Linear Algebra
Problem Set 2 - Solutions

Abolfazl Ranjbar

Course Instructors: Dr. Elham Ghasrodashti and Dr. Peyman Adibi

Computer Engineering Department
University of Isfahan

Fall Semester 2024

۱- S را یک **زیر فضا** از V می نامند هرگاه،

1. $\forall \mathbf{s},\mathbf{t} \in S \rightarrow \quad \mathbf{s}+\mathbf{t} \in S$
2. $\forall \mathbf{s} \in S, \quad \forall a \in F \rightarrow \quad a\mathbf{s} \in S$

نشان دهید، در فضای برداری دو بعدی $\Re^2$ هر خط راستی که از مبدأ عبور کند، یک زیر فضای برداری از $\Re^2$ است،

$$S = \{(x, y) \in \Re^2 : ax + by = 0\}$$

بسته بودن نسبت به جمع :

$$\begin{cases} s = (x_1, y_1) \in S \\ t = (x_2, y_2) \in S \end{cases} \longrightarrow s + t \in S \hookrightarrow (x_1 + x_2, y_1 + y_2) \in S$$

we know that $\begin{cases} ax_1 + by_1 = 0 \\ ax_2 + by_2 = 0 \end{cases} \longrightarrow a(x_1 + x_2) + b(y_1 + y_2) = 0 + 0 = 0$

$ax_1 + ax_2 + by_1 + by_2$

بسته بودن نسبت به ضرب :

if $s = (x_1, y_1) \in S, \ c \in R \xrightarrow{\text{so}} cs \in S \longrightarrow (cx_1, cy_1) \in S$

$$a(cx_1) + b(cy_1) = c(ax_1) + c(by_1) = c(ax_1 + by_1) = 0$$

$$\longrightarrow a(cx_1) + b(cy_1) = 0 \longrightarrow (cx, cy) \in S$$

بسته در یک فضای برداری از R^2 است

آیا در فضای برداری دو بعدی $\Re^2$ هر خط راستی که از مبدأ عبور نکند، یک زیر فضای برداری از $\Re^2$ است؟

$$S = \{(x, y) \in \Re^2 : ax + by = k\}$$

if $(x_1, y_1), (x_2, y_2) \in S \longrightarrow (x_1 + x_2, y_1 + y_2) \in S$

$$\begin{cases} (x_1, y_1) \in S \longrightarrow ax_1 + by_1 = k \\ (x_2, y_2) \in S \longrightarrow ax_2 + by_2 = k \end{cases} \rightarrow a(x_1 + x_2) + b(y_1 + y_2) = (ax_1 + by_1) + (ax_2 + by_2) \longrightarrow = k + k = 2k$$

$\longrightarrow k \neq 2k \rightarrow$ نسبت به جمع بسته نیست

خط راستی که از مبدأ عبور نکند زیر فضای برداری R^2 نیست .

۲-استقلال خطی یا وابستگی خطی بردارهای زیر را بررسی کنید.

الف) $u_1 = \begin{bmatrix}-2\\1\end{bmatrix}, u_2 = \begin{bmatrix}-1\\-3\end{bmatrix}, u_3 = \begin{bmatrix}4\\-2\end{bmatrix}$

$a_1u_1 + a_2u_2 + a_3u_3$

$= a_1\begin{bmatrix}-2\\1\end{bmatrix} + a_2\begin{bmatrix}-1\\-3\end{bmatrix} + a_3\begin{bmatrix}4\\-2\end{bmatrix}$ ← همواره جواب غیر صفر دارد

$a_1 = 2 \rightsquigarrow \begin{cases} -a_2 + 4a_3 = 4 \\ -3a_2 - 2a_3 = -2 \end{cases} \rightsquigarrow -5a_2 = 0 \rightarrow a_2 = 0, a_3 = 0$ ← وابسته خطی

ب) $u_1 = \begin{bmatrix}1\\-2\\3\\-4\end{bmatrix}, u_2 = \begin{bmatrix}-1\\3\\4\\2\end{bmatrix}, u_3 = \begin{bmatrix}1\\1\\-2\\-2\end{bmatrix}$

$\left.\begin{array}{l} a_1 - a_2 + a_3 = 0 \\ -2a_1 + 3a_2 + a_3 = 0 \end{array}\right\} \rightarrow 3a_1 - 4a_2 = 0$

$\left.\begin{array}{l} 3a_1 + 4a_2 - 2a_3 = 0 \\ 4a_1 + 2a_2 - 2a_3 = 0 \end{array}\right\} \rightarrow 7a_1 + 2a_2 = 0$

$\left.\begin{array}{l} 3a_1 - 4a_2 = 0 \\ 7a_1 + 2a_2 = 0 \end{array}\right\} \rightarrow 14a_1 = 0$

