Problem 1: Clustering with K-Means

0. Importing and understanding the dataset

In [1]:
```
from sklearn import datasets
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0)
```

In [2]:
```
data = datasets.load_breast_cancer()
print(data.keys())
# ['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename']
# 'data': feature data to determine how to classify the sample, given in np array
# 'target': target data (ie: is the tumor malignant or benign), given in np array
# 'feature_names': names of the columns in data
# 'target_names': names of the target columns

training_set = data.data
print(training_set.shape) # (569, 30)
```

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
(569, 30)
```

1. Implement K-means

- Input: array containing dataset, value of K
- Output: cluster centroids, cluster assignment for each data point

In [3]:
```
def initialize_centroids(training_set, m, k):
    # pick k cluster centroids randomly in the training set
    # return a 1xk array where each element is at most 569 (aka training_set.shape[0])
    # don't allow the same value to be picked multiple times (ie: replace = False)

    random_row_indices = np.random.choice(np.shape(training_set)[0], k, replace=False)
    centroids = training_set[random_row_indices, :]
    return centroids

def distance(x, u, m, k):
    # squared norm error between a data point x(i) and a centroid u(k)
    # ||x(i) - u(k)||^2 = (sqrt((x1(i) - u1(k))^2 + (x2(i) - u2(k))^2 + ...))^2
    #                   = (x1(i) - u1(k))^2 + (x2(i) - u2(k))^2 + ...
    distance = np.zeros((m, k))
    for i in range(m):
        for j in range(k):
            distance[i][j] = (np.linalg.norm(x[i] - u[j]))**2 # distance from point x(i) to centroid u(k)

    return distance

def centroid_assignment(distances):
    # based on the squared norm distance from x(i) to u(i), determine which cluster with index k the point x(i) should belong to
    # find the min. distance for each point (ie: find the min element in a row of the array)

    c = np.empty(distances.shape[0], dtype=np.int8)

    for i in range(distances.shape[0]):
        smallest_error = np.min(distances[i])
        smallest_error_index = np.argwhere(distances[i] == smallest_error)[0][0]
        c[i] = smallest_error_index

    return c

def move_centroids(centroids, c, training_set, k):
    # find all training data indices i for which c[i] is the same (i.e.: the value is k) -> np.argwhere()
    # average the values of x[i]
    # assign the value to centroid[k]

    old_centroids = centroids

    for j in range(k):
        same_centroid = np.argwhere(c == j)
        centroids[j] = np.mean(training_set[same_centroid], axis=0)

    return centroids, old_centroids

def should_stop(old_centroids, new_centroids):
    # stop if the old cluster centroid assignments are the same as the new cluster centroid assignments
    if (old_centroids == new_centroids).all():
        return True
```

```
    else:
        return False
```

In [4]:
```
def k_means(training_set, k):
    m = training_set.shape[0]

    # initialize cluster centroids
    centroids = initialize_centroids(training_set, m, k)

    old_centroids = np.empty(centroids.shape)
    c = np.empty(m)

    while not should_stop(old_centroids, centroids): # if no centroid position changed, then the algorithm is complete

        # find the squared norm distance between each data point in the training set and all cluster centroids
        distances = distance(training_set, centroids, m, k)

        # cluster assignment
        c = centroid_assignment(distances)

        # move centroids
        centroids, old_centroids = move_centroids(centroids, c, training_set, k)

    return centroids, c
```

In [5]:
```
# sanity check to make sure that my k_means implementation returns a result on a trivial dataset
test_training_set = np.random.random((10, 2))
centroids, cluster_assignment = k_means(test_training_set, 2)
print(centroids)
print(cluster_assignment)
```

```
[[0.07103606 0.0871293 ]
 [0.57051404 0.70425608]]
[1 1 1 1 1 1 1 0 1 1]
```

2. Run K-means algorithm for k varying between 2 and 7

Explanation of `k_means` function inputs:

The `training_set` (`sklearn.datasets.load_breast_cancer().data`) input variable is a 569 x 30 NumPy array containing feature data that describes the characteristics of the cell nuclei. Note that the `sklearn.datasets.load_breast_cancer().target` data is not needed because K-means is an unsupervised algorithm, which means it does not use the dataset's labels.

`K` is the number of cluster centroids that the K-Means algorithm should form around the points in the input dataset.

In [6]:
```
for k in range(2, 8):
    centroids, cluster_assignment = k_means(training_set, k)
    print("When k = %d, the centroids are:" % k)
    print(centroids)
    print("\nand the cluster assignments are:")
    print(cluster_assignment)
    print("\n")
```

```
When k = 2, the centroids are:
[[1.74791266e+01 2.08779913e+01 1.15178603e+02 9.74493886e+02
  1.00080699e-01 1.36085109e-01 1.47086026e-01 8.24311354e-02
  1.88402183e-01 6.13818777e-02 5.86226638e-01 1.15645764e+00
  4.15768996e+00 6.96300000e+01 6.50541485e-03 3.02648690e-02
  3.92163712e-02 1.45209083e-02 1.98221659e-02 3.85201659e-03
  2.09672489e+01 2.80111354e+01 1.39900873e+02 1.39594105e+03
  1.38713362e-01 3.39472227e-01 4.07594017e-01 1.70624410e-01
  3.09772489e-01 8.68513537e-02]
 [1.18697324e+01 1.82198529e+01 7.63367059e+01 4.39625882e+02
  9.38544706e-02 8.29603824e-02 4.95415021e-02 2.63478353e-02
  1.76285294e-01 6.37511471e-02 2.83226471e-01 1.25753176e+00
  1.99610794e+00 2.06074353e+01 7.40169706e-03 2.22541353e-02
  2.69616929e-02 9.96092353e-03 2.10273294e-02 3.75643676e-03
  1.31049088e+01 2.41052647e+01 8.52774412e+01 5.33474412e+02
  1.28095206e-01 1.96875500e-01 1.80988874e-01 7.68763265e-02
  2.76809118e-01 8.19888529e-02]]

and the cluster assignments are:
[0 0 0 1 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 1 1 0 1 1 1 1 1 0
 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 0 0 1 0 0 0 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 1 1 1 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 1 1 0 0 1 1
 1 0 1 0 1 0 1 1 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1
 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 0 1 1 0 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 0
 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0
```

```
 1 0 1 1 1 1 0 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 1 1 1 1 0 1 1
 0 1 0 0 0 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1
 0 1 1 0 1 0 0 1 0 1 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1 1 1 0 0 1 1 0 0 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]


