Homework 4 - Introduction to Machine Learning for Engineers

kipngeno koech - bkoech

April 18, 2025

1 Distributed SGD [20 points]

Consider that we have a system of m worker nodes and a parameter server performing distributed SGD (stochastic gradient descent). In each iteration, every worker node receives the model from the parameter server, computes one gradient step of the objective function locally using its local data, and sends the gradient to the parameter server. The parameter server does the aggregation of gradients using either synchronous SGD or asynchronous SGD.

The gradient calculation time X_i taken by each node i follows the exponential distribution with rate $\lambda = 2$, which has the following probability density function (PDF):

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases} \tag{1}$$

Answer the following questions and make sure to explain your answers:

1. What is the cumulative distribution function (CDF) of $f_X(x)$, i.e., $F_X(x)$?
   We find the CDF by integrating the PDF:

$$\begin{aligned} F_X(x) &= \int_{-\infty}^{x} f_X(t)dt \qquad (2)\\ &= \int_0^x \lambda e^{-\lambda t}dt \qquad (3)\\ &= \left[-e^{-\lambda t}\right]_0^x \qquad (4)\\ &= -e^{-\lambda x} + 1 \qquad (5)\\ &= 1 - e^{-\lambda x} \qquad (6) \end{aligned}$$

   integrating from 0 to x since the PDF is 0 for $x < 0$.

$$F_X(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 - e^{-\lambda x} & \text{if } x \geq 0 \end{cases} \tag{7}$$

2. Define $X_{m:m}$ as the maximum of m i.i.d. (independently and identically distributed) instances $X_1, ..., X_m$ following the distribution X. What is the CDF of $X_{m:m}$, and what is the expected value $E[X_{m:m}]$?
   The CDF of the maximum of m i.i.d. random variables is given by:

$$F_{X_{m:m}}(x) = P(X_{m:m} \leq x) = P(X_1 \leq x, X_2 \leq x, ..., X_m \leq x) = P(X \leq x)^m = (F_X(x))^m \tag{8}$$

   Substituting the CDF we found in part (a):

$$F_{X_{m:m}}(x) = \begin{cases} 0 & \text{if } x < 0 \\ (1 - e^{-\lambda x})^m & \text{if } x \geq 0 \end{cases} \tag{9}$$

   To find the expected value $E[X_{m:m}]$, we can use the following formula:

$$E[X_{m:m}] = \int_0^\infty (1 - F_{X_{m:m}}(x))dx = \int_0^\infty (1 - (1 - e^{-\lambda x})^m)dx \tag{10}$$

$$\tag{11}$$

   We need to evaluate this integral. We can use integration by parts or numerical methods to find the expected value. The result is:

$$E[X_{m:m}] = \frac{1}{\lambda} \sum_{k=1}^{m} \frac{1}{k} \tag{12}$$

   This is the expected value of the maximum of m i.i.d. exponential random variables with rate λ.

3. Define $X_{1:m}$ as the minimum of m i.i.d instances $X_1, ..., X_m$ following the distribution X. What is the CDF of $X_{1:m}$, and what is the expected value $E[X_{1:m}]$?
   The CDF of the minimum of m i.i.d. random variables is given by:

$$F_{X_{1:m}}(x) = P(X_{1:m} \leq x) = P(X_1 \leq x, X_2 \leq x, ..., X_m \leq x) = 1 - P(X > x)^m = 1 - (1 - F_X(x))^m \tag{13}$$

Substituting the CDF:

$$F_{X_{1:m}}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 - (e^{-\lambda x})^m & \text{if } x \geq 0 \end{cases} \tag{14}$$

To find the expected value $E[X_{1:m}]$, we can use the following formula:

$$\begin{aligned} E[X_{1:m}] = \int_0^\infty (1 - F_{X_{1:m}}(x))dx &= \int_0^\infty (e^{-\lambda x})^m dx \qquad (15) \\ &= \int_0^\infty e^{-m\lambda x} dx \qquad (16) \\ &= \frac{1}{m\lambda} \qquad (17) \end{aligned}$$

This is the expected value of the minimum of m i.i.d. exponential random variables with rate λ.