$a_1 = 0 \rightsquigarrow a_2 = 0$

$\rightsquigarrow a_3 = 0 \rightsquigarrow$ مستقل خطی

۳- آیا مجموعه زیر برای فضای برداری مورد نظر تشکیل یک پایه می دهد

الف) $\mathbf{v}_1 = \begin{bmatrix}1\\-1\\1\end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix}-1\\2\\-2\end{bmatrix}, \quad \mathbf{v}_3 = \begin{bmatrix}-1\\4\\-4\end{bmatrix}$ (فضای برداری $\Re^3$)

$a_1v_1 + a_2v_2 + a_3v_3 = 0 \begin{cases} a_1 - a_2 - a_3 = 0 \\ -a_1 + 2a_2 + 4a_3 = 0 \\ a_1 - 2a_2 - 4a_3 = 0 \end{cases} \xrightarrow{+} a_2 + 3a_3 = 0 \begin{cases} a_2 = 3 \\ a_3 = -1 \end{cases}$

$a_1 = 2$ ← وابسته خطی ← پایه نیست

۴- بردارهای زیر را در نظر بگیرید. با استفاده از الگوریتم گرام-اشمیت بردارهای پایه یکه متعامد را بدست آورید.

$$u_1 = \begin{bmatrix}1\\2\\1\\0\end{bmatrix}, u_2 = \begin{bmatrix}3\\3\\3\\0\end{bmatrix}, u_3 = \begin{bmatrix}2\\-10\\0\\0\end{bmatrix}, u_4 = \begin{bmatrix}-2\\1\\-6\\2\end{bmatrix}$$

$$\tilde{q}_k = \vec{a}_k - \sum_{i=1}^{k-1} (q_i^T a_k) q_i$$

$$q_1 = \frac{\tilde{q}_1}{\|\tilde{q}_1\|} = \frac{u_1}{\|u_1\|} = \begin{bmatrix}1/\sqrt{6}\\2/\sqrt{6}\\1/\sqrt{6}\\0\end{bmatrix}$$

$$\tilde{q}_2 = u_2 - (q_1^T u_2) q_1 = \begin{bmatrix}3\\3\\3\\0\end{bmatrix} - 2\sqrt{6}\begin{bmatrix}1/\sqrt{6}\\2/\sqrt{6}\\1/\sqrt{6}\\0\end{bmatrix} = \begin{bmatrix}3\\3\\3\\0\end{bmatrix} - \begin{bmatrix}2\\4\\2\\0\end{bmatrix} = \begin{bmatrix}1\\-1\\1\\0\end{bmatrix} \quad q_2 = \begin{bmatrix}1/\sqrt{3}\\-1/\sqrt{3}\\1/\sqrt{3}\\0\end{bmatrix}$$

$$\tilde{q}_3 = u_3 - (q_2^T u_3) q_2 - (q_1^T u_3) q_1 = \begin{bmatrix}2\\-10\\0\\0\end{bmatrix} - (4\sqrt{3})\begin{bmatrix}1/\sqrt{3}\\-1/\sqrt{3}\\1/\sqrt{3}\\0\end{bmatrix} + (3\sqrt{6})\begin{bmatrix}1/\sqrt{6}\\2/\sqrt{6}\\1/\sqrt{6}\\0\end{bmatrix}$$

$$= \begin{bmatrix}2\\-10\\0\\0\end{bmatrix} - \begin{bmatrix}4\\-4\\4\\0\end{bmatrix} + \begin{bmatrix}3\\6\\3\\0\end{bmatrix} = \begin{bmatrix}1\\0\\-1\\0\end{bmatrix} \rightsquigarrow q_3 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\-1\\0\end{bmatrix} = \begin{bmatrix}1/\sqrt{2}\\0\\-1/\sqrt{2}\\0\end{bmatrix}$$

$$\tilde{q}_4 = u_4 - (q_3^T u_4) q_3 - (q_2^T u_4) q_2 - (q_1^T u_4) q_1$$

$$= \begin{bmatrix}-2\\1\\6\\2\end{bmatrix} - (2\sqrt{2})\begin{bmatrix}1/\sqrt{2}\\0\\-1/\sqrt{2}\\0\end{bmatrix} + (3\sqrt{2})\begin{bmatrix}1/\sqrt{3}\\-1/\sqrt{3}\\1/\sqrt{3}\\0\end{bmatrix} + (\sqrt{6})\begin{bmatrix}1/\sqrt{6}\\2/\sqrt{6}\\1/\sqrt{6}\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\\2\end{bmatrix} \rightsquigarrow q_4 = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

۵- اگر فرایند گرام اشمیت را به یک دسته بردار وابسته خطی اعمال نماییم. نتیجه نهایی چه خواهد شد؟ با انتخاب سه بردار وابسته خطی پاسخ خود را بررسی نمایید.

example: $u_1 = \begin{bmatrix}0\\0\\1\end{bmatrix}, u_2 = \begin{bmatrix}0\\1\\1\end{bmatrix}, u_3 = \begin{bmatrix}0\\1\\2\end{bmatrix}$

$$q_1 = \frac{u_1}{\|u_1\|} = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

$$\tilde{q}_2 = u_2 - (q_1^T u_2) q_1 = \begin{bmatrix}0\\1\\1\end{bmatrix} - \begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix} \rightsquigarrow q_2 = \frac{\tilde{q}_2}{\|\tilde{q}_2\|} = \begin{bmatrix}0\\0\\0\end{bmatrix} \rightsquigarrow$$

یکی از بردارها صفر شد.