When k = 3, the centroids are:
[[1.95430400e+01 2.17314400e+01 1.29295200e+02 1.20446880e+03
  1.01212240e-01 1.48352560e-01 1.77952640e-01 1.01437040e-01
  1.91270400e-01 6.04592800e-02 7.47665600e-01 1.22007600e+00
  5.28069600e+00 9.70044000e+01 6.55269600e-03 3.23240560e-02
  4.25852000e-02 1.56708400e-02 2.04582400e-02 3.98989600e-03
  2.38914400e+01 2.88933600e+01 1.59595200e+02 1.77789600e+03
  1.39904320e-01 3.56494880e-01 4.50019200e-01 1.92622960e-01
  3.11418400e-01 8.61172000e-02]
 [1.42819704e+01 1.94118719e+01 9.29795074e+01 6.32965517e+02
  9.54984236e-02 1.06713892e-01 8.62499754e-02 4.73187783e-02
  1.79087192e-01 6.18801478e-02 3.37761084e-01 1.07925665e+00
  2.41458079e+00 3.02404433e+01 6.08706404e-03 2.50067143e-02
  3.04668571e-02 1.15277833e-02 1.83680197e-02 3.48998571e-03
  1.62645813e+01 2.61111330e+01 1.07581429e+02 8.17306404e+02
  1.31508916e-01 2.80893744e-01 2.95969813e-01 1.21273887e-01
  2.94476355e-01 8.51564532e-02]
 [1.11880041e+01 1.79202075e+01 7.17578423e+01 3.88304149e+02
  9.45696680e-02 7.95146473e-02 4.47053349e-02 2.30276017e-02
  1.77666390e-01 6.47832365e-02 2.84312033e-01 1.33108299e+00
  1.99394523e+00 1.94499502e+01 8.09774274e-03 2.23244440e-02
  2.75502183e-02 1.00124772e-02 2.24173444e-02 3.95060664e-03
  1.23196224e+01 2.36436100e+01 7.98473029e+01 4.68471369e+02
  1.29184149e-01 1.78811286e-01 1.59921058e-01 6.85247801e-02
  2.75298755e-01 8.17998340e-02]]

and the cluster assignments are:
[0 0 0 2 0 1 0 1 1 1 1 0 0 1 1 1 1 0 0 1 1 2 1 0 0 0 1 0 1 0 0 1 0 0 0 0 1
 2 1 1 1 2 0 1 1 0 2 1 2 1 2 1 2 0 1 2 0 1 1 2 2 2 1 2 1 1 2 2 2 2 0 2 0 1
 2 1 1 0 0 1 2 1 0 0 2 0 1 0 2 1 1 1 1 1 1 0 2 2 2 1 1 2 2 2 2 1 2 2 0 2 2
 2 1 2 2 2 2 1 0 0 2 0 0 1 1 1 1 0 1 0 2 1 1 1 0 1 2 2 1 2 2 1 2 1 2 2 2 1
 1 1 1 2 2 2 1 2 0 1 2 2 2 0 0 2 0 1 2 0 0 1 2 1 1 2 2 2 2 1 1 2 0 0 1 2 1
 2 0 2 2 2 1 2 2 1 1 2 1 0 0 1 2 0 0 1 1 1 2 0 1 1 0 2 0 1 1 1 2 2 0 0 1 1
 2 1 1 1 2 1 2 1 1 2 2 0 2 1 0 0 1 0 1 2 2 1 0 2 1 1 2 2 0 2 0 0 0 1 0 1 1
 1 0 0 0 1 0 0 2 1 2 2 1 2 0 2 0 2 2 0 1 1 0 2 0 1 2 2 2 2 2 2 2 1 1 2 2 2 1
 2 2 1 2 0 2 0 2 2 2 1 2 1 1 2 1 2 2 2 2 2 0 2 2 2 0 2 0 2 2 1 2 1 1 1 1 2
 2 2 0 2 0 2 0 1 2 2 0 2 2 2 1 2 2 2 1 0 1 2 2 2 1 2 2 2 1 2 1 1 0 0 2 0 0
 1 1 0 0 1 1 2 1 1 2 2 2 2 2 1 1 2 1 2 0 2 2 0 0 2 1 1 2 2 2 0 2 1 2 2 2 1
 1 0 2 2 2 2 1 1 2 2 0 2 2 2 1 2 1 2 2 2 2 2 2 1 2 0 0 1 1 1 1 1 1 2 0 1 2
 0 2 0 1 1 0 2 0 2 1 1 1 2 1 1 2 0 0 1 2 1 1 1 2 0 2 2 2 1 2 2 2 1 1 2 1 2
 1 1 1 1 2 1 0 2 1 2 0 0 2 1 1 2 2 0 0 1 1 2 0 2 2 2 2 1 1 2 1 1 1 1 2 0 0
 1 2 2 0 2 1 2 2 1 2 1 2 2 2 1 0 2 0 1 2 2 2 2 1 1 1 1 1 2 2 2 2 2 2 2 2 2
 2 2 2 1 2 1 2 1 0 0 0 1 0 2]


When k = 4, the centroids are:
[[1.86433333e+01 2.18255556e+01 1.23087879e+02 1.10033232e+03
  1.00925455e-01 1.44982121e-01 1.64867879e-01 9.31973737e-02
  1.91317172e-01 6.06556566e-02 7.03007071e-01 1.09958081e+00
  4.84673737e+00 8.91811111e+01 6.10022222e-03 3.03844444e-02
  3.87474747e-02 1.43712020e-02 1.81937374e-02 3.72207071e-03
  2.34732323e+01 2.96264646e+01 1.56382828e+02 1.72818182e+03
  1.43825253e-01 3.78693939e-01 4.61301010e-01 1.91489596e-01
  3.23242424e-01 8.91446465e-02]
 [1.39927619e+01 1.91716667e+01 9.09324286e+01 6.06775714e+02
  9.48709048e-02 1.02447952e-01 7.95837381e-02 4.38978190e-02
  1.77711905e-01 6.18334762e-02 3.20195714e-01 1.07569619e+00
  2.28762810e+00 2.77836190e+01 5.96571429e-03 2.43428762e-02
  2.93760571e-02 1.11640857e-02 1.85091810e-02 3.44532905e-03
  1.58322381e+01 2.57248095e+01 1.04444667e+02 7.72665714e+02
  1.30448048e-01 2.70945762e-01 2.81362200e-01 1.15717995e-01
  2.91449524e-01 8.46661905e-02]
 [1.10464045e+01 1.78439545e+01 7.08175909e+01 3.78081364e+02
  9.48637273e-02 7.95483636e-02 4.42052986e-02 2.24546000e-02
  1.78278636e-01 6.51847727e-02 2.84570455e-01 1.35245045e+00
  1.99521273e+00 1.92316273e+01 8.28141364e-03 2.24043682e-02
  2.75531936e-02 9.98128636e-03 2.27092273e-02 4.02266000e-03
  1.21529500e+01 2.35402273e+01 7.87425000e+01 4.55227273e+02
  1.29437273e-01 1.75772909e-01 1.55314477e-01 6.63507818e-02
  2.74650455e-01 8.20724545e-02]
 [2.06012500e+01 2.15840000e+01 1.36725000e+02 1.32745500e+03
  1.01111750e-01 1.50052000e-01 1.94178500e-01 1.11247500e-01
  1.89997500e-01 6.00312500e-02 7.77465000e-01 1.50239500e+00
  5.79030000e+00 1.01433750e+02 8.19210000e-03 3.62009000e-02
  5.20212500e-02 1.87228000e-02 2.51107500e-02 4.55777500e-03
  2.33725000e+01 2.74065000e+01 1.57325000e+02 1.68880000e+03
  1.30218500e-01 2.90436500e-01 3.98780000e-01 1.83888000e-01
  2.85612500e-01 7.76002500e-02]]

