ACM 104 **October 24, 2023**

Problem Set 3

Problem 1. (10 points) Inner Products vs Norms

part (a)
Yes, it is possible to reconstruct the inner product $\langle \cdot, \cdot \rangle$ from the norm of V. By the inner product axioms we have (note $u, v \in V$):

$$\begin{aligned} ||u+v||^2 &= \langle u+v, u+v \rangle \\ &= \langle u, u+v \rangle + \langle v, u+v \rangle \\ &= \langle u, u \rangle + \langle u, v \rangle + \langle v, u \rangle + \langle v, v \rangle \\ &= \langle u, u \rangle + 2\langle u, v \rangle + \langle v, v \rangle \\ &= ||u||^2 + 2\langle u, v \rangle + ||v||^2 \end{aligned}$$

Solving for $\langle u, v \rangle$ yields

$$\langle u, v \rangle = \frac{1}{2}\left(||u+v||^2 - ||u||^2 - ||v||^2\right)$$

Hence, as shown above, the inner product can be constructed, if we know the norm of any vector in V.

part (b)
No, there are no two distinct inner products that induce the same norm.

Proof:
Assume that there are two distinct inner products $\langle \cdot, \cdot \rangle_1 \neq \langle \cdot, \cdot \rangle_2$ that induce the same norm $|| \cdot ||_1 = || \cdot ||_2$. From part A we have:

$$||u+v||_1^2 = \langle u, u \rangle_1 + 2\langle u, v \rangle_1 + \langle v, v \rangle_1$$
$$||u+v||_2^2 = \langle u, u \rangle_2 + 2\langle u, v \rangle_2 + \langle v, v \rangle_2$$

Subtracting one equation from the other we have

$$||u+v||_1^2 - ||u+v||_2^2 = \langle u, u \rangle_1 + 2\langle u, v \rangle_1 + \langle v, v \rangle_1 - (\langle u, u \rangle_2 + 2\langle u, v \rangle_2 + \langle v, v \rangle_2)$$

$$||u+v||_1^2 - ||u+v||_2^2 = (\langle u, u \rangle_1 - \langle u, u \rangle_2) + (2\langle u, v \rangle_1 - 2\langle u, v \rangle_2) + (\langle v, v \rangle_1 - \langle v, v \rangle_2)$$

$$||u+v||_1^2 - ||u+v||_2^2 = (||u||_1^2 - ||u||_2^2) + (2\langle u, v \rangle_1 - 2\langle u, v \rangle_2) + (||v||_1^2 - ||v||_2^2)$$

If they induce the same norm, then:

$$0 = 2\langle u, v \rangle_1 - 2\langle u, v \rangle_2 \Rightarrow \langle u, v \rangle_1 - \langle u, v \rangle_2 = 0 \Rightarrow \langle u, v \rangle_1 = \langle u, v \rangle_2$$

Hence we have a contradiction. Thus, there are no two distinct inner products that induce the same norm

Problem 2. (10 points) Continuously Differentiable Functions

part (a)
$\langle f, g\rangle_1$ is not an innner product. This is because $\langle f, g\rangle_1$ fails to satisfy the positive definite axiom of an inner product. For example, let $f(x) = -1$. So we have $f'(x) = 0$, hence:

$$\langle f, f\rangle_1 = \int_0^1 f'(x)f'(x)\, dx = \int_0^1 0\, dx = 0$$

In other words $\langle f, f\rangle_1 = 0$ does not imply that f is the zero vector. In contrast, $\langle f, g\rangle_2$ does define an inner product on $C^1[0, 1]$ as it satisfies all three inner product axioms (bilinear, symmetric, positive-definite).

part (b)
Inner product:

$$\langle f, g\rangle = \int_0^1 (f(x)g(x) + f'(x)g'(x))\, dx$$

Cauchy-Scharz inequality:

$$|\langle f, g\rangle| \le \sqrt{\langle f, f\rangle} \cdot \sqrt{\langle g, g\rangle}$$

$$\left|\int_0^1 (f(x)g(x) + f'(x)g'(x))\, dx\right| \le \sqrt{\int_0^1 (f(x)f(x) + f'(x)f'(x))\, dx} \cdot \sqrt{\int_0^1 (g(x)g(x) + g'(x)g'(x))\, dx}$$

Triangle inequality:

$$\sqrt{\langle f + g, f + g\rangle} \le \sqrt{\langle f, f\rangle} + \sqrt{\langle g, g\rangle}$$

$$\sqrt{\int_0^1 (\,(f(x) + g(x))^2 + (f'(x) + g'(x))^2\,)\, dx} \le \sqrt{\int_0^1 (f(x)f(x) + f'(x)f'(x))\, dx} + \sqrt{\int_0^1 (g(x)g(x) + g'(x)g'(x))\, dx}$$

part (c)
Given $f(x) = 1$ and $g(x) = e^x$ (so $f'(x) = 0$ and $g'(x) = e^x$) we have:

$$\langle f, g\rangle = \int_0^1 (f(x)g(x) + f'(x)g'(x))\, dx = \int_0^1 e^x\, dx = e - 1$$

$$\langle f, f\rangle = \int_0^1 (f(x)f(x) + f'(x)f'(x))\, dx = \int_0^1 1\, dx = 1$$

$$\langle g, g\rangle = \int_0^1 (g(x)g(x) + g'(x)g'(x))\, dx = \int_0^1 2e^{2x}\, dx = e^2 - 1$$

