AEGIS: Image and Signal Processing

Image Segmentation using K-Means Algorithm Unit Lesson

Chinyen Chuo
NSF RET Fellow
Seminole High School, Seminole County Public Schools
chinyen_chuo@scps.k12.fl.us

Dr. Michael Georgiopoulos
Faculty Mentor
University of Central Florida
michaelg@ucf.edu

Tiantian Zhang
Graduate Student Mentor
University of Central Florida
watera427@gmail.com

July 20, 2012

Table of Contents

Executive Summary **4**

K-Means Lesson Plan **5**

Teaching Notes **9**
Introduction to Image Segmentation 9
Colors and Numbers 9
Key Concepts of K-Means 11
Procedure of K-Means 11
Example 12
Optimization of K-Means 15
Weaknesses of K-Means 15
Exercises 16
Further Reading 16

Result Analysis - Teacher Notes **18**

Project #1 - Result Analysis **23**

Project #2 - Keeping Track of Student Performance **25**

Appendices **27**
A Project2: Sample Grades 27
B Sample Bellwork 28
C Worksheet #1: Connecting Geometry to K-Means 31
D MATLAB source file: `randpts.m` 34
E MATLAB source file: `scatterpts_group.m` 35
F MATLAB source file: `kMeansCluster_CC2.m` 36
G MATLAB source file: `kMeansCluster.m` 38
H MATLAB source file: `K_means.m` 40
I MATLAB source file: `DistMatrix2D.m` 42
J MATLAB source file: `DistMatrix.m` 43
K MATLAB source file: `Scattergrades2.m` 44
L ANSWERS 45

Acknowledgement **53**

References **53**

Executive Summary

This unit plan was written as an exercise in exploratory learning, and as an introduction to concepts central to image and signal processing, in particular, image segmentation. This lesson can be used as an application of matrices, and of the Euclidean distance or vector magnitude, and as an introduction to the concept of optimization for the Algebra II students. For higher level students in Pre-calculus and Calculus, this is an excellent topic for discussion of optimization.

This unit lesson consists of two parts: the procedural and the conceptual of the K-Means algorithm. Materials included are: Teaching Notes with exercise problems, suggested bellwork problems, a worksheet, two projects, and relevant MATLAP and TI-84 programs.

The worksheet can be used to introduce or review the concept of centroid. Project #1 focuses heavily on the concept of the algorithm, especially on the spirit of optimization. Teaching notes and some sample experimental results are also included for teacher reference. Project #2 is an application of the optimization process to real world situation involving the student performance as measured by exam grades and possibly other factors.

K-Means Algorithm
Lesson Plan
C. Chuo

Content Area: Precalculus

Grade Level: 9 - 12

Title: Application of Vectors

Next Generation Sunshine State Standards:

- MA.912.D.9.3 - Use vectors to model and solve application problems
- MA.912.C.3.4 - Find local and absolute maximum and minimum points
- MA.912.A.7.9 - Solve optimization problems
- MA.912.D.8.1 - Use matrices to organize and store data. Perform matrix operations (addition, subtraction, scalar multiplication, multiplication).
- MA.912.D.8.2 - Use matrix operations to solve problems.
- MA.912.D.8.4 - Find the inverse of a matrix and use the inverse to solve problems with and without the use of technology.

Common Core State Standard Mathematics:

- MACC.912.F-IF.2.4 - Interpret functions that arise in applications in terms of the context such as local and global minimum and maximum.
- MACC.912.N-VM.2.5 - Operations on vectors, compute magnitude of a vector
- MACC.912.N-VM.2.6 - Use matrices to represent and manipulate data
- MACC.912.N-VM.2.7 - Multiply matrices by scalars to produce new matrices
- MACC.912.N-VM.3.8 - Add, subtract, and multiply matrices of appropriate dimensions
- MACC.912.N-VM.3.9 - Understand that, unlike multiplication of numbers, matrix multiplication for square matrices is not a commutative operation, but still satisfies the associative and distributive properties
- MACC.912.N-VM.3.10 - Understand that the zero and identity matrices play a role in matrix addition and multiplication similar to the role of 0 and 1 in the real numbers. The determinant of a square matrix is nonzero if and only if the matrix has a multiplicative inverse

- MACC.912.N-VM.3.11 - Multiply a vector (regarded as a matrix with one column) by a matrix of suitable dimensions to produce another vector. Work with matrices as transformations of vectors
- MACC.912.N-VM.3.12 - Work with 2 2 matrices as transformations of the plane, and interpret the absolute value of the determinant in terms of area.

Unit: Vectors (IB Mathematics SL Syllabus: Unit 4.2)

Learning Goal: Student will understand the application aspect of vectors and matrices in image processing, in particular, image segmentation using the K-means clustering algorithm.

Objectives:

- Student will be able to use the K-Means algorithm to organize data values into clusters analytically and using technology such as Excel Spreadsheet and MATLAB.
- Student will be able to implement the K-Means algorithm using MATLAB. Student will use the implemented K-Means algorithm to cluster (segment) images.
- Student will be able to compare and identify strengths and weaknesses of Excel and MATLAB programs.

Materials:

- Computer or laptop with wired or wireless internet connection
- Computer software MS Excel, MATLAB, TI calculators,
- Worksheet

Prior Knowledge:

- Magnitude of a vector from given points in 2D and 3D, mid-point formula (average values of coordinate points),
- Operations of vectors
- Sequences
- Sum and sigma notation
- Matrices
- Basic knowledge of Excel and TI calculator programming

- Basic MatLab knowledge such as entering points as an $m \times n$ matrix, input images, and running script and function files

Procedures: (Time needed - Four 49-minute periods)

Day 1:

- Introduction to Image Segmentation using Power Point Presentation, paying special attention to the mathematics.
- **Key Words:** mean, vector components, magnitude (norm), cluster, segmentation, inter-distance, intra-distance, optimization, (limit, derivative - optional)
- **Homework Assignment:** Have student read the Teaching Notes (Optional), and bring one of their favorite photos to class next day.

Day 2:

- **Bell-work:** This should include review problems for finding midpoint between points, magnitudes of vectors, and evaluating sum using sigma notation. For sample questions, see Teaching Notes.
- Tracing MATLAB programs for image segmentation, in particular clustering programs.
- **Motivation:** Let students work in groups of two or three and go on the website: `http://befunky.com`. Have students use the online program to edit their photos and then ask students the following questions:
  1. What features they like the most, the least?
  2. If they have the option (power) to change the features that they don't like, how would they change them?
  3. Do they know how the program works?
  4. Make a conjecture as to what mathematics they think is behind the photo editing program. Explain their reasoning.
  5. Have students work in groups and trace the included image segmentation programs, especially the cluster program.

Day 3:

- **Group Activity:**
  1. Have students work in groups of two or three to partition analytically a small set of data points in 2D into 2 clusters.

2. Use either a TI calculator or Excel spreadsheet to check their answers and explore different clusters using different centroids.
3. Use MatLab and a given K-Means algorithm code to cluster an image into two or more clusters.
4. If possible, have students modify the code to see the change in results.

- **Lecture:** Introduce basic concept of the K-Means Algorithm (procedure). Then, move on to emphasize the following key ideas/concepts of K-Means:
  1. Vectors
  2. Definition of distance between two vectors **u** and **v**
  3. K-Means minimizes an objective function
  4. Minimizing an objective function by adjustment of centroids and reassignment of points
  5. The average of the points in a cluster minimizes the sum of the squared-distances of the points to this average point (the center).
  6. The objective function $d = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \parallel x_i - c_{n_j} \parallel^2$ defines a sequence, and the limit of this sequence converges.
- **Visualization** Segment images (photos) using MatLab programs.

Day 4:

- Continue lecture or computer lab work, whichever is needed.
- Further study and investigation on the proof of optimization (Optional, if time permits).
- Further discussion on issues, questions, and discoveries on the project
- **Evaluation:** Bell-work, worksheets, and project.

Day 5:

- Student presentation using posters or Power Point.

• Introduction: The Basic Idea of Image Segmentation

The K-Means clustering is one of the simplest learning algorithms in solving clustering problems. "Clustering" is the process of organizing a set of objects into classes of similar objects. By doing so, we can partition an image into separate but edge-connected regions, called image segmentation, and as a result, the image can be simplified for easy analysis and faster computer processing. This is especially useful in the medical field where the area of interest can be accurately separated and quickly identified, for instance, cancer cell identification.

Following are two images that illustrate this idea. Figure 1 is the original image that consists of two main colors: red and green, but notice the different shades of green.

Figure 1: Original Image

Figure 2: 2-Clustered Image

If our focus is to separate the green color from the red color, then we may cluster the image into two groups. Here we are using the Euclidean distance-based K-Means algorithm to classify data values into K clusters (groups) with initial randomly guessed (chosen) centroids in each cluster. Since each point in an m-dimensional plane/space can be represented by vectors, we use magnitude, or norm to calculate distances between the data points and the centroids.

• Image Segmentation - Colors and Numbers

Images can be thought of as a collection of 3-dimensional vectors that represent the color at a particular point. More precisely, at a given point, the color is given by an RGB vector $[r, g, b]$, where r, g, and b are the intensity of red, green, and blue respectively. *Distance* in this case measures the difference or closeness of the colors. For example, red is closer to pink than to violet. That is, the vector $[255, 0, 0]$ represents red which is closer to the vector $[255, 0, 255]$ that is pink, than it is to the vector $[238, 130, 238]$, which is violet.

If d_1 is the distance from red to pink and d_2 is the distance from red to violet, then

$$\begin{aligned} d_1 &= \sqrt{(255-255)^2 + (0-0)^2 + (0-255)^2} \\ &= 255 \end{aligned}$$

and

$$\begin{aligned} d_2 &= \sqrt{(255-238)^2 + (0-130)^2 + (0-238)^2} \\ &= 271.72 \end{aligned}$$

Since $d_1 < d_2$, pink and red would be assigned to the same cluster.

Let's take a closer look at the peppers image.