and the cluster assignments are:
[0 0 0 2 3 1 0 1 1 1 0 0 3 1 1 1 1 0 0 1 1 2 1 0 0 0 1 3 0 3 0 1 0 0 0 0 1
 2 1 1 1 2 0 1 1 0 2 1 2 1 2 1 2 3 1 2 0 1 1 2 2 2 1 2 1 1 2 2 2 2 0 2 0 1
 1 0 1 0 3 1 2 1 3 3 2 0 1 0 2 1 1 1 1 1 1 0 2 2 2 1 1 2 2 2 2 1 2 2 0 2 2
 2 1 2 2 2 2 1 0 3 2 0 3 1 1 1 1 0 1 3 2 0 0 1 0 1 2 2 1 2 2 0 2 1 2 2 2 1
 1 1 1 2 2 2 1 2 0 1 2 2 2 3 0 2 3 1 2 0 0 1 2 1 1 2 2 2 2 1 1 2 3 0 0 2 1
 2 0 2 2 2 1 1 2 1 1 1 1 3 0 1 1 0 3 1 1 1 2 0 1 1 3 2 3 3 1 1 2 2 0 0 1 1
```

```
 2 1 1 1 2 1 1 1 0 2 2 0 2 1 0 3 1 0 1 2 2 1 3 2 1 1 2 2 0 2 0 0 0 1 0 1 0
 1 0 0 0 1 0 0 2 1 1 2 1 2 0 2 0 2 2 3 1 1 0 2 0 1 1 2 2 2 2 2 1 1 1 2 2 1
 2 2 1 2 0 2 3 2 2 2 1 2 1 1 2 1 1 2 2 2 2 0 2 2 2 3 1 0 2 2 1 2 0 1 1 1 2
 2 2 0 2 0 2 0 1 2 2 3 2 2 2 1 2 2 2 1 0 1 2 2 1 1 2 2 2 1 2 1 1 3 3 1 0 0
 0 1 3 0 1 1 2 1 1 2 2 2 2 2 1 1 2 1 2 3 2 2 0 0 2 1 1 1 2 2 0 2 1 2 2 2 1
 1 0 1 2 2 2 1 1 2 2 0 2 2 2 1 2 1 2 2 2 2 2 2 1 2 3 0 1 1 1 1 1 1 2 0 1 2
 0 2 0 1 1 0 2 3 2 1 1 1 2 1 1 2 0 0 1 2 1 1 1 2 0 2 2 2 1 2 2 1 1 1 2 1 2
 1 1 1 1 2 1 0 2 0 1 3 0 2 1 1 1 1 0 3 1 1 2 0 2 2 2 2 1 1 2 1 1 1 1 2 0 3
 1 1 2 3 2 1 2 2 1 2 1 2 2 2 1 3 2 3 1 2 2 2 2 1 1 1 1 1 2 2 2 2 2 2 1 2 1
 2 2 2 1 2 1 2 1 3 3 3 0 0 2]


When k = 5, the centroids are:
[[9.91444681e+00 1.81442553e+01 6.33778723e+01 3.00280851e+02
  9.71493617e-02 8.00557447e-02 5.01998085e-02 2.05064468e-02
  1.86670213e-01 6.84570213e-02 3.07485106e-01 1.65032340e+00
  2.06835532e+00 1.88682128e+01 9.89636170e-03 2.51836383e-02
  3.92240638e-02 1.11818511e-02 2.53246809e-02 5.22582979e-03
  1.09011277e+01 2.36212766e+01 7.01338298e+01 3.62121277e+02
  1.32285957e-01 1.61085745e-01 1.54981702e-01 5.81372340e-02
  2.73925532e-01 8.59040426e-02]
 [1.22568966e+01 1.80093966e+01 7.87550862e+01 4.62145690e+02
  9.30346552e-02 7.84006897e-02 4.36533276e-02 2.59744310e-02
  1.73820690e-01 6.22196552e-02 2.83616379e-01 1.18059828e+00
  2.00751638e+00 2.11018707e+01 6.78400000e-03 1.95304397e-02
  2.31006284e-02 9.75981034e-03 2.06669828e-02 3.22615086e-03
  1.35364655e+01 2.38965517e+01 8.77768103e+01 5.59591379e+02
  1.26614397e-01 1.84831638e-01 1.67751888e-01 7.79932845e-02
  2.76490517e-01 7.89426724e-02]
 [8.52250000e+00 1.76015000e+01 5.43680000e+01 2.19720000e+02
  9.80675000e-02 9.08395000e-02 4.61210500e-02 1.43068500e-02
  1.87240000e-01 7.39020000e-02 3.08230000e-01 1.37852000e+00
  2.19670000e+00 1.61834500e+01 1.17647000e-02 3.04406000e-02
  3.24495500e-02 9.08700000e-03 2.74645000e-02 6.14990000e-03
  9.43830000e+00 2.21265000e+01 6.08965000e+01 2.68875000e+02
  1.37908500e-01 1.79224000e-01 1.36583500e-01 4.03495000e-02
  2.81485000e-01 9.28875000e-02]
 [1.11840625e+01 1.77878125e+01 7.17253125e+01 3.83421875e+02
  9.42537500e-02 7.76015625e-02 4.16369172e-02 2.26815625e-02
  1.75962500e-01 6.34951562e-02 2.60684375e-01 1.32371406e+00
  1.83628437e+00 1.79905625e+01 7.75795312e-03 2.20376250e-02
  2.45098234e-02 9.72442187e-03 2.16942187e-02 3.48779688e-03
  1.22562500e+01 2.37664062e+01 7.97057812e+01 4.58042187e+02
  1.28247656e-01 1.82069687e-01 1.57404891e-01 6.89443750e-02
  2.75123438e-01 7.99278125e-02]
 [1.63491304e+01 2.03213975e+01 1.07261646e+02 8.57069565e+02
  9.77558075e-02 1.23383944e-01 1.22721894e-01 6.86968944e-02
  1.83658385e-01 6.13513975e-02 4.97960248e-01 1.13536304e+00
  3.53803385e+00 5.63419565e+01 6.28087267e-03 2.80393758e-02
  3.54245373e-02 1.31994224e-02 1.91404286e-02 3.70570124e-03
  1.92590683e+01 2.72191304e+01 1.28056304e+02 1.19387360e+03
  1.34928571e-01 3.11889348e-01 3.58156227e-01 1.49726211e-01
  3.00832298e-01 8.57055901e-02]]