4.1 *Minimizing mean square distance to a set of vectors.* Let $x_1, \ldots, x_L$ be a collection of n-vectors. In this exercise you will fill in the missing parts of the argument to show that the vector z which minimizes the sum-square distance to the vectors,

$$J(z) = \|x_1 - z\|^2 + \cdots + \|x_L - z\|^2,$$

is the average or centroid of the vectors, $\overline{x} = (1/L)\,(x_1 + \cdots + x_L)$. (This result is used in one of the steps in the k-means algorithm. But here we have simplified the notation.)

(a) Explain why, for any z, we have

$$J(z) = \sum_{i=1}^{L} \|x_i - \overline{x} - (z - \overline{x})\|^2 = \sum_{i=1}^{L} \left(\|x_i - \overline{x}\|^2 - 2(x_i - \overline{x})^T(z - \overline{x})\right) + L\|z - \overline{x}\|^2.$$

(b) Explain why $\sum_{i=1}^{L}(x_i - \overline{x})^T(z - \overline{x}) = 0$. *Hint.* Write the left-hand side as

$$\left(\sum_{i=1}^{L}(x_i - \overline{x})\right)^T (z - \overline{x}),$$

and argue that the left-hand vector is 0.

(c) Combine the results of (a) and (b) to get $J(z) = \sum_{i=1}^{L} \|x_i - \overline{x}\|^2 + L\|z - \overline{x}\|^2$. Explain why for any $z \neq \overline{x}$, we have $J(z) > J(\overline{x})$. This shows that the choice $z = \overline{x}$ minimizes $J(z)$.

4.2 *k-means with nonnegative, proportions, or Boolean vectors.* Suppose that the vectors $x_1, \ldots, x_N$ are clustered using k-means, with group representatives $z_1, \ldots, z_k$.

(a) Suppose the original vectors x_i are nonnegative, *i.e.*, their entries are nonnegative. Explain why the representatives z_j are also nonnegative.

(b) Suppose the original vectors x_i represent proportions, *i.e.*, their entries are nonnegative and sum to one. (This is the case when x_i are word count histograms, for example.) Explain why the representatives z_j also represent proportions, *i.e.*, their entries are nonnegative and sum to one.

(c) Suppose the original vectors x_i are Boolean, *i.e.*, their entries are either 0 or 1. Give an interpretation of $(z_j)_i$, the ith entry of the j group representative.

Hint. Each representative is the average of some of the original vectors.

(a) as $z_j = \frac{1}{|G_j|} \sum_{k \in G_j} x_k$ if x are nonnegative

$\hookrightarrow z_j$ is nonnegative too

(b)

$$1^T z_j = \frac{1}{|G_j|} \sum_{k \in G_j} 1^T x_k = \frac{|G_j|}{|G_j|} = 1$$

(c) because z_j is a fraction of the vectors in the group $\rightarrow$ if $z_j = 1$ then all the vectors in that group are 1 or if it close to 1 $\rightsquigarrow$ most of vectors are one.

4.3 *Linear separation in 2-way partitioning.* Clustering a collection of vectors into $k = 2$ groups is called 2-way partitioning, since we are partitioning the vectors into 2 groups, with index sets G_1 and G_2. Suppose we run k-means, with $k = 2$, on the n-vectors $x_1, \ldots, x_N$. Show that there is a nonzero vector w and a scalar v that satisfy

$$w^T x_i + v \geq 0 \text{ for } i \in G_1, \qquad w^T x_i + v \leq 0 \text{ for } i \in G_2.$$

In other words, the affine function $f(x) = w^T x + v$ is greater than or equal to zero on the first group, and less than or equal to zero on the second group. This is called *linear separation* of the two groups (although *affine separation* would be more accurate).

Hint. Consider the function $\|x - z_1\|^2 - \|x - z_2\|^2$, where z_1 and z_2 are the group representatives.

$\rightsquigarrow$ $\|x_i - z_2\|^2 - \|x_i - z_1\|^2 \geq 0$

$\|x_i - z_2\|^2 - \|x_i - z_1\|^2 \leq 0$

expanding: $\|x_i\|^2 - 2 x_i^T z_2 + \|z_2\|^2 - \|x_i\|^2 + 2 x_i^T z_1 - \|z_1\|^2$

$= -2 x_i^T (z_2 - z_1) + \|z_2\|^2 - \|z_1\|^2$

format $\rightsquigarrow$ $= -2 (z_2 - z_1)^T x_i + \|z_2\|^2 - \|z_1\|^2$

$(z_2 - z_1)^T$ → w; $\|z_2\|^2 - \|z_1\|^2$ → v

4.4 *Pre-assigned vectors.* Suppose that some of the vectors $x_1, \ldots, x_N$ are assigned to specific groups. For example, we might insist that x_{27} be assigned to group 5. Suggest a simple modification of the k-means algorithm that respects this requirement. Describe a practical example where this might arise, when each vector represents n features of a medical patient.