Figure 3:

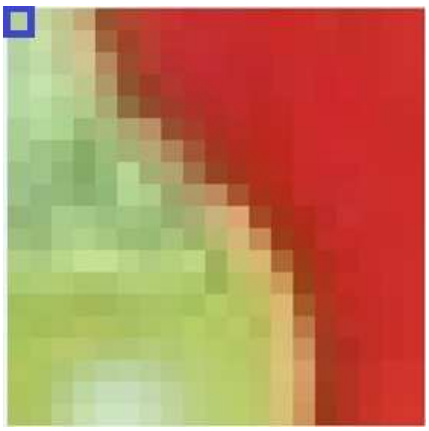


Figure 4: Part of Original Image - 19^2 Pixels

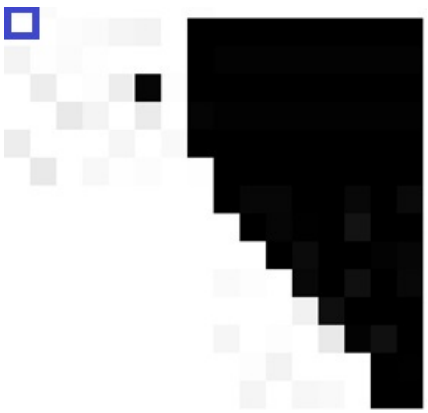


Figure 5: 2-Clusters - 19^2 Pixels

The boxed pixel in Figure 4 has vector values of $[20, 202, 0]$, which is closer to the green values $[15, 149, 0]$ than to the red color $[253, 26, 0]$, and therefore, this pixel would be clustered in the color white group with other shades of green as indicated in Figure 5.

- **The Mathematical Key Concepts of the K-Means Algorithm**

  K-Means algorithm is one of the simplest clustering algorithms. However it is quite often used as a stepping-stone to design more sophisticated clustering algorithms. The main idea of the K-Means algorithm revolves around the following two steps:

  – Computing distances between points and the centroid of a cluster

  – Finding new centroids that minimize an objective function.

  The second step essentially defines the algorithm since it minimizes the objective function,

$$d = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \parallel x_i - c_{n_j} \parallel^2 .$$

  We will elaborate on this key idea later.

- **Procedure of the K-Means Algorithm**

  1. Decide the number of clusters desired.

  2. Choose K-number of centroids

  3. Iteration #1: Let p_i be a member in the sample problem and c_j be the centroid in each cluster.

  A) Calculate the distances between each point p_i and the centroids c_j.

  B) For each p_i, find the shortest of the distances to the points c_j.

  C) Classify the point, that is, assign p_i to the group that contains the centroid to which p_i is closest.

  4. Iteration #2: Within each cluster, find a new centroid by taking the average of the corresponding components from all points in that cluster. The formula for calculating the coordinates of the new centroid is:

$$c_j = \left(\frac{x_{n_1} + x_{n_2} + \cdots + x_{n_j}}{n_j}, \frac{y_{n_1} + y_{n_2} + \cdots + y_{n_j}}{n_j}, \frac{z_{n_1} + z_{n_2} + \cdots + z_{n_j}}{n_j} \right),$$

  assuming we are working in the 3-dimensional space with n_j points in cluster j.

  In general, if there are n points in cluster j of dimension m, then we may express this set of data in a matrix:

$$\begin{array}{c} \text{Data/Dimension} \\ \text{Point } P_1 \\ \text{Point } P_2 \\ \vdots \\ \text{Point } P_n \end{array} \begin{array}{c} \begin{array}{cccc} D_1 & D_2 & \ldots & D_m \end{array} \\ \begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,m} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \ldots & a_{n,m} \end{bmatrix} \end{array}$$

and the general formula for finding the coordinate-averages of all points is:

$$\text{Average} = \left(\frac{a_{1,1} + a_{2,1} + \cdots + a_{n,1}}{n}, \frac{a_{1,2} + a_{2,2} + \cdots + a_{n,2}}{n}, \ldots, \frac{a_{1,m} + a_{2,m} + \cdots + a_{n,m}}{n} \right)$$

5. If no points changed groups or the centroids remained the same, the algorithm is done. Otherwise, go to step 3 until no points change groups.

- **Example:**

1. Given below are the coordinates of six points:

P_i	P_1	P_2	P_3	P_4	P_5	P_6
(x, y)	$(1, 0)$	$(3, 2)$	$(5, 1)$	$(6, 5)$	$(7, 3)$	$(8, 6)$

Suppose we want to divide these points into 2 groups. Following the K-Means procedure, we have

A) **Iteration #1:**

a) Let us choose P_1 and P_3 as initial centroids, denoted c_1, c_2, and let $d_{c_j P_i}$ be the distances between each centroid c_j and point P_i, were $i = 1, 2, \ldots 6$ and $j = 1, 2$. Then, we obtain the distances as follows:

$$\begin{aligned}
d_{c_1P_1} &= \sqrt{(1-1)^2 + (0-0)^2} \\
&= 0 \\
d_{c_1P_2} &= \sqrt{(3-1)^2 + (2-0)^2} \\
&= \sqrt{8} \\
&\approx 2.83 \\
d_{c_1P_3} &= \sqrt{(5-1)^2 + (1-0)^2} \\
&= \sqrt{17} \\
&\approx 4.12 \\
d_{c_1P_4} &\approx 7.07 \\
d_{c_1P_5} &\approx 6.71 \\
d_{c_1P_6} &\approx 9.22
\end{aligned}$$

$$\begin{aligned}
d_{c_2P_1} &= \sqrt{(1-5)^2 + (0-1)^2} \\
&= \sqrt{17} \\
&\approx 4.12 \\
d_{c_2P_2} &= \sqrt{(3-5)^2 + (2-1)^2} \\
&= \sqrt{5} \\
&\approx 2.24 \\
d_{c_2P_3} &= \sqrt{(5-5)^2 + (1-1)^2} \\
&= 0 \\
d_{c_2P_4} &\approx 4.12 \\
d_{c_2P_5} &\approx 2.83 \\
d_{c_2P_6} &\approx 5.83
\end{aligned}$$

b) Compare each corresponding pairs: $d_{c_1P_i}$ and $d_{c_2P_i}$, for $1 \leq i \leq 6$, and assign the point P_i to appropriate group. That is, compare:

$d_{c_1P_1}$ and $d_{c_2P_1}$
$d_{c_1P_2}$ and $d_{c_2P_2}$

and so on ....

Organize the data and classify, we have the initial groups:

P_i	$d_{c_1 P_i}$	$d_{c_2 P_i}$	Cluster j
1	0	4.12	1
2	2.83	2.24	2
3	4.12	0	2
4	7.07	4.12	2
5	6.71	2.83	2
6	9.22	5.83	2

Figure 6: Data Clusters

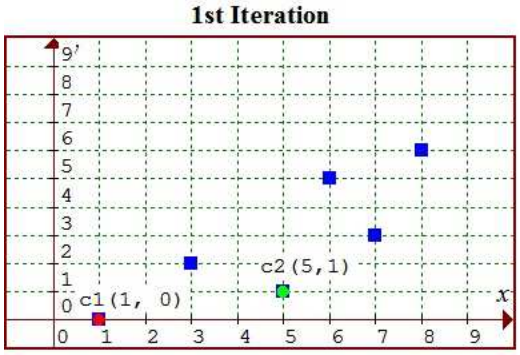


Figure 7: Data Plot

B) **Iteration #2:**

a) Within each cluster, find a new centroid:
$c_1 = (1, 0)$ since there is only one point in cluster 1.

$$\begin{aligned} c_2 &= \left(\frac{x_{P_2} + x_{P_3} + x_{P_4} + x_{P_5} + x_{P_6}}{n_2}, \frac{y_{P_2} + y_{P_3} + y_{P_4} + y_{P_5} + y_{P_6}}{n_2} \right) \\ &= \left(\frac{3+5+6+7+8}{5}, \frac{2+1+5+3+6}{5} \right) \\ &= \left(\frac{29}{5}, \frac{17}{5} \right) \\ &= (5.8, 3.4) \end{aligned}$$

b) Calculate the distances between all points to the new centroids and classify, we have

P_i	$d_{c_1 P_i}$	$d_{c_2 P_i}$	Cluster j	Change ?
1	0	5.88	1	No
2	2.83	3.13	1	Yes
3	4.12	2.53	2	No
4	7.07	1.61	2	No
5	6.71	1.26	2	No
6	9.22	3.41	2	No

Figure 8: Data Clusters

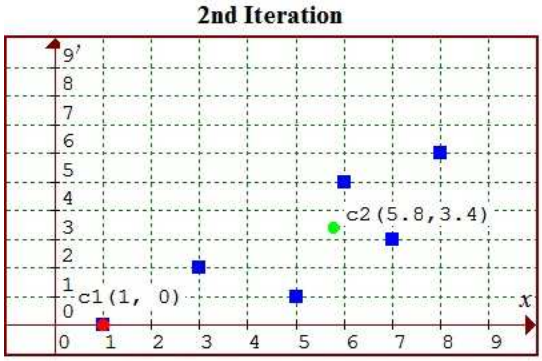


Figure 9: Data Plot

We can see that P_2 switched from cluster 1 to cluster 2, so we repeat the process again.

C) **Iteration #3:**
New centroids:

$$c_1 = \left(\frac{1+3}{2}, \frac{0+2}{2}\right) = (2, 1),$$
$$c_2 = \left(\frac{5+6+7+8}{4}, \frac{1+5+3+6}{4}\right) = (6.5, 3.75)$$

Find new distances and classify:

P_i	$d_{c_1 P_i}$	$d_{c_2 P_i}$	Cluster j	Change ?
1	1.41	6.66	1	No
2	1.41	3.91	1	No
3	3.00	3.13	1	Yes
4	5.66	1.35	2	No
5	5.39	0.90	2	No
6	7.81	2.70	2	No

Figure 10: Data Clusters

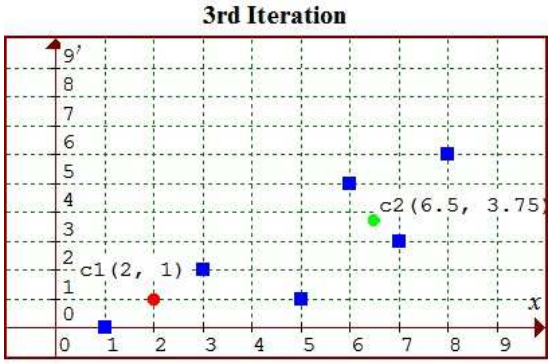


Figure 11: Data Plot

Once again, we see that P_3 switched from cluster 2 to 1; hence, we repeat the process again.

