**Clustering Optimization using k-means**

**Preface**

In this paper, we shall demonstrate two approaches for optimizing the k-means algorithm for clustering.

The problem of clustering is a good example of an optimization problem. The problem is, first, how to divide a set of n elements into a given number of $k$ groups. We have many ways to do this, and we need to find the optimal grouping, assigning to each group its best matching elements.

Moreover, in real-life problems, $k$ itself is unknown and needs to be optimized by the machine itself. We can theoretically declare the whole set of elements as one cluster, or consider each element as its own cluster, but that would be pointless. Thus, we need to find a way to optimize the choice of the number $k$, assuming the first problem itself (finding the optimal grouping, for a given $k$) has an optimal solution.

There are two kinds of clustering that we would like to consider, which may affect the algorithm that we run,

**1. Clustering of scattered details**

In this kind of clustering, we try to identify clusters from details that are scattered along the image. This can have various applications, such as **Military Intelligence** (clusters of troops, aircraft, armored vehicles), **Nature Science** (clustered structures of birds, insects, fish), and many more.

**2. Identification of objects as clusters**

In this kind of clustering, we try to identify large objects, from samples found

in the image. A major example of this kind of clustering would be face recognition, where we have many features in some image, and we try to isolate and identify the face of a person, or of several people.

**Definition of the problem**

First, we shall observe the obvious fact that there is a finite number of choosing $k$ clusters out of $n$ elements.

In fact, this number is known as a **Stirling number of the second kind**, it is marked as $\left\{ {n \atop k} \right\}$ and is given by the formula

$S(n,k) = \left\{ {n \atop k} \right\} := \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n$

Moreover, if we are not given the value of $k$, then the total number of options, to choose any $1 \leq k \leq n$ from $n$, is known as **Bell number**, and is given by $B_n := \sum_{k=0}^{n} \left\{ {n \atop k} \right\}$

In addition, we shall be able to attach a score to each partition of $k$ from $n$, which measures the match between the center point and the locations of the associated elements

This score is known as **WCSS** (Within-Cluster Sum of Square),

which means, we calculate the squared distances of all the elements from their associated clusters,

That is (using $w(k)$ instead of WCSS($k$)),

$w(k) := \sum_{j=1}^{k} \sum_{i=1}^{m_j} (c_j - p_{j_i})^2 = \sum_{j=1}^{k} \sum_{i=1}^{m_j} ((x_j - x_{j_i})^2 + (y_j - y_{j_i})^2)$

Where

$m_j$ - the number of elements associated to cluster $j$

$c_j = (x_j, y_j)$ - the center point of cluster $j$

$p_{j_i} = (x_{j_i}, y_{j_i})$ - the point $i$ of cluster $j$

So, for a given $k$, our goal is to find arg min(w($k$)), which is a classical optimization problem.

**The basic k-means algorithm**

One of the very well-known algorithms for clustering is the k-means algorithm.

This algorithm is based on a very simple concept of acquiring initial data, then adjusting this data until the algorithm stables. This algorithm is called k-means, because we are trying to find $k \in \mathbb{N}$ clusters, out of $n \in \mathbb{N}$ elements, which are supposed to give the optimal clustering because each cluster has a center point, which is the mean of all the points that are grouped together in this cluster. We call this cluster, or the center point of this cluster, a **centroid**.

The basic description of this algorithm, for a given $k$, is,

**1. Initialization** Initialize a set of $k$ centroids within the pixels of the image.

**2. Association** For each element, calculate its distance from the center of each cluster, find the centroid with the minimal distance from this element, and associate the element to this centroid.

**3. Recalculation** Using the association of the elements, calculate the center point of each centroid again, by taking the mean point of all its associated elements.

**4. Iteration** Iterate steps 2 and 3 until the algorithm turns stable, that is, there are no more moves of associated elements between centroids.

