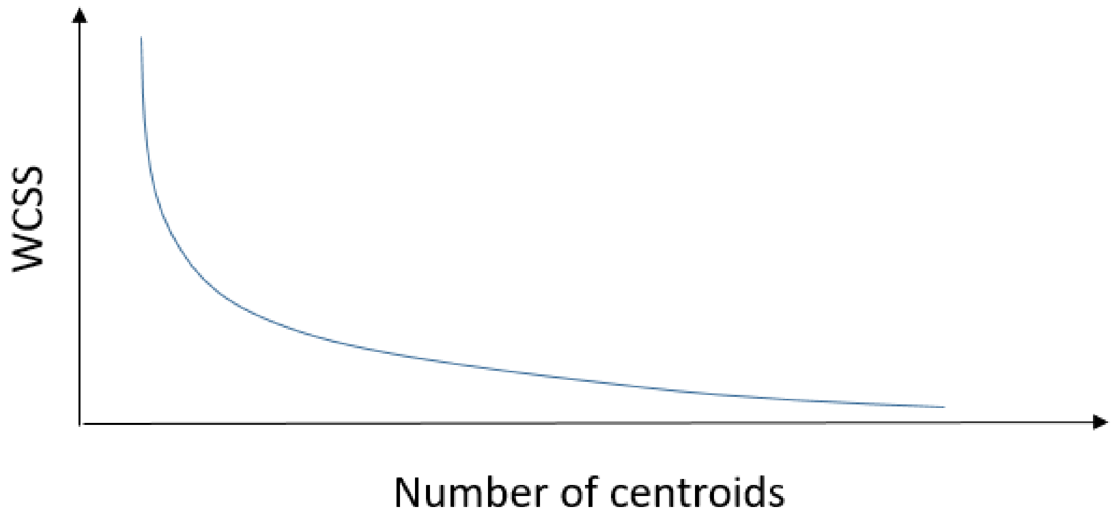
Now, how can we decide the number of centroids?

There are many methods that could be employed for this task. However, in this article, I’m going to explain and use the so-called ‘Elbow method’. The idea is that what we would like to observe within our clusters is a low level of variation, which is measured with the within-cluster sum of squares (WCSS):

$$WCSS = \sum_{xi \in c} (x_i - \bar{x})^2$$

And it is intuitive to understand that, the higher the number of centroids, the lower the WCSS. In particular, if we have as many centroids as the number of our observations, each WCSS will be equal to zero. However, if we remember the law of parsimony, we know that setting the highest number possible of centroids would be inconsistent.

The idea is picking that number of centroids after which the reduction in WCSS is irrelevant. The relation I've just described can be represented with the following graph:



The idea is that, if the plot is an arm, the elbow of the arm is the optimal number of centroids.

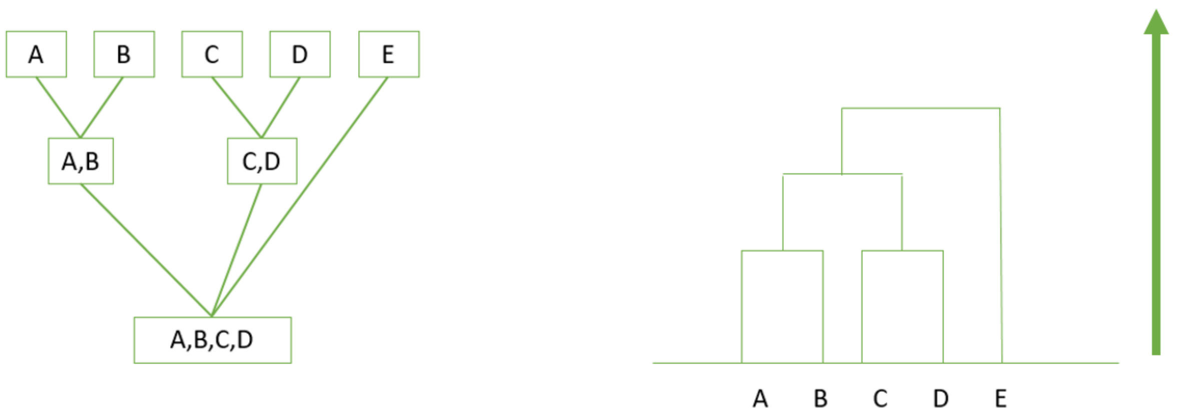
Hierarchical Clustering

This algorithm can use two different techniques:

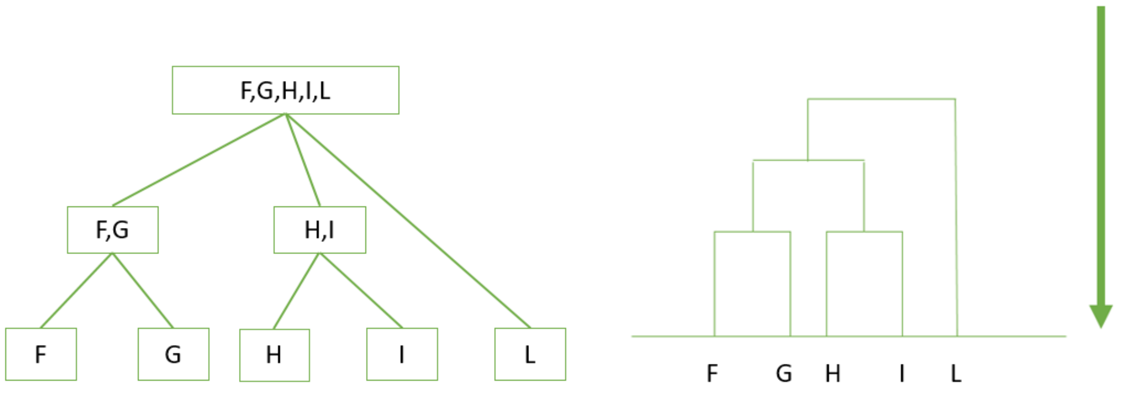
- Agglomerative
- Divisive

Those latter are based on the same ground idea, yet work in the opposite way: being *K* the number of clusters (which can be set exactly like in K-means) and *n* the number of data points, with *n*>*K*, agglomerative HC starts from *n* clusters, then aggregates data until it obtains K clusters; divisive HC, on the other hand, starts from just one cluster and then splits it depending, again, on similarities, until it obtains *K* clusters. Note that, when I say similarities, I'm referring to the distance among data points, which can be computed in different ways (I will dwell on this later on).

Let's visualize both the agglomerative and divisive techniques:



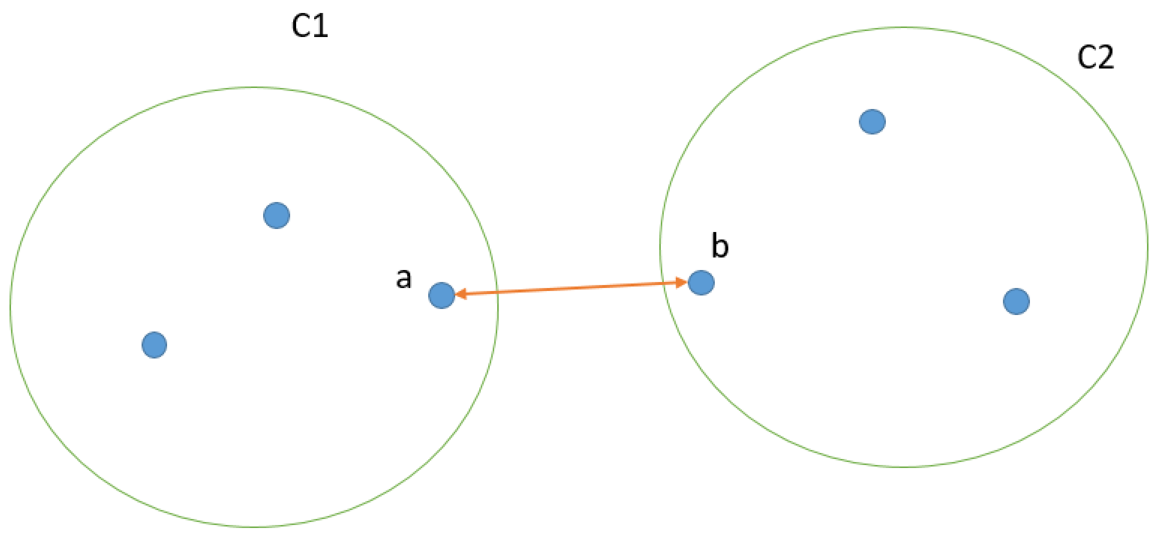
Agglomerative HC



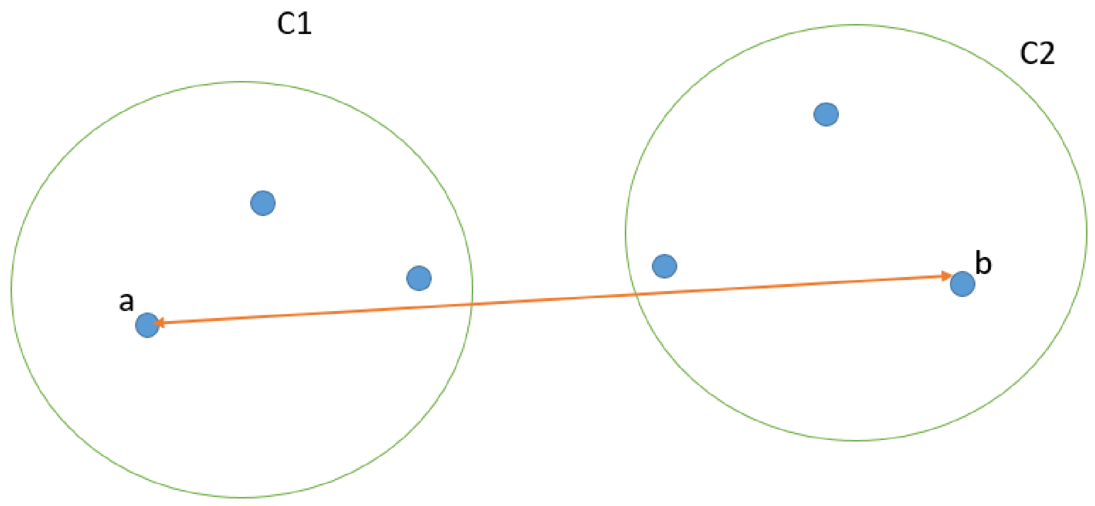
Divisive HC

As anticipated, the key element of discrimination here is similarity among data points. In mathematical terms, similarity mainly refers to distance, and it can be computed with different approaches. Here, I will propose three of them:

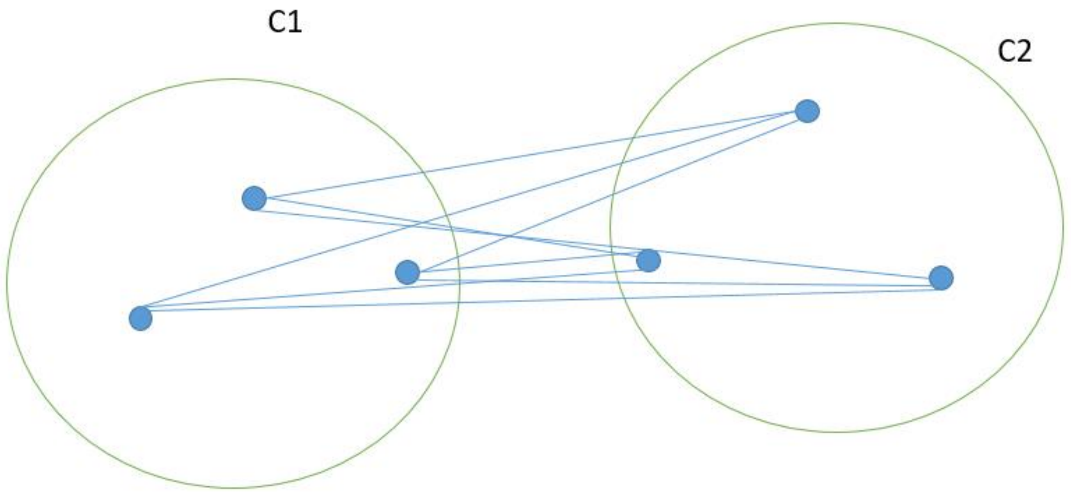
- Min: it states that, given two clusters C1 and C2, the similarity between them is equal to the minimum of similarity (translated: distance) between point a and b, such that a belongs to C1 and b belongs to C2.



- Max: it states that, given two clusters C1 and C2, the similarity between them is equal to the maximum of similarity between point a and b, such that a belongs to C1 and b belongs to C2.



- Average: it takes all the pairs of points, compute their similarities and then calculate the average of the similarities. That latter is the similarity between the clusters C1 and C2.



So, to recap, both the algorithms look for similarities among data and both use the same approaches to decide the number of clusters. Choosing the one rather than the other really depends on the kind of task you're facing.

b) Briefly explain the steps of the K-means clustering algorithm.

Clustering is an *unsupervised* machine learning technique, with several valuable applications in healthcare.

Consider the situation where a model is trying to predict a patient's likelihood of having diabetes. In most cases, a set of outputs (label of whether a person has diabetes or not) is associated with inputs (attributes like age, weight, blood pressure and so on). One can train a supervised classification model, like random forest to learn the patterns associated with the output. Then, if a new patient comes in, the trained classifier can predict if the patient has diabetes or not. This is an example of a *supervised* learning problem because outputs, or labels, are provided with the training examples.

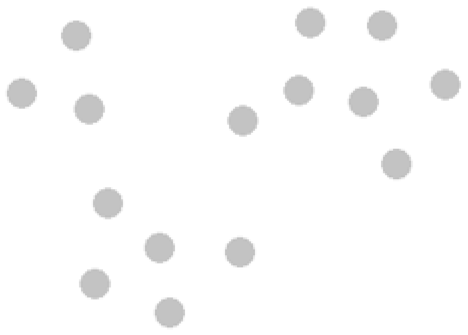
But what if there are no outputs provided, or they are unreliable? Data is often gathered and divided for model training based on ICD-10 codes, the "ground truth" for things like who has diabetes. However, it's possible that there are many missing ICD-10 codes for diabetes. If the code can't be trusted for model training, the diabetic/non-diabetic group structure must be discovered from the input values alone. When the label is unreliable or absent, we must turn to unsupervised learning.

Clustering is exactly that. We are trying to find a structure in a collection of data assuming that a reliable label output is not provided.

Clustering via K-means

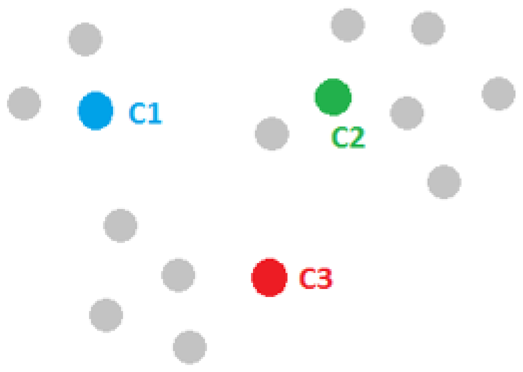
Among all the unsupervised learning algorithms, clustering via k-means might be one of the simplest and most widely used algorithms. Briefly speaking, k-means clustering aims to find the set of k clusters such that every data point is assigned to the closest center, and the sum of the distances of all such assignments is minimized.

Let's walk through a simple 2D example to better understand the idea. Imaging we have these gray points in the following figure and want to assign them into three clusters. K-means follows the four steps listed below.