D) **Iteration #4:**
New centroids: $c_1 = (3, 1)$ and $c_2 = (7, 4.67)$.

P_i	$d_{c_1 P_i}$	$d_{c_2 P_i}$	Cluster j	Change ?
1	2.24	7.60	1	No
2	1.00	4.80	1	No
3	2.00	4.18	1	No
4	5.00	1.05	2	No
5	4.47	1.67	2	No
6	7.07	1.67	2	No

Figure 12: Data Clusters

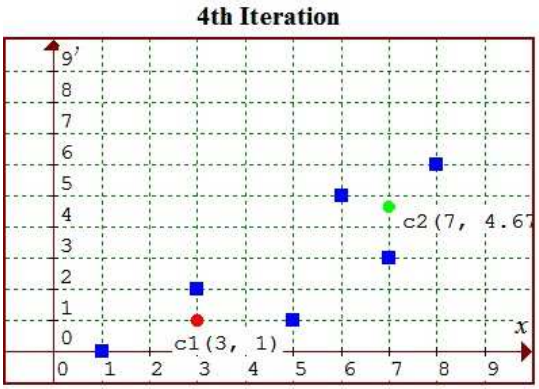


Figure 13: Data Plot

Note that there is no change in the groups, hence the algorithm is completed and has assigned P_1, P_2, P_3 to one group and P_4, P_5, P_6 to another group.

Also notice the movement of the centroids: At each iteration, each new centroid gets closer and closer to the points in its group, and this is achieved by taking averages of the coordinates of the points in each group.

Remark: Until now, our focus has been on the "grouping" aspect of the algorithm, but during the process the algorithm produces centroids that minimize the distances within each cluster, which is the single most important idea. While grouping is an image processing point of view, the primary mathematical idea is producing new centroids that minimize the the sum of the squared-distances within each cluster, and consequently minimizes the objective function

$$d = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \parallel x_{(n_j)i} - c_j \parallel^2$$

The inner sum is the sum of the squared-distances within each cluster; the outer sum is the total squared-distances from all clusters. Ideally, we would like d to be as small as possible. When this is achieved, the algorithm is done.

Let's look at the numerical values and behavior of the above example:

	Cluster 1	Cluster 2	
Iteration # $\downarrow$	$\sum_{i=1}^{n_1}(x_i - c_1)^2$	$\sum_{i=1}^{n_2}(x_i - c_2)^2$	$\sum_{j=1}^{K}\sum_{i=1}^{n_j}(x_{(n_j)i} - c_j)^2$
I_1	0	64	64
I_2	8	22.2	30.2
I_3	13	9.9	22.9
I_4	10	6.7	16.7

Figure 14: Optimization

The clusters stabilized after the 4^{th} iteration; we say that the objective function converges to 16.7 - the smallest value of d.

• Weaknesses of K-Means Algorithm

1. Results are highly sensitive to initialization of centroids.
2. The number of clusters, K, must be determined beforehand.
3. Results may get trapped in a local optimum.
4. We never know which attribute contributes more to the grouping process since we assume that each attribute has the same weight.
5. The algorithm is sensitive to outliers. That is, data that are far from the centroid may pull the centroid away from the real one.

[1]

- **Experiment and Sample Results of K-Means**

Figure 15: Original Image

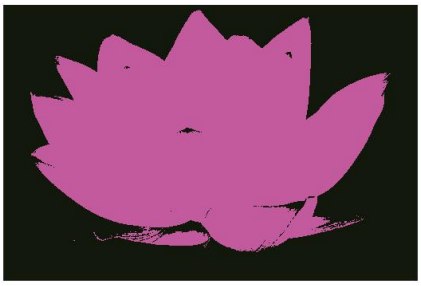


Figure 16: 2-Clusters

Figure 17: 4-Clursters

Figure 18: 8-Clusters

- **Exercises:**

1. Using a graphing program - MATLAB, Excel, Graphmatica, Graph, Geogebra, etc. - plot the points $(0, 1)$, $(2, 5)$, $(3, 5)$, $(5, 4)$, and $(7, 6)$.
2. Using the K-Means algorithm and the first two points as initial centroids,
   A) find the distance matrix with x-values in column 1, y-values in column 2, distances to point 1 in column 3, and distances to point 2 in column 4.
   B) find the 2 clusters for the first two iterations.

   Show your numerical calculations.
3. Given six points $P_1(1, 3, 0)$, $P_2(2, 0, 5)$, $P_3(3, 1, 2)$, $P_4(4, 0, 2)$, $P_5(1, 0, 5)$, $P_6(2, 4, 1)$, find 2 clusters of the first 2 iterations. (Hint: What must you do to start the procedure?)

- **Applications for Further Study**

Here are some suggested readings for interesting applications that implement the K-Means clustering algorithms for "tracking objects" as well as discussion on the weaknesses of K-Means algorithm.

1. K-Means Tracker: Ageneral Algorithm for Tracking People by Chunsheng Hua, et al.

2. Detecting Unusual Activity in Video by Mirko Visontai, et al.

3. On-chip pulse based parallel neural circuits for object-tracking system by H. Zhuang, et al.

- ## Weaknesses of K-Means

  The weaknesses of K-Means were briefly presented in the previous notes. Results of K-Means on the same data set may vary and are highly dependent on the choice of initial centroids, and because the number of clusters K is fixed, the results may not be ideal if there are any outliers. Furthermore, the use of gradient-descent concept may cause the objective function to reach a local minimum but may not be able to attain a global minimum. For more information on this, please see the technical report.

  This investigative activity/project will focus on choices of initial centroids and size of the data set, and use the results to gain a better understanding of the objective function:

$$d = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \parallel x_i - c_{n_j} \parallel^2 .$$

- ## Directions on Implementation of MATLAB Programs

  Below are MATLAB programs that show results of each iteration including optimization values and new centroids for each iteration. These programs are designed for teaching purposes and are different than the student programs indicated in project #1. Teachers may choose to give these programs to students if time is an issue.

  - `randpts.m`
  - `scatterpts_groupRandC2.m`
  - `kMeansClusterCC_randC_distinct_distances.m` - This file uses random K points for initial centroids
  - `kMeansClusterCC_randC_distinct_distances1.m` - This file uses the first K points as initial centroids.
  - `DistMatrix2D(m,c).m`

  Notes: Be aware that the MATLAB program `randpts.m` generates different data sets each time it is run. The key is to use the same data set to generate multiple results for analysis; hence, run `randpts.m` only once for each set of points.

  To obtain results for the same set of data, run `randpts.m` once and then `scatterpts_groupRandC2.m` as many times as needed. The last three files are embedded in the file `scatterpts_groupRandC2.m`; hence all five files must be stored in the same folder. The variable `cc` gives a matrix that contains the following data:

  Columns #1, 2: data ponts in 2D
  Column #3: cluster
  Column #4, 5: centroids in their corresponding cluster in 2D

Column #6: distances between a point and its centroid
Column #7: squared distances between a points and its centroid.

If data points are 3-dimensional, then the columns will be pushed off one column. That is, columns #1, 2, and 3 will display values for (x, y, z), and thus column #4 displays the Kth cluster, and so on.

Students' MATLAB Programs:

- `randpts.m`
- `scatterpts_group.m`
- `kMeansClusterCC2.m` - This program does not keep track of identical centroids.
- `DistMatrix2D(m,c).m`

• Sample Results

1. **Results 1:** 10 data points. See Figures 19-23

Points	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_10
x	4.36	4.47	3.06	5.09	5.11	8.18	7.95	6.44	3.79	8.10
y	5.32	3.50	4.31	9.39	8.76	5.50	6.22	5.87	2.07	4.70

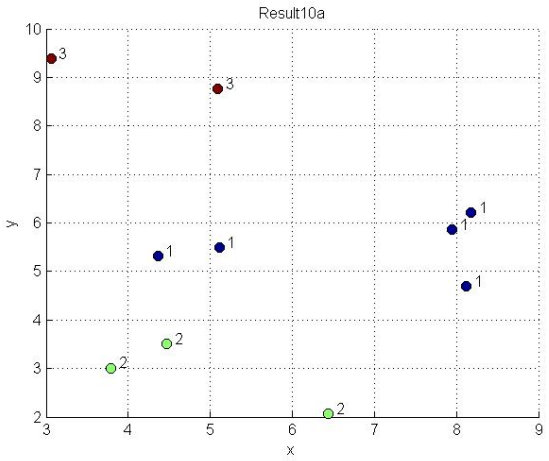


Figure 19: Initial Centroids: P_1, P_2, P_3

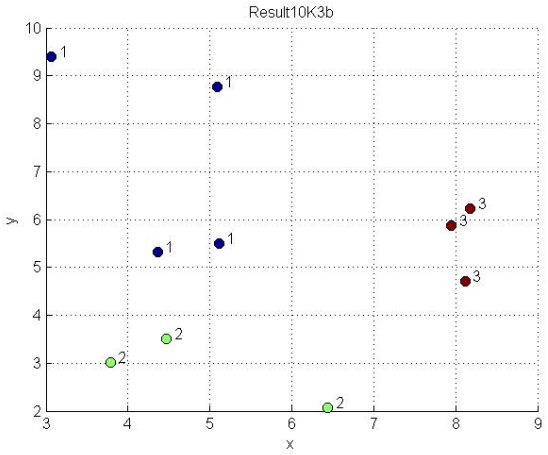


Figure 20: Initial Centroids: P_2, P_6, P_10

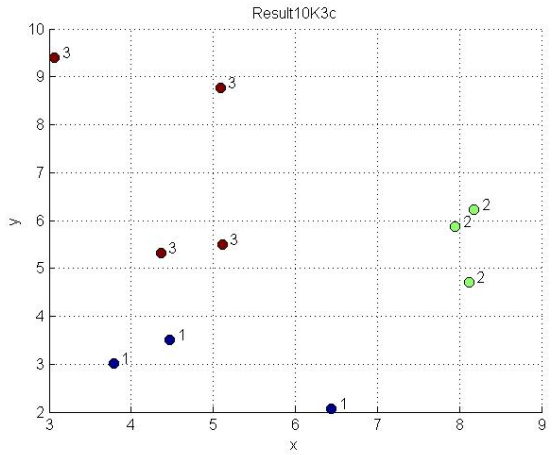


Figure 21: Initial Centroids P_5, P_6, P_7

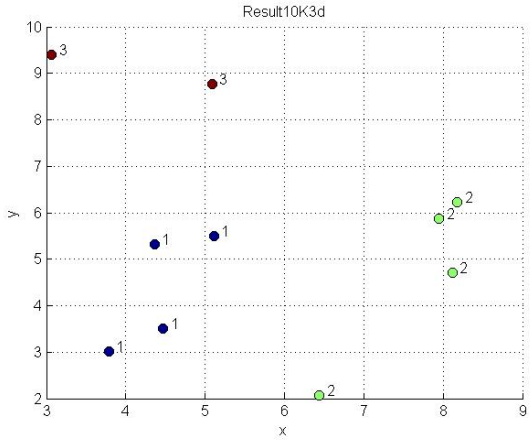


Figure 22: Initial Centroids P_1, P_6, P_7

2. **Results 2**: 50 data points. See Figures 24-27.