By substitution we have:

$$cos(\theta) = \frac{\langle f, g\rangle}{\sqrt{\langle f, f\rangle} \cdot \sqrt{\langle g, g\rangle}} = \frac{e - 1}{\sqrt{1} \cdot \sqrt{e^2 - 1}} = \sqrt{\frac{e - 1}{e + 1}}$$

Solving for theta gives $\theta \approx 0.8233$ radians.

Problem 3. (10 points) The K-means Algorithm for Clustering

See code attached.

Problem 4. (10 points) Gram Matrices

part (a)

$$G = \begin{bmatrix} \langle 1,1\rangle & \langle 1,e^x\rangle & \langle 1,e^{2x}\rangle \\ \langle e^x,1\rangle & \langle e^x,e^x\rangle & \langle e^x,e^{2x}\rangle \\ \langle e^{2x},1\rangle & \langle e^{2x},e^x\rangle & \langle e^{2x},e^{2x}\rangle \end{bmatrix} = \begin{bmatrix} \int_0^1 1\,dx & \int_0^1 e^x dx & \int_0^1 e^{2x} dx \\ \int_0^1 e^x dx & \int_0^1 e^{2x} dx & \int_0^1 e^{3x} dx \\ \int_0^1 e^{2x} dx & \int_0^1 e^{3x} dx & \int_0^1 e^{4x} dx \end{bmatrix}$$

Solving the definite integrals we have:

$$G = \begin{bmatrix} 1 & e-1 & \frac{1}{2}(e^2-1) \\ e-1 & \frac{1}{2}(e^2-1) & \frac{1}{3}(e^3-1) \\ \frac{1}{2}(e^2-1) & \frac{1}{3}(e^3-1) & \frac{1}{4}(e^4-1) \end{bmatrix}$$

part (b)
Because the functions $1, e^x, e^{2x}$ are linearly independent, then G must be positive definite. Note that the functions $1, e^x, e^{2x}$ are linearly independent as you cannot express any one of them as a linear combination of the other two.

part (c)
Using the inner product from part 2 we have:

$$\langle f, g\rangle = \int_0^1 \left(f(x)g(x) + f'(x)g'(x)\right) dx$$

$$G = \begin{bmatrix} \langle 1,1\rangle & \langle 1,e^x\rangle & \langle 1,e^{2x}\rangle \\ \langle e^x,1\rangle & \langle e^x,e^x\rangle & \langle e^x,e^{2x}\rangle \\ \langle e^{2x},1\rangle & \langle e^{2x},e^x\rangle & \langle e^{2x},e^{2x}\rangle \end{bmatrix} = \begin{bmatrix} \int_0^1 1\,dx & \int_0^1 e^x dx & \int_0^1 e^{2x} dx \\ \int_0^1 e^x dx & \int_0^1 2e^{2x} dx & \int_0^1 3e^{3x} dx \\ \int_0^1 e^{2x} dx & \int_0^1 3e^{3x} dx & \int_0^1 5e^{4x} dx \end{bmatrix}$$

Solving the definite integrals:

$$G = \begin{bmatrix} 1 & e-1 & \frac{1}{2}(e^2-1) \\ e-1 & (e^2-1) & (e^3-1) \\ \frac{1}{2}(e^2-1) & (e^3-1) & \frac{5}{4}(e^4-1) \end{bmatrix}$$

Note that G is still positive definite for the same reasoning as in part a (the functions $1, e^x, e^{2x}$ are linearly independent).

part (d)
No, you cannot find $\langle\cdot,\cdot\rangle_1$ and $\langle\cdot,\cdot\rangle_2$ such that one of the Gram matrices is positive definite and the other one is not. This is because the functions $1, e^x, e^{2x}$ are linearly independent regardless of what the inner product is. Hence, any Gram Matrix associated with these functions must be positive definite.

Problem 5. (10 points) Gram Matrices for Text Classification

See code attached.