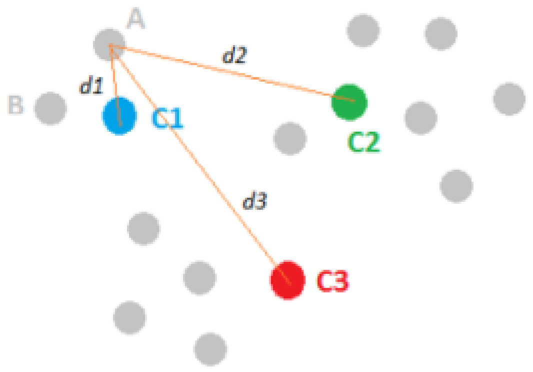


Step one: Initialize cluster centers

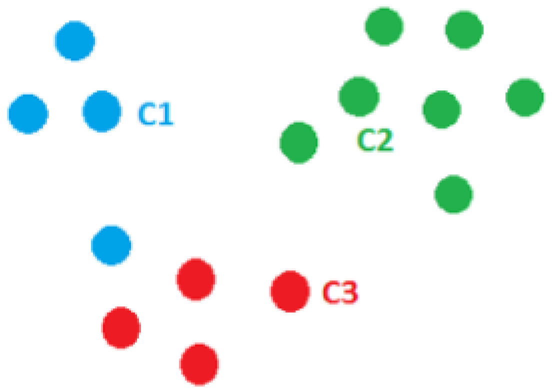
We randomly pick three points C1, C2 and C3, and label them with blue, green and red color separately to represent the cluster centers.



Step two: Assign observations to the closest cluster center

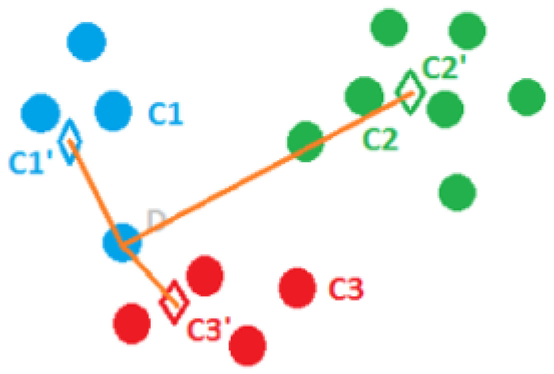


Once we have these cluster centers, we can assign each point to the clusters based on the minimum distance to the cluster center. For the gray point A, compute its distance to C1, C2 and C3, respectively. And after comparing the lengths of *d1*, *d2* and *d3*, we figure out that *d1* is the smallest, therefore, we assign point A to the blue cluster and label it with blue. We then move to point B and follow the same procedure. This process can assign all the points and leads to the following figure.



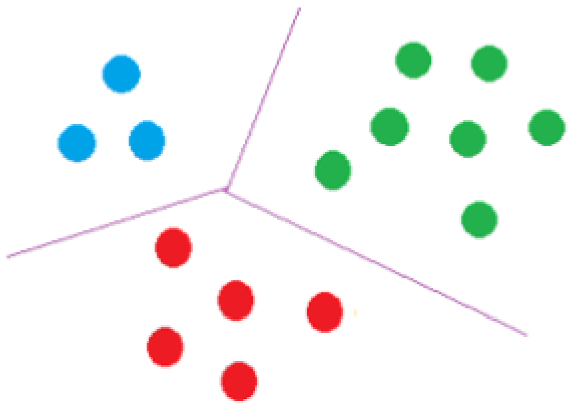
Step three: Revise cluster centers as mean of assigned observations

Now we've assigned all the points based on which cluster center they were closest to. Next, we need to update the cluster centers based on the points assigned to them. For instance, we can find the center mass of the blue cluster by summing over all the blue points and dividing by the total number of points, which is four here. And the resulted center mass C1', represented by a blue diamond, is our new center for the blue cluster. Similarly, we can find the new centers C2' and C3' for the green and red clusters.



Step four: Repeat step 2 and step 3 until convergence

The last step of k-means is just to repeat the above two steps. For example, in this case, once C1', C2' and C3' are assigned as the new cluster centers, point D becomes closer to C3' and thus can be assigned to the red cluster. We keep on iterating between assigning points to cluster centers, and updating the cluster centers until convergence. Finally, we may get a solution like the following figure. Well done!

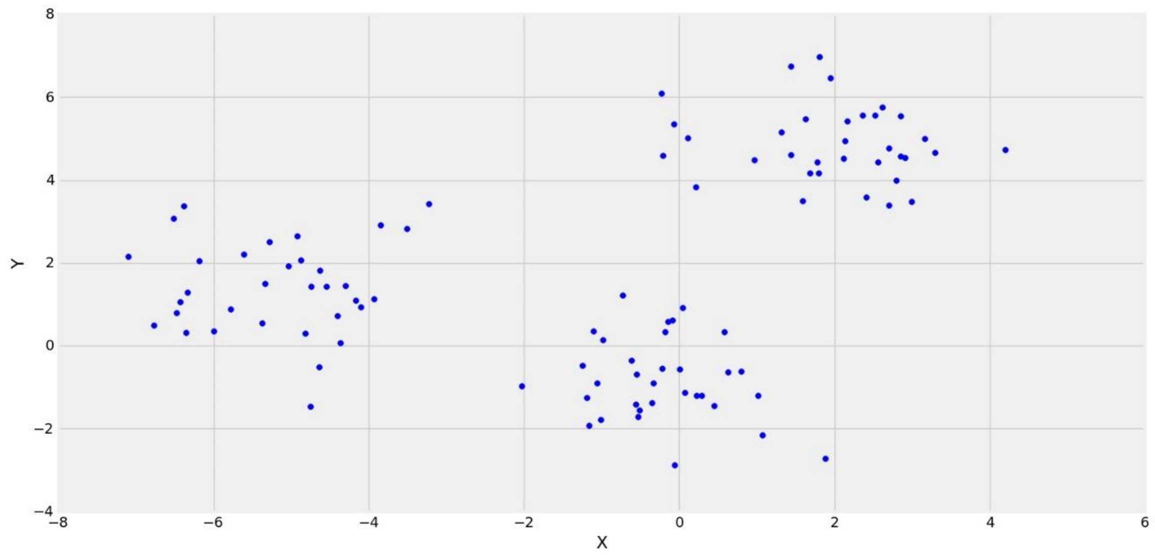


Some Additional Remarks about K-means

- The k-means algorithm converges to local optimum. Therefore, the result found by K-means is not necessarily the most optimal one.
- The initialization of the centers is critical to the quality of the solution found. There is a smarter initialization method called K-means++ that provides a more reliable solution for clustering.
- The user has to select the number of clusters ahead of time.