From the data values in Figure 23, we have the following:

- **Observation**
  - All four sets have different initial and final centroids.
  - 10b and 10c have the same objective value, although 10c takes more iterations to achieve that value.
  - 10d uses six iterations, but it has the smallest objective value.
- **Conclusion** - Different initial partition gives different clusters that may achieve different levels of optimization.

	Result10a		Result10b	
	Iteration: 1		Iteration: 1	
	initial centroids:		initial centroids:	
	4.3600	5.3200	5.1080	5.4970
	4.4700	3.5000	4.4680	3.5010
	3.0600	9.3900	8.1160	4.7040
	Iteration: 2		Iteration: 2	
	Centroids		Centroids	
	6.7440	5.5220	4.4038	7.2425
	4.9000	2.8600	4.8990	2.8583
	4.0750	9.0750	8.0807	5.5970
	Objective function =22.1923		**Objective function =22.5517**	

Result10c		**Result10d**			
Iteration: 1		Iteration: 1		Iteration: 4	
initial centroids:		initial centroids:		Centroids	
6.4430	2.0690	4.3590	5.3240	4.1568	5.3434
8.1770	6.2210	7.9490	5.8660	7.5027	4.2130
5.1080	5.4970	8.1770	6.2210	6.6310	7.4900
Iteration: 2		Iteration: 2		Iteration: 5	
Centroids		Centroids		Centroids	
5.1145	2.5370	4.6160	5.3636	4.4303	4.3317
8.0807	5.5970	8.0325	5.2850	7.5027	4.2130
4.4166	6.4942	7.1990	7.1990	5.4417	8.1233
Iteration: 3		Iteration: 3		Iteration: 6	
Centroids		Centroids		Centroids	
4.8990	2.8583	4.1568	5.3434	4.4303	4.3317
8.0807	5.5970	7.6712	4.7150	7.6712	4.7150
4.4038	7.2425	6.9220	6.9220	4.0740	9.0745
Objective function =22.5517				**Objective function =20.5506**	

Figure 23: Data for 10 points

Iteration: 1		
initial centroids:		P1
2.3050	7.9600	P2
8.4430	0.9780	P3
1.9470	2.6110	P4
2.2590	3.3470	P5
1.7070	6.7940	P6
Iteration: 9		
Centroids		
8.7696	7.4460	
7.5227	0.9131	
2.0879	1.8872	
5.4065	5.5435	
2.8742	7.5189	
Objective function =132.7959		

Figure 24: Data - Result50a

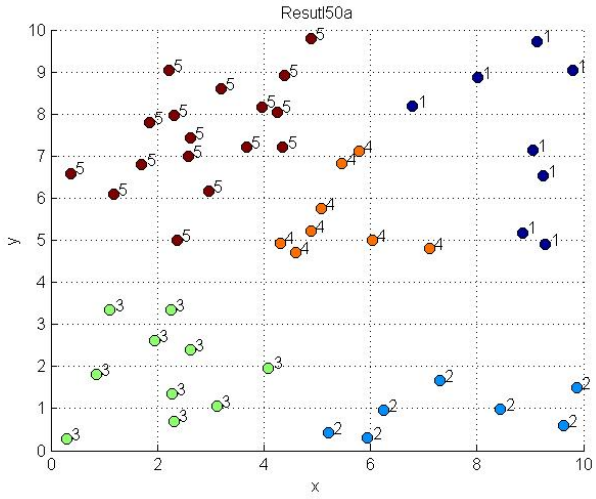


Figure 25: Plot - Result50a

Iteration: 1		
initial centroids:		
7.3040	1.6710	P33
2.3160	0.7050	P42
1.1740	6.0950	P23
2.2170	9.0470	P22
4.8890	5.2120	P34
Iteration: 5		
Centroids		
7.5227	0.9131	
2.0879	1.8872	
3.0036	5.8258	
3.4257	8.1984	
7.8792	6.9415	
Objective function =151.54		

Figure 26: Data - Result50b

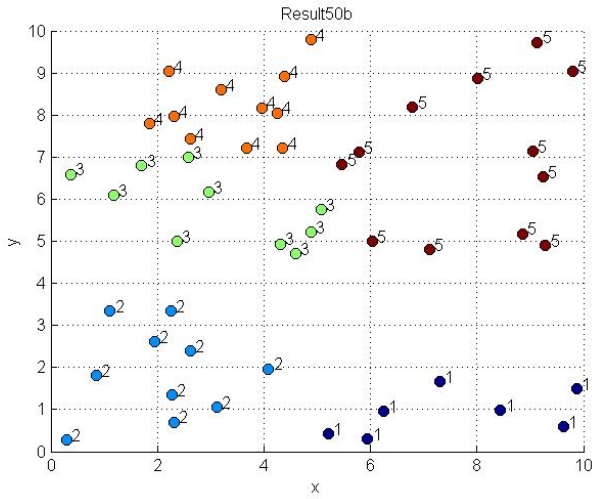


Figure 27: Plot - Result50b

Analysis of K-Means Results
Project #1
C. Chuo

Objective

We talked about the weaknesses of K-Means algorithm in the previous section. Due to these weaknesses, the results of K-Means algorithm may not be unique. The aim of this project is to implement the K-Means algorithm using the provided MATLAB code, obtain the results, and analyze the results to gain a better understanding of the K-Means clustering algorithm, especially the purpose of the objective function

$$d = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \| x_i - c_{n_j} \|^2 .$$

Perform the following and write a detailed report that addresses the questions.

- **Part I**

  1. Explain the differences between local and global extremum.
  2. Use a graphing calculator, MATLAB, or graphing program to graph the function $f(x) = 2x^4 - 6x^3 + 5x - 2$ and then find the local and global minimum.
  3. Use the same points from Example #1 of the Teaching Notes 1, choose points 1 and 4 as initial centroids, and find the 2 clusters.
  4. Is your result different than the example? Are there any differences in the computing process?

- **Part II**

  1. Open and run the script file "`randpts.m`" to generate 10 points. In your report, be sure to show your data point values.
  2. Modify "`kMeansClusterCC2(m,k)`", if needed, to assign the first K points as initial centroids.
  3. Run script file "`scatterpts_group.m`" to obtain a scatter plot of 3 clusters.
  4. State the initial centroids and describe the results.
  5. Modify "`kMeansClusterCC2(m,k)`" again, to assign random K points as initial centroids.
  6. Run the script file "`scatterpts_group.m`" three times and obtain three sets of results.
  7. Analyze the data and describe the differences in results obtained from step #3 and #4

8. Using the distance formula program on TI84 and/or Excel spreadsheet program, calculate the value for the objective function for results from steps #3 and #6. You may write a MATLAB script program for this.
9. Based on your practice and understanding from part I exercise problems, what are the causes of the discrepancies?

- **Part III**

1. Modify the "`randpts.m`" file to have 20 points.
2. Modify "`scatterpts_group.m`" to obtain 4 clusters.
3. Repeat all steps in Part II.
4. In addition to all steps in part II, describe any similarities and differences between results in part II and part III.

- **Part IV**

1. Modify the "`randpts.m`" file to have 50 points.
2. Modify "`scatterpts_group.m`" to obtain 5 clusters.
3. Repeat all steps in Part II.
4. In addition to all steps in part II, describe any similarities and differences of results in all parts.

- **MATLAB programs needed:**
  - `randpts.m`
  - `scatterpts_group.m`
  - `kMeansClusterCC2.m`
  - `DistMatrix2D.m`

K-Means Algorithm
Project #2
C. Chuo

- **Materials/Technology Needed:** Computer or laptop, MatLab, Excel Spreadsheet, TI calculator, any document processing program, e.g. MS Office Word, Open Office Word, etc.

- **Goal:** To provide students an opportunity for practice and develop a true appreciation and understanding of the applications of K-Means algorithm.

- **Motivation:** Grade Point Averages (GPA) are commonly used as indicators of academic performance. Although high schools in Florida have credit hour requirement for graduation, universities set a minimum GPA that should be maintained in order to continue in a degree program. In some Universities, the minimum GPA requirement set for the students is 1.5. Nonetheless, for any graduate program, a GPA of 3.0 and above is considered an indicator of good academic performance. Therefore, GPA remains the most common factor used by the academic planners to evaluate progression in an academic environment[3][4].

  In this exercise, we will be using nine tests of 15 students from the first three quarters of a high school senior math class. The objectives of this exercise are manifold:

  - Monitor students' progress
  - Interpret results
  - Determine the effectiveness of the K-Means algorithm

- **Directions:** Given nine test scores of 15 students from three quarters of a math course. Perform the following and write a detailed report and discussion of your findings. Be sure that your discussion addresses the questions.

  1. Run the `scattergrades.m` file provided using MatLab; obtain the results of

  A) 4 clusters of the test scores of the first set,

  B) 4 clusters of the test scores taking both first and second sets into account,

  C) 4 clusters of the data of all three semesters

  2. Describe the performance changes of a particular student during the three semesters.

  3. Analyze and describe the general behavior of the cluster changes.

  4. The K-Means algorithm groups objects that are similar in the same group. What similarity do you think this is? Are the results the same as you would expect? Describe in detail.