ACM/IDS 104 - Problem Set 3 - MATLAB Problems

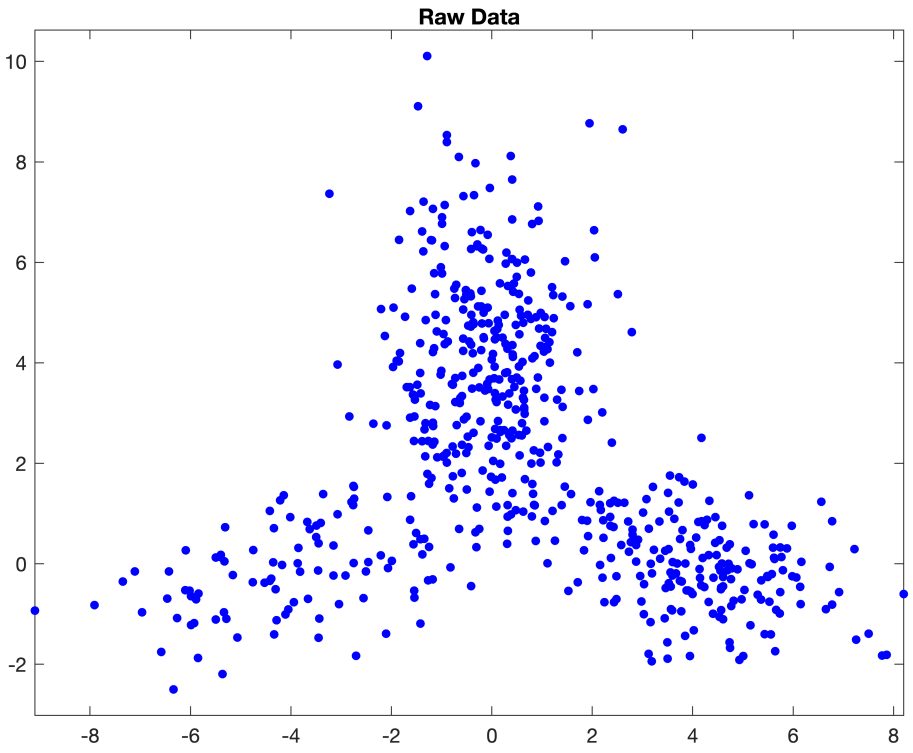
Before writing your MATLAB code, it is always good practice to get rid of any leftover variables and figures from previous scripts.

```
clc; clear; close all;
```

Problem 3 (10 points) The K-Means Algorithm for Clustering

In this problem, we will be implementing the K-Means Algorithm for clustering vectors in $\mathbb{R}^2$. To start, let us load and visualize our dataset.

```
load clustering_data.mat
figure;
plot(x(:,1),x(:,2),'.b','MarkerSize',15);
title("Raw Data")
axis equal;
```




```
rng(2016); % for reproducibility
```

Part (a) The Algorithm

In this part, we will implement the algorithm for $K = 3$, which seems to be a reasonable number of clusters for this dataset. Show the clusters after the 1st, 5th, 10th and last iteration. This will illustrate the cluster evolution as the algorithm runs. `subplot()` is very useful here.

To further aid our understanding of the algorithm, we will be keeping track of a few things:

1. Keep track of the evolution of the representatives for each cluster. You can do so by updating the variable `z` after each iteration.
2. Keep track of the objective function value at each iteration.

Once you are done implementing the algorithm, feel free to change the value of `K` and see how the clusterings are altered.

```
%{
Some useful setup done for you
%}
K = 3; % number of clusters
n = length(x); % number of vectors
c = zeros(n, 1); % clustering : c(i) is the cluster number for vector x(i, :)
previous_cluster = c;
iter = 0; % iteration number
z = datasample(x, K); % initial representatives
iter_num = 1;
objective_func = [];
plotpos = 1;

%keeps track of how the representatives change
reps_history = {z(1,:), z(2,:), z(3,:)}
```

reps_history = 1×3 cell

	1	2	3
1	[-0.9896,6....	[-0.3253,7....	[-0.5055,5....

```
%colors and style for each cluster (assumes a max of five cluster, hence only 5 colors
colors = [".b", ".g", ".r", ".m", ".c"]
```