→ first we override the partitionny step, then we enforce the given assignment instead of allowing the algorithem to resign it. The remaininy vectors are arrigned to graups as usual bered on their position

☆ stack vector: refers to vectors formed with concatenating two or more vectors { vertically = column vector; horizontally = row vector

5.1 *Linear independence of stacked vectors.* Consider the stacked vectors

$$c_1 = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix}, \quad \ldots, \quad c_k = \begin{bmatrix} a_k \\ b_k \end{bmatrix},$$

where $a_1, \ldots, a_k$ are n-vectors and $b_1, \ldots, b_k$ are m-vectors.

(a) Suppose $a_1, \ldots, a_k$ are linearly independent. (We make no assumptions about the vectors $b_1, \ldots, b_k$.) Can we conclude that the stacked vectors $c_1, \ldots, c_k$ are linearly independent?

(b) Now suppose that $a_1, \ldots, a_k$ are linearly dependent. (Again, with no assumptions about $b_1, \ldots, b_k$.) Can we conclude that the stacked vectors $c_1, \ldots, c_k$ are linearly dependent?

(a) yes, suppose: $\alpha^T C = 0 \rightarrow \alpha_1 c_1 + \cdots + \alpha_n c_n = 0$

$$\Leftrightarrow \alpha_1 \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} + \cdots + \alpha_n \begin{bmatrix} a_n \\ b_n \end{bmatrix} = 0$$

as you can see → $\alpha_1 a_1 + \cdots + \alpha_n a_n = 0$ → is linear independent

so → $\alpha_1, \ldots, \alpha_n = 0$ → then adding "b" vector doesn't change anything.

(b) nope!, lets use the example above:

$\alpha_1 a_1 + \cdots + \alpha_n a_n = 0$ → is linear independent

→ so there is some "α" that are not zero

↳ so by adding "b" vector, those coefficients might not work

→ $\alpha_1 \beta_1 + \cdots + \alpha_n \beta_n \neq 0$

5.2 *A surprising discovery.* An intern at a quantitative hedge fund examines the daily returns of around 400 stocks over one year (which has 250 trading days). She tells her supervisor that she has discovered that the returns of one of the stocks, Google (GOOG), can be expressed as a linear combination of the others, which include many stocks that are unrelated to Google (say, in a different type of business or sector).

Her supervisor then says: "It is overwhelmingly unlikely that a linear combination of the returns of unrelated companies can reproduce the daily return of GOOG. So you've made a mistake in your calculations."

Is the supervisor right? Did the intern make a mistake? Give a very brief explanation.

→ The intern is correct, with 400 stocks in a 250-dimentional space (trading days), linear dependence guarantees that at least one stock's returns can be expressed as linear combination of the other.

5.4 *Norm of linear combination of orthonormal vectors.* Suppose $a_1, \ldots, a_k$ are orthonormal n-vectors, and $x = \beta_1 a_1 + \cdots + \beta_k a_k$, where $\beta_1, \ldots, \beta_k$ are scalars. Express $\|x\|$ in terms of $\beta = (\beta_1, \ldots, \beta_k)$.

$\|x\|^2 = x^T x = (\beta_1 a_1 + \cdots + \beta_k a_k)^T (\beta_1 a_1 + \cdots + \beta_k a_k) =$

as a_i is orthonormal → $a_i^T a_i = 1$

so → $\|x\|^2 = \beta_1^2 + \cdots + \beta_k^2 = \|\beta\|^2$ → $\|x\| = \|\beta\|$

5.5 *Orthogonalizing vectors.* Suppose that a and b are any n-vectors. Show that we can always find a scalar γ so that $(a - \gamma b) \perp b$, and that γ is unique if $b \neq 0$. (Give a formula for the scalar γ.) In other words, we can always subtract a multiple of a vector from another one, so that the result is orthogonal to the original vector. The orthogonalization step in the Gram–Schmidt algorithm is an application of this.

→ $(a - \gamma b)^T b = 0$ → $a^T b - \gamma b^T b = 0$

→ $\gamma = a^T b / b^T b$

5.6 *Gram–Schmidt algorithm.* Consider the list of n n-vectors

$$a_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad a_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \ldots, \quad a_n = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

(The vector a_i has its first i entries equal to one, and the remaining entries zero.) Describe what happens when you run the Gram–Schmidt algorithm on this list of vectors, *i.e.*, say what $q_1, \ldots, q_n$ are. Is $a_1, \ldots, a_n$ a basis?