and the cluster assignments are:
[4 4 4 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 0 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 1 4 4 4 3 4 4 4 4 4 2 4 1 4 1 4 4 1 4 4 4 2 0 2 4 0 4 4 0 3 0 1 4 2 4 4
 1 4 4 4 4 1 1 1 4 4 1 4 4 4 1 4 4 4 4 4 4 4 1 0 1 4 4 2 1 0 0 4 1 1 4 1 0
 1 4 0 2 1 2 4 4 4 1 4 4 4 4 4 4 4 4 1 4 4 4 4 4 1 3 4 3 0 4 3 4 3 1 1 4
 4 4 1 2 0 3 4 1 4 4 1 3 1 4 4 1 4 4 3 4 4 4 1 4 4 3 3 2 0 4 1 1 4 4 4 3 4
 3 4 1 3 1 4 1 0 4 4 1 4 4 4 4 1 4 4 4 4 4 0 4 4 4 4 1 4 4 4 4 1 0 4 4 4 4
 0 4 4 4 0 4 1 4 4 3 3 4 0 4 4 4 4 4 4 1 3 4 4 0 4 4 3 3 4 1 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 3 4 1 3 4 3 4 0 4 3 3 4 4 4 4 1 4 4 1 1 1 1 3 3 4 4 1 1 1 4
 3 1 4 0 4 1 4 0 3 3 4 2 4 4 3 4 1 3 2 1 1 4 2 1 0 4 1 4 1 1 4 1 4 4 4 4 3
 3 1 4 1 4 0 4 4 0 3 4 1 0 1 4 3 1 1 4 4 4 3 1 1 4 2 0 1 4 1 4 4 4 4 1 4 4
 4 4 4 4 4 4 0 4 4 3 3 3 1 1 4 4 1 4 3 4 0 2 4 4 1 4 4 1 3 1 4 1 4 1 1 3 4
 4 4 1 1 3 0 4 4 1 0 4 1 3 1 4 3 4 0 0 3 3 3 1 4 1 4 4 4 4 1 4 4 4 3 4 4 0
 4 1 4 4 4 4 1 4 1 4 1 4 1 4 4 0 4 4 4 3 4 4 4 0 4 1 0 1 4 1 3 1 4 4 3 4 1
 4 4 4 4 1 4 4 1 4 1 4 4 1 1 4 4 4 4 1 4 2 0 1 3 4 4 3 4 4 4 4 3 4 4
 4 1 0 4 3 4 0 2 4 1 4 1 1 1 4 4 3 4 4 3 2 2 3 4 4 4 4 4 0 0 0 3 3 3 1 2 1
 0 0 0 4 3 4 3 4 4 4 4 4 4 2]


When k = 6, the centroids are:
[[1.55863158e+01 1.99178947e+01 1.02098105e+02 7.55688421e+02
  9.80267368e-02 1.21732316e-01 1.15168947e-01 6.34394737e-02
  1.85509474e-01 6.15702105e-02 4.19084211e-01 1.06367053e+00
  2.96353684e+00 4.12287368e+01 6.34540000e-03 2.72663263e-02
  3.53423684e-02 1.31552105e-02 1.89480632e-02 3.49038737e-03
  1.82547368e+01 2.71502105e+01 1.21162105e+02 1.02814632e+03
  1.37486105e-01 3.23767263e-01 3.70383789e-01 1.50834000e-01
  3.13362105e-01 8.71746316e-02]
 [1.89580000e+01 2.20505714e+01 1.25287143e+02 1.11868429e+03
  1.00934143e-01 1.45814857e-01 1.69363000e-01 9.65170000e-02
  1.93317143e-01 6.08728571e-02 6.75890000e-01 1.29106571e+00
  4.79855714e+00 8.04302857e+01 6.65897143e-03 3.24334429e-02
  4.29055714e-02 1.62123857e-02 2.12532857e-02 4.15674286e-03
  2.26651429e+01 2.90467143e+01 1.51354286e+02 1.56951429e+03
  1.38891429e-01 3.44308571e-01 4.33510000e-01 1.83958571e-01
  3.09655714e-01 8.56488571e-02]
 [2.42315385e+01 2.27000000e+01 1.61176923e+02 1.85638462e+03
  1.01446923e-01 1.61216923e-01 2.27800000e-01 1.35273077e-01
  1.81169231e-01 5.80138462e-02 1.24866154e+00 1.13635385e+00
```