  5. Do you think these results are a good representation of group clustering?

  6. Do you think the K-Means algorithm can be used to keep track of students' academic progression? To what extent?

7. Are there any inadequacies of the algorithm? If there are any, what are the remedies?

- **MatLab Programs Needed:**
  1. Excel spreadsheet of grades
  2. `scattergrades2.m`
  3. `kMeansClusterCC2(m, k).m`
  4. `DistMatrix2D.m`

Appendices

A Project2: Sample Grades

Sample Grades

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9
Student 1	72	91	40	59	72	70	91	70	100
Student 2	74	83	61	100	78	92	46	51	78
Student 3	61	75	89	74	78	98	47	82	40
Student 4	67	78	93	63	47	42	77	52	49
Student 5	46	61	43	41	54	82	48	56	83
Student 6	50	50	39	80	88	66	36	78	62
Student 7	59	97	96	88	76	89	92	57	100
Student 8	54	66	69	85	77	84	87	92	90
Student 9	72	35	48	96	90	76	61	68	49
Student 10	88	40	63	85	47	51	52	58	53
Student 11	67	72	82	100	44	93	100	58	91
Student 12	40	92	51	38	83	47	93	94	94
Student 13	41	79	92	35	89	47	73	46	90
Student 14	98	55	96	80	66	96	83	99	38
Student 15	77	88	71	98	41	68	61	36	40

Figure 28: 9-Test Grades

B Sample Bellwork

K-Means Algorithm
Bell-work

Day 1:

1. Find the mid-point between the points: $(2, 5)$ and $(6, 8)$.

2. Given the vectors $\mathbf{u} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} 0 \\ 2 \\ -1 \end{bmatrix}$, find $\| \mathbf{u} \|$ and $\| \mathbf{v} \|$.

3. Find the sum: $\sum_{i=3}^{6} 2i^2 - 4i + 5$.

Day 2:

1. Given the following four colors, conjuecture which cluster that figure (a) belongs if (b), (c) and (d) represent the colors of the three clusters.

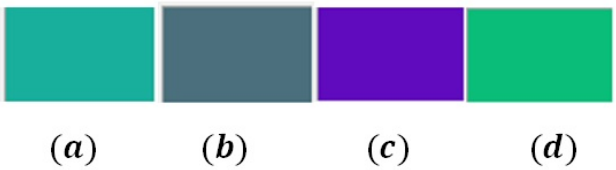


Figure 29: Caption for Color2Numbers1

2. Let $P = (24, 175, 156)$ be a point in 3D. Given that the centroids of three clusters $1, 2$ and 3 are $c_1 = (75, 111, 124)$, $c_2 = (96, 11, 189)$, and $c_3 = (10, 190, 123)$ respectively, determine the cluster to which P belongs.

3. Consider the MatLab code below.

```
1    function d=DistMatrix2D(A,B)
2    %%%
3    [row,data] = size(A);
4    [r,data] = size(B);
5    d = zeros(row,r);
6    for i = 1:row
7       for j = 1:r
8         d(i, j)=sqrt(sum((A(i,:)-B(j, :)).^2));
9       end
10   end
```

A) Is this a script or function file? How can you tell?

B) Describe what the code does between line #6 and #10?

C) What do you think the result is?

Day 3:

1. Find the magnitude of the difference of the vectors $\mathbf{u}$ and $\mathbf{v}$, where $\mathbf{u} = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} -2 \\ 1 \\ -5 \end{bmatrix}$.

2. Given the points $P_1(3,4)$, $P_2(4,1)$, $P_3(5,5)$ and $P_4(6,2)$, using the K-Means algorithm, find the 2 clusters after two iterations. Show your calculations and fill in the answers below.

Distances:	Centroids		Clusters	New Centroids		Clusters
P_i to c_j	$c_1(3,4)$	$c_2(5,5)$	1 or 2	$c_{1_{new}}$ (,)	$c_{2_{new}}$ (,)	1 or 2
P_1	0	$\sqrt{5}$	1			
P_2						
P_3						
P_4						

C Worksheet #1: Connecting Geometry to K-Means

Pre-Calculus **Name:** _____ **Period:** ____
Worksheet #1 **Date:**_____

Materials Needed: Straight-edge, protractor, graph paper, and calculator

1. Using the triangle given below and the construction tools, sketch and find the centroid of the triangle.

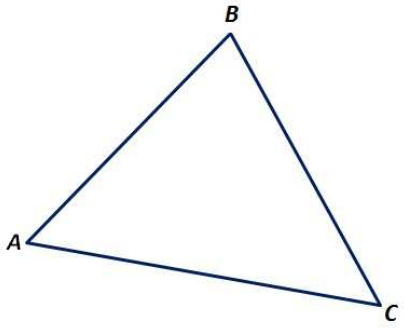


Figure 30: Triangle 1

2. Use the Euclidean distance to find the centroid of the triangle. Is this centroid the same as from problem #1? Explain why or why not? If they are the same, what are the central ideas?

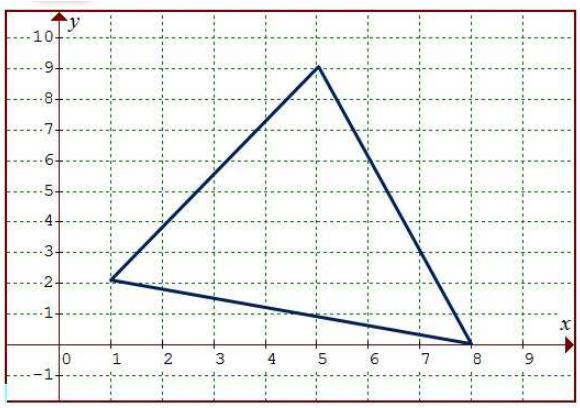
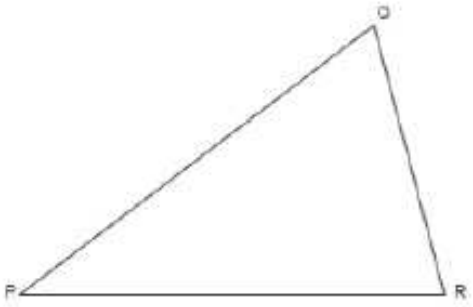
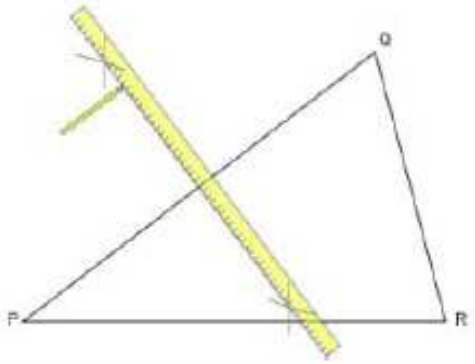
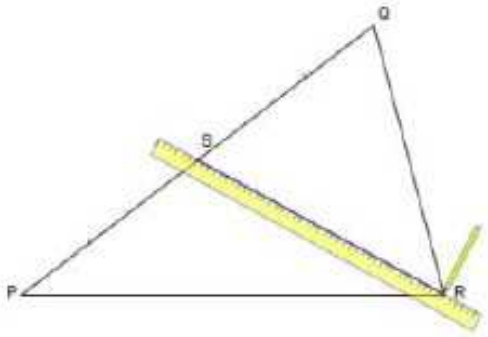
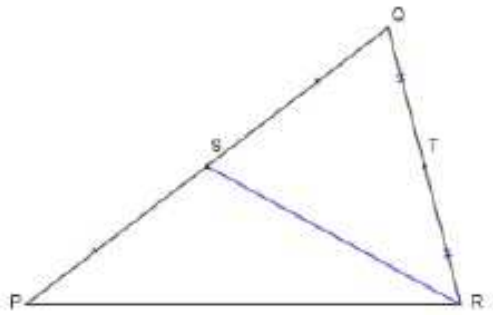
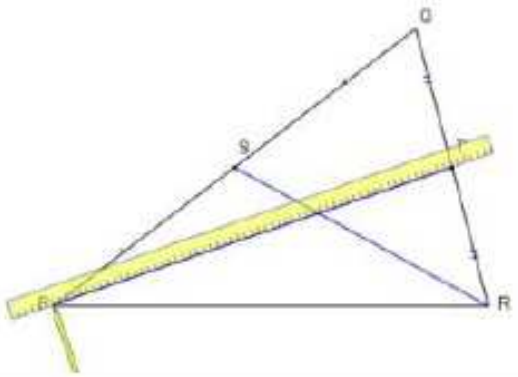
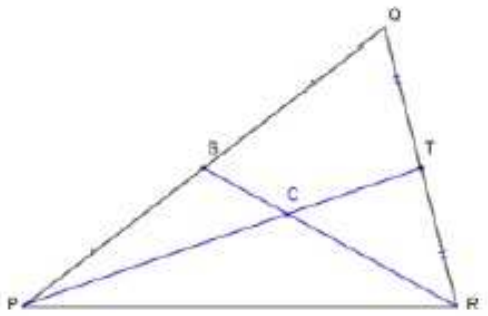


Figure 31: Triangle 2

3. Can you think of another way to find the centroid given that you have only a straight edge?

This construction assumes you are already familiar with Constructing the Perpendicular Bisector of a Line Segment.

After doing this	Your work should look like this
We start with a triangle PQR.	
First, we draw the median of the triangle through R	
1. Construct the bisector of the line segment PQ. Label the midpoint of the line S. See Constructing a perpendicular bisector of a line segment	
2. Draw the median from the midpoint S to the opposite vertex R	
Next, we draw the second median of the triangle through P	

After doing this	Your work should look like this
3. In the same manner, construct T, the midpoint of the line segment QR. See Constructing a perpendicular bisector of a line segment	
4. Draw the median from the midpoint T to the opposite vertex P	
(Optional step) Repeat for the third side. This will convince you that the three medians do in fact intersect at a single point. But two are enough to find that point.	
5. Done. The point C where the two medians intersect is the centroid of the triangle PQR.	