$\tilde{q}_1 = a_1 \longrightarrow q_1 = e_1$

$\tilde{q}_2 = a_2 - (q_1^T a_2) q_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = e_2 \rightarrow q_2 = e_2$

($q_1^T a_2 = e_1$)

$\tilde{q}_3 = a_3 - (q_2^T a_3) q_2 - (q_1^T a_3) q_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = e_3$

($q_2^T a_3 = e_2$)

$\rightarrow q_3 = e_3$

$\rightarrow q_i = e_i$

basis ?! $\rightarrow \beta_1 a_1 + \cdots + \beta_k q_k = 0 \longrightarrow$ because $a_n = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$ then

$\beta_n = 0 \longrightarrow$ and $a_{n-1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \rightarrow \beta_{n-1} = 0 \rightarrow$ so a is basis

5.7 *Running Gram–Schmidt algorithm twice.* We run the Gram–Schmidt algorithm once on a given set of vectors $a_1, \ldots, a_k$ (we assume this is successful), which gives the vectors $q_1, \ldots, q_k$. Then we run the Gram–Schmidt algorithm on the vectors $q_1, \ldots, q_k$, which produces the vectors $z_1, \ldots, z_k$. What can you say about $z_1, \ldots, z_k$?

since q vector is already orthogonal then applying Gram-Schmidt does nothing and it gaves us $z_i = q_i$

5.8 *Early termination of Gram–Schmidt algorithm.* When the Gram–Schmidt algorithm is run on a particular list of 10 15-vectors, it terminates in iteration 5 (since $\tilde{q}_5 = 0$). Which of the following must be true?

(a) a_2, a_3, a_4 are linearly independent.

(b) a_1, a_2, a_5 are linearly dependent.

(c) a_1, a_2, a_3, a_4, a_5 are linearly dependent. ✓

(d) a_4 is nonzero.

5.9 *A particular computer can carry out the Gram–Schmidt algorithm on a list of $k = 1000$ n-vectors, with $n = 10000$, in around 2 seconds. About how long would you expect it to take to carry out the Gram–Schmidt algorithm with $\tilde{k} = 500$ $\tilde{n}$-vectors, with $\tilde{n} = 1000$?*

$k = 1000$, $n = 10000$ $\rightsquigarrow$ $A_{10000 \times 1000}$ $\rightsquigarrow$ 2 sec

book solution: comp speed $\rightsquigarrow$ flop/sec

Gram-schmidt requires $2nk^2$ flops

$k = 10^3, n = 10^4 \rightsquigarrow \approx \frac{2 \times 10^{10} \text{ flop}}{2 \text{ sec}} = 10^{10}$ flop/sec

second prob: $k = 500, n = 10^3 \rightsquigarrow 2 \times 10^3 \times 250000$ flops $= 5 \times 10^8$ flop

$\rightsquigarrow \frac{5 \times 10^8}{10^{10}} = 0.05$ second

Calculate the weighted linear combination using Python code.

$$\lambda_1 = 1, \lambda_2 = 2, \lambda_3 = -3$$

$$v_1 = \begin{bmatrix} 4 \\ 5 \\ 1 \end{bmatrix}, v_2 = \begin{bmatrix} -4 \\ 0 \\ -4 \end{bmatrix}, v_3 = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

$$w = \lambda_1 v_1 + \lambda_2 v_2 + \lambda_3 v_3 = \begin{bmatrix} -7 \\ -4 \\ -13 \end{bmatrix}$$

```
import numpy as np
v1 = np.array([4, 5, 1])
v2 = np.array([-4, 0, -4])
v3 = np.array([1, 3, 2])
lambda1, lambda2, lambda3 = 1, 2, -3
weighted_sum = lambda1 * v1 + lambda2 * v2 + lambda3 * v3
print(f"weighted sum: {weighted_sum}")
```

Run the following code multiple times and describe your results and observations.