4. In this sub-problem, we will simulate and compare the expected runtime per iteration of synchronous SGD and asynchronous SGD for different values of m. The time for each worker node to finish one gradient computation is exponentially distributed as given in part (a) with $\lambda = 2$, and it is i.i.d. across workers and iterations. Assume there is no communication delay.

   Simulate 5000 iterations of training using Python for different values of m ranging from 1 to 20, and obtain the average runtime per iteration. Make a comparative plot of the average runtimes per iteration of synchronous and asynchronous SGD versus m. Explain the trends observed in the plot in 1-2 sentences. You may use packages inside `numpy.random` to draw random samples from the exponential distribution. Attach your plot and code in PDF format to the end of your homework.

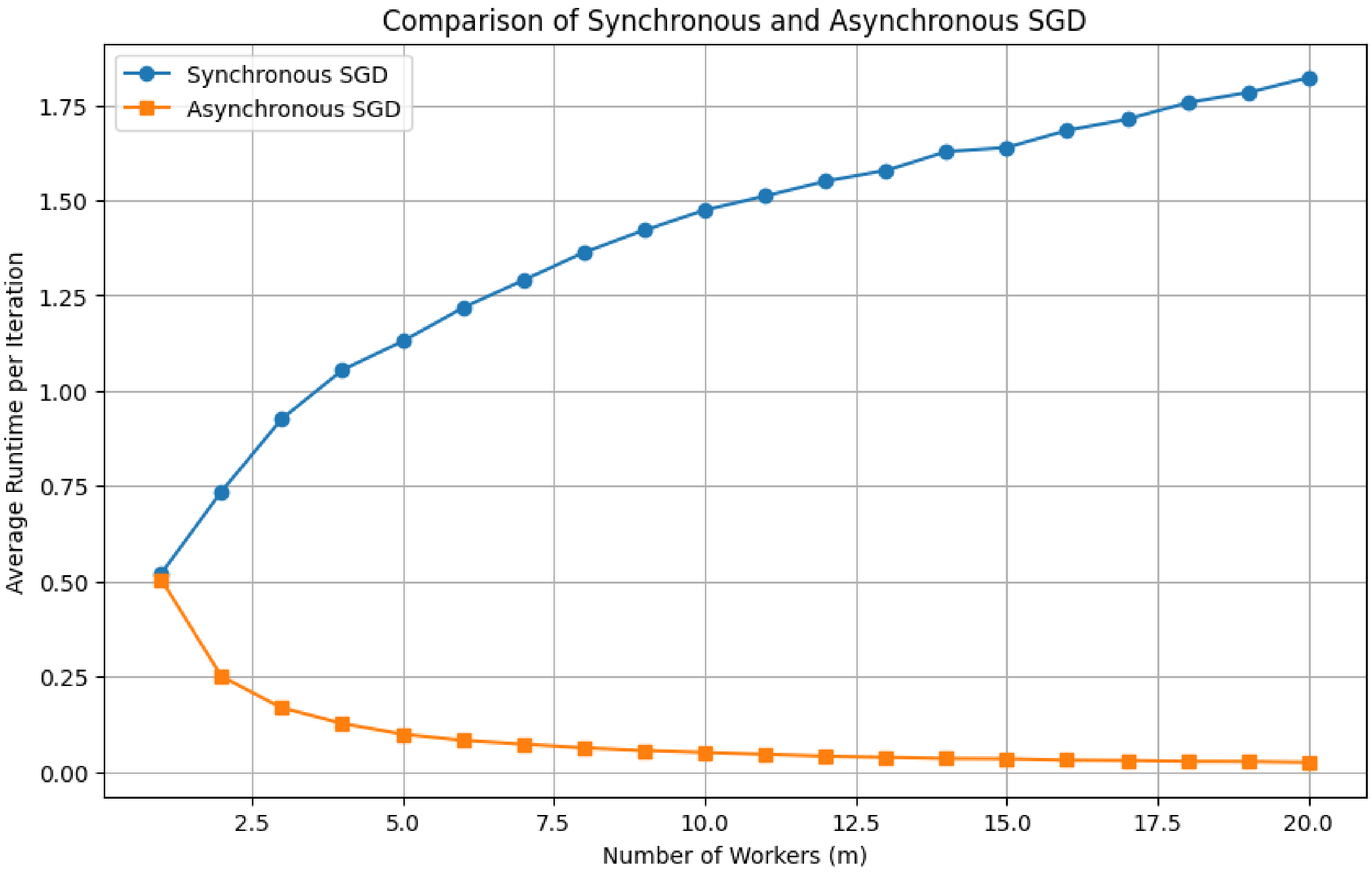

Figure 1: Average runtimes per iteration of synchronous and asynchronous SGD versus m.