**Algorithm 1** Calculate k-means

**Require:**
**Ensure:**
$r \leftarrow true$

$L \leftarrow size(samples)$

**while** $r$ is $true$ **do**
$a \leftarrow 0$

$i \leftarrow 0$
**while** $i < L$ **do**
$s \leftarrow samples[i]$

$f \leftarrow null$
$m \leftarrow null$

$j \leftarrow 0$
**while** $j < k$ **do**
$c \leftarrow centroids[j]$

$dx \leftarrow s.x - c.x$
$dy \leftarrow s.y - c.y$
$d2 \leftarrow dx^2 + dy^2$

**if** $m$ is null **or** $d2 < m$ **then**
$m \leftarrow d2$
$f \leftarrow c$
**end if**

**if** $s.c \neq f$ **then**
$a \leftarrow a + 1$
**end if**

$j \leftarrow j + 1$
**end while**

$i \leftarrow i + 1$
**end while**



**if** $a < 1$ **then**
$r \leftarrow false$
**end if**

**end while**

This algorithm, as described, is promised to converge, that is, to achieve a stable state, where the stopping condition (no more moves between centroids) is satisfied.

The proof of convergence is given by the basic observation that the number of grouping options, for a given number of $k$ clusters, out of $n$ elements, is obviously finite, and that on each step, we get a better score on the clustering. A full proof can be found at [1]

**Proposition** The k-means algorithm does not necessarily give an optimal solution, for a given $k$

**Explanation** The given proof only proves that if we start from some initial setting of the system, we are sure to converge, at some point. However, this convergence is to a local minimum only, because, if we start from a different setting, we may converge to another local minimum, possibly to the best existing solution, which we shall refer to as the global minimum.

**The Elbow method**

Recall from above, one major problem that we have, with the k-means algorithm, is that the native algorithm requires an input number of k, for running. We also recall that the total number of ways to group $n$ points to $1 \leq k \leq n$ clustering is given by the Bell number, which is a sum of the Stirling number, for each $k$, thus significantly larger.

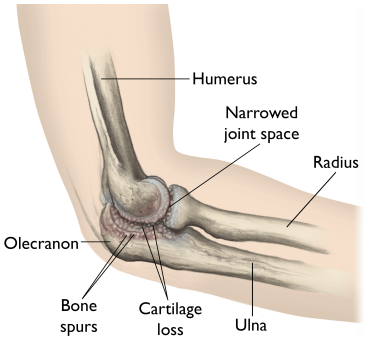
To automatically choose the optimal number of $k$, we need a way to compare the scores of different values of $k$, running under the same conditions. This brings us to an optimization method, called the **Elbow Method**. The basic concept of this method is that we can compute some score on each $k$, and then present this score as a function of $k$.

At first glance, we are looking for a $k$ value that will yield the minimal

WCSS.
However, we can observe, as mentioned before, that we are not looking for a minimal value, but rather for the optimal value. Indeed, if we continue to calculate the k-means for higher values of $k$,
The WCSS will decrease to the minimum, without giving us any benefit, but actually ruining the clustering.
It is a trivial observation since we can always take $k = n$, that is, declare each element as a separate cluster, thus obtaining $w(k = n) := \sum_{j=1}^{k} \sum_{i=1}^{m_j} 0 = \sum_{j=1}^{k} 0 = 0$,
which is clearly the minimal WCSS possible, while it is obvious that this clustering result is absolutely wrong.

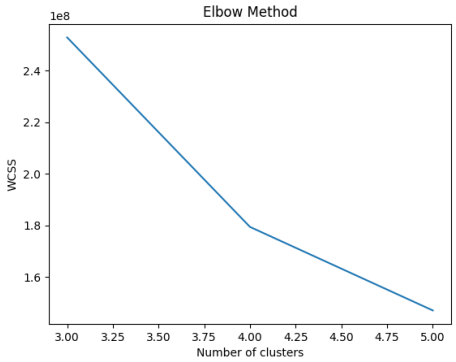
Taking a range of $k$ values, that is, $\{k_1, k_2, k_3, \ldots, k_m\}$, we shall observe that for the lower values of $k$, the function is decreasing rapidly, while after a certain value of $k$, the function is taking a significant turn, from a high (negative) slope, into a nearly asymptotic graph. This means that for less than the optimum $k$ value, our clusters are too large, and for more than the optimum, the more $k$ clusters we calculate, in the k-means algorithm running, we do not add any improvement for the clustering, but exactly the opposite, meaning the output clusters will split the real clusters in the image, and not give us any beneficial clustering information.

