K-Means Clustering

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition **n observations** into **k clusters** in which each observation belongs to the cluster with the nearest **mean** (cluster centers or cluster **centroid**), serving as a prototype of the cluster.

https://en.wikipedia.org/wiki/k-means_clustering

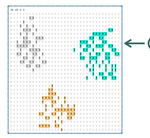
K-means Clustering is a clustering algorithm that works on data that has not been labeled yet. This means that K-means Clustering is an ***Unsupervised Learning*** *algorithm. K-means simply partitions the given dataset into groups (clusters) where each group has different features. It refers to the total number of clusters existing in the given dataset. The boundaries of each cluster are determined by the euclidean proximity between each of the cluster's members and the cluster's centroid.*

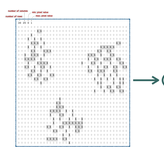
K-means Clustering works best for 2-dimensional numerical data. In real applications, feature vectors (i.e. data points) usually have multiple dimensions, hence, dimensionality reduction has to be performed before K-means can be applied.

Task:

Given the following binary image (2-dimensions) containing 3 objects, where the background pixels have the value of 0 and the objects the value of 1, detect the objects by homogeneously labeling their constituent pixels. E.g.: label pixels belonging to:

object-1 as 1
object-2 as 2
object-3 as 3





PointSet

x	y
x_1	y_1
x_2	y_2
x_3	y_3
⋮	⋮
x_n	y_n

Cartesian coordinates of each pixel of value == 1

1 - randomly select centroids
2 - read closest
3 - find centroids
4 - reset closest each centroid for each p
5 - find the optimal distance (sum)
6 - repeat steps 2 to 5 until
7 - on step 4 fix for all points
8 - repeat 2 to 7 until no more changes

1 Extract points from the binary image. → **PointSet** → **2** K-means clustering method → **3** Display the labeled points

Extract the x_i, y_i-coordinates of ALL the points belonging to objects (i.e. pixels==1) from the above binary image into an output.txt file.

extracting points

Algorithm 1

Algorithm - steps for extracting points from a binary image:

- step-0: infile ← open input.txt image file (and all other initializations)
- step-1: read the image L ←R & T ←B — i.e. left to right , top to bottom
- step-2: p(x,y) ← next point from image — i.e. point (pixel) **p** coordinates x, y
  output x , y — i.e. output x,y coordinates to PointSet out-file
- step-3: repeat steps 1 to 2 until eof — i.e. eof means 'end of file'

Sometimes there will be an oscillating point that toggles e.g. blue to burgundy to blue. These points are very rare, increasing the precision of distance measuring (e.g. float → double) should fix that.

Algorithm 2

k-mean clustering

Algorithm steps for k-mean clustering:

step-0: instantiations (allocation), initializations
pointSet ← open input image file and extract all points
n ← number of points (count the points when extracting)
k ← is given by the user
change ← 0 ; *1 obviously approved*
2 "Forgy" method, or random partition: for the Centroids linear alpha value each Closest/Assign another logic (randomly) then there is no change and we are done
3. define "Forgy" as the number in the "total distance" from all labels to a cluster) is then the sum of squares, which is the "variance" of the points in the cluster, and the "means" has been used as well as from corresponding "total distance" not have to be stored.

$c_1, c_2, \ldots, c_k$ ← clusters
$c_1, c_2, \ldots, c_k$ ← centroids

step-1: **randomly** select **k** different points from among the points $\mathbf{p}(x,y)$ in the **pointSet** and assign them as starting **centroids**, where $1 \le i, j \le n$
$c_1, c_2, \ldots, c_k$ ← some random $\mathbf{p}(x,y)$
go to step-4

step-2: Reset changes
change ← 0

step-3: Find centroids
for each cluster $c_1, c_2, \ldots c_k$ ← compute **centroid** c_i ($x(c_i)$, $y(c_i)$), where $1 \le i \le n$.

$$x(c_i) = \frac{1}{|c_i|}\sum_{p \in c_i} x(p) \qquad y(c_i) = \frac{1}{|c_i|}\sum_{p \in c_i} y(p)$$

sum of the x-coordinates of points belonged to cluster c_i divided by the cardinality of cluster c_i (number of points in cluster c_i)

sum of the y-coordinates of points belonged to cluster c_i divided by the cardinality of cluster c_i (number of points in cluster c_i)

e.g. Centroid c_1 ($x(c_1)$, $y(c_1)$) — Note: The Center of Mass of $\mathbf{p}(x,y)$ belonged to c_1