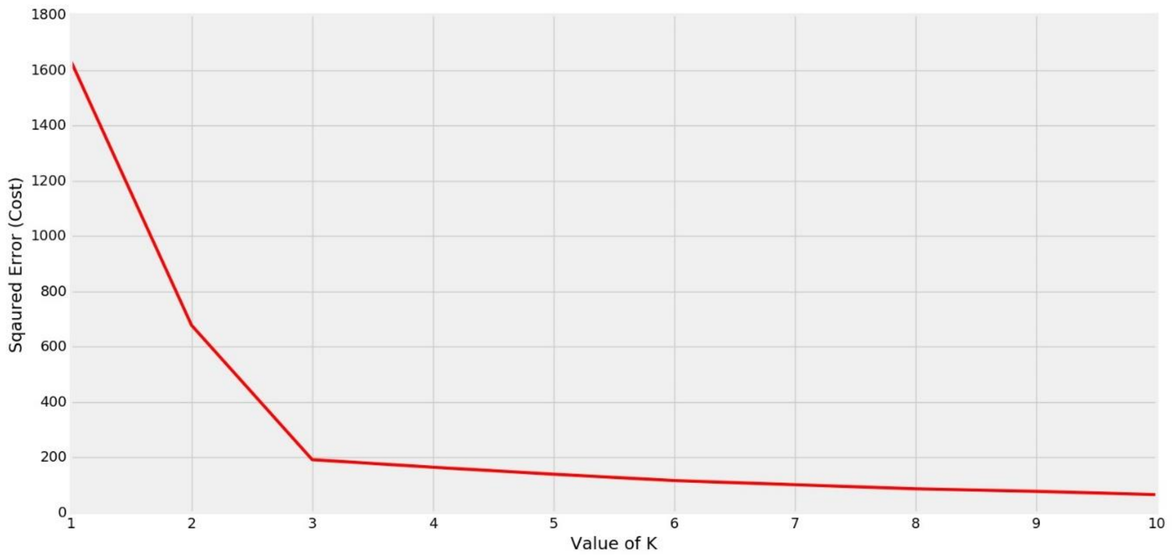
c) How is the value of 'k' chosen in K-means clustering? Explain both the statistical as well as the business aspect of it.

There is a popular method known as **elbow method** which is used to determine the optimal value of K to perform the K-Means Clustering Algorithm. The basic idea behind this method is that it plots the various values of cost with changing *k*. As the value of *K* increases, there will be fewer elements in the cluster. So average distortion will decrease. The lesser number of elements means closer to the centroid. So, the point where this distortion declines the most is the **elbow point**.



3 clusters are forming

In the above figure, its clearly observed that the distribution of points are forming 3 clusters. Now, let’s see the plot for the squared error(Cost) for different values of K.



Elbow is forming at K=3

Clearly the elbow is forming at K=3. So the optimal value will be 3 for performing K-Means.

d) Explain the necessity for scaling/standardization before performing Clustering.

Standardizing multivariate data is important, particularly when variable scales differ significantly.

Model stability and parameter estimate precision are influenced during multivariate analysis when multi-scaled variables are used. For example, in boundary detection, a variable that ranges between 0 and 100 will outweigh a variable that ranges between 0 and 1. Using variables without standardization can give variables with larger ranges greater importance in the analysis. Transforming the data to comparable scales can prevent this problem.

Standardizing continuous predictor variables in neural network is extremely important.

When performing regression analysis, standardizing multi-scale variables can help reduce multicollinearity issues for models containing interaction terms.

Standardizing your data prior to cluster analysis is also extremely critical. Clustering is an unsupervised learning technique that classifies observations into similar groups or clusters. A commonly used measure of similarity is Euclidean distance. The Euclidean distance is calculated by taking the square root of the sum of the squared differences between observations. This distance can be greatly affected by differences in scale among the variables. Generally variables with large variances have a larger effect on this measure than variables with small variances. For this reason, standardizing multi-scaled variables is advised prior to performing clustering.

Standardizing or normalizing data is also important in principal component analysis (PCA) since it projects your original data onto orthogonal directions which maximize the variance.

However, tree-based analyses are not sensitive to outliers and do not require variable transformations. As a result, standardization of multi-scaled data is not necessary for Decision Trees, Random Forest and/or Gradient Boosting algorithms.

e) Explain the different linkages used in Hierarchical Clustering.

Hierarchical clustering Technique:

Hierarchical clustering is one of the popular and easy to understand clustering technique. This clustering technique is divided into two types:

1. Agglomerative
2. Divisive

Agglomerative Hierarchical clustering Technique: In this technique, initially each data point is considered as an individual cluster. At each iteration, the similar clusters merge with other clusters until one cluster or K clusters are formed.

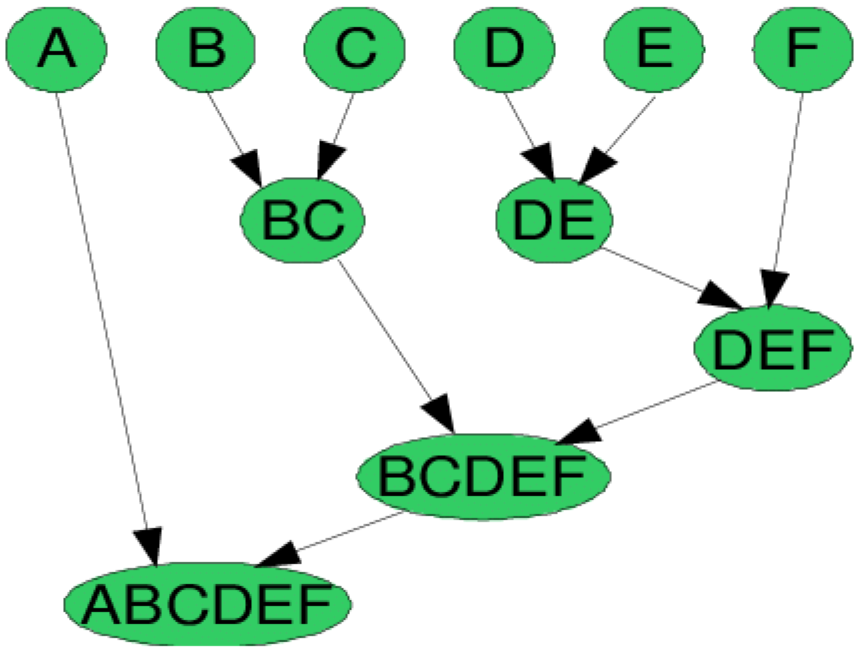
The basic algorithm of Agglomerative is straight forward.

- Compute the proximity matrix
- Let each data point be a cluster
- Repeat: Merge the two closest clusters and update the proximity matrix
- Until only a single cluster remains

Key operation is the computation of the proximity of two clusters

To understand better let's see a pictorial representation of the Agglomerative Hierarchical clustering Technique. Lets say we have six data points {A,B,C,D,E,F}.

- Step- 1: In the initial step, we calculate the proximity of individual points and consider all the six data points as individual clusters as shown in the image below.

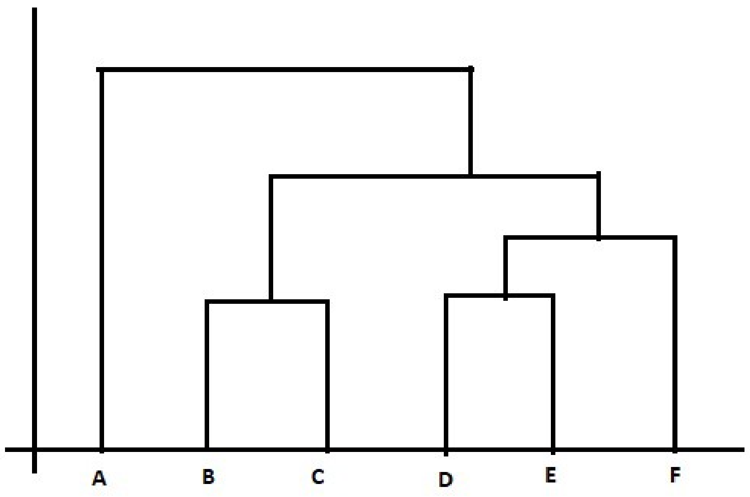


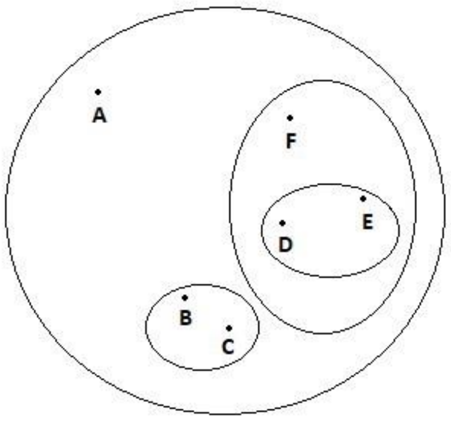
Agglomerative Hierarchical Clustering Technique

- Step- 2: In step two, similar clusters are merged together and formed as a single cluster. Let's consider B,C, and D,E are similar clusters that are merged in step two. Now, we're left with four clusters which are A, BC, DE, F.
- Step- 3: We again calculate the proximity of new clusters and merge the similar clusters to form new clusters A, BC, DEF.
- Step- 4: Calculate the proximity of the new clusters. The clusters DEF and BC are similar and merged together to form a new cluster. We're now left with two clusters A, BCDEF.
- Step- 5: Finally, all the clusters are merged together and form a single cluster.