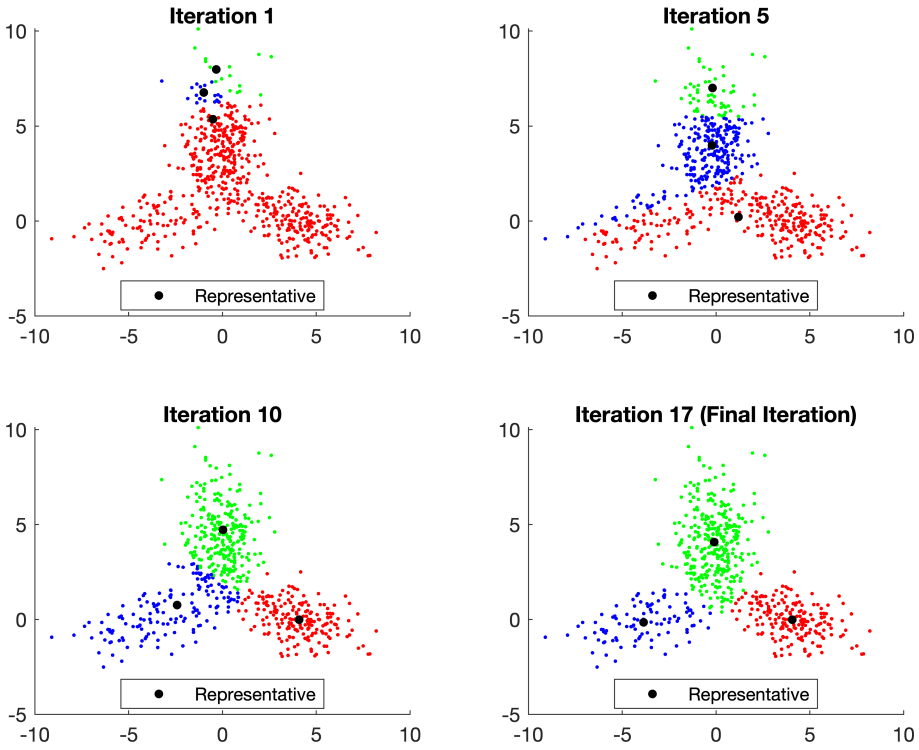
colors = 1×5 string

".b" ".g" ".r" ".m" ".c"

```
while true
    %{
    Step 1: Partitioning into clusters given our initial representatives
    %}

    % sum of distances between each vector and its representative divided by n is diffs
    diffs = 0;
    for idx = 1:n
        % calculating distance between initial representatives
        % and vector in dataset
        dists = zeros(1, K);
        for rep = 1:K
            dists(rep) = norm(x(idx, :) - z(rep, :));
        end
        [M, I] = min(dists);
```

```
        c(idx, 1) = I;

        diffs = diffs + (1/n) * (norm(x(idx, :) - z(I, :)))^2;
    end

    %plot clusters
    if iter_num == 1 || iter_num == 5 || iter_num == 10
        subplot(2,2, plotpos);
        hold on
        for idx = 1:n
           cluster_num = c(idx);
           plot(x(idx, 1), x(idx, 2),colors(cluster_num),'MarkerSize',5);

        end
        h = plot(z(:,1), z(:,2),".k",'MarkerSize',15, 'DisplayName','Representative');
        legend(h, "Representative", "location", "best");
        title(compose("Iteration %d", iter_num))
        hold off
        plotpos = plotpos + 1;

    end

    %{
    Increment the iteration number
    Update the objective function
    %}
    objective_func = [objective_func diffs];

    %update representatives
    reps = zeros(K, 2); %used to update representatives
    for idx = 1:n
        cluster_num = c(idx);
        reps(cluster_num, :) = reps(cluster_num, :) + x(idx, :);
    end
    for i = 1:K
        nk = sum(c(:, 1) == i);
        reps(i, :) = (1/nk) * reps(i, :);
    end
    z = reps;

    for i = 1:K
        reps_history{i} = [ reps_history{i}; z(i, :)];
    end

    %{
    Iterate until convergence:
    %}
    if isequal(previous_cluster, c)
        subplot(2,2, plotpos);
        hold on
```

```
        for idx = 1:n
           cluster_num = c(idx);
           plot(x(idx, 1), x(idx, 2),colors(cluster_num),'MarkerSize',5);
        end
        h = plot(z(:,1), z(:,2),".k",'MarkerSize',15, 'DisplayName','Representative');
        legend(h, "Representative", "location", "best");
        title(compose("Iteration %d (Final Iteration)", iter_num))
        hold off
        break
    end
    previous_cluster = c;
    iter_num = iter_num + 1;

end
```