```
  8.91046154e+00 2.09633846e+02 6.42192308e-03 2.95492308e-02
  3.99623077e-02 1.56461538e-02 1.86715385e-02 3.43969231e-03
  3.10161538e+01 3.02569231e+01 2.08676923e+02 2.97169231e+03
  1.38492308e-01 3.60700000e-01 4.67676923e-01 2.29123077e-01
  2.82615385e-01 8.08823077e-02]
 [2.05108000e+01 2.16268000e+01 1.36540000e+02 1.31602400e+03
  1.06411200e-01 1.70701600e-01 2.21052000e-01 1.19939600e-01
  1.96216000e-01 6.18932000e-02 8.37464000e-01 1.13392400e+00
  5.85844000e+00 1.12390000e+02 6.74552000e-03 3.61324000e-02
  4.93444000e-02 1.59316000e-02 1.90560000e-02 4.15308000e-03
  2.61104000e+01 2.89276000e+01 1.75184000e+02 2.09400000e+03
  1.47364000e-01 4.18052000e-01 5.51704000e-01 2.22628000e-01
  3.22888000e-01 9.12496000e-02]
 [1.08045691e+01 1.79257447e+01 6.92488298e+01 3.61061702e+02
  9.54575532e-02 8.04265957e-02 4.48460410e-02 2.22685426e-02
  1.79607447e-01 6.58394149e-02 2.87899468e-01 1.38491596e+00
  2.01335106e+00 1.90514734e+01 8.60159574e-03 2.31447181e-02
  2.85773548e-02 1.01758138e-02 2.30759043e-02 4.17844787e-03
  1.19013777e+01 2.36825000e+01 7.71360638e+01 4.35812766e+02
  1.30652021e-01 1.76901330e-01 1.56254229e-01 6.58537766e-02
  2.75530319e-01 8.28544149e-02]
 [1.33237640e+01 1.87317978e+01 8.61425843e+01 5.48429213e+02
  9.28424719e-02 9.05328652e-02 6.07392978e-02 3.43175730e-02
  1.73588202e-01 6.14733146e-02 2.92827528e-01 1.10944607e+00
  2.09295337e+00 2.40914663e+01 6.00085955e-03 2.24593371e-02
  2.61850949e-02 1.01834101e-02 1.87829888e-02 3.38567584e-03
  1.48482022e+01 2.48817978e+01 9.73733146e+01 6.77511798e+02
  1.26331854e-01 2.32693652e-01 2.25252202e-01 9.59539382e-02
  2.81246067e-01 8.19034831e-02]]

and the cluster assignments are:
[3 3 1 4 1 5 1 0 5 5 0 0 1 0 5 0 0 0 3 5 5 4 0 2 3 1 0 1 0 0 1 5 1 1 0 0 5
 5 5 5 4 1 0 5 1 4 5 5 5 4 5 4 1 0 4 3 0 5 4 4 4 0 4 5 0 4 4 4 5 1 4 1 5
 5 0 5 1 1 5 4 5 2 1 5 1 5 1 5 5 5 0 5 5 0 1 4 4 4 4 0 0 4 4 4 4 5 4 4 3 4 4
 4 5 4 4 5 4 0 0 0 4 1 3 5 5 5 5 1 5 1 4 0 0 0 1 5 4 4 0 4 4 0 4 5 4 4 5 0
 5 5 5 4 4 4 5 5 1 0 4 4 4 1 3 5 2 5 4 0 1 5 4 0 0 4 4 4 4 0 5 5 2 3 0 4 0
 4 1 4 4 4 5 5 4 5 5 5 5 0 1 0 5 0 3 0 5 0 4 0 5 0 1 4 2 0 5 5 4 4 3 3 5 5
 4 0 5 0 4 0 5 5 0 4 4 3 4 5 2 1 5 1 5 5 4 5 1 4 5 5 4 4 3 4 3 0 3 5 3 0 0
 0 1 0 0 0 1 2 4 5 5 4 5 4 3 4 1 4 4 1 5 5 1 4 1 0 5 5 4 5 4 4 5 0 5 4 5 5
 4 4 5 4 3 5 1 4 4 4 5 4 5 5 4 0 5 4 4 5 4 1 4 4 4 1 5 3 5 5 5 4 0 0 0 5 4
 4 4 1 5 1 4 2 0 4 4 1 4 4 4 0 4 4 4 0 2 0 4 5 5 5 4 4 5 5 5 0 5 1 1 5 2 3
 0 0 1 3 5 0 4 5 5 4 4 4 4 5 5 5 4 5 4 1 4 4 0 3 4 5 5 5 4 4 1 5 5 5 4 4 0
 5 1 5 4 4 4 0 0 4 4 1 5 4 4 5 4 5 4 4 4 4 4 5 0 4 1 1 5 0 5 5 5 5 4 0 5 4
 1 4 1 0 0 3 4 1 4 5 5 5 4 5 5 4 1 2 5 4 5 5 5 4 1 4 4 5 0 5 4 5 5 5 4 0 4
 5 5 5 0 5 0 1 4 0 5 0 1 4 5 5 5 5 1 1 0 5 5 2 4 4 4 4 0 0 4 5 5 0 0 4 1 1
 5 5 4 2 4 5 4 4 5 5 5 4 4 4 5 1 4 1 5 4 4 4 4 5 0 5 5 5 4 4 4 4 4 4 5 4 5
 4 4 4 5 4 5 4 0 1 3 1 0 1 4]


When k = 7, the centroids are:
[[8.71415385e+00 1.72869231e+01 5.56203846e+01 2.30553846e+02
  9.74557692e-02 8.91150000e-02 5.36742692e-02 1.54532692e-02
  1.87842308e-01 7.33019231e-02 3.07900000e-01 1.50781538e+00
  2.15876923e+00 1.64877308e+01 1.09951923e-02 3.10556923e-02
  4.13492692e-02 1.02191538e-02 2.88815385e-02 5.82765385e-03
  9.60842308e+00 2.17973077e+01 6.19557692e+01 2.79176923e+02
  1.34688462e-01 1.81895000e-01 1.73030769e-01 4.58996154e-02
  2.85473077e-01 9.17442308e-02]
 [2.31418182e+01 2.27081818e+01 1.54581818e+02 1.68931818e+03
  1.05533182e-01 1.75010000e-01 2.46609091e-01 1.36777273e-01
  1.89577273e-01 5.96786364e-02 1.11561364e+00 1.24347273e+00
  8.01477273e+00 1.77216364e+02 6.98304545e-03 3.65136364e-02
  4.94527273e-02 1.60586364e-02 1.99854545e-02 3.85600000e-03
  2.93990909e+01 3.01554545e+01 1.99195455e+02 2.67500000e+03
  1.42559091e-01 3.98722727e-01 5.35881818e-01 2.30145455e-01
  2.98822727e-01 8.34613636e-02]
 [1.26503623e+01 1.79758696e+01 8.13502899e+01 4.93083333e+02
  9.16384058e-02 7.86634783e-02 4.46079565e-02 2.69401522e-02
  1.72857246e-01 6.13634783e-02 2.74703623e-01 1.13326377e+00
  1.94775290e+00 2.11517246e+01 6.34836232e-03 1.94039855e-02
  2.25894848e-02 9.41640580e-03 1.97656522e-02 3.10364130e-03
  1.39268841e+01 2.37302174e+01 9.05647101e+01 5.94002174e+02
  1.24037826e-01 1.90557391e-01 1.71489812e-01 7.89674710e-02
  2.76900725e-01 7.81404348e-02]
 [1.48621429e+01 1.95977551e+01 9.72074490e+01 6.84465306e+02
  9.86887755e-02 1.18722653e-01 1.06442449e-01 5.77154082e-02
  1.84036735e-01 6.24872449e-02 3.73345918e-01 1.06780510e+00
  2.67615306e+00 3.41547959e+01 6.48238776e-03 2.75628469e-02
  3.46627449e-02 1.30658571e-02 1.91473061e-02 3.72447755e-03
  1.71612245e+01 2.65138776e+01 1.14115306e+02 9.02895918e+02
  1.36865510e-01 3.11894694e-01 3.47059490e-01 1.40759388e-01
  3.04588776e-01 8.76646939e-02]
 [1.85378761e+01 2.15801770e+01 1.22333628e+02 1.07520265e+03
  1.00112301e-01 1.42169646e-01 1.60748407e-01 9.21464602e-02
  1.91737168e-01 6.07131858e-02 6.58974336e-01 1.21544336e+00
  4.63015929e+00 7.79939823e+01 6.47119469e-03 3.11249204e-02
  4.07986726e-02 1.54774690e-02 2.03118584e-02 3.95092035e-03
  2.24440708e+01 2.88039823e+01 1.49470796e+02 1.55230088e+03
  1.39578230e-01 3.48060708e-01 4.29768142e-01 1.83564779e-01
  3.15304425e-01 8.66069027e-02]
 [1.37147917e+01 1.96195833e+01 8.89547917e+01 5.80504167e+02
  9.32902083e-02 1.00370625e-01 7.31112083e-02 3.83433958e-02
  1.71370833e-01 6.20237500e-02 2.93468750e-01 1.10628750e+00
```