The Hierarchical clustering Technique can be visualized using a **Dendrogram.**

A **Dendrogram** is a tree-like diagram that records the sequences of merges or splits.





Dendrogram representation

2. Divisive Hierarchical clustering Technique: Since the Divisive Hierarchical clustering Technique is not much used in the real world, I'll give a brief of the Divisive Hierarchical clustering Technique.

In simple words, we can say that the Divisive Hierarchical clustering is exactly the opposite of the **Agglomerative Hierarchical clustering.** In Divisive Hierarchical clustering, we consider all the data points as a single cluster and in each iteration, we separate the data points from the cluster which are not similar. Each data point which is separated is considered as an individual cluster. In the end, we'll be left with n clusters.

As we're dividing the single clusters into n clusters, it is named as **Divisive Hierarchical clustering.**

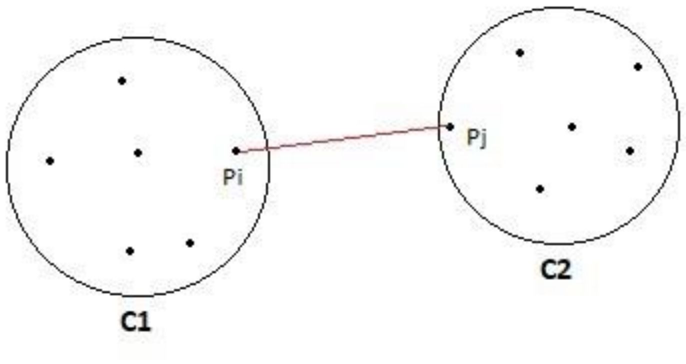
Calculating the similarity between two clusters is important to merge or divide the clusters. There are certain approaches which are used to calculate the similarity between two clusters:

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Ward's Method
- **MIN:** Also known as single linkage algorithm can be defined as the similarity of two clusters C1 and C2 is equal to the **minimum** of the similarity between points Pi and Pj such that Pi belongs to C1 and Pj belongs to C2.

Mathematically this can be written as,

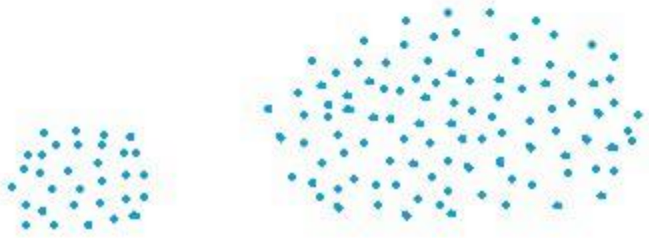
$\mathrm{Sim}(C1,C2) = \mathrm{Min}\ \mathrm{Sim}(Pi,Pj)$ such that $Pi \in C1$ & $Pj \in C2$

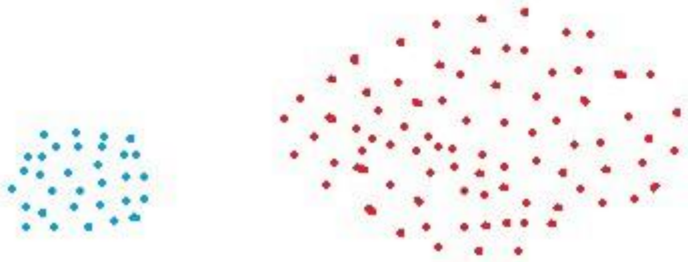
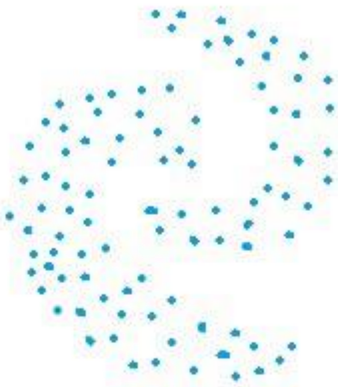
In simple words, pick the two closest points such that one point lies in cluster one and the other point lies in cluster 2 and take their similarity and declare it as the similarity between two clusters.

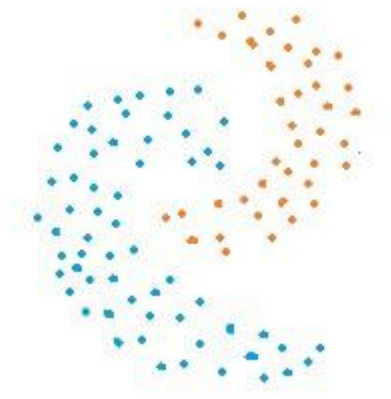


Pros of MIN:

- This approach can separate non-elliptical shapes as long as the gap between two clusters is not small.



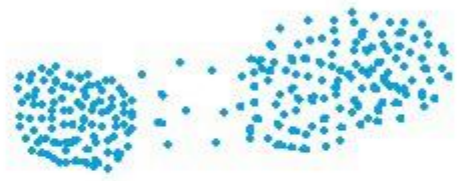



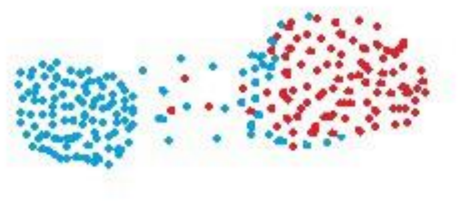


Original data vs Clustered data using MIN approach

Cons of MIN:

- MIN approach cannot separate clusters properly if there is noise between clusters.





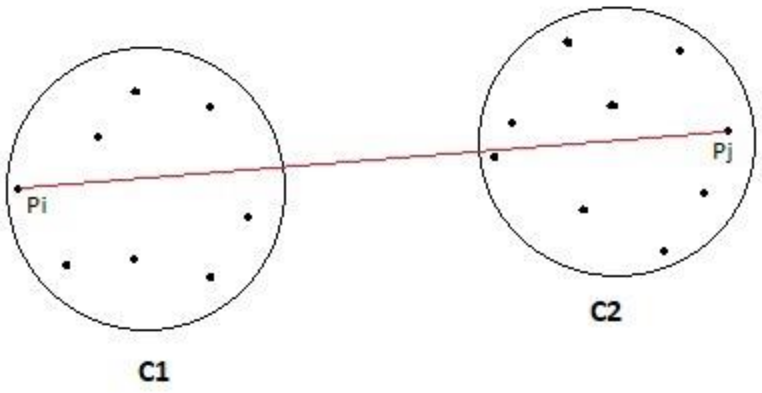
Original data vs Clustered data using MIN approach

- **MAX:** Also known as the complete linkage algorithm, this is exactly opposite to the **MIN** approach. The similarity of two clusters C1 and C2 is equal to the **maximum** of the similarity between points Pi and Pj such that Pi belongs to C1 and Pj belongs to C2.