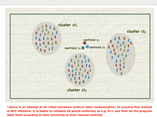
step-4: For each point **p** in **pointSet**, find distance from **p** to each centroid c_i
p ← get a point from among **pointSet**
compute **k** distances:
dist$_1$(distance from **p** to centroid c_1)
dist$_2$(distance from **p** to centroid c_2)
⋮
dist$_k$(distance from **p** to centroid c_k)

$$\mathbf{distance} = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2} \qquad \mathbf{dist}_i\ (x_i, y_i)$$

step-5: Find min distance's label
minLbl ← min(**dist**$_1$, **dist**$_2$, ..., **dist**$_k$)

step-6: Relabel point **p** and update the total **change**, where **change** keeps track of how many points have been relabeled
if(**p.label** ≠ **minLbl**)
p.label ← **minLbl**
change+=1

step-7: Repeat step-4 to step-6 until ALL **n** points have been processed



OR use

Intel.:
Centroid (the center of mass) in mathematics and physics, the centroid or geometric center of a plane figure is the arithmetic mean position of all the points in the figure. Informally, it is the point at which a cutout of the shape could be perfectly balanced on the tip of a pin.

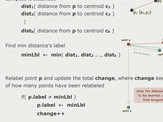
step-1: c_i ← get a cluster from among the clusters
step-2 compute centroid c_i for c_i
step-3 repeat step 3.1 to 3.2 until all clusters are processed

step-3.1 c_i ← get a cluster from among the clusters
step-3.2 compute centroid c_i for c_i
step-3.3 repeat step 3.1 to 3.2 until all clusters are processed

extra dimensions

Note that our example computes only in 2-dimensions and so it will not be necessary to increase the number of dimensions (e.g. to 3-dimensions because our feature vector has 2 attributes), we must compute the centroid for each of these extra dimensions (e.g. d_3, d_4, ..., d_n) and the "dist" function will extend the distance calculation to those extra dimensions:

distance $= \sqrt{(x_1-x_2)^2 + (x_2-x_2)^2 + (x_3-x_3)^2}$



**where is an attempt of an initial assistance uniform initial randomization. In practice this method is NOT effective it is better to initialize all points uniformly (e.g. K-11) and then let the program build them according to their proximity to their nearest centroid.*

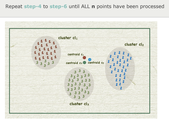
step-4: For each point **p** in **pointSet**, find distance from **p** to each centroid c_i
p ← get a point from among **pointSet**
compute **k** distances:
dist$_1$(distance from **p** to centroid c_1)
dist$_2$(distance from **p** to centroid c_2)
⋮
dist$_k$(distance from **p** to centroid c_k)

$$\mathbf{distance} = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2} \qquad \mathbf{dist}_i\ (x_i, y_i)$$

step-5: Find min distance's label
minLbl ← min(**dist**$_1$, **dist**$_2$, ..., **dist**$_k$)

step-6: Relabel point **p** and update the total **change**, where **change** keeps track of how many points have been relabeled
if(**p.label** ≠ **minLbl**)
p.label ← **minLbl**
change+=1

step-7: Repeat step-4 to step-6 until ALL **n** points have been processed



standard deviation If in a measure of how spread out the numbers are.

If a set of **variance**

variance is the average of the squared differences from the mean. steps for computing variance:
step1 find the mean (find the simple average of the numbers)
step2 for each number: subtract the mean and square the result
step3 find the average of the squared differences (from the mean. Informally it measures how far a set of (random) numbers are spread out from their average (mean).
step4 compute square root

$$\text{variance} = \frac{|\text{mean}_1 - \text{mean}_2|^2 + |\text{mean}_2 - \text{mean}_2|^2}{n}$$

In probability theory and statistics, **variance** is the expectation of the squared deviation of a random variable from its mean. Informally, it measures how far a set of (random) numbers are spread out from their average (mean).

variance = large variance
compact = small variance

What if we do NOT know k?
- in step3 k (variance) = start with 1
- in step7 compute **variance** for each k-cluster then sum-up
- repeat steps 2 to 7 until variance is acceptable (to the user). otherwise k++.