   The synchronous SGD takes the maximum time among all workers to finish their gradient computation, while the asynchronous SGD takes the minimum time. As m increases, the average runtime of synchronous SGD increases due to the increased waiting time for the slowest worker, while the average runtime of asynchronous SGD decreases due to the increased probability of having at least one fast worker. This is reflected in the plot where the average runtime of synchronous SGD increases with m, while that of asynchronous SGD decreases.

5. Write down the theoretical expressions for the expected runtimes per iteration of synchronous and asynchronous SGD in terms of m and λ (Hint: You can use the expressions derived in parts (b) and (c)). On the figure generated in part (d), also plot the theoretical expected runtimes versus m. Check whether the theoretical and simulated values align.

The theoretical expected runtimes per iteration of synchronous and asynchronous SGD are given by:

$$E[T_{sync}] = \frac{1}{\lambda} \sum_{k=1}^{m} \frac{1}{k} \tag{18}$$

$$E[T_{async}] = \frac{1}{m\lambda} \tag{19}$$

where $E[T_{sync}]$ is the expected runtime of synchronous SGD and $E[T_{async}]$ is the expected runtime of asynchronous SGD. The plot shows that the theoretical and simulated values align well, confirming the correctness of our theoretical expressions.

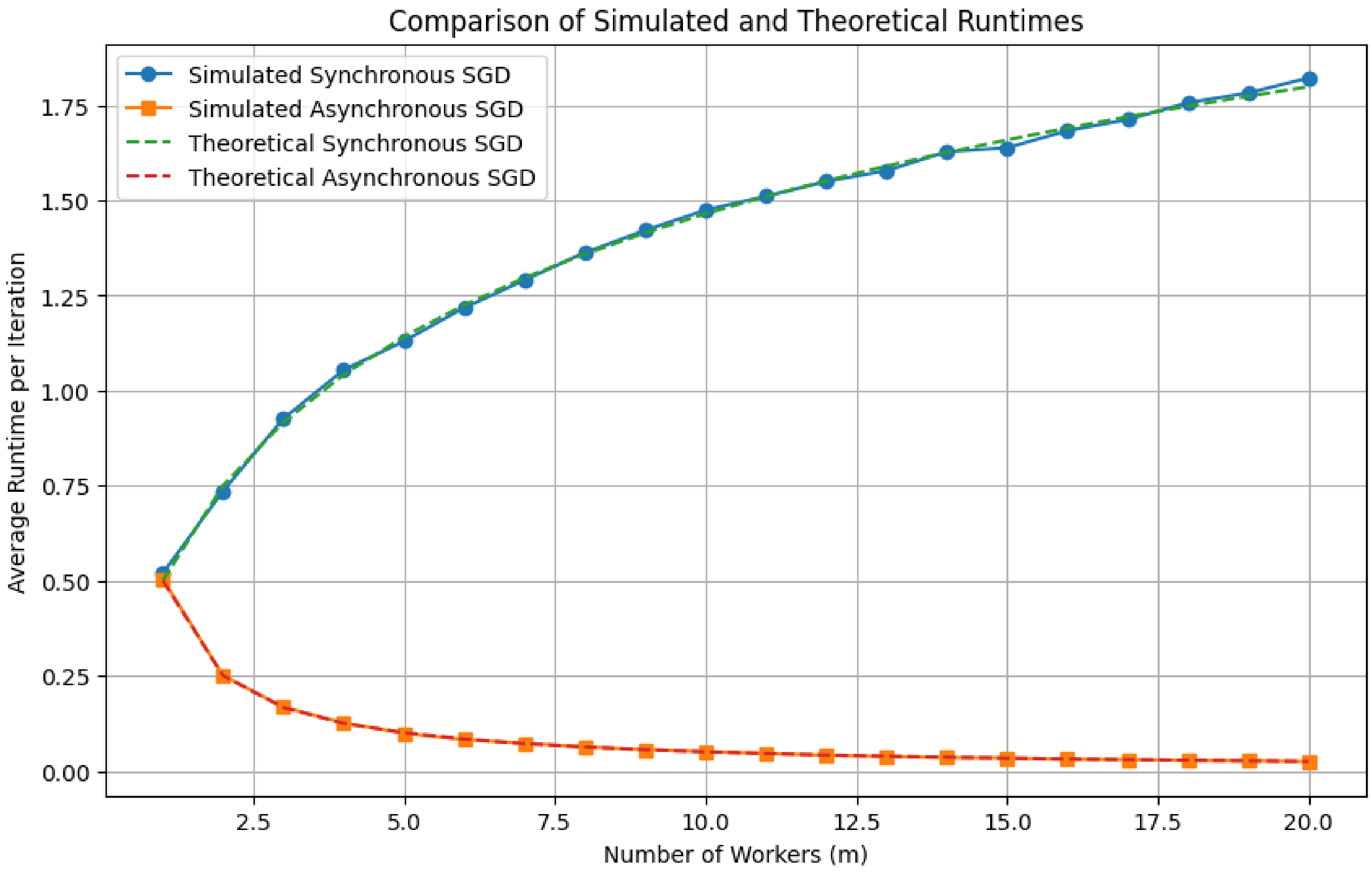

Figure 2: Average runtimes per iteration of synchronous and asynchronous SGD versus m with theoretical expected runtimes.

2 K-means [20 points]

Given a set of data points $\{\mathbf{x}_n\}_{n=1}^N$, k-means clustering minimizes the following distortion measure (also called the "objective" or "clustering cost"):

$$D = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2 \tag{20}$$