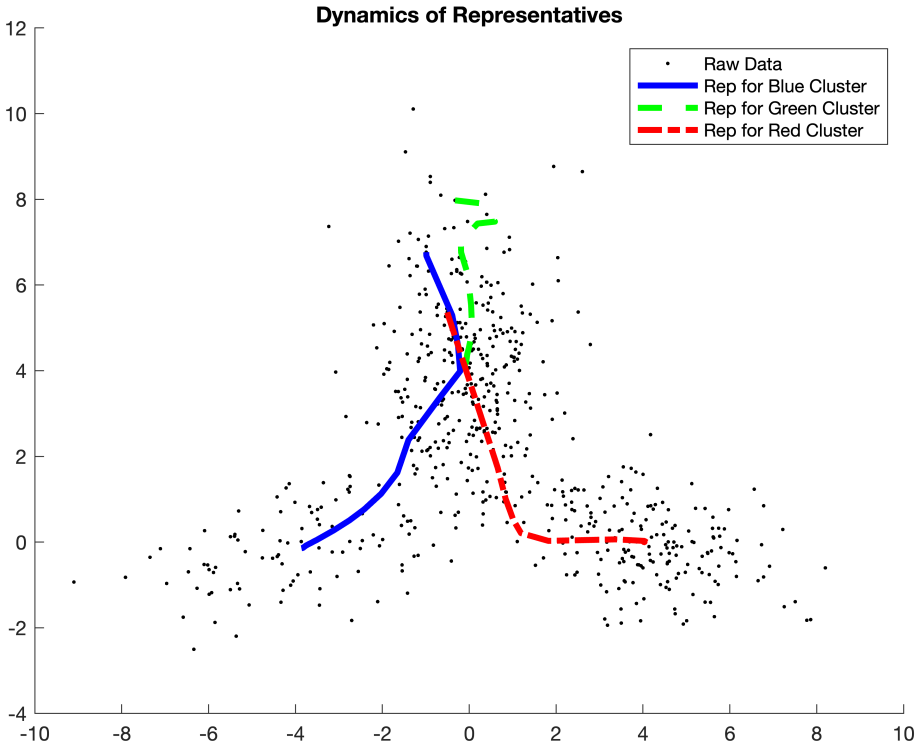



```
%{
Plotting
%}
```

Part (b) Evolution of Representatives

Now, let us see the dynamics of the representatives. Plot the original (raw) data and the trajectories of the cluster representatives in one figure. `squeeze()` might be useful here.

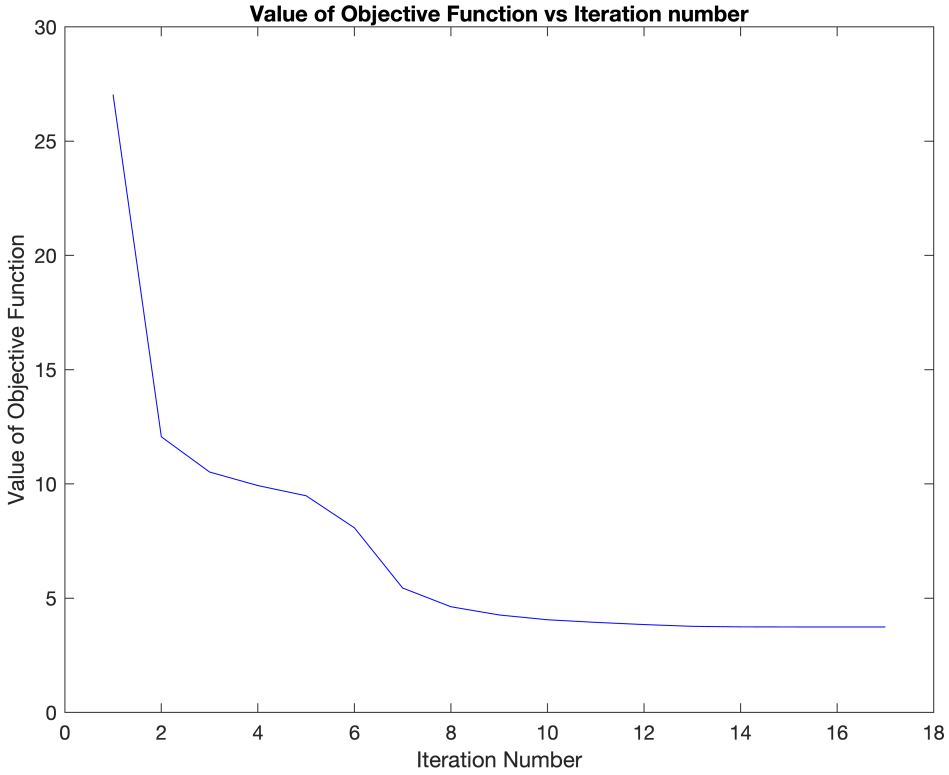
```
figure;
hold on
plot(x(:,1),x(:,2),'.k','MarkerSize',5, "DisplayName", "Raw Data");
plot(reps_history{1}(:,1),reps_history{1}(:,2),'-b','MarkerSize',15, "LineWidth", 3, "
plot(reps_history{2}(:,1),reps_history{2}(:,2),'--g','MarkerSize',15, "LineWidth", 3, '
plot(reps_history{3}(:,1),reps_history{3}(:,2),'-.r','MarkerSize',15, "LineWidth", 3, '
hold off
legend("location", "best")
title("Dynamics of Representatives")
```



Part (c) Objective Function

Plot the values of the objective function against the iteration number. Does the trend match what we expect? Report the minimum value of the objective function using `disp()`.

```
figure;
plot(1 : iter_num, objective_func(1, :),'-b','MarkerSize',10);
title("Value of Objective Function vs Iteration number")
xlabel("Iteration Number")
ylabel("Value of Objective Function")
```




```
[M,I] = min(objective_func)
```

```
M = 3.7420
I = 17
```

```
disp(M)
```

```
    3.7420
```