```
  2.10762292e+00 2.52431250e+01 5.53175000e-03 2.46980000e-02
  2.88122917e-02 1.04014375e-02 1.70940000e-02 3.60154167e-03
  1.53797917e+01 2.64939583e+01 1.01192708e+02 7.26502083e+02
  1.27421458e-01 2.76886458e-01 2.76117500e-01 1.09161042e-01
  2.80472917e-01 8.74033333e-02]
 [1.08662177e+01 1.81066129e+01 6.96552419e+01 3.62800806e+02
  9.56870968e-02 7.92699194e-02 4.39089815e-02 2.25582661e-02
  1.79391129e-01 6.51888710e-02 2.81825806e-01 1.40603065e+00
  1.95893468e+00 1.88166048e+01 8.53788710e-03 2.26192258e-02
  2.80398847e-02 1.02005887e-02 2.24041935e-02 4.11548387e-03
  1.19552419e+01 2.40362903e+01 7.74983871e+01 4.37137097e+02
  1.31136452e-01 1.74932500e-01 1.53969218e-01 6.67731452e-02
  2.73407258e-01 8.21549194e-02]]

and the cluster assignments are:
[4 4 4 2 4 5 4 3 5 5 4 4 4 3 5 3 3 4 1 5 2 0 3 1 4 4 3 4 4 4 4 3 4 4 4 4 3
 2 5 5 5 6 4 3 5 4 0 5 2 5 6 2 2 4 3 6 4 3 2 0 6 0 3 0 3 3 6 6 0 2 4 0 4 3
 2 4 2 4 4 2 6 2 1 4 2 4 3 4 2 3 3 3 3 5 3 4 6 6 6 3 3 0 2 6 6 3 6 2 1 6 6
 2 5 6 0 2 0 3 4 4 6 4 1 5 2 5 3 4 3 4 2 3 4 3 4 2 2 6 3 6 6 4 6 2 6 6 2 3
 3 5 2 0 6 6 2 2 4 3 2 6 2 4 1 2 1 3 6 4 4 3 2 3 3 6 6 0 6 3 2 2 1 4 4 6 3
 6 4 6 6 2 5 2 0 5 3 2 3 4 4 3 2 4 1 3 2 3 6 4 2 3 4 2 1 3 3 5 2 6 4 1 5 2
 6 3 5 3 6 3 2 5 4 6 6 4 6 5 1 4 5 4 2 2 6 5 4 6 2 2 6 6 4 6 4 4 4 3 4 3 4
 3 4 4 4 3 4 1 6 2 2 6 5 6 1 6 4 6 6 4 5 2 4 2 4 3 2 2 2 2 6 6 3 3 2 2 2 2
 6 2 3 6 4 2 4 6 6 6 2 0 5 2 6 3 2 6 0 2 2 4 0 2 6 4 2 4 2 2 5 2 3 3 3 2 6
 6 2 4 2 4 6 1 3 6 6 4 6 6 2 3 6 6 2 3 1 3 6 2 2 5 0 6 2 2 2 3 2 4 4 2 1 1
 4 3 4 4 2 3 6 2 2 6 6 6 6 2 2 3 2 5 6 4 6 0 4 1 2 2 2 2 6 2 4 2 2 2 2 6 3
 2 4 2 6 6 0 3 3 2 6 4 2 6 6 3 6 5 6 6 6 6 6 2 3 2 4 4 3 3 2 5 5 5 6 4 5 6
 4 2 4 3 3 4 6 4 2 5 2 5 6 2 2 6 4 1 5 6 2 5 2 6 4 6 6 2 3 2 6 2 3 5 6 3 2
 3 2 2 3 2 3 4 2 3 2 4 4 2 2 3 2 2 4 4 3 3 2 1 0 6 2 6 3 3 6 3 3 3 3 6 4 4
 2 2 0 1 6 5 6 0 5 2 2 2 2 2 5 4 6 4 5 6 0 0 6 3 3 2 2 5 6 6 6 6 6 6 2 0 2
 6 6 6 5 6 5 6 3 4 4 4 4 4 0]
```

3. Plot K-means distortion for values of K ranging between 2 and 7

In [7]:
```
def k_means_distortion(training_set, c, centroids):
    num_points = training_set.shape[0]

    errors_sum = 0
    for i in range(num_points):
        errors_sum += (np.linalg.norm(training_set[i] - centroids[c[i]]))**2

    error = errors_sum / num_points

    return error
```

In [8]:
```
# sanity check to check that my k_means_distortion implementation returns a result
centroids, cluster_assignment = k_means(test_training_set, 3)
error = k_means_distortion(test_training_set, cluster_assignment, centroids)
print(error)
```

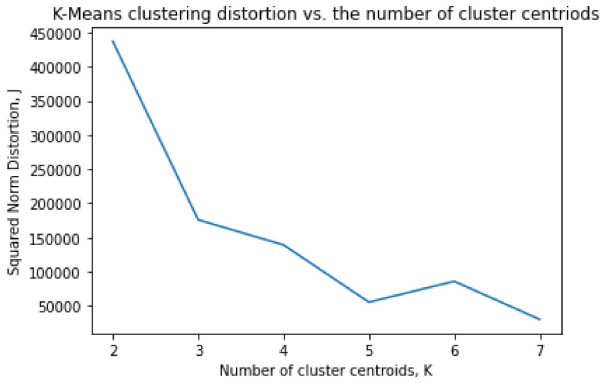
```
0.04657992134047305
```

In [17]:
```
x_axis = np.arange(2, 8, 1) # values of k
y_axis = np.empty(x_axis.shape)

print("k | error")
print("--|------")
for k in range(2, 8):
    centroids, cluster_assignment = k_means(training_set, k=k)
    error = k_means_distortion(training_set, cluster_assignment, centroids)
    y_axis[k-2] = error
    print("%d | %d" % (k, error))

plt.plot(x_axis, y_axis)
plt.xlabel('Number of cluster centroids, K')
plt.ylabel('Squared Norm Distortion, J')
plt.title("K-Means clustering distortion vs. the number of cluster centriods")
plt.show()
```