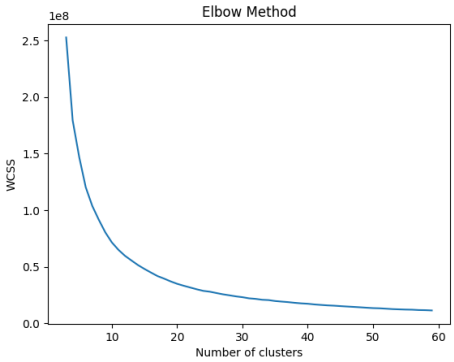
In other words, the optimal number of $k$ is the turning point of the graph, from the high slope to the asymptote. This is why it is called "elbow" because it resembles a folded arm and the elbow that is the outmost point in the arm.




So, if we can compute different ranges of numbers, for different maximal $k$ values, $\{m_1, m_2, m_3, \ldots, m_l\}$, and we get the same elbow (that is, a specific value of $k$), for each $m_i$, then we have the optimal number of $k$, for this image. We can even assume that the optimum will move up and down, but will maintain some boundaries, from which we can take the average $k$, with or without some weight or probability considerations.

We would like to have a clear elbow point, for each image, so we can have a simple algorithm to calculate it,
but in reality, this is hardly the case. For example, we compare two computations of the elbow method, on the same image,
one is computing a range of $\{3, 4, 5, 6\}$ clusters, while the other is computing a range of $\{3, 4, ..., 60\}$.



While the 6 clusters computation has a clear elbow point (at $k = 4$),
the 60 clusters graph is schematically looking like exponential decay,
which clearly does not have a minimum.

So, we need to find a way to compute the optimal elbow point.
Our approach, for this problem, is based on the following observation,
If we draw a straight line, $A$, between the first $k$ and last $k$ points $(k_0, w_0), (k_m, w_m)$,
where $w_i = w(k_i)$
We can see that the elbow is the most distant point from this line (when the distance from a point $(k_i, w_i)$ to $A$ is given by $\min\{\|(k_i, w_i) - a\| : a \in A\}$

So, a simple way for us to calculate the most distant point, would be to rotate the set of points,
so the first and last $k$ have the same value of $y$, and calculate the lowest $y$ of all the $k$ clusters that we have computed.
This can be achieved by rotating all the $(k_i, w(k_i))$ around $(k_0, w(k_0))$
This can be done simply by using a rotation matrix, of the form,

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

where $\theta = tan^{-1}(\frac{w_m - w_0}{k_m - k_0})$
So, for $1 \leq i \leq m$, mark $(x'_i, y'_i)$ as $(x_i, y_i)$ rotated by $\theta$,

$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Going back to the elbow method from above, we need to construct an algorithm that will automatically determine the optimal $k$ values, by finding the elbow. So, basically, our algorithm outline will look like this,
**1.** set $k$, an initial number of desired clusters
**2.** calculate $k$ centroids, using the k-means algorithm from above

**3.** calculate $w(k)$, if it is the optimal value, return $k$
**4.** if $k = n$, return $k$. otherwise, increment $k$, and repeat step 3.

So, this algorithm can run, theoretically, until $k = n$.
However, we can observe that this is not exactly the desired algorithm, because,
how do we know that we have achieved the elbow, that is, the optimal value?

So, we edit this algorithm in a slightly different manner,
**1.** set $s$, the number of successful optimal $k$ calculations
**2.** set $t$, the tolerance for optimal $k$ values.
**3.** set $q$, a number representing a quant of $k$ values to calculate
**4.** set $k$, an initial number of desired clusters
**5.** calculate $k$ centroids, using the k-means algorithm from above
**6.** calculate $w(k)$, and store it in an array
**7.** if $k$=0 mod $q$, calculate the optimal $k$ using all the stored WCSS values and store it in an array
**8.** if the array of optimal $k$ values has $s$ identical values of $k$, or if their difference is in the range of $t$, we take their value (in case they are identical), or some average between them, and return this value as the final $k$ value.
**9.** if $k = n$, return $k$. otherwise, increment $k$, and repeat step 5.