Part (d) Comparing with `kmeans( )`

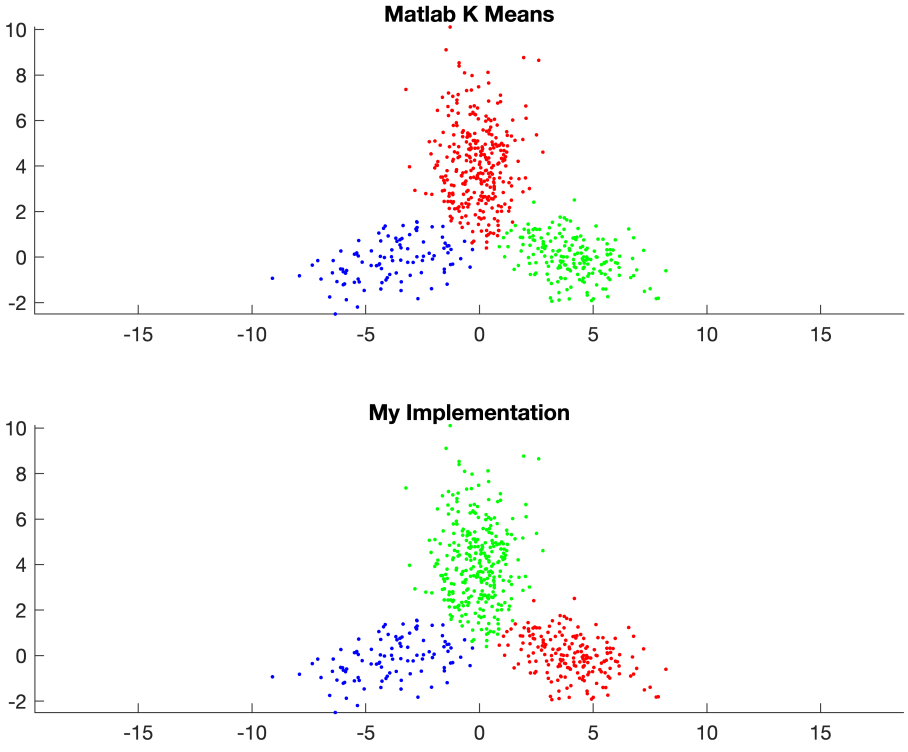
The K-means algorithm is a very popular clustering algorithm and it is often used in various applications. As such, it is implemented in most numerical software packages. In MATLAB, it exists as the built-in function `kmeans()`. In the simplest form, `idx = kmeans(X, k)` performs K-means clustering to partition the rows in the data matrix `X` into `k` clusters, and returns a vector `idx` containing cluster indices for each row. To compare your implementation with the built-in one, plot the clustering your obtained in (a) and the clustering obtained by using `kmeans()`.

```
idx = kmeans(x, K)
```

```
idx = 600×1
     1
     1
     1
     1
     1
     1
     1
```

```
1
3
3
:
:
```

```
figure;
subplot(2,1, 1);
hold on
for i = 1:n
   cluster_num = idx(i);
   plot(x(i, 1), x(i, 2),colors(cluster_num),'MarkerSize',5);
end
%legend(h, "Representative", "location", "best");
title("Matlab K Means")
axis equal;
hold off
subplot(2,1, 2);
hold on
for i = 1:n
   cluster_num = c(i);
   plot(x(i, 1), x(i, 2),colors(cluster_num),'MarkerSize',5);
end
hold off
title("My Implementation")
axis equal;
```



Problem 5 (10 points) Gram Matrices for Text Classification

Gram matrices are used in many different applications. Here is a simple example of application to classification of text documents. Suppose we have a collection of n text documents and we want to measure similarity between them. First, we predefine a list of words (the "dictionary") $w_1, w_2, w_3, \cdots, w_m$ appearing in the documents. Then, for each text i we construct a vector $x_i \in \mathbb{R}^m$ whose j^{th} component is the number of times the word w_j appears in the text. This way, each text is represented as a vector in $\mathbb{R}^m$. This representation is often referred to as the "bag of words" representation.

Let $\langle \cdot, \cdot \rangle$ be the usual dot product in $\mathbb{R}^m$ and $\| \cdot \|$ be the Euclidean norm. Let us normalize all vectors $z_i = \dfrac{x_i}{\|x_i\|}$ and define:

$$Z = [z_1 \cdots z_n] \in \mathbb{M}_{m \times n}$$

Let G be the Gram matrix associated with $z_1 \cdots z_n$:

$$G = Z^T Z \in \mathbb{M}_{n \times n}$$

Since all z_i are unit vectors:

$$G_{ij} = z_i^T z_j = \|z_i\| \|z_j\| \cos\theta_{ij} = \cos\theta_{ij}$$

where θ_{ij} is the angle between z_i and z_j, which of course is the same as the angle between x_i and x_j. This is widely use as a *measure of similarity* between documents. Hence, the more similar two texts i and j are, the closer G_{ij} is to 1.

Now, let us apply this method to real texts. The task is to compare the text of the US Constitution (found here) with the Wikipedia pages of Bernie Sanders, Hillary Clinton, Donald Trump, Ted Cruz, and John Kasich, and to find the page which is "closest" to the Constitution.