Mathematically this can be written as,

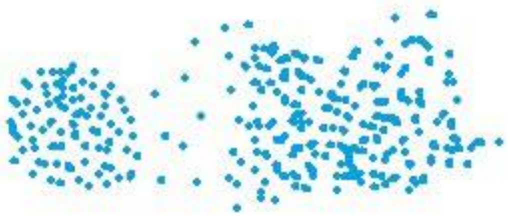
Sim(C1,C2) = Max Sim(Pi,Pj) such that Pi ∈ C1 & Pj ∈ C2

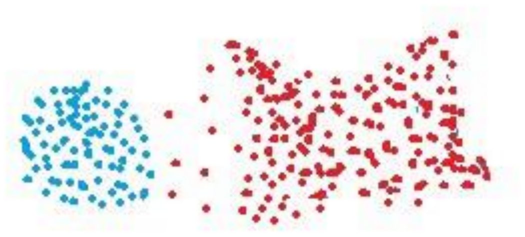
In simple words, pick the two farthest points such that one point lies in cluster one and the other point lies in cluster 2 and take their similarity and declare it as the similarity between two clusters.



Pros of MAX:

- MAX approach does well in separating clusters if there is noise between clusters.

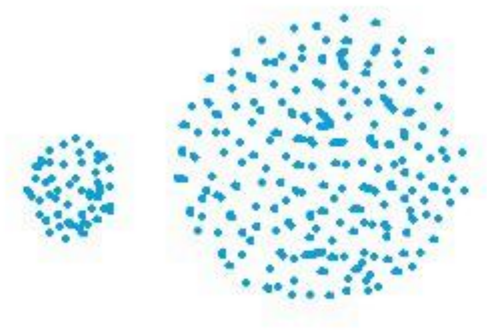


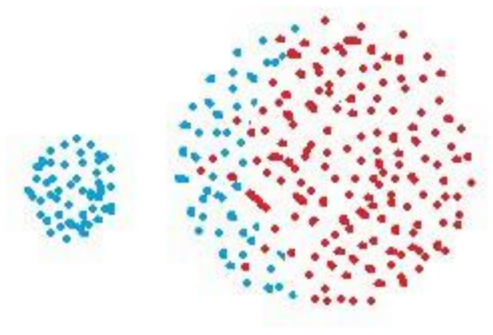


Original data vs Clustered data using MAX approach

Cons of Max:

- Max approach is biased towards globular clusters.
- Max approach tends to break large clusters.





Original data vs Clustered data using MAX approach

- **Group Average:** Take all the pairs of points and compute their similarities and calculate the average of the similarities.

Mathematically this can be written as,

sim(C1,C2) = ∑ sim(Pi, Pj)/|C1|*|C2|

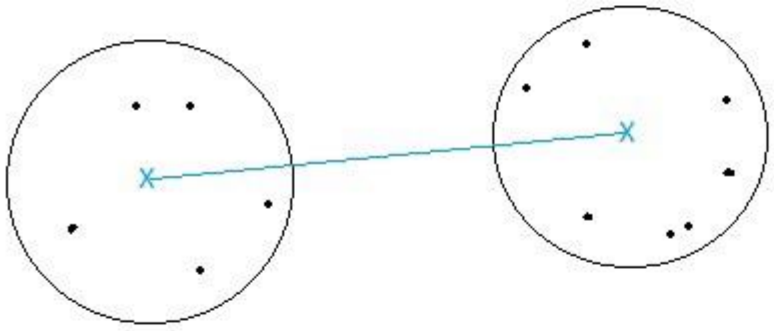
where, Pi ∈ C1 & Pj ∈ C2

Pros of Group Average:

- The group Average approach does well in separating clusters if there is noise between clusters.

Cons of Group Average:

- The group Average approach is biased towards globular clusters.
- **Distance between centroids:** Compute the centroids of two clusters C1 & C2 and take the similarity between the two centroids as the similarity between two clusters. This is a less popular technique in the real world.



- **Ward's Method:** This approach of calculating the similarity between two clusters is exactly the same as Group Average except that Ward's method calculates the sum of the square of the distances Pi and PJ.

Mathematically this can be written as,

sim(C1,C2) = $\sum$ (dist(Pi, Pj))^2/|C1|*|C2|

Pros of Ward's method:

- Ward's method approach also does well in separating clusters if there is noise between clusters.

Cons of Ward's method:

- Ward's method approach is also biased towards globular clusters.

Space and Time Complexity of Hierarchical clustering Technique:

Space complexity: The space required for the Hierarchical clustering Technique is very high when the number of data points are high as we need to store the similarity matrix in the RAM. The space complexity is the order of the square of n.

Space complexity = O(n^2) where n is the number of data points.

Time complexity: Since we've to perform n iterations and in each iteration, we need to update the similarity matrix and restore the matrix, the time complexity is also very high. The time complexity is the order of cube of n.

Time complexity = O(n^3) where n is the number of data points.

Limitations of Hierarchical clustering Technique:

1. There is no mathematical objective for Hierarchical clustering.
2. All the approaches to calculate the similarity between clusters has its own disadvantages.
3. High space and time complexity for Hierarchical clustering. Hence this clustering algorithm cannot be used when we have huge data.

Question 3: Principal Component Analysis

a) Give at least three applications of using PCA.

PCA is predominantly used as a dimensionality reduction technique in domains like facial recognition, computer vision and image compression. It is also used for finding patterns in data of high dimension in the field of finance, data mining, bioinformatics, psychology, etc.
You must be wondering many a times show can a machine read images or do some calculations using just images and no numbers. We will try to answer a part of that now. For simplicity, we will be restricting our discussion to square images only. Any square image of size ***NxN*** pixels can be represented as a ***NxN*** matrix where each element is the intensity value of the image. (The image is formed placing the rows of pixels one after the other to form one single image.) So if you have a set of images, we can form a matrix out of these matrices, considering a row of pixels as a vector, we are ready to start principal component analysis on it. How is it useful?

Say you are given an image to recognize which is not a part of the previous set. The machine checks the differences between the to-be-recognized image and each of the principal components. It turns out that the process performs well if PCA is applied and the differences are taken from the 'transformed' matrix. Also, applying PCA gives us the liberty to leave out some of the components without losing out much information and thus reducing the complexity of the problem.

For image compression, on taking out less significant eigenvectors, we can actually decrease the size of the image for storage. But to mention, on reproducing the original image from this will lose out some information for obvious reasons.

For application of PCA, you can hard-code the whole process in any programming language, be it C++, R, Python, etc. or directly use the libraries made available by contributors. However, it is recommended to hard-code in case the problem is not too complex so that you actually get to see what exactly is happening in the back-end when the analysis is being done and also understand the corner cases. Just for instance, in R, there are libraries called princomp, HSAUR, prcomp, etc. which can be used for direct application.

b) Briefly discuss the 2 important building blocks of PCA - Basis transformation and variance as information.

Introduction

Thanks to the astonishing advances of data analysis software in the last few years, data scientists have the possibility of using highly complex statistical methods by just typing the .fit() command in the prompt. This is very helpful for reducing the time needed to develop a project, but it can also have a dangerous drawback: *the need to fully understand the statistical method behind .fit() does not seem to be that important anymore*. However, what if the output of that function does not seem to make sense? What tools do we have to deal with this situation? Well, from my perspective there is only one way to deal with this scenario: *understanding the basis of the statistical method that is behind the programming command*.

This was me admiring the "magic" of the .fit() command when I first started using PCA without understanding its mathematical reasoning

More concretely, in this article I would like to explain one commonly used mathematical transformation in the Data Science field, called **Principal Component Analysis** or **PCA**, which is not transparent for those who do not understand the mathematical reasoning behind it. I am going to reduce the technicality to the minimum in order to make this article understandable to the maximum number of readers possible.