©Copyright by Martin Gregory Sendrovicz 2023
All Rights Reserved

"IntelliSeed" algorithm by Martin Gregory Sendrovicz (May 15, 2023)

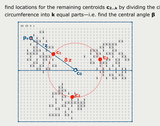
step-1: consider ALL the data-points and find their collective centroid $\mathbf{c_0}$
$\mathbf{c_0}(x_0, y_0)$ ← coordinates x_0 , y_0 where:

$$x_0 = \frac{\sum_{i=1}^{n} x_i}{n} \qquad y_0 = \frac{\sum_{i=1}^{n} y_i}{n} \qquad \text{where } n = \text{total number of data points}$$

step-2: find point $\mathbf{p_0}$ such that $\mathbf{p_0}$ is the farthest point away from centroid $\mathbf{c_0}$
(let's call this distance $\mathbf{z}$)
$\mathbf{p_0}(x_i, y_i)$ ← farthest point from $\mathbf{c_0}$
$\mathbf{z}$ ← 0
for ALL points $\mathbf{p_i}$, $1 \le i \le n$:
$\mathbf{z_{temp}}$ ← dist($\mathbf{c_0}, \mathbf{p_i}$)
if $\mathbf{z_{temp}} > \mathbf{z}$:
$\mathbf{z} \leftarrow \mathbf{z_{temp}}$
$\mathbf{p_0} \leftarrow \mathbf{p_i}$

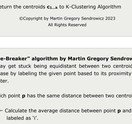
step-3: use $\mathbf{c_0}$ as the center of the circle with radius $\mathbf{r} = \mathbf{S}\,\mathbf{z}$ where $\mathbf{S} = e.g.\ ⅓$
Equation of the circle with center (h,k) and radius r
$(x-h)^2 + (y-k)^2 = r^2$; *S can be a parameter adjusted heuristically.*
so: **circle** = $(x - x_0)^2 + (y - y_0)^2 = (\mathbf{S}\,\mathbf{z})^2$

step-4: place 1+ centroid $\mathbf{c_i}$ at $\mathbf{S}$ **z distance away** $\mathbf{c_0}$ at the intersection of the circumference of the **circle** and the segment joining $\mathbf{c_0}$ and $\mathbf{p_0}$.



$\mathbf{c_1}$ ← coordinates ($\mathbf{x_1}$, $\mathbf{y_1}$)

step-5: find locations for the remaining centroids $\mathbf{c_2}, \mathbf{c_3}$ by dividing the circle's circumference into 4 equal parts—i.e. find the central angle $\boldsymbol{\beta}$



step-6: return the centroids $\mathbf{c_0}, \mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}$ to K-Clustering Algorithm

©Copyright by Martin Gregory Sendrovicz 2023
All Rights Reserved

"K-Means Tie-Breaker" algorithm by Martin Gregory Sendrovicz (May 17,2023)

A given point may get stuck being equidistant between two centroids. This algorithm will break such case by labeling the given point based to its proximity to ALL the points of the closer cluster.

(1) Detect which point **p** has the same distance between two centroids $\mathbf{c_i}$ and $\mathbf{c_j}$

(2) avg_dist$_i$ ← Calculate the average distance between point **p** and ALL points labeled as 'i'.

(3) avg_dist$_j$ ← Calculate the average distance between point **p** and ALL points labeled as 'j'.

(4) assign label
if(avg_dist$_i$ < avg_dist$_j$)
p.label ← i
else
p.label ← j

(5) recalculate centroids

©Copyright by Martin Gregory Sendrovicz 2023
All Rights Reserved

Note that, if there is noise in a given data-image, such noise may or may not be considered as a cluster. If the noise is not considered, then K will be smaller. It is possible to capture such noise by evaluating the avg_dist$_i$ and avg_dist$_j$ against each cluster's variance. If the avg_dist is above a certain threshold (away from variance), it may be a good idea to either pinpoint the presence of noise or to create a new k+1 cluster.

"Intelligent Seed" Implementation (steps 4,5,6)

Collective Centroid $\mathbf{c_0} = (\ 13.5056,\ 17.3315\ )$; r_{max} (cut)
Farthest Point $\mathbf{p_0} = (\ 3,\ 3\ )$; z_{max} (cut)
$\mathbf{dist_{c_0,p_0}} = \sqrt{(13.5056-3)^2 + (17.3315-3)^2} = 17.76962362$
$S = ⅓$
$\beta = 0.5$ (Greek lowercase letter delta)

Step-4.1

- Perform the Coordinate System Translation from 'cross2-coordinate' Image Space into 'x,y-coordinate' Cartesian Space. (note that, without any loss of generality, we can also do this for multiple dimensions)

To adjust our image's 'cross2-coordinate' system to be congruent with the 'x,y-coordinate' system we need to flip the coordinates, i.e. (flipping and shifting).