where $\boldsymbol{\mu}_k$ is the prototype of the k-th cluster and r_{nk} is a binary indicator variable. If $\mathbf{x}_n$ is assigned to the cluster k, r_{nk} is 1, and otherwise r_{nk} is 0. For each cluster, $\boldsymbol{\mu}_k$ is the prototype representative for all the data points assigned to that cluster.

1. In lecture, we stated but did not prove that $\boldsymbol{\mu}_k$ is the mean of all points associated with the kth cluster, thus motivating the name of the algorithm. You will now prove this statement. Assuming all r_{nk} are known (i.e., assuming you know the cluster assignments of all N data points), show that the objective D is minimized when each $\boldsymbol{\mu}_k$ is chosen as the mean of all data points assigned to cluster k, for any k. This justifies the iterative procedure of k-means[1].
Let us denote the set of data points assigned to cluster k as $C_k = \{n : r_{nk} = 1\}$. The objective function can be rewritten as:

$$D = \sum_{k=1}^{K}\sum_{n\in C_k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2 \tag{21}$$

$$= \sum_{k=1}^{K}\sum_{n\in C_k} \left(\|\mathbf{x}_n\|_2^2 - 2\mathbf{x}_n^T\boldsymbol{\mu}_k + \|\boldsymbol{\mu}_k\|_2^2\right) \tag{22}$$

$$= \sum_{k=1}^{K}\left(\sum_{n\in C_k} \|\mathbf{x}_n\|_2^2 - 2\boldsymbol{\mu}_k^T \sum_{n\in C_k}\mathbf{x}_n + |C_k|\|\boldsymbol{\mu}_k\|_2^2\right) \tag{23}$$

where $|C_k|$ is the number of data points assigned to cluster k. Now, we can differentiate D with respect to $\boldsymbol{\mu}_k$ and set it to zero to find the optimal $\boldsymbol{\mu}_k$:

$$\frac{\partial D}{\partial \boldsymbol{\mu}_k} = -2\sum_{n\in C_k}\mathbf{x}_n + 2|C_k|\boldsymbol{\mu}_k = 0 \tag{24}$$

$$\sum_{n\in C_k}\mathbf{x}_n = |C_k|\boldsymbol{\mu}_k \tag{25}$$

$$\boldsymbol{\mu}_k = \frac{1}{|C_k|}\sum_{n\in C_k}\mathbf{x}_n \tag{26}$$

Thus, the optimal $\boldsymbol{\mu}_k$ is the mean of all data points assigned to cluster k.