Part (a) Setup

To start, we will define a list of $m = 10$ words. To find the most "relevant words", copy the text of the Constitution and paste it into a word-cloud generator (such as this one). High frequency words appear large and we can form the dictionary from these words. Note that common sense is needed: words like "amendment" and "section" would appear large, but it makes no sense to include them into the dictionary. Also, it makes sense to include "United States" as a single "word", not two separate words, etc. But you are free to chose your own dictionary. In the code box below, please fill out the dictionary with your chosen words.

```
%Dictionary:
%    1) states
%    2) any
%    3) all
%    4) law
%    5) person
%    6) house
%    7) other
```

```
%    8) number
%    9) each
%    10) one
```

Part (b) Fun

With your specified dictionary, let us see which Wikipedia page is closest. First, construct $n = 6$ vectors (as specified above), one for each Wikipedia page and one for the Constitution. No need to get fancy here; just go to the related page and search for the number of occureneces of the words in your dictionary. Put all your results in a column vector for each candidate:

```
%{
Report your results here
%}
sanders = [68 14 50 26 9 50 31 9 5 30];
clinton = [59 15 51 44 19 63 40 9 2 36];
trump = [90 26 101 66 42 71 46 15 6 35];
cruz = [97 13 52 83 19 22 26 3 4 25];
kasich = [29 8 34 36 7 53 21 5 1 18];
constitution =[132 80 43 39 34 34 31 26 26 25];
vecs = {sanders, clinton, trump, cruz, kasich, constitution};
Z = []
```

```
Z =

     []
```

```
%{
Normalize your vectors
%}
for i = 1:6
    vecs{i} = (1/norm(vecs{i})) * vecs{i}
    Z = [Z ; vecs{i}]
end
```

```
vecs = 1×6 cell
```

	1	2
1	[0.6071,0.125,0.4464,0.2321,0.0804,0.4464,0.2768,0.0804,...	[59,15,51,44,19,63,40,...

```
Z = 1×10
    0.6071    0.1250    0.4464    0.2321    0.0804    0.4464    0.2768    0.0804 ···
vecs = 1×6 cell
                                                                                  ...
```

	1
1	[0.6071,0.125,0.4464,0.2321,0.0804,0.4464,0.2768,0.0804,...

```
Z = 2×10
    0.6071    0.1250    0.4464    0.2321    0.0804    0.4464    0.2768    0.0804 ···
    0.4731    0.1203    0.4089    0.3528    0.1523    0.5051    0.3207    0.0722
vecs = 1×6 cell
                                                                                  ...
```

	1
1	[0.6071,0.125,0.4464,0.2321,0.0804,0.4464,0.2768,0.0804,...

```
Z = 3×10
    0.6071    0.1250    0.4464    0.2321    0.0804    0.4464    0.2768    0.0804 ···
    0.4731    0.1203    0.4089    0.3528    0.1523    0.5051    0.3207    0.0722
    0.4900    0.1415    0.5499    0.3593    0.2287    0.3865    0.2504    0.0817
vecs = 1×6 cell
                                                                                ···
```

	1
1	[0.6071,0.125,0.4464,0.2321,0.0804,0.4464,0.2768,0.0804,...

```
Z = 4×10
    0.6071    0.1250    0.4464    0.2321    0.0804    0.4464    0.2768    0.0804 ···
    0.4731    0.1203    0.4089    0.3528    0.1523    0.5051    0.3207    0.0722
    0.4900    0.1415    0.5499    0.3593    0.2287    0.3865    0.2504    0.0817
    0.6640    0.0890    0.3559    0.5681    0.1301    0.1506    0.1780    0.0205
vecs = 1×6 cell
                                                                                ···
```

	1
1	[0.6071,0.125,0.4464,0.2321,0.0804,0.4464,0.2768,0.0804,...

```
Z = 5×10
    0.6071    0.1250    0.4464    0.2321    0.0804    0.4464    0.2768    0.0804 ···
    0.4731    0.1203    0.4089    0.3528    0.1523    0.5051    0.3207    0.0722
    0.4900    0.1415    0.5499    0.3593    0.2287    0.3865    0.2504    0.0817
    0.6640    0.0890    0.3559    0.5681    0.1301    0.1506    0.1780    0.0205
    0.3465    0.0956    0.4062    0.4301    0.0836    0.6332    0.2509    0.0597
vecs = 1×6 cell
                                                                                ···
```

	1
1	[0.6071,0.125,0.4464,0.2321,0.0804,0.4464,0.2768,0.0804,...

```
Z = 6×10
    0.6071    0.1250    0.4464    0.2321    0.0804    0.4464    0.2768    0.0804 ···
    0.4731    0.1203    0.4089    0.3528    0.1523    0.5051    0.3207    0.0722
    0.4900    0.1415    0.5499    0.3593    0.2287    0.3865    0.2504    0.0817
    0.6640    0.0890    0.3559    0.5681    0.1301    0.1506    0.1780    0.0205
    0.3465    0.0956    0.4062    0.4301    0.0836    0.6332    0.2509    0.0597
    0.7328    0.4441    0.2387    0.2165    0.1888    0.1888    0.1721    0.1443
```

```
Z = Z'
```