$(\ \mathbf{row_c}, \mathbf{col_c}\ ) = (\ 13.5056,\ 17.3315\ )$ ⇒ $(\mathbf{x_c}, \mathbf{y_c}) = (\ 17.3315,\ 13.5056\ )$; $(\ \mathbf{x_c}, \mathbf{y_c}\ )$

Since we appointed $\mathbf{c_0}$ to be the center of the circle $(x-h)^2 + (y-k)^2 = r^2$, let's also adjust our notation accordingly:

$\mathbf{c_0} = (\ 17.3315,\ 13.5056\ ) \Rightarrow (\ \mathbf{x_0}, \mathbf{y_0}\ ) = (\ h,k\ )$
$\mathbf{z} = 17.3315,\ \mathbf{r_{max}} = 13.5056$

Now let's apply the same transform to $\mathbf{p_0}$:
$\mathbf{p_0} = (\ row_0\ col\ ) = (\ 3_{row},\ 3_{col}\ ) \Rightarrow (\ \mathbf{x_0}, \mathbf{y_0}\ ) = (\ x,y\ )$ *Invertibly flip them*

So from this point on we can speak of each point in terms of the standard x,y coordinate system.

Step-4.2

- Determine the Quadrant (there are 4 cases):

Quadrant I
if $x > h$ ⇒ so x is to the Right of h
if $y > k$ ⇒ so x is to the Up of k
⇒ **Right + Up = Quadrant I**

Quadrant II
so $x \le h$ = 13.5315 ⇒ so x is to the Left of h
$y > k$ = 13.5056 ⇒ so x is to the Up of k
⇒ **Left + Up = Quadrant II**

Quadrant III
if $x \le h$ ⇒ so x is to the Left of h
if $y \le k$ ⇒ so x is to the Down of k
⇒ **Left + Down = Quadrant III**

Quadrant IV
if $x > h$ ⇒ so x is to the Right of h
if $y \le k$ ⇒ so x is to the Down of k
⇒ **Right + Down = Quadrant IV**

Result: The $\mathbf{p_0}_{,case\ 1}$ is in **Quadrant II**

Step-4.3

- Adjust the coordinates of so that so, from coordinates (17.3315, 13.5056) translates to position $(x_0=0, y_0=0)$. Then, based on this translation, find the corresponding position for $\mathbf{p_0}$ which is currently at (3,3).

if **Quadrant I** based on $(x_0=0, y_0=0)$ we want: $[+x_0, +y_0]$
and above based on **Quadrant I**: $x > h$, $y > k$ so: $[+x_0, +y_0]$
$[+x_0, +y_0]$

if **Quadrant II** based on $(x_0=0, y_0=0)$ we want: $[-x_0, +y_0]$
and above based on **Quadrant II**: $x < h$, $y > k$ so: $[-x_0, +y_0]$
$[-x_0, +y_0]$

$[+x_0, +y_0] = (17.3315+3, 13.5056+3) = (14.3315\ ,\ 10.5056)$ ⇒ $[-x_0, +y_0] = (-14.3315\ ,\ 10.5056)$
so if $\mathbf{c_0}$ is origin (0,0), the new $\mathbf{p_0}$ location is $(-14.3315\ ,\ 10.5056)$

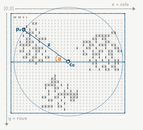
if **Quadrant III** based on $(x_0=0, y_0=0)$ we want: $[-x_0, -y_0]$
and above based on **Quadrant III**: $x < h$, $y < k$ so: $[-x_0, -y_0]$
$[-x_0, -y_0]$

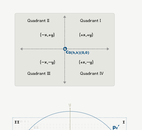
if **Quadrant IV** based on $(x_0=0, y_0=0)$ we want: $[+x_0, -y_0]$
and above based on **Quadrant IV**: $x > h$, $y < k$ so: $[+x_0, -y_0]$
$[+x_0, -y_0]$