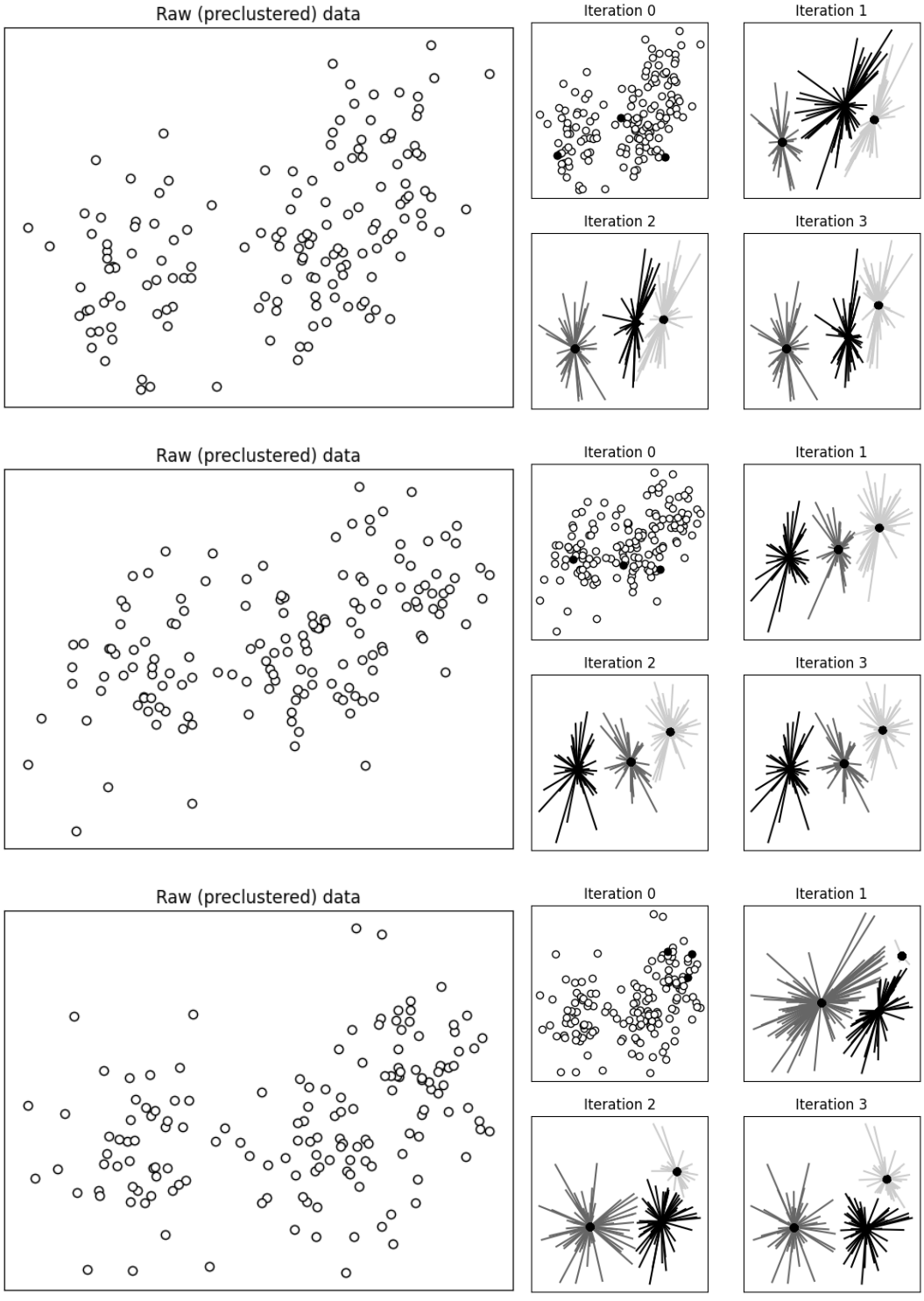
```
nPerClust = 50
# blur around centroid (std units)
blur = 1
# XY centroid locations
A = [ 1, 1 ]
B = [ -3, 1 ]
C = [ 3, 3 ]
# generate data
a = [ A[0]+np.random.randn(nPerClust)*blur ,
A[1]+np.random.randn(nPerClust)*blur ]
b = [ B[0]+np.random.randn(nPerClust)*blur ,
B[1]+np.random.randn(nPerClust)*blur ]
c = [ C[0]+np.random.randn(nPerClust)*blur ,
C[1]+np.random.randn(nPerClust)*blur ]
# concatanate into a matrix
data = np.transpose( np.concatenate((a,b,c),axis=1) )
# plot data
plt.plot(data[:,0],data[:,1],'ko',markerfacecolor='w')
plt.title('Raw (preclustered) data')
plt.xticks([])
plt.yticks([])
plt.show()


## initialize random cluster centroids
k = 3 # extract three clusters
```

```
# random cluster centers (randomly sampled data points)
ridx = np.random.choice(range(len(data)),k,replace=False)
centroids = data[ridx,:]
# setup the figure
fig,axs = plt.subplots(2,2,figsize=(6,6))
axs = axs.flatten()
lineColors = [ [0,0,0],[.4,.4,.4],[.8,.8,.8] ]#'rbm'
# plot data with initial random cluster centroids
axs[0].plot(data[:,0],data[:,1],'ko',markerfacecolor='w')
axs[0].plot(centroids[:,0],centroids[:,1],'ko')
axs[0].set_title('Iteration 0')
axs[0].set_xticks([])
axs[0].set_yticks([])
# loop over iterations
for iteri in range(3):
  # step 1: compute distances
  dists = np.zeros((data.shape[0],k))
  for ci in range(k):
    dists[:,ci] = np.sum((data-centroids[ci,:])**2,axis=1)
  # step 2: assign to group based on minimum distance
  groupidx = np.argmin(dists,axis=1)
  # step 3: recompute centers
  for ki in range(k):
    centroids[ki,:] = [ np.mean(data[groupidx==ki,0]),
  np.mean(data[groupidx==ki,1]) ]

  # plot data points
  for i in range(len(data)):
    axs[iteri+1].plot([ data[i,0],centroids[groupidx[i],0] ],[
    data[i,1],centroids[groupidx[i],1] ],color=lineColors[groupidx[i]])
    axs[iteri+1].plot(centroids[:,0],centroids[:,1],'ko')
    axs[iteri+1].set_title(f'Iteration {iteri+1}')
    axs[iteri+1].set_xticks([])
    axs[iteri+1].set_yticks([])
plt.show()
```

The given code implements the k-means clustering algorithm and visualizes the clustering process for a 2D dataset over four iterations. Here's a breakdown of the observations and results when running the code multiple times:



K-means Clustering on Handwritten Digits (1-5) Dataset

This project explores the application of the K-means clustering algorithm to group handwritten digits from 1 to 5 using the MNIST dataset. The digits were extracted, flattened, and normalized before running the clustering algorithm. The goal was to cluster the data for varying values of k = 3, 4, 5, 6 and visualize the resulting clusters and centroids.