Notation

Let's start with some notation before going into the explanation of PCA. Let **X** be a matrix of dimension *n x p,* where *n* is the number of observations of a given data set and *p* is the number of predictors. Now let **S** denote the **covariance matrix** of **X**, that is,

$$\mathbf{S} = \mathbf{Cov}(\mathbf{X})$$

Remember that a covariance matrix is a *p x p* matrix where the diagonal elements correspond to the **variance** (*dispersion measure*) of the covariates in **X** and the off-diagonal elements correspond to the **covariance** (*similarity measure*) between two specific covariates. Keep in mind this straightforward detail: *the covariance between a covariate X1 and a covariate X2 is the same as the covariance between X2 and X1*. Therefore, the covariance matrix **S** is **symmetric**, which means the **S** is equivalent to its **transposed** version,

$$\mathbf{S} = \mathbf{S}^T$$

Motivating PCA

Imagine a scenario where *p* is very large and you would like to perform some kind of **dimensionality reduction**, that is, reduce the amount of predictors, in order to do some **exploratory analysis** or **fit a machine learning algorithm**. We all know that reducing the dimensionality of the problem can lead to decreasing the **variance** of any statistical model, which might compensate the increase of **bias** (**bias-variance trade-off**). However, one drawback of performing dimensionality reduction is that we might **lose important information**. At the end of the day, there is no way that reducing, say, from 1,000 covariates to 10, does not imply any kind of information loss. So the next question that raises is, *how do we measure the amount of information that each variable has?* Well, the **variance** of the covariate seems to be a good measure of information. The larger the variance the larger the amount of information the variable contains.

Let's make sure all of us understand this point because it is critical to understand PCA. *Why is a high variance of a covariate good?* Well, assume, for example, you want to understand the effect of salary on a given outcome. Do you prefer to have a set of observations whose values for salary range from 1,000 to 2,000 or from 0 to 10,000? The larger the range of the variable salary, the more information we have about how such variable affects the outcome, right? Note that in the first scenario, we have no way of understanding the effect of salary on the outcome when the salary is smaller than 1,000 or larger than 2,000. The reason is that there is no data in such part of the space. Therefore, the larger the range of the covariate the more information. And, what is the effect of the range of a variable on its variance? Exactly, the larger the range, the larger the variance (*the dispersion increases*).

Some people tend to identify the word **variance** as something negative, which makes sense because there are several contexts where it can mean something bad. For example, not only the data has variance; **estimators** such as the sample mean also have **dispersion measure**, which is called **standard error** and is defined as the square root of the variance of the estimator. Of course in this case the variance is unwanted because the larger it is, the more uncertain the estimate is. Another example where the variance might be something bad is when it is present in the outcome variable. That is, if the variable that we want to predict has very high variance, it might be more complicated to make precise predictions (under some circumstances that will not be covered in this article). *But, remember that, as explained before, predictors with high variance provide more information than predictors with low variance!*

So far we have come up with a very intuitive explanation of why we want to the variance of the original predictors to be kept when performing dimensionality reduction. But, wait… *is the variance of the predictors all we care about when measuring the amount of information?* No! Think of this, what if you have two covariates that both have a high variance, but they are extremely similar to each other (say their correlation is close to 1)? In this case, what is the additional information that the second predictor provides with respect to the first one? The answer is that that the additional information is almost null. Therefore, when performing dimensionality reduction we not only want the transformed predictors to keep the variance of the original ones, but also to make them uncorrelated.

Once we understand how a dimensionality reduction should be, let me introduce you to the definition of PCA: **PCA is a mathematical approach that transforms the matrix of predictors X into another one of the same dimension, call it Y, such that**

- **the covariance matrix of Y is diagonal,** meaning that all the transformed predictors are uncorrelated, and
- **the transformed predictors are sorted by a decreasing amount of information**, meaning that the diagonal entries of the covariance matrix of the transformed predictors, which contain their variances (amount of information) as explained earlier, decrease as we move to the right (or down) throughout the matrix.

Is there a mathematical way to achieve such a transformation? Well, there is! Let’s deepen a bit into how PCA work mathematically without getting very technical.

Simplified Mathematical Development
Some of you might feel like this from this moment on. Stay with me, read it a second time if necessary and you will see it is not that hard

As we said before, let **X** be the original predictor matrix and **Y** the transformed predictor matrix, both of dimensions *n x p*. Remember the ideal properties that **Y** should have (uncorrelated predictors, which must be ordered in decreasing order of information). In order to transform **X**, we perform what is known in **linear algebra** a **change of variable**, which implies to multiply **X** by another unknown matrix **P** of dimensions *p x p* in order to come up with **Y**. That is:

$$\mathbf{XP} = \mathbf{Y}$$
$$\mathbf{X} = \mathbf{YP}^{-1}$$

So far we know nothing about **P** besides that it has to be invertible in order to reconstruct **X**. Hence, the goal is to find a matrix **P** that performs a change of variable with the ideal characteristics that we are looking for.

In order to move forward we have to make use of a linear algebra theorem called the **Diagonalization Theorem** which will not be proven (do not worry, this is this only moment where I make use of something that I do not explain where it comes from; you can look for a mathematical proof, but it will not be necessary to understand PCA). It says that a symmetric matrix, like the **covariance matrix of X**, also written as **S**, is **diagonalizable** as follows

$$\mathbf{S} = \mathbf{PDP}^{-1}$$

where **D** is a diagonal matrix of dimensions *p x p* and the matrix **P** represents the same matrix used in the change of variable from above.

Note that we still do not know anything about **P**. Let's show an important property of this matrix that will be used later on to come up with PCA. It turns out that **P** is an **orthogonal matrix**, which means that its transposed version is equivalent to its inverse version. That is,

$$\mathbf{P}^T = \mathbf{P}^{-1}$$

Why does the previous equation hold? To show this, we need to remember that the covariance matrix is symmetric and, therefore, it is equivalent to its transpose

$$\mathbf{S} = \mathbf{S}^T$$

Let’s see how the needed condition that makes the previous equation hold is that **P** has to be orthogonal:

$$
\begin{aligned}
\mathbf{S}^T &= (\mathbf{P}\mathbf{D}\mathbf{P}^{-1})^T && \text{(Diagonalization Theorem)}\\
&= (\mathbf{P}^{-1})^T\mathbf{D}^T\mathbf{P}^T && \text{(Apply transpose to all matrices)}\\
&= (\mathbf{P}^{-1})^T\mathbf{D}\mathbf{P}^T && \text{(Transpose of diagonal matrix does not change the original)}\\
&= (\mathbf{P}^T)^T\mathbf{D}\mathbf{P}^T && \text{(If P is orthonormal)}\\
&= \mathbf{P}\mathbf{D}\mathbf{P}^T && \text{(Transpose of transpose is equal to original)}\\
&= \mathbf{S} && \text{(Covariance matrix is symmetric)}
\end{aligned}
$$

Hence, it has been proven that **P** is orthogonal given that it is the only way to prove that the covariance matrix of **X** is symmetric. But, *how does this finding help up to fulfill creating uncorrelated predictors?* To see this, let’s derive the covariance matrix of the transformed predictors:

$$
\begin{aligned}
\mathbf{Cov}(\mathbf{Y}) &= \mathbf{Cov}(\mathbf{X}\mathbf{P}) && \text{(Definition change of variable)}\\
&= \mathbf{P}^T\mathbf{Cov}(\mathbf{X})\mathbf{P} && \text{(Apply covariance properties)}\\
&= \mathbf{P}^T\mathbf{S}\mathbf{P}
\end{aligned}
$$