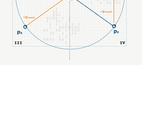
Step-4.3

- Find the angle $\boldsymbol{\theta}$: Using Unit Circle Symmetry, consider point $\mathbf{p_0}'$ located in Quadrant I. The coordinates of such $\mathbf{p_0}'$ must be mirrored across the x axis (since $\mathbf{p_0}$ is in Quadrant II), the $\mathbf{p_0}'$ was in Quadrant III, then the mirroring would be across the y axis; thus, note that, to find $\boldsymbol{\theta}$, we only need the x-coordinate.
  *Exception: $\mathbf{p_0}$ may happen to be in Quadrant I to begin with—in that case, just calculate angle $\boldsymbol{\theta}$.
  - if **Quadrant I** : just calculate angle $\boldsymbol{\theta}$ using $\mathbf{p_0}_{,case\ 1}$ (above below).
  - if **Quadrant II** : $\mathbf{p_0}_{,case} \Rightarrow \mathbf{p_0}'_{,Q1}$ since $\mathbf{p_0}$ is in Quadrant II, $\mathbf{p_0}'_{,Q1}$ $\mathbf{x} = \mathbf{p_0}_{,case}\ \mathbf{x} = 10.5056$
  - if **Quadrant III** : $\mathbf{p_0}' \Rightarrow \mathbf{p_0}'_{,Q1}$ $\mathbf{p_0}'_{,Q1}$
  - if **Quadrant IV** : $\mathbf{p_0}' \Rightarrow \mathbf{p_0}'_{,Q1}$ $\mathbf{p_0}'_{,Q1}$

now calculate angle $\boldsymbol{\theta}$:
sin $\boldsymbol{\theta} = \mathbf{y_{case}} / \mathbf{dist_{c_0,p_0}} \Rightarrow \boldsymbol{\theta} = \sin^{-1}(\mathbf{y_{case}} / \mathbf{dist_{c_0,p_0}}) = 36.24305963°$
$\boldsymbol{\theta} = 36.24°$







Step-4.4

- Find placement for c_1 centroid: To place c_1, we must scale the $\mathbf{dist_{c_0,p_0}}$ using parameter S—let S=⅓ ; So know that $\mathbf{p_0}$ is in Quadrant II so c_1 must also be in Quadrant II.

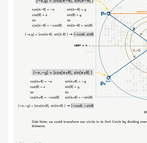
$\mathbf{r_{max}} = \mathbf{z_{max}} = \mathbf{p_0}'$
let radius = $\mathbf{r} = \mathbf{dist_{c_0,p_0}} \times S$, $\mathbf{dist_{c_0,p_0}} = 8.85481181$ ⇒ $\mathbf{r} = 8.85481181$

if **Quadrant I** then: $[+x_0, +y_0]$ [+cos θ, +sin θ]
$x_1 = r \times \cos\theta$; $x_1 = r \times \cos\theta$
$y_1 = r \sin\theta$; $y_1 = r \sin\theta$
$x_1 = [+\cos\theta]$
$y_1 = [+\sin\theta]$

if **Quadrant II** then: $[-x_0, +y_0]$ [−cos θ, +sin θ] $[+x_0, +y_0]$ [+cos θ, +sin θ]
$x_1 = r \times \cos\theta$; $x_1 = r \times \cos\theta$
$y_1 = r \sin\theta$; $y_1 = r \sin\theta$
$x_1 = -7.165748851$; $x_1 = [-\cos\theta]$
$y_1 = 5.252806982$; $y_1 = [+\sin\theta]$
$\boldsymbol{\theta} = 72.\ 5.5$; ⇒ remember we are using so all [0,0] as origin

if **Quadrant III** then: $[-x_0, -y_0]$ [−cos θ, −sin θ] $[-x_0, +y_0]$ [−cos θ, +sin θ]
$x_1 = r \times \cos\theta$; $x_1 = r \times \cos\theta$
$y_1 = r \sin\theta$; $y_1 = r \sin\theta$
$x_1 = [-\cos\theta]$
$y_1 = [-\sin\theta]$

if **Quadrant IV** then: $[+x_0, -y_0]$ [+cos θ, −sin θ] $[+x_0, -y_0]$ [+cos θ, −sin θ]
$x_1 = r \times \cos\theta$; $x_1 = r \times \cos\theta$
$y_1 = r \sin\theta$; $y_1 = r \sin\theta$
$x_1 = [+\cos\theta]$
$y_1 = [-\sin\theta]$



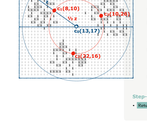
Step-4.5

- Find the central angle $\boldsymbol{\beta}$: by dividing the circumference into 4 equal parts (2π by 4) or [360° by 4]—based on which we'll place the other 3 c_i centroids.