```
k | error
--|------
2 | 437074
3 | 175908
4 | 139120
5 | 55457
6 | 85947
7 | 30214
```



Note on the plot above: The distortion plot should be a smooth, monotonically decreasing curve that looks similar to the plot of y = 1/x for x > 0. However, when I run the code block above, my plots will sometimes have "bumps" in the distortion for higher values of K. An example of such a bump can be seen in the plot above when K = 6. I think these bumps occur because of noise caused by the randomness in my initialize_centroids() function.

4. Picking a value for K

Based on the plot above, pick K = 7.

This is the value of K for which the K-means algorithm has the lowest distortion. When the distortion is low, the error (distance between the cluster centroid and the points in the dataset that are assigned to that cluster centroid) is minimized. This means that the cluster centroid is a good fit for the points in the training set.

Problem 2: Lack of optimality of K-Means

From the hint in the question, let $K=2$ for the dataset, with $\mu_1=2$ $\mu_2=4$

$m=4$

$K=2$

Assume towards a contradiction that these μ will converge towards a globally optimal solution.

K-Means

1. Distance from $x[i]$ to $\mu[k]$, $i \in [1,4]$, $k \in [1,2]$. $d = \|x_i - \mu_k\|^2 = [\sqrt{(x_i-\mu_k)^2}]^2 = (x_i-\mu_k)^2$

$x[i]$	$\|x_i-\mu_1\|^2$	$\|x_i-\mu_2\|^2$
$i=1$	1	9
2	0	4
3	1	1
4	4	0

2. Cluster assignment step: $c_i = \min_k \{\|x_i-\mu_k\|^2\}$

	$i=1$	$i=2$	$i=3$	$i=4$
$c[i]$	1	1	1	2

3. Move centroids: μ_k' = avg. of all $x[i]$ assigned to cluster k.

k	avg. value of $x[i]$	μ_k'
1	$\frac{1}{3}(1+2+3)$	2
2	4	4

Since $\mu_1' = \mu_1 = 2$ and $\mu_2' = \mu_2 = 2$, we stop the algorithm.

4. Calculate the distortion J.

$$J(c^{(1)}=1, c^{(2)}=1, c^{(3)}=1, c^{(4)}=2, \mu_1=2, \mu_2=4) = \frac{1}{4}\sum_{i=1}^{4} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

$$= \frac{1}{4}\left[(1-2)^2 + (2-2)^2 + (3-2)^2 + (4-4)^2\right]$$

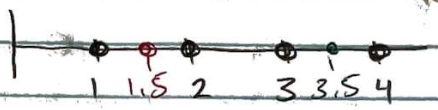
$$= \frac{1}{4}(1+0+1+0)$$

$$J = \frac{1}{2}$$

If $\mu_1=2$ and $\mu_2=4$ were the globally optimal solution, then $J=\frac{1}{2}$ should be the lowest value of the distortion for any other cluster centroids.

But we can achieve a lower distortion if we pick different initializations.

$\mu_1' = 2$
$\mu_2' = 3$



K-Means

1. Distance

i	$\|\|x_i - \mu_1\|\|^2$	$\|\|x_i - \mu_2\|\|^2$
1	1	4
2	0	1
3	1	0
4	4	1

i	$\|\|x_i - \mu_1\|\|^2$	$\|\|x_i - \mu_2\|\|^2$
1	0.25	6.25
2	0.25	2.25
3	2.25	0.25
4	6.25	0.25

2. Cluster assignment

	$i=1$	$i=2$	$i=3$	$i=4$
$c[i]$	1	1	2	2

	$i=1$	$i=2$	$i=3$	$i=4$
$c[i]$	1	1	2	2

3. Move centroid

k	avg. value of $x[i]$	μ_k''
1	$\frac{1}{2}(1+2) = 1.5$	1.5
2	$\frac{1}{2}(3+4) = 3.5$	3.5

Since $\mu_k'' \neq \mu_k'$, repeat the algorithm with the new values of μ_k

k	avg. value of $x[i]$	μ_k''
1	$\frac{1}{2}(1+2) = 1.5$	1.5
2	$\frac{1}{2}(3+4) = 3.5$	3.5

Since $\mu_k'' = \mu_k'$, we stop K-means.

4. Calculate the distortion, J'

$$J'(c^{(1)}=1, c^{(2)}=1, c^{(3)}=2, c^{(4)}=2, \mu_1=1.5, \mu_2=3.5) = \frac{1}{4}\sum_{i=1}^{4} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

$$= \frac{1}{4}\left[(1-1.5)^2 + (2-1.5)^2 + (3-3.5)^2 + (4-3.5)^2\right]$$

$$= \frac{1}{4}\left[4(0.5)^2\right]$$

$$= 0.25$$

$$J' = \frac{1}{4}.$$

Since $J' = \frac{1}{4} < J = \frac{1}{2}$, the cluster assignment $\mu_1' = 1.5$, $\mu_2' = 3.5$ is more optimal than $\mu_1 = 2$, $\mu_2 = 4$. But K-means converged for $\mu_1 = 2$, $\mu_2 = 4$, implying that this was a global optimum. This is a contradiction $\square$.

Problem 3: SVD

1. $A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 1 \end{bmatrix}$

$\downarrow$ $2R_1 - R_2 \to R_2$ to obtain leading 0 in second row

$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ Matrix is now in row echelon form

Since A in row echelon form has 2 non-zero rows, rank(A) = 2. □

2. $A^TA = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 8 & 3 \\ 8 & 13 & 5 \\ 3 & 5 & 2 \end{bmatrix}$

$A - \lambda I = \begin{bmatrix} 5 & 8 & 3 \\ 8 & 13 & 5 \\ 3 & 5 & 2 \end{bmatrix} - \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} = 0.$