In order to prove that the covariance matrix of Y is diagonal, which would mean that the predictors are uncorrelated, I am going to demonstrate that the last expression of the previous derivation is equal to a diagonal matrix. Let’s start:

$$
\begin{aligned}
\mathbf{S} &= \mathbf{P}\mathbf{D}\mathbf{P}^{-1} && \text{(Diagonalization Theorem)}\\
\mathbf{S} &= \mathbf{P}\mathbf{D}\mathbf{P}^T && \text{(P is orthonormal)}\\
\mathbf{P}^{-1}\mathbf{S} &= \cancel{\mathbf{P}^{-1}\mathbf{P}}\mathbf{D}\mathbf{P}^T && \text{(Multiply both sides by inverse of P)}\\
\mathbf{P}^{-1}\mathbf{S} &= \mathbf{D}\mathbf{P}^T\\
\mathbf{P}^{-1}\mathbf{S}(\mathbf{P}^T)^{-1} &= \mathbf{D}\cancel{\mathbf{P}^T(\mathbf{P}^T)^{-1}} && \text{(Multiply both sides by inverse of transpose of P)}\\
\mathbf{P}^{-1}\mathbf{S}(\mathbf{P}^T)^{-1} &= \mathbf{D}\\
\mathbf{P}^T\mathbf{S}(\mathbf{P}^T)^T &= \mathbf{D} && \text{(P is orthonormal)}\\
\mathbf{P}^T\mathbf{S}\mathbf{P} &= \mathbf{D} && \text{(Transpose of transpose is equal to original)}\\
\mathbf{Cov}(\mathbf{Y}) &= \mathbf{D} && \text{(Proven above)}
\end{aligned}
$$

The last mathematical issue to prove is that the **total information** of the principal components is the same as the total information of the original predictors. That is, let's show that by applying PCA we have not lost information (we have just allocated it differently). Remember the definition of change of variable from the beginning. Given that we have proven the **P** is orthogonal we can write the change of variable as:

$$\begin{aligned} \mathbf{X} &= \mathbf{Y}\mathbf{P}^{-1} && \text{(Definition change of variable)} \\ &= \mathbf{Y}\mathbf{P}^{T} && \text{(P is orthogonal)} \end{aligned}$$

This is a special case called **orthogonal change of variable**, which allows the total variance of the data to be kept unchanged because the multiplication of an orthogonal matrix does not change the lengths of the vectors nor their angles (*the vectors of an orthogonal matrix are orthonormal, which means that they have length one and they are perpendicular to each other*). Therefore, the total variance (or total amount of information) or the original predictors can be written as:

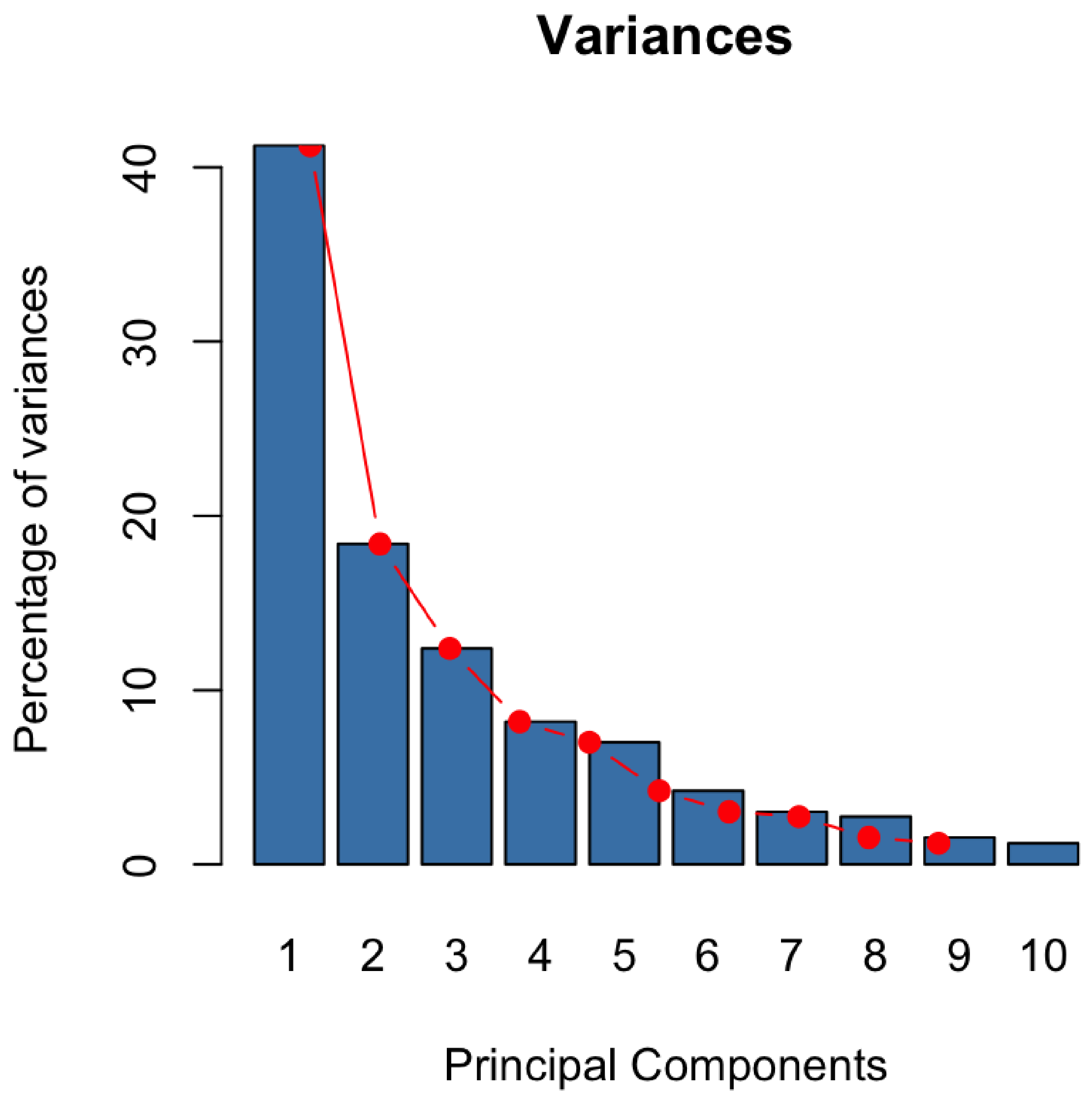
$$\textbf{Total Variance of } \mathbf{X} = \textbf{Total Variance of } \mathbf{Y} = trace(\mathbf{D})$$

where **trace** is simply the sum of the diagonal entries of a given matrix. Given that **D** contains all the variances of the principal components, their sum measures the total amount of information contained in **X**.

Last Clarifications

Done! We have just proven that PCA leads to uncorrelated predictors! This way, every transformed covariate contains unique information that the others do not. Each of these transformed covariates are called **principal components**. The **first principal component** corresponds to the first column of **Y**, which is also the one that has the most information because we order the transformed matrix **Y** by decreasing order of the amount of contained information (the first diagonal entry of the covariance matrix of **Y** is the highest one). Likewise, the second column of **Y** is called the **second principal component**. So on and so forth.

The next and final step is to decide how many principal components to use in our analysis. This decision will be different in each scenario. A common approach is to create a barplot with the amount of information that each principal components has and see if there is a point where including more principal components leads to very small increase in information which does not compensate the increase in dimensionality. This barplot has the following shape:



Conclusion

The goal of this article was not to explain all the technicalities about PCA. In fact, several critical concepts such as the role of **eigenvectors** and **eigenvalues** have not been mentioned even though they play an important role in PCA. The goal instead was to help beginners in PCA get an intuitive overview of what it does without losing ourselves too much in mathematical details.

c) State at least three shortcomings of using Principal Component Analysis.

As noted above, the results of PCA depend on the scaling of the variables. This can be cured by scaling each feature by its standard deviation, so that one ends up with dimensionless features with unital variance

The applicability of PCA as described above is limited by certain (tacit) assumptions made in its derivation. In particular, PCA can capture linear correlations between the features but fails when this assumption is violated. In some cases, coordinate transformations can restore the linearity assumption and PCA can then be applied.

Another limitation is the mean-removal process before constructing the covariance matrix for PCA. In fields such as astronomy, all the signals are non-negative, and the mean-removal process will force the mean of some astrophysical exposures to be zero, which consequently creates unphysical negative fluxes, and forward modeling has to be performed to recover the true magnitude of the signals. As an alternative method, non-negative matrix factorization focusing only on the non-negative elements in the matrices, which is well-suited for astrophysical observations.