$360°/4 = 360°/3 = 120°$ ⇒ $\boldsymbol{\beta}$ = central angle of rotation

Step-4.6

- Find the c_i's angle θ :
  - if c_1 is in **Quadrant I** : $\boldsymbol{\theta}_1$ angle = θ
  - if c_1 is in **Quadrant II** : $\boldsymbol{\theta}_1$ angle = 180°− θ OR 1− θ
    $\boldsymbol{\theta}_1$ = 180°− θ = 180°−36.24° = **143.7569404°**
  - if c_1 is in **Quadrant III** : $\boldsymbol{\theta}_1$ angle = 180°+ θ
  - if c_1 is in **Quadrant IV** : $\boldsymbol{\theta}_1$ angle = 360°− θ



Step-6

- Return the centroids: c_0(row, col), c_1(row, col), c_2(row, col), c_3(row, col).

Step-5.1

- Find the remaining k=1 centroids:

Starting from centroid c_1's quadrant, determine the quadrant in which to place centroids c_i, where $2 \le i \le$ k) (i.e. All the remaining centroids)

- For i=2 → k
- Find the c_i's angle θ:
  $\boldsymbol{\theta}_i$ angle = $\boldsymbol{\theta}_{i-1}$ + $\boldsymbol{\beta}$
  if $\boldsymbol{\theta}_i > 360°$ ⇒ $\boldsymbol{\theta}_i = \boldsymbol{\theta}_i - 360°$, so basically $\boldsymbol{\theta}_{i+1} = [\boldsymbol{\theta}_i + \boldsymbol{\beta}]$ mod 360
  - Determine the Quadrant c_i case for c_i using $\boldsymbol{\theta}_i$ angle :
    - if $0° < \boldsymbol{\theta}_i < 90°$ ⇒ $c_{i,case}$ = **Quadrant I**
    - if $90° \le \boldsymbol{\theta}_i < 180°$ ⇒ $c_{i,case}$ = **Quadrant II**
    - if $180° \le \boldsymbol{\theta}_i < 270°$ ⇒ $c_{i,case}$ = **Quadrant III**
    - if $270° \le \boldsymbol{\theta}_i < 360°$ ⇒ $c_{i,case}$ = **Quadrant IV**
  - Calculate the c_i's (i.e. the angle required to calculate the c_i's x,y-coordinates) :
    - if $c_{i,case}$ = **Quadrant I** : $\boldsymbol{\theta}_i^{\,I} = \boldsymbol{\theta}_i$
    - if $c_{i,case}$ = **Quadrant II** : $\boldsymbol{\theta}_i^{\,II} = 180° - \boldsymbol{\theta}_i$
    - if $c_{i,case}$ = **Quadrant III** : $\boldsymbol{\theta}_i^{\,III} = \boldsymbol{\theta}_i - 180°$
    - if $c_{i,case}$ = **Quadrant IV** : $\boldsymbol{\theta}_i^{\,IV} = 360° - \boldsymbol{\theta}_i$
  - Calculate the c_i's x,y-coordinates
    - Quadrant I : $x_{c_i} = r \cos(\boldsymbol{\theta}_i^{\,I})$; $y_{c_i} = r \sin(\boldsymbol{\theta}_i^{\,I})$
    - Quadrant II : $x_{c_i} = r \cos(\boldsymbol{\theta}_i^{\,II})$; $y_{c_i} = r \sin(\boldsymbol{\theta}_i^{\,II})$
    - Quadrant III : $x_{c_i} = r \cos(\boldsymbol{\theta}_i^{\,III})$; $y_{c_i} = r \sin(\boldsymbol{\theta}_i^{\,III})$
    - Quadrant IV : $x_{c_i} = r \cos(\boldsymbol{\theta}_i^{\,IV})$; $y_{c_i} = r \sin(\boldsymbol{\theta}_i^{\,IV})$
    - i++ , (i.e. calculate the next centroid)

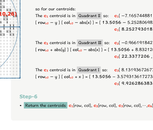
Step-5.2

- Adjust the coordinates of c_i, c_0, c_1, c_2, so that the origin is back in the Top-Left corner (i.e. perform transform to the original image's coordinates system)
  Note that: y-coordinates correspond to row
  x-coordinates correspond to col.