$\begin{bmatrix} 5-\lambda & 8 & 3 \\ 8 & 13-\lambda & 5 \\ 3 & 5 & 2-\lambda \end{bmatrix} = 0$ let this matrix be B'

$\det(B') = (5-\lambda)[(13-\lambda)(2-\lambda)-25] - 8[8(2-\lambda)-15] + 3[40-3(13-\lambda)]$

$= (5-\lambda)(\lambda^2 - 15\lambda + 1) - 8(1-8\lambda) + 3(1+3\lambda)$

$= -\lambda^3 + 15\lambda^2 - \lambda + 5\lambda^2 - 75\lambda + 5 - 8 + 64\lambda + 3 + 9\lambda$

$= -\lambda^3 + 20\lambda^2 - 3\lambda$

$= -\lambda(\lambda^2 - 20\lambda + 3)$. $\lambda_0 = 0$. since rank(A) = 2.

$\lambda_1, \lambda_2 = \frac{20 \pm \sqrt{400 - 12}}{2} = 10 \pm \sqrt{97}.$

$\boxed{\sigma_0 = 0, \ \sigma_1 = \sqrt{10 + \sqrt{97}}, \ \sigma_2 = \sqrt{10 - \sqrt{97}}}$

3. From (2), the singular values of A are $\sigma_1 = \sqrt{10+\sqrt{97}}$ and $\sigma_2 = \sqrt{10-\sqrt{97}}$.

To find V, calculate the orthonormal set of eigenvectors of A^TA
From (2), the eigenvalues are $\lambda_1 = 10+\sqrt{97}$, $\lambda_2 = 10-\sqrt{97}$, $\lambda_3 = 0$.
Since A^TA is symmetrical, the eigenvectors will be orthogonal.

For $\lambda_1 = 10+\sqrt{97}$: $A^TA - (10+\sqrt{97})I = \begin{bmatrix} 5-(10+\sqrt{97}) & 8 & 3 \\ 8 & 13-(10+\sqrt{97}) & 5 \\ 3 & 5 & 2-(10+\sqrt{97}) \end{bmatrix}$

an eigenvector $\vec{v}_1$ exists such that $[A^TA - (10+\sqrt{97})I]\vec{v}_1 = 0$

$$\begin{bmatrix} 5-(10+\sqrt{97}) & 8 & 3 \\ 8 & 13-(10+\sqrt{97}) & 5 \\ 3 & 5 & 2-(10+\sqrt{97}) \end{bmatrix} \begin{bmatrix} v_{1,1} \\ v_{2,1} \\ v_{3,1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

which produces a system of equations:

$[5-(10+\sqrt{97})]v_{1,1} + 8v_{2,1} + 3v_{3,1} = 0$
$8v_{1,1} + [13-(10+\sqrt{97})]v_{2,1} + 5v_{3,1} = 0$
$3v_{1,1} + 5v_{2,1} + [2-(10+\sqrt{97})]v_{3,1} = 0$

Let $v_{1,1} = 1$. Then solving the system of equations yields

$v_{2,1} = \frac{49+5\sqrt{97}}{31+3\sqrt{97}}$, $v_{3,1} = \frac{18+2\sqrt{97}}{31+3\sqrt{97}}$ so $\vec{v}_1 = \begin{bmatrix} 1 \\ 49+5\sqrt{97}/31+3\sqrt{97} \\ 18+2\sqrt{97}/31+3\sqrt{97} \end{bmatrix}$

$\|v_1\| = \sqrt{v_{1,1}^2 + v_{2,1}^2 + v_{3,1}^2} \simeq 2.01$.

unit length vector $\vec{v}_1 = \begin{bmatrix} 0.499 \\ 0.809 \\ 0.311 \end{bmatrix}$

Doing the same process for $\lambda_2 = 10-\sqrt{97}$ yields

$\vec{v}_2 = \begin{bmatrix} 1 \\ -49+5\sqrt{97}/-31+3\sqrt{97} \\ -18+2\sqrt{97}/-31+3\sqrt{97} \end{bmatrix}$ $\xrightarrow[\text{vector}]{\text{unit length}}$ $\vec{v}_2 = \begin{bmatrix} 0.646 \\ -0.109 \\ -0.755 \end{bmatrix}$

$\|v_2\| = 1.547$.

Doing the same process with $\lambda_3 = 0$ yields

$$\vec{v}_3 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \xrightarrow[\text{vector}]{\text{unit length}} \vec{v}_3 = \begin{bmatrix} 1/\sqrt{3} \\ -1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix}$$

$$\text{So } V = \begin{bmatrix} 0.499 & 0.646 & 0.577 \\ 0.809 & -0.109 & -0.577 \\ 0.311 & -0.755 & 0.577 \end{bmatrix} \quad \text{(orthonormal)}$$

To find U_i we use the formula $u_i = \frac{1}{\sigma_i} A v_i$

$$u_1 = \frac{1}{\sqrt{10+\sqrt{97}}} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 1 \end{bmatrix} \begin{bmatrix} 0.499 \\ 0.809 \\ 0.311 \end{bmatrix} = \frac{1}{\sqrt{10+\sqrt{97}}} \begin{bmatrix} 0.499+1.618+0.311 \\ 0.998+2.427+0.311 \end{bmatrix}$$

$$= \begin{bmatrix} 0.545 \\ 0.839 \end{bmatrix}$$

$$u_2 = \frac{1}{\sqrt{10-\sqrt{97}}} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 1 \end{bmatrix} \begin{bmatrix} 0.646 \\ -0.109 \\ 0.755 \end{bmatrix} = \frac{1}{\sqrt{10-\sqrt{97}}} \begin{bmatrix} 0.646-0.218+0.755 \\ 1.292-0.327-0.755 \end{bmatrix}$$

$$= \begin{bmatrix} -0.838 \\ 0.545 \end{bmatrix}$$

$$U = \begin{bmatrix} 0.545 & -0.838 \\ 0.839 & 0.545 \end{bmatrix}$$

$$V = \begin{bmatrix} 0.499 & 0.646 & 0.577 \\ 0.809 & -0.109 & -0.577 \\ 0.311 & -0.755 & 0.577 \end{bmatrix} \qquad U = \begin{bmatrix} 0.545 & -0.838 \\ 0.839 & 0.545 \end{bmatrix}$$

Σ is a 2×3 matrix where the singular values are on the diagonal.

$$\Sigma = \begin{bmatrix} \sqrt{10+\sqrt{97}} & 0 & 0 \\ 0 & \sqrt{10-\sqrt{97}} & 0 \end{bmatrix}$$