**Algorithm 2** Calculate elbow for k-means

**Require:**
$q$: quant of $k$ for optimum calculations
$s$: number of successful elbow calculations
$t$: tolerance of $k$ value

**Ensure:**
$L \leftarrow size(samples)$

$a \leftarrow$ float[]
$b \leftarrow$ integer[]

$r \leftarrow true$

**while** $r$ is $true$ **do**
  $a \leftarrow 0$

  $i \leftarrow 0$
  **while** $i < L$ **do**
    $s \leftarrow samples[i]$

    $f \leftarrow null$
    $m \leftarrow null$

    $j \leftarrow 0$
    **while** $j < k$ **do**
      $c \leftarrow centroids[j]$

      $dx \leftarrow s.x - c.x$
      $dy \leftarrow s.y - c.y$
      $d2 \leftarrow dx^2 + dy^2$

      **if** $m$ is null **or** $d2 < m$ **then**
        $m \leftarrow d2$
        $f \leftarrow c$
      **end if**

      **if** $s.c \neq f$ **then**
        $a \leftarrow a + 1$
      **end if**

      $j \leftarrow j + 1$
    **end while**

    $i \leftarrow i + 1$
  **end while**

**A statistical k-means hybrid algorithm**

Our approach is a hybrid method, which is using the basic k-means algorithm, together with
statistical considerations.

The basic concept of our method is making practical use of the **CLT** – the Central Limit Theorem.
CLT is a well-known fundamental theorem in probability and statistics, which says that given a sequence of $n$ random variables, $\{X_i\}_{i=1}^n$, with mean $\mu$ and variance $\sigma^2$, we mark the average $\bar{X_n} := \frac{\sum_{i=1}^n X_i}{n}$, then,
the sequence $\frac{\bar{X_n}-\mu}{\sigma/\sqrt{n}}$ converges in distribution to the normal (Gaussian) distribution, with $\mu$ and $\sigma^2$ as parameters.

This theorem, although appearing abstract, since we are referring to the convergence of the sequence as $n$ tends to infinity, is actually very useful in applications, since we practically do not need a significantly large $n$, but it works also on small values of $n$.

This allows us to utilize CLT for our purposes. We can run the basic k-means algorithm on the elements, but then we can automatically split clusters by a very basic statistical observation.

The normal distribution, in statistics, is used for calculating, for each random variable a score, which is called a **Z-score** or a **standard score**. This means that we have a linear scale, for measuring the distance of each random variable from the mean, in units of standard deviation, and there exists a formula, which gives us the probability for each Z-score. For each sample, $X_i$, its Z-score is $Z_i := \frac{X_i-\mu}{\sigma}$. The probability, $\mathbb{P}(Z_i < z)$, can be found in a standard table of already calculated values.
Also, we will be interested in calculating the absolute value of the Z-score, since, for our specific problem, there is no meaning to the question is the sample is located "before" or "after" the center point of the centroid, so there is no difference between $Z_i = \frac{X_i-\mu}{\sigma}$ and -$Z_i = \frac{\mu-X_i}{\sigma}$

And so, if we say that a specific Z-score is higher or lower, we shall refer to its absolute value.

Thus, the algorithm, after combining this statistical calculation, is,

**1.** set an initial value of k, which can be quite small.

**2.** associate all the elements to their closest centroid.

**3.** for each centroid, calculate its mean, std, and the Z-score of each associated element.

**4.** for each centroid, count the number of elements that have a Z-score higher than X. If the count is higher than Y, create a new centroid, and associate all these elements to this centroid. If the count is lower than Y, then check the "ignore" flag. If it is true, then remove the association of these elements to the current centroid. If it is false, then keep these elements associated with the current centroid.

**5.** for each centroid, calculate its center point again, as the mean of all the associated elements.

**6.** while the number of clusters is increasing, or associated elements are still moving from one cluster to another, repeat step 2.

**7.** (after the algorithm turned stable) return the current list of centroids.

**Proposition** The algorithm above is converging to a local minimum.

**Explanation** This is trivial since each iteration is computing the k-means algorithm, for a given $k$, and this algorithm is promised to converge to a local minimum, as stated above.

The splitting of the clusters, using the statistical criteria, must obviously stop, at some point, because, theoretically, we can split the clusters until each element is declared a separate cluster (in this case, we will get a convergence, but the whole clustering will be, of course, useless). However, using the minimal size of a cluster we are setting for the algorithm, the splitting is expected to stop when the resulting clustering is a reasonable one.

## Bibliography

[1] [2] https://towardsdatascience.com/elbow-method-is-not-sufficient-to-find-best