The origin returns to (row=0, col=0) and c_0 gets back its original row,col-coordinates:
$\mathbf{c_0}$ (7.3315, 13.5056) = (h,k) ⇒ (row$_{c_0}$, col$_{c_0}$) = (13.5056, 7.3315)

Now based on ($\mathbf{row_c}$, $\mathbf{col_c}$) we adjust each centroid's row,col-coordinates according to the table given below:

Quadrant I	$\mathbf{row_c} - \mathbf{y}$	$\mathbf{col_c} + \mathbf{x}$	
Quadrant II	$\mathbf{row_c} - \mathbf{y}$	$\mathbf{col_c} - \mathrm{abs}(\mathbf{x})$	
Quadrant III	$\mathbf{row_c} + \mathrm{abs}(\mathbf{y})$	$\mathbf{col_c} - \mathrm{abs}(\mathbf{x})$	
Quadrant IV	$\mathbf{row_c} + \mathrm{abs}(\mathbf{y})$	$\mathbf{col_c} + \mathbf{x}$	



so:
if centroid c_1's $\mathbf{y}$ is in **Quadrant I** : $\mathbf{row_c}$ (row, col) = ($\mathbf{row_c} - \mathbf{y}$] , $\mathbf{col_c} + \mathbf{x}$]
if centroid c_1's $\mathbf{y}$ is in **Quadrant II** : $\mathbf{row_c}$ (row, col) = ($\mathbf{row_c} - \mathbf{y}$] , $\mathbf{col_c} - \mathrm{abs}(\mathbf{x})$]
if centroid c_1's $\mathbf{y}$ is in **Quadrant III** : $\mathbf{row_c}$ (row, col) = ($\mathbf{row_c} + \mathrm{abs}(\mathbf{y})$] , $\mathbf{col_c} - \mathrm{abs}(\mathbf{x})$]
if centroid c_1's $\mathbf{y}$ is in **Quadrant IV** : $\mathbf{row_c}$ (row, col) = ($\mathbf{row_c} + \mathrm{abs}(\mathbf{y})$] , $\mathbf{col_c} + \mathbf{x}$]

so for our centroids.
The c_1 centroid is in **Quadrant II** so: row$_{c_1}$ = −7.165748851 , 5.252806982]
[$\mathbf{row_c} - \mathbf{y}$] [$\mathbf{col_c} - \mathrm{abs}(\mathbf{x})$] = [13.5056 − 5.252806982] [7.3315 − 7.165748851] =
c_1(22.3337286 , 14.3663081%) ; c_1(7.5748851)

The c_2 centroid is in **Quadrant III** so: c_2(−0.964119827 , −8.832120596]
[$\mathbf{row_c} + \mathrm{abs}(\mathbf{y})$] [$\mathbf{col_c} - \mathrm{abs}(\mathbf{x})$] = [13.5056 + 8.832120596] [7.7.3315 − 0.964119827] =
c_2(22.3337286 , 16.3663081%) ; c_2(6.367119827)

The c_3 centroid is in **Quadrant I** so: c_3(8.131936726777% , 3.579313617273%)
[$\mathbf{row_c} - \mathbf{y}$] [$\mathbf{col_c} + \mathbf{x}$] = [13.5056 − 3.579313617273%] [7.7.3315 + 8.131936726777%] =
c_3(9.926286383 , 25.46434673) ; c_3(15.81236726777)

Step-6

- Return the centroids: c_0(row, col), c_1(row, col), c_2(row, col), c_3(row, col).

Resources
+ Unit Circle
https://www.khanacademy.org/math/trigonometry/unit-circle-trig-func/unit-circle-definition-of-trig-functions/v/unit-circle-definition-of-trig-functions-1

C2

Starting from centroid c_1's quadrant, determine the quadrant in which to place centroid c_2.