Data Preprocessing

- **Dataset**: The MNIST dataset was loaded using torchvision, which provides images of handwritten digits. Only the digits 1 through 5 were selected from the dataset.
- **Normalization**: Each image was flattened into a 28x28 (784) vector and normalized by dividing pixel values by 255.
- **GPU Acceleration**: The clustering algorithm leverages PyTorch to compute Euclidean distances and perform clustering operations on the GPU for faster processing.

K-means Implementation

The K-means algorithm was implemented using PyTorch. The steps involved include:

1. **Initialization**: Centroids were randomly initialized from the dataset.
2. **Assignment**: Each data point was assigned to the nearest centroid based on the Euclidean distance.
3. **Update**: The centroids were updated by taking the mean of the points in each cluster.
4. **Convergence**: The algorithm iterates until the centroids no longer change or the maximum number of iterations is reached.

Code Implementation

```
# Function to compute Euclidean distance using PyTorch
def euclidean_distance_gpu(a, b):
    a = a.unsqueeze(1)  # Shape: (n_samples, 1, n_features)
    b = b.unsqueeze(0)  # Shape: (1, n_clusters, n_features)
    return torch.sum((a - b) ** 2, dim=2)
# K-means algorithm using PyTorch
def run_kmeans_gpu(data, k, max_iter=500):
    # Initialize centroids randomly
    centroids = data[torch.randperm(data.shape[0])[:k]]
    for _ in tqdm(range(max_iter), desc="Running k-means"):
        # Compute distances and assign labels
        distances = euclidean_distance_gpu(data, centroids)
        labels = torch.argmin(distances, dim=1)
        # Compute new centroids
        new_centroids = torch.stack([
            data[labels == i].mean(dim=0) if (labels == i).sum() > 0 else centroids[i]
            for i in range(k)
        ])
```

```
        # Check for convergence        if torch.allclose(new_centroids, centroids, atol=1e-8):
            print("K-means converged.")
            break
        centroids = new_centroids
    return centroids.cpu().numpy(), labels.cpu().numpy()
# Test K-means with multiple cluster sizes
k_to_test = [3, 4, 5, 6]
for k in k_to_test:
    print(f"Running k-means with k={k}...")
    centroids, labels = run_kmeans_gpu(x_normalized.cuda(), k)
    visualize_clusters(centroids, labels, x, k)
    print(f"Finished k-means with k={k}.")
```

Visualization of Results

For each k value, the following visualizations were generated:

1. **Centroids**: Representative centroids for each cluster were displayed as images, reshaped into 28x28 matrices.
2. **Cluster Samples**: Random samples from each cluster were displayed in 28x28 image format to visually inspect the grouping.

Results

For each k = 3, 4, 5, 6, the results were as follows:

- **Centroid Images**: The centroids for each cluster (shown as images) represented the "average" of the digits in that cluster. The centroids for different values of k displayed varying levels of differentiation, with more clusters yielding finer distinctions between different handwritten styles.
- **Cluster Samples**: For each value of k, at least 5 sample images from each cluster were displayed. This provided insight into how well the algorithm grouped similar-looking digits.

Conclusion

The K-means algorithm was successful in clustering the digits 1 to 5 into meaningful groups. The quality of the clusters improved with higher values of k, although some overlap in cluster membership was observed when using fewer clusters. This demonstrates how K-means can be used for unsupervised learning tasks in digit classification.

Example Visualizations

Centroid and Sample Images:

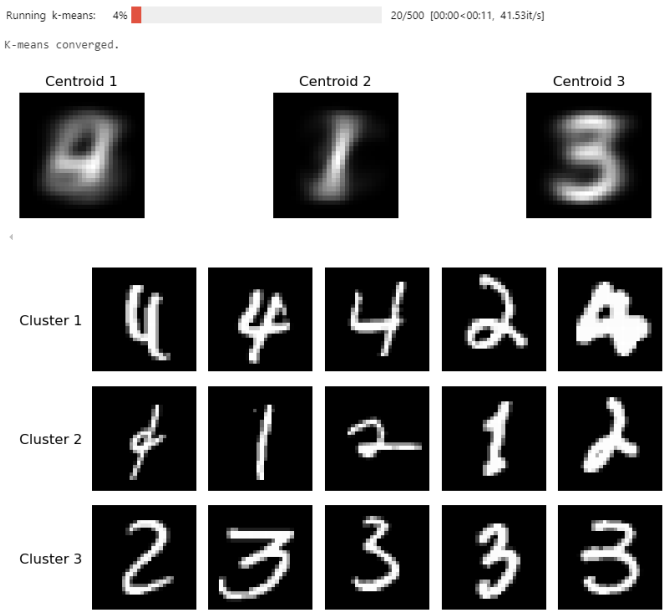


Figure 3- K = 3

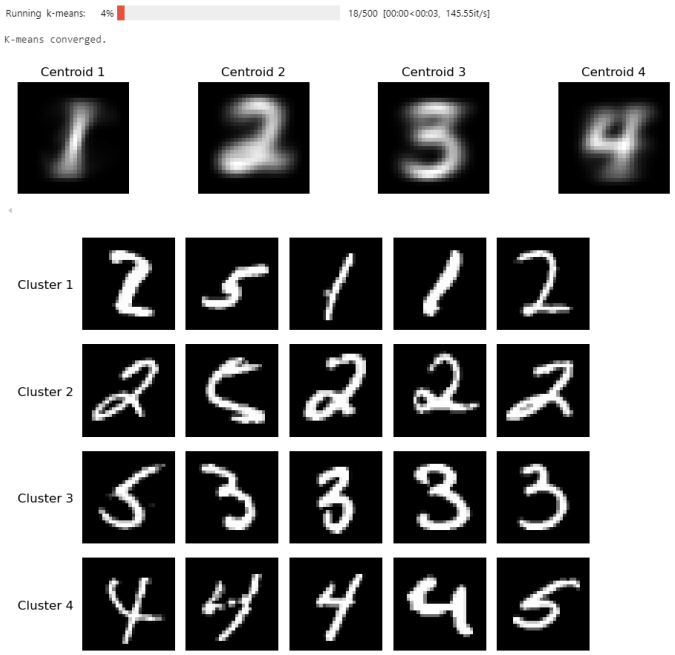


Figure 2- K = 4

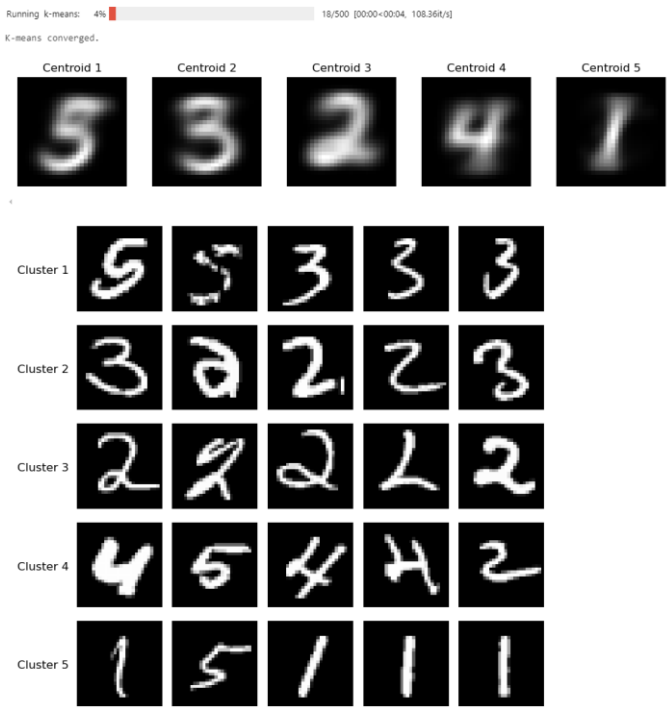


Figure 4- K = 5

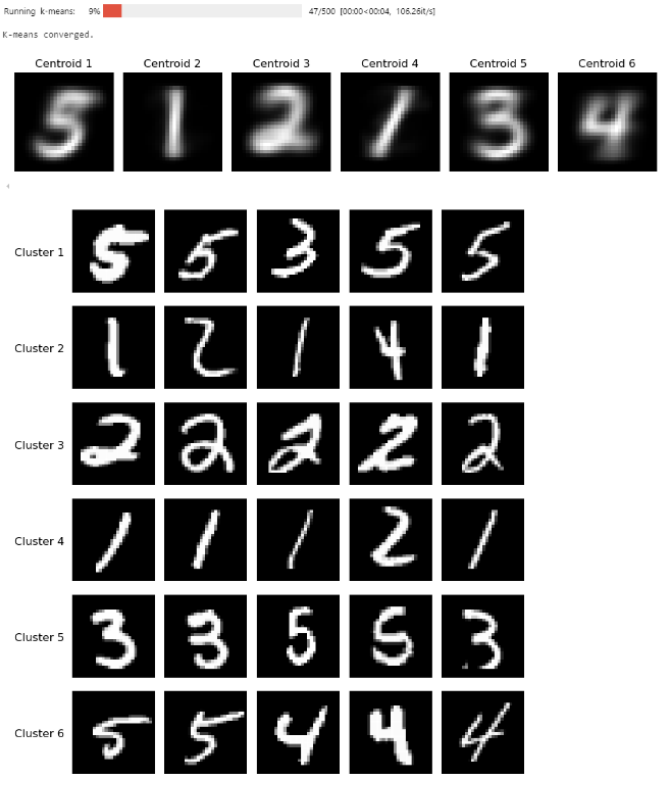


Figure 1- K = 6