Reference: http://www.mathopenref.com/constcentroid.html

[2]

D MATLAB source file: `randpts.m`

```
%%%%%%  This script file generates "pt" number of random points
%
pt=10;        % number of points
%x= randi([0,10],[pt,1]);    %Generates integers only
%y= randi([-10,10],[pt,1]);
x= randi([0,10000],[pt,1])/1000  %Random rational numbers between 0 and 10
y= randi([-10,10000],[pt,1])/1000 %with an nx2 matrix, where pt=n
%
%Sample data for Project #1: 10 points
%x=[7.7250; 3.1190; 1.7900; 3.3890; 2.1010; 5.1020; 9.0640; 6.2890; 1.0150; 3.9080];
%y=[0.5360; 5.0080; 4.3110; 9.9760; 8.1140; 4.8510; 8.9440; 1.3660; 3.8940; 9.2730];

save('x','x')
save('y','y')
%A=[x y]
```

TI-84 program for computing distances

```
PROGRAM:D2
:Input A
:Input B
:Input C
:Input D
:√((A-C)²+(B-D)²
)→F
:Disp F
```

E MATLAB source file: scatterpts_group.m

```
%%% scatterpts_group.m
    %%% This script file does the following:
    %%%(1) plots the points with group indexes
    %%%(2) uses "random" k points as centroids
    %%%(3) Displays nth iterations and centroids for each iteration
    %
    %
k=3;        %%% number of clusters
load('x')
load('y')
    %
    %x=[7.7250; 3.1190; 1.7900; 3.3890; 2.1010; 5.1020; 9.0640; 6.2890; 1.0150; 3.9080];
    %y=[0.5360; 5.0080; 4.3110; 9.9760; 8.1140; 4.8510; 8.9440; 1.3660; 3.8940; 9.2730];
m=[x y]   %%% points displayed in matrix form
    %
    %kMeansClusterCC uses the first k pixels as initial centroid
cc=kMeansClusterCC2(m,k);  %% This shows centroids for EACH iteration
%cc=kMeansClusterCC(m,k);  %% This shows ony initial centroids
    %
    %plot all points with different colors
scatter(cc(:,1),cc(:,2), 60, cc(:,3),'o', 'filled', 'LineWidth', 1, 'MarkerEdgeColor', 'k')
    %
    %Plot all points without the points shown
    %scatter(cc(:,1),cc(:,2), 30, cc(:,3),'*', 'filled')
    %
    %Plot group indexes
for k=1:length(cc(:, 1))
    text(cc(k,1)+.1,cc(k,2)+.1, num2str(cc(k,3)))
end
    %axis off
grid on
```

F MATLAB source file: kMeansCluster_CC2.m

```
function y=kMeansClusterCC2(m,k) %m= matrix of points, k= number of clusters
           %%%%%%%%%%%%%%%%%%%%%%%%%%%%%% % %
           %%% This shows centroids for EACH iteration
           % kMeansCluster - Simple k means clustering algorithm
           % Author: Kardi Teknomo, Ph.D.
           %
           % Purpose: classify the objects in data matrix based on the attributes
           % Criteria: minimize Euclidean distance between centroids and object points
           % For more explanation of the algorithm, see http://people.revoledu.com/kardi/tutoria
           % Output: matrix data plus an additional column represent the group of each object %
           %
           % Example: m = [ 1 1; 2 1; 4 3; 5 4] or in a nice form
           %                  m = [ 1 1;
           %                            2 1;
           %                            4 3;
           %                            5 4]
           %                  k = 2
           % kMeansCluster(m,k) produces m = [ 1 1 1;
           %                                   2 1 1;
           %                                   4 3 2;
           %                                   5 4 2]
           %
           %
           % Input:
           % m - matrix data: objects in rows and attributes in columns
           % k - number of groups
           %
           % Local Variables
           % c - centroid coordinate size (1:k, 1:maxCol)
           % g - current iteration group matrix size (1:maxRow)
           % i - scalar iterator
           % maxCol - scalar number of rows in the data matrix m = number of attributes
           % maxRow - scalar number of columns in the data matrix m = number of objects
           % temp - previous iteration group matrix size (1:maxRow)
           % z - minimum value (not needed)
           %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  [maxRow, maxCol]=size(m); %m = Matrix of data to be clustered
  iteration=0;
  A=[];
  B=[];
  if maxRow<=k,
         y=[m, 1:maxRow];
  else

         % initial value of centroid
         iteration=iteration+1;
```

```
        disp(['Iteration: ' num2str(iteration)]);
        disp('initial centroids:');
        for i=1:k
             c(i,:)=m(i,:); %using first k pixels as centroid
             %c(i,:)= m(randi(maxRow,1), :);  %ith centroid randomly chosen
             disp(c(i,:));
        end

        temp=zeros(maxRow,1); % initialize as zero vector
        while 1,
             d=DistMatrix2D(m,c); % calculate objcets-centroid distances
            [z,g]=min(d,[],2); % find group matrix g
            if g==temp
                    break; % stop the iteration
            else
                   temp=g; % copy group matrix to temporary variable
           end
           for i=1:k
                c(i,:)=mean(m(find(g==i),:));
           end
           iteration=iteration+1;
           %display([%iteration iteration')
           disp(['Iteration: ' num2str(iteration)])
           disp('Centroids');
           disp(c);
        end

       y=[m,g];
end
```

G MATLAB source file: kMeansCluster.m

```
%%%%%%%%%%%%%%%%%%%  kMeansCluster.m
function y=kMeansCluster(m,k)
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% % %
     % kMeansCluster - Simple k means clustering algorithm
     % Author: Kardi Teknomo, Ph.D.
     %
     % Purpose: classify the objects in data matrix based on the attributes
     % Criteria: minimize Euclidean distance between centroids and object points
     % For more explanation of the algorithm, see http://people.revoledu.com/kardi/tutorial/kMea
     % Output: matrix data plus an additional column represent the group of each object %
     %
     % Example: m = [ 1 1; 2 1; 4 3; 5 4] or in a nice form
     %                 m = [ 1 1;
     %                           2 1;
     %                           4 3;
     %                           5 4]
     %                 k = 2
     % kMeansCluster(m,k) produces m = [ 1 1 1;
     %                                                                 2 1 1;
     %                                                                 4 3 2;
     %                                                                 5 4 2]
     % Input:
     % m - matrix data: objects in rows and attributes in columns
     % k - number of groups
     %
     % Local Variables
     % c - centroid coordinate size (1:k, 1:maxCol)
     % g - current iteration group matrix size (1:maxRow)
     % i - scalar iterator
     % maxCol - scalar number of rows in the data matrix m = number of attributes
     % maxRow - scalar number of columns in the data matrix m = number of objects
     % temp - previous iteration group matrix size (1:maxRow)
     % z - minimum value (not needed)
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   [maxRow, maxCol]=size(m);
   if maxRow<=k,
          y=[m, 1:maxRow];
   else

          % initial value of centroid
          for i=1:k
               c(i,:)=m(i,:);
          end

          temp=zeros(maxRow,1); % initialize as zero vector
          while 1,
```

```
            d=DistMatrix(m,c); % calculate objcets-centroid distances
            [z,g]=min(d,[],2); % find group matrix g
            if g==temp
                    break; % stop the iteration
            else
                   temp=g; % copy group matrix to temporary variable
            end
            for i=1:k
                 c(i,:)=mean(m(find(g==i),:));
            end
            end
        y=[m,g];
end
```

H MATLAB source file: K_means.m

```
%%%%%%% Source file: K_Means.m
tic;
clear all
clc
close all

% Input the name of test image
prompt = {'Enter test image name'};
dlg_title = 'Input of K-means image segmentation';
num_lines= 1;
def = {'1'};

TestImage = inputdlg(prompt,dlg_title,num_lines,def);
TestImage = strcat('.\',char(TestImage));
% TestImage = 'dawnlight.jpg';
m = imread(TestImage);
figure;
imshow(m);
m = double(m);


% Input the number of clusters
k = str2double(inputdlg('input the number of clusters','clusters'));
% k = 5;

% Default intensity value for different clusters
% clusters = zeros(1,10,3);
% clusters(1,1,:) = [0,0,0];
% clusters(1,2,:) = [255,255,255];
% clusters(1,3,:) = [255,0,0];
% clusters(1,4,:) = [255,255,0];
% clusters(1,5,:) = [0,255,0];
% clusters(1,6,:) = [0,0,255];
% clusters(1,7,:) = [128,128,255];
% clusters(1,8,:) = [128,0,128];
% clusters(1,9,:) = [128,128,0];
% clusters(1,10,:) = [0,255,255];

[maxRow,maxCol,rgb]=size(m);

% initial value of centroid
c = zeros(1,k,rgb);
for i = 1:k
    c(1,i,:)= m(i,i,:);
end
temp=zeros(maxRow,maxCol); % initialize as zero vector
```

```
while(1)
    d=DistMatrix(m,c); % calculate objcets-centroid distances
    [z,g]=min(d,[],3); % find group matrix g
    if(g==temp)
        for i = 1:k
            [x,y] = find(g == i);
            num = size(x,1);
            for j = 1:num
                m(x(j),y(j),:) = c(1,i,:);
            end
        end
        break; % stop the iteration
    else
        temp=g; % copy group matrix to temporary variable
        % Recalculate the centroids
        c = zeros(1,k,rgb);
        for i = 1:k
            [x,y] = find(g == i);
            num = size(x,1);
            for j = 1:num
                c(1,i,:) = c(1,i,:) + m(x(j),y(j),:)/num;
            end
        end
    end
end

m = uint8(m);
figure;
imshow(m);
toc;
```

I MATLAB source file: DistMatrix2D.m

```
%%%%%%%%%%%%%%%%%%%  DistMatrix2D.m
%This calculates distances between points in 2D
%Define A=[x1 y1; x2 y2; x3 y3; x4 y4]
%Define B=[a1 b1] - point or centroid
%B could be multi-row, then this calculates the distance betwee each point
%in A to each point in B.  Exp: C=[1 4; 2 0]
%Example: Points: (1,4), (2,0), (3,1) and (4,2)
%Define  A=[1 4; 2 0; 3 1; 4 2] and B=[1 4] or C=[1 4; 2 0]
%This returns: G =
%        0  - Distance between (1,4) and (1,4)
%   4.1231  - Distance between (1,4) and (2,0)
%   3.6056  - Distance between (1,4) and (3,1)
%   3.6056  - Distance between (1,4) and (4,2)
%Now try DistMatrix2D(A,C)

function d=DistMatrix2D(A,B)

[row,data] = size(A); %data is feature of data: color rgb or vector values
                      %but in 2D, this is the same as y
[k,data] = size(B);  % k is number of centroids
d = zeros(row,k);  %d=Distance Matrix. d should have same no. of row and
                   %k number of columns in B (same # of centroids)

for i = 1:row     %loop through # of points which is the same as # of rows in A
    for j = 1:k    %loop through # of centroids = # of rows in B
        d(i, j)=sqrt(sum((A(i,:)-B(j, :)).^2));

    end
end
```