- Find the c_2's angle θ:
  $\boldsymbol{\theta}_2$ angle = $\boldsymbol{\theta}_1$ + $\boldsymbol{\beta}$ = 143.7569404° + 120° = 263.7569404°
  if $\boldsymbol{\theta}_2 > 360°$ ⇒ $\boldsymbol{\theta}_2 = \boldsymbol{\theta}_2 - 360°$ ⇒ $\boldsymbol{\theta}_2$ = 263.7569404°
  so basically $\boldsymbol{\theta}_{i+1} = [\boldsymbol{\theta}_i + \boldsymbol{\beta}]$ mod 360
  - Determine the Quadrant for c_2' angle :
    - if $0° < \boldsymbol{\theta}_2 < 90°$ ⇒ Quadrant I
    - if $90° \le \boldsymbol{\theta}_2 < 180°$ ⇒ Quadrant II
    - if $180° \le \boldsymbol{\theta}_2 < 270°$ ⇒ Quadrant III
    - if $270° \le \boldsymbol{\theta}_2 < 360°$ ⇒ Quadrant IV
  - Calculate the c_2' (i.e. the angle required to calculate the c_2's x,y-coordinates) :
    - if $\boldsymbol{\theta}_2$ = Quadrant I : $\boldsymbol{\theta}_2^{\,I} = \boldsymbol{\theta}_2$
    - if $\boldsymbol{\theta}_2$ = Quadrant II : $\boldsymbol{\theta}_2^{\,II} = 180° - \boldsymbol{\theta}_2$
    - if $\boldsymbol{\theta}_2$ = Quadrant III : $\boldsymbol{\theta}_2^{\,III} = \boldsymbol{\theta}_2 - 180°$
    - if $\boldsymbol{\theta}_2$ = Quadrant IV : $\boldsymbol{\theta}_2^{\,IV} = 360° - \boldsymbol{\theta}_2$
    - = 263.7569404° − 180° = 83.7569404°
  - Calculate the c_2's x,y-coordinates
    - Quadrant I : $x_{c_2} = r \cos(\boldsymbol{\theta}_2^{\,I})$; $y_{c_2} = r \sin(\boldsymbol{\theta}_2^{\,I})$
    - Quadrant II : $x_{c_2} = r \cos(\boldsymbol{\theta}_2^{\,II})$; $y_{c_2} = r \sin(\boldsymbol{\theta}_2^{\,II})$
    - Quadrant III : $x_{c_2} = r \cos(\boldsymbol{\theta}_2^{\,III})$; $y_{c_2} = r \sin(\boldsymbol{\theta}_2^{\,III})$
    - Quadrant IV : $x_{c_2} = r \cos(\boldsymbol{\theta}_2^{\,IV})$; $y_{c_2} = r \sin(\boldsymbol{\theta}_2^{\,IV})$
    - x_{c_2} = −9.664119827 ; y_{c_2} = 8.832120596
    - $\boldsymbol{\theta}_2$ = 0.97 ; −8.83 ; ⇒ remember we are using so all [0,0] as origin
  - perform the above steps for all the remaining k-2 centroids, where: c_i's (instead of c_1's) quadrant is now our starting point. Ergo, in general, use centroid c_i's quadrant to find location for centroid c_{i+1}.

C3

Starting from centroid c_2's quadrant, determine the quadrant in which to place centroid c_3.

- Find the c_3's angle θ:
  $\boldsymbol{\theta}_3$ angle = $\boldsymbol{\theta}_2$ + $\boldsymbol{\beta}$ = 263.7569404° + 120° = 383.7569404°
  if $\boldsymbol{\theta}_3 > 360°$ ⇒ $\boldsymbol{\theta}_3 = \boldsymbol{\theta}_3 - 360°$ ⇒ $\boldsymbol{\theta}_3$ = 23.7569404°
  so basically $\boldsymbol{\theta}_{i+1} = [\boldsymbol{\theta}_i + \boldsymbol{\beta}]$ mod 360
  - Determine the Quadrant for c_3' angle :
    - if $0° < \boldsymbol{\theta}_3 < 90°$ ⇒ Quadrant I
    - if $90° \le \boldsymbol{\theta}_3 < 180°$ ⇒ Quadrant II
    - if $180° \le \boldsymbol{\theta}_3 < 270°$ ⇒ Quadrant III
    - if $270° \le \boldsymbol{\theta}_3 < 360°$ ⇒ Quadrant IV
  - Calculate the c_3' (i.e. the angle required to calculate the c_3's x,y-coordinates) :
    - if $\boldsymbol{\theta}_3$ = Quadrant I : $\boldsymbol{\theta}_3^{\,I} = \boldsymbol{\theta}_3$
    - if $\boldsymbol{\theta}_3$ = Quadrant II : $\boldsymbol{\theta}_3^{\,II} = 180° - \boldsymbol{\theta}_3$