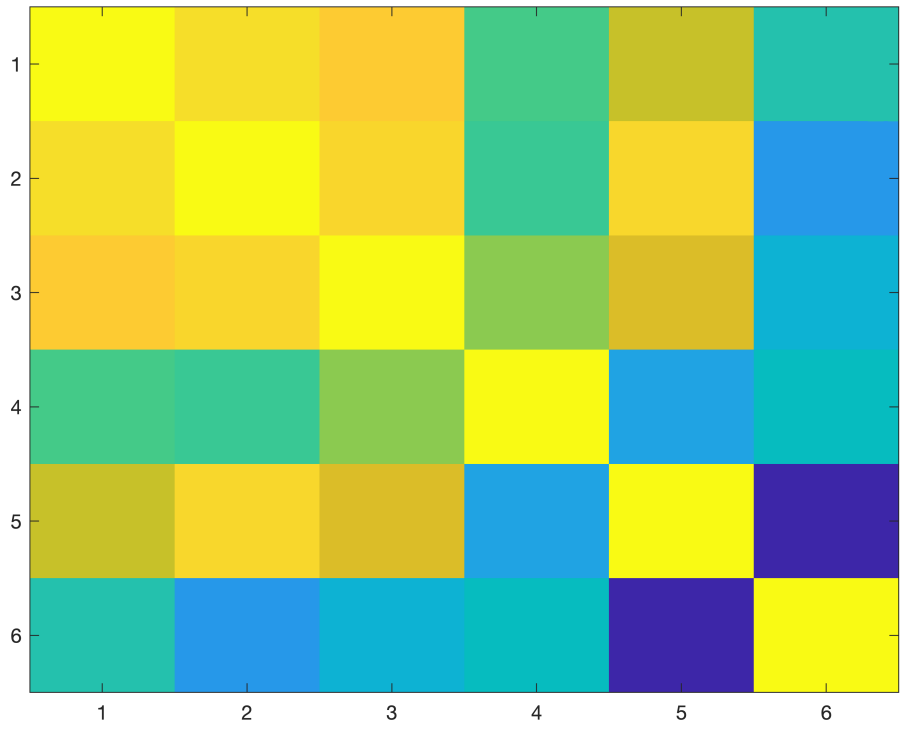
```
Z = 10×6
    0.6071    0.4731    0.4900    0.6640    0.3465    0.7328
    0.1250    0.1203    0.1415    0.0890    0.0956    0.4441
    0.4464    0.4089    0.5499    0.3559    0.4062    0.2387
    0.2321    0.3528    0.3593    0.5681    0.4301    0.2165
    0.0804    0.1523    0.2287    0.1301    0.0836    0.1888
    0.4464    0.5051    0.3865    0.1506    0.6332    0.1888
    0.2768    0.3207    0.2504    0.1780    0.2509    0.1721
    0.0804    0.0722    0.0817    0.0205    0.0597    0.1443
    0.0446    0.0160    0.0327    0.0274    0.0119    0.1443
    0.2679    0.2887    0.1905    0.1711    0.2150    0.1388
```

```
%{
Construct the Gram matrix G as discussed
%}
```

```
G = Z' * Z
```

```
G = 6×6
    1.0000    0.9771    0.9634    0.8807    0.9253    0.8596
    0.9771    1.0000    0.9723    0.8751    0.9729    0.8062
    0.9634    0.9723    1.0000    0.9055    0.9341    0.8332
    0.8807    0.8751    0.9055    1.0000    0.8167    0.8484
    0.9253    0.9729    0.9341    0.8167    1.0000    0.7051
    0.8596    0.8062    0.8332    0.8484    0.7051    1.0000
```

```
%{
You can visualize G by uncommenting the following lines of code
%}
figure;
imagesc(G);
```




```
angles = []
```

```
angles =

     []
```

```
for i = 1:5
    cosine = dot(Z(:,i), Z(:,6));
    angles = [angles acos(cosine)]
end
```

```
angles = 0.5363
angles = 1×2
```

```
    0.5363    0.6331
angles = 1×3
    0.5363    0.6331    0.5860
angles = 1×4
    0.5363    0.6331    0.5860    0.5579
angles = 1×5
    0.5363    0.6331    0.5860    0.5579    0.7882
```

```
[M, I] = min(angles);

%{
Find the page closest to the Constitution text and report your
answer using disp()
%}
pages = ["Bernie Sanders", "Hillary Clinton", "Donald Trump", "Ted Cruz", "John Kasich"
disp(append(pages(I,1), " Page is the closest to the Constitution"));
```

```
Bernie Sanders Page is the closest to the Constitution
```

```
disp(Z(:,I));
```

```
    0.6071
    0.1250
    0.4464
    0.2321
    0.0804
    0.4464
    0.2768
    0.0804
    0.0446
    0.2679
```

Submit the obtained result to the corresponding piazza poll. Remember: this is not about your political preferences, it is about your data analysis :)