J MATLAB source file: DistMatrix.m

```
%%%%%%%%%%%%%%%%%%%  DistMatrix.m
%This finds the distances between points in 3D
function d=DistMatrix(A,B)

[row,col,rgb] = size(A);
[r,c,rgb] = size(B);
d = zeros(row,col,c);
% for i = 1:row
%     for j = 1:col
%         for t = 1:c
%             d(i,j,t) = sqrt(sum((A(i,j,:)-B(1,t,:)).^2));
%         end
%     end
% end
for i = 1:c
    temp = zeros(row,col,rgb);
    for j = 1:rgb
        temp(:,:,j) = A(:,:,j) - B(1,i,j);
    end
    d(:,:,i) = sqrt(sum(temp.^2,3));
end
```

K MATLAB source file: `Scattergrades2.m`

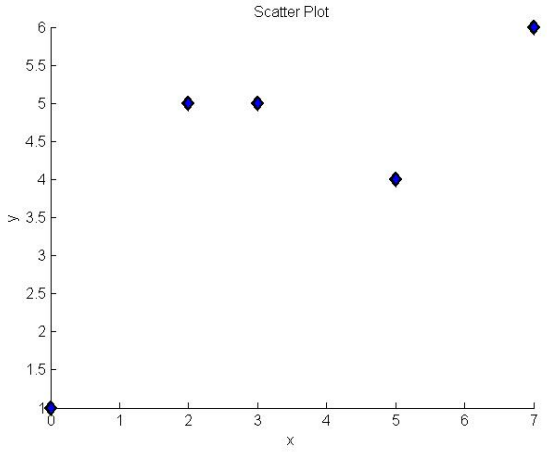
Create an Excel spreadsheet of grades before implementing this program.

```
%%%%%%%%%%%%%%%%%%%  scattergrades2.m
    %This file clusters data into k groups
    %Plots average vs. student
    %Creates a matrix of data values plus clusters in the last column in m1

load('m')
m=xlsread('grades.xlsx');
cols=length(m(1,:));   %# of grades
rows=length(m(:, 1));  %# of students
k=5;
m1=kMeansClusterCC2(m, k);

av=sum(m(:,1:cols), 2)/cols;  %2 is to sum along the columns and 1 to sum along the rows

scatter(1:rows, av, 40, m1(:, cols+1), 'filled','LineWidth', 1, 'MarkerEdgeColor', 'k')


for j=1:rows %# of students
    text(j+.1,av(j)+.1, num2str(m1(j,cols+1))) %reads and labels last column of m1 which is the
    %(x, y)=(j+0.1, av(j)+0.1)
end
    %axis off
title('Average Vs. Student');
xlabel('Student');
ylabel('Average');
```

L ANSWERS

- **Teaching Notes Exercises**

1. Scatter plot done by MATLAB:



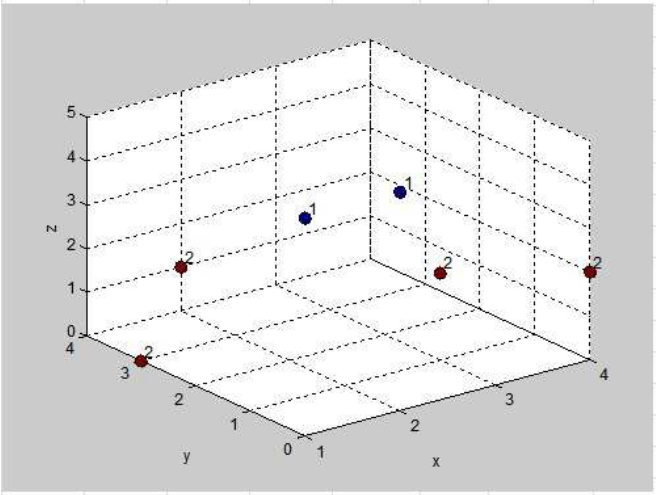
2. (a)
$$\begin{bmatrix} 0 & 1 & 0 & 4.47 \\ 2 & 5 & 4.47 & 0 \\ 3 & 5 & 5.00 & 1.00 \\ 5 & 4 & 5.83 & 3.16 \\ 7 & 6 & 8.60 & 5.10 \end{bmatrix}$$

(b) First iteration:
$$\begin{bmatrix} 0 & 1 & 0 & 4.47 & 1 \\ 2 & 5 & 4.47 & 0 & 2 \\ 3 & 5 & 5.00 & 1.00 & 2 \\ 5 & 4 & 5.83 & 3.16 & 2 \\ 7 & 6 & 8.60 & 5.10 & 2 \end{bmatrix}$$

Second iteration:
$$\begin{bmatrix} 0 & 1 & 0 & 4.85 & 1 \\ 2 & 5 & 4.47 & 2.80 & 2 \\ 3 & 5 & 5.00 & 2.08 & 2 \\ 5 & 4 & 5.83 & 1.00 & 2 \\ 7 & 6 & 8.60 & 3.83 & 2 \end{bmatrix}$$

3. Answer vary depending on the initial centroids.
Sample answer using P_3 and P_5 as initial centroids:

x	y	z	Cluster	x	y	z	distances	sq. dist
1	3	0	2	2.5	2	1.25	2.19	4.81
2	0	5	1	1.5	0	5	0.50	0.25
3	1	2	2	2.5	2	1.25	1.35	1.81
4	0	2	2	2.5	2	1.25	2.61	6.81
1	0	5	1	1.5	0	5	0.50	0.25
2	4	1	2	2.5	2	1.25	2.08	4.31



Iteration: 1
initial centroids:
1 0 5 P5

3 1 2 P3

Iteration: 2
Centroids
1.5000 0 5.0000
2.5000 2.0000 1.2500

Objective function =18.25
>>

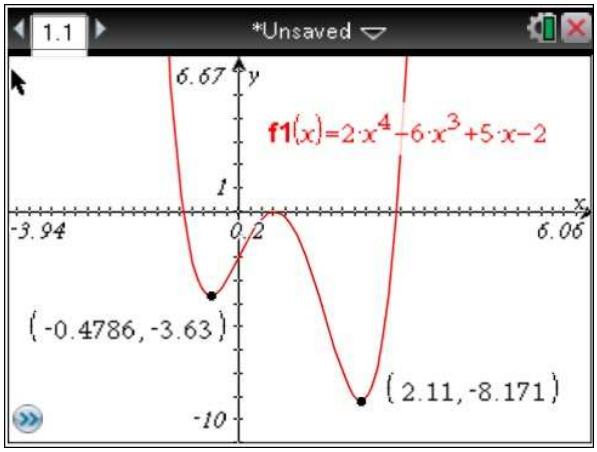
MATLAB code:

```
%%%  scatterpts_group3DRandC.m
%%%
%%% This script file does the following:
%%%(1) plots the points with group indexes
%%%(2) uses "random" k points as centroids
%%%(3) Displays nth iterations and centroids for each iteration

%%% cc: 9-column matrix: col8: distances, col9: squared-distances
%%% col #5 -6: coordinates of centroids
%
k=2;        %%% number of clusters
%load('x')
%load('y')
x=[1; 2; 3; 4; 1; 2];
y=[3; 0; 1; 0; 0; 4];
z=[0; 5; 2; 2; 5; 1];
%x=[7.7250; 3.1190; 1.7900; 3.3890; 2.1010; 5.1020; 9.0640; 6.2890; 1.0150; 3.9080];
%y=[0.5360; 5.0080; 4.3110; 9.9760; 8.1140; 4.8510; 8.9440; 1.3660; 3.8940; 9.2730];
m=[x y z]   %%% points displayed in matrix form
%
%kMeansClusterCC uses the first k pixels as initial centroid
cc=kMeansClusterCC_randC_distinct_distances(m,k);  %cc is a 4-column matrix
%
%plot all points with different colors
scatter3(cc(:,1),cc(:,2),cc(:,3), 60, cc(:,4),'o', 'filled', 'LineWidth', 1, 'MarkerEdgeColor', 'k')
%
%Plot all points without the points shown
%scatter(cc(:,1),cc(:,2), 30, cc(:,3),'*', 'filled')
%
%Plot group indexes
for k=1:length(cc(:,1))
    text(cc(k,1)+.1,cc(k,2)+.1,cc(k, 3)+.1, num2str(cc(k,4)))
end
%axis off
grid on
```

- **Project #1 Results Analysis**
  (Also see Teacher's Notes for direction on MATLAB codes & sample results)

  1. Answer vary. Also see definition of extrema

  2.



  3.

			Distance to				Distance to				Distance to				Distance to		
	x	y	c1	c2	Assign		c1	c2	Assign		c1	c2	Assign		c1	c2	Assign
P1	**1**	**0**	0.00	7.07	k1		1.41	6.66	k1		2.24	7.60	k1		2.24	7.60	k1
P2	**3**	**2**	2.83	4.24	k1		1.41	3.91	k1		1.00	4.81	k1		1.00	4.81	k1
P3	**5**	**1**	4.12	4.12	k2		3.00	3.13	k1	change	2.00	4.18	k1		2.00	4.18	k1
P4	**6**	**5**	7.07	0.00	k2		5.66	1.35	k2		5.00	1.05	k2		5.00	1.05	k2
P5	**7**	**3**	6.71	2.24	k2		5.39	0.90	k2		4.47	1.67	k2		4.47	1.67	k2
P6	**8**	**6**	9.22	2.24	k2		7.81	2.70	k2		7.07	1.67	k2		7.07	1.67	k2
Iteration	**1**			**2**			**3**				**4**				**5**		
c1	1.00	0.00		2.00	1.00		3.00	1.00			3.00	1.00			3.00	1.00	
c2	6.00	5.00		6.50	3.75		7.00	4.67			7.00	4.67			7.00	4.67	

When points 1 and 4 are chosen as the initial centroids, it takes only three iterations to finalize the clusters while it takes four iterations when points 1 and 3 are used.

- **Project #1: MATLAB Code for teaching and demonstrations**

  1.

```
%%% scatterpts_groupRand2.m
%%% This script file does the following:
%%%(1) plots the points with group indexes
%%%(2) uses "random" k points as centroids
%
k=2;        %%% number of clusters
load('x')
load('y')
%x=[7.7250; 3.1190; 1.7900; 3.3890; 2.1010; 5.1020; 9.0640; 6.2890; 1.0150; 3.9080];
%y=[0.5360; 5.0080; 4.3110; 9.9760; 8.1140; 4.8510; 8.9440; 1.3660; 3.8940; 9.2730];
m=[x y]   %%% points displayed in matrix form
%
%kMeansClusterCC uses the first k pixels as initial centroid
%
cc=kMeansClusterCC_randC_distinct_distances1(m,k); %modify here to get random centroids
%cc=kMeansClusterCC2(m,k);
%
%plot all points with different colors
scatter(cc(:,1),cc(:,2), 50, cc(:,3),'o', 'filled', 'LineWidth', 1, 'MarkerEdgeColor', 'k')
%
%Plot all points without the points shown
%scatter(cc(:,1),cc(:,2), 30, cc(:,3),'*', 'filled')
%
%Plot group indexes
for k=1:length(cc(:, 1))
    text(cc(k,1)+.1,cc(k,2)+.1,num2str(cc(k,3)))
end
%axis off
grid on
```

  2.

```
function y=kMeansClusterCC_randC_distinct_distances(m,k) %m= matrix of points, k= number of clusters
          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% % %
          % kMeansCluster - Simple k means clustering algorithm
          % Author: Kardi Teknomo, Ph.D.
          %
          % Purpose: classify the objects in data matrix based on the attributes
          % Criteria: minimize Euclidean distance between centroids and object points
          % For more explanation of the algorithm, see http://people.revoledu.com/kardi/tutorial/kMean/index.html %
          % Output: matrix data plus an additional column represent the group of each object %
          %
          % Example: m = [ 1 1; 2 1; 4 3; 5 4] or in a nice form
          %                 m = [ 1 1;
          %                           2 1;
          %                           4 3;
          %                           5 4]
          %                 k = 2
          % kMeansCluster(m,k) produces m = [ 1 1 1;
          %                                   2 1 1;
          %                                   4 3 2;
          %                                   5 4 2]
          %
          %
          % Input:
          % m - matrix data: objects in rows and attributes in columns
          % k - number of groups
          %
          % Local Variables
          % c - centroid coordinate size (1:k, 1:maxCol)
          % g - current iteration group matrix size (1:maxRow)
          % i - scalar iterator
```

```
          % maxCol - scalar number of rows in the data matrix m = number of attributes
          % maxRow - scalar number of columns in the data matrix m = number of objects
          % temp - previous iteration group matrix size (1:maxRow)
          % z - minimum value (not needed)
          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   [maxRow, maxCol]=size(m); %m = Matrix of data to be clustered
   iteration=0;
   %
   %
   if maxRow<=k,    %% k = number of centroids
          y=[m, 1:maxRow]; %% Ensuring no. of centroids is less than no. of points
   else
  %
          % initial value of centroid
          iteration=iteration+1;  %% count initial picking as first iteration
  indices = zeros (1,k);  %% keeping track of which point is centroid
          disp(['Iteration: ' num2str(iteration)]);
          disp('initial centroids:');
          %
          for i=1:k   %% picking k centroids
               %c(i,:)=m(i,:); %using first k pixels as centroid
   j=randi(maxRow,1); %choosing one no. between 1 and maxrow

               %%% Ensuring for distinct centroids
   while (find(indices == j))
      j = randi(maxRow,1);
               end

   indices(i) = j;  %store and keeps track of points picked
               c(i,:)= m(j, :);  %ith centroid randomly chosen
               disp(c(i,:));
               %a=[c(i,:);c(i+1,:)]
          end
          %
          temp=zeros(maxRow,1); % initialize as zero vector
          while 1,
               d=DistMatrix2D(m,c); % calculate objcets-centroid distances
              [z,g]=min(d,[],2); % find group matrix g

              if g==temp
                      break; % stop the iteration
              else
                     temp=g; % copy group matrix to temporary variable
             end
             for i=1:k
                  c(i,:)=mean(m(find(g==i),:));
             end
             iteration=iteration+1;
             %display([%iteration iteration')
             disp(['Iteration: ' num2str(iteration)])
             disp('Centroids');
             disp(c);
              %plot(c(:,1),(:,2));
           %
          end
distances = zeros(maxRow,1);  %create a column for distances
    centroids = zeros(maxRow,maxCol);  %creates columns for final centroids
for i = 1:maxRow
distances(i) = d(i,g(i));  %d: distMatrix // g(i): index of ith centroid
                                     %stores the distance of the ith
                                     %point to the correspoinding
                                     %centroid
    centroids(i,:) = c(g(i),:); %store the corresponding centroid
        end
%
         y=[m,g,centroids, distances, distances.^2]; %% change to distances.^2 for squared distances
         disp(['Objective function =' num2str(sum(distances.^2))]);
end
```

- **Project #2: Sample Results for 5 clusters**

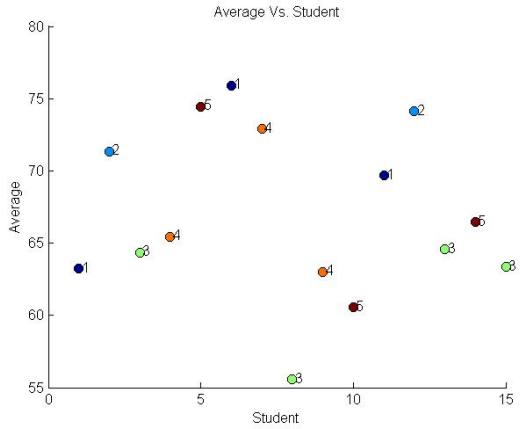
  Centroids and iterations:

```
 Iteration: 1
initial centroids:
    69    64    51    61    60    94    36    40    94

    86    57    78    41    97    48    92    70    73

    84    66    54    38    46    83    79    44    85

    95    85    43    55    89    38    77    43    64

    92    39    92    90    40    81    72    66    98

Iteration: 2
Centroids
   69.0000   74.0000   59.0000   81.0000   68.0000   86.0000   59.3333   46.0000   84.0000
   85.0000   53.5000   73.5000   56.5000   88.0000   66.0000   93.0000   72.0000   67.0000
   65.2500   51.5000   55.0000   52.5000   52.0000   74.0000   83.2500   49.0000   75.0000
   80.0000   77.3333   50.0000   70.3333   84.0000   54.6667   80.3333   45.3333   62.0000
   80.3333   41.0000   76.6667   94.0000   45.0000   73.0000   75.6667   53.6667   65.0000
```

  Grades and clusters: These are randomly generated grades by Excel.

```
69 64 51 61 60 94 36 40 94 1
86 57 78 41 97 48 92 70 73 2
84 66 54 38 46 83 79 44 85 3
95 85 43 55 89 38 77 43 64 4
92 39 92 90 40 81 72 66 98 5
82 78 63 85 82 97 81 42 73 1
79 85 67 89 88 64 64 46 74 4
59 61 63 45 51 56 71 43 51 3
66 62 40 67 75 62 100 47 48 4
55 47 81 98 55 44 56 58 51 5
56 80 63 97 62 67 61 56 85 1
84 50 69 72 79 84 94 74 61 2
53 36 45 72 56 96 86 46 91 3
94 37 57 94 40 94 99 37 46 5
65 43 58 55 55 61 97 63 73 3
```

Data plot: 5 clusters



- **Sample Bellwork: Day 1**

  1. $\left(4, \frac{13}{2}\right)$

  2. $\| \mathbf{u} \| = \sqrt{13}$; $\| \mathbf{v} \| = \sqrt{5}$

  3. 120

- **Sample Bellwork: Day 2**

  1. (d)

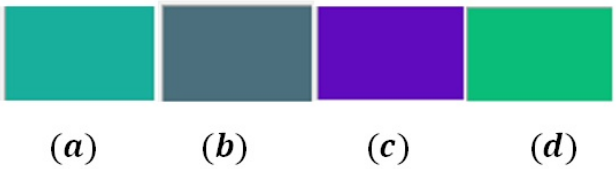


Figure 32: Color2Numbers1

  2. $d_{Pc_1} = 87.9$
  $d_{Pc_2} = 182.1$
  $d_{Pc_3} = 38.9$
  Hence, $P \in k_3$.

  Note: This is the same problem as part (A). It would be helpful to use the program "Paint" to show students the colors and their vector values. P is the vector value of color (a), c_1 is the vector value of color (b), and so on. The answer is the same as for part (A).

Using the "Paint" program, here are the vector values of the colors:

3. A) A function file

B) The code calculates the distances from all points to the chosen r points.

C) The result is a distance matrix with values obtained from the previous question

- **Sample Bellwork: Day 3**

1. $\| \mathbf{u} - \mathbf{v} \| = \sqrt{53}$

2.

Distances	$c_1(3,4)$	$c_2(5,5)$	k_1/k_2	$c_{1_{new}}\left(\frac{7}{2},\frac{5}{2}\right)$	$c_{2_{new}}\left(\frac{11}{2},\frac{7}{2}\right)$	k_1/k_2
P_1	0	$\sqrt{5}$	k_1	1.58	2.55	k_1
P_2	$\sqrt{10}$	$\sqrt{17}$	k_1	1.58	2.92	k_1
P_3	$\sqrt{5}$	0	k_2	2.92	1.58	k_2
P_4	$\sqrt{13}$	$\sqrt{10}$	k_2	2.55	1.58	k_2

- **Worksheet**

1. See Construction page

2. $A = (1,2)$, $B = (5,9)$, $C = (8,0)$

$$\text{Centroid} = \left(\frac{1+5+8}{3}, \frac{2+9+0}{3}\right) = \left(\frac{14}{3}, \frac{11}{3}\right) \approx (4.67, 3.67)$$

3. Find centroid of a triangle by "folding": Steps -

- Find midpoint of all sides by overlapping the verteces
- Fold and create creases along the midpoint and each vertex.
- The intersection is the centroid.

For a video intruction, see `http://www.youtube.com/watch?v=mFwFGkd1Q-s`

Acknowledgement

We acknowledge support of the NSF grant #1200566

References

[1] http://people.revoledu.com/kardi/tutorial/kMean/Weakness.htm

[2] http://www.mathopenref.com/constcentroid.html

[3] O.J. Oyelade, O.O. Oladipupo, I.C. Obagbuwa, *Application of k-Means Clustering algorithm for prediction of Students' Academic Performance'*, International Journal of Computer Science and Information Security, 2010.

[4] S. Sujit Sansgiry, M. Bhosle, and K. Sail, *Factors that affect academic performance among pharmacy students*, American Journal of Pharmaceutical Education, 2006.