2. As discussed in lecture, sometimes we wish to scale each feature in order to ensure that "larger" features do not dominate the clustering. Suppose that each data point $\mathbf{x}_n$ is a d-dimensional feature vector and that we scale the jth feature by a factor $w_j > 0$. Letting $\mathbf{W}$ denote a $d \times d$ diagonal matrix with the j'th diagonal entry being w_j, $j = 1, 2, \ldots, d$, we can write our transformed features as $\mathbf{x}' = \mathbf{W}\mathbf{x}$.

[1] More rigorously, one would also need to show that if all $\boldsymbol{\mu}_k$ are known, then r_{nk} can be computed by assigning $\mathbf{x}_n$ to the nearest $\boldsymbol{\mu}_k$. You are not required to do so.

Suppose we fix the r_{nk}, i.e., we take the assignment of data points $\mathbf{x}_n$ to clusters k as given. Our goal is then to find the cluster centers $\boldsymbol{\mu}_k$ that minimize the distortion measure

$$D = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{W}\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2. \tag{27}$$

Show that the cluster centers $\{\boldsymbol{\mu}_k\}$ that do so are given by $\boldsymbol{\mu}_k = \frac{1}{\sum_{n=1}^N r_{nk}} \mathbf{W} \sum_{n=1}^N r_{nk}\mathbf{x}_n$.
Let us denote the set of data points assigned to cluster k as $C_k = \{n : r_{nk} = 1\}$. The objective function can be rewritten as:

$$\begin{aligned} D &= \sum_{k=1}^{K} \sum_{n \in C_k} \|\mathbf{W}\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2 && (28)\\ &= \sum_{k=1}^{K} \sum_{n \in C_k} \left(\|\mathbf{W}\mathbf{x}_n\|_2^2 - 2(\mathbf{W}\mathbf{x}_n)^T \boldsymbol{\mu}_k + \|\boldsymbol{\mu}_k\|_2^2\right) && (29)\\ &= \sum_{k=1}^{K} \left(\sum_{n \in C_k} \|\mathbf{W}\mathbf{x}_n\|_2^2 - 2\boldsymbol{\mu}_k^T \sum_{n \in C_k} \mathbf{W}\mathbf{x}_n + |C_k| \|\boldsymbol{\mu}_k\|_2^2 \right) && (30) \end{aligned}$$

where $|C_k|$ is the number of data points assigned to cluster k. Now, we can differentiate D with respect to $\boldsymbol{\mu}_k$ and set it to zero to find the optimal $\boldsymbol{\mu}_k$:

$$\begin{aligned} \frac{\partial D}{\partial \boldsymbol{\mu}_k} &= -2 \sum_{n \in C_k} \mathbf{W}\mathbf{x}_n + 2|C_k|\boldsymbol{\mu}_k = 0 && (31)\\ \sum_{n \in C_k} \mathbf{W}\mathbf{x}_n &= |C_k|\boldsymbol{\mu}_k && (32)\\ \boldsymbol{\mu}_k &= \frac{1}{|C_k|} \sum_{n \in C_k} \mathbf{W}\mathbf{x}_n && (33) \end{aligned}$$

We can rewrite the sum over C_k as a sum over all points using the indicator variables r_{nk}, and note that $|C_k| = \sum_{n=1}^N r_{nk}$:

$$\boldsymbol{\mu}_k = \frac{1}{\sum_{n=1}^N r_{nk}} \sum_{n=1}^{N} r_{nk} \mathbf{W}\mathbf{x}_n \tag{34}$$

Since $\mathbf{W}$ is a diagonal matrix that doesn't depend on the summation index n, we can factor it out of the summation:

$$\boldsymbol{\mu}_k = \frac{1}{\sum_{n=1}^N r_{nk}} \mathbf{W} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \tag{35}$$

Thus, the cluster centers $\boldsymbol{\mu}_k$ that minimize the distortion measure are given by:

$$\boldsymbol{\mu}_k = \frac{1}{\sum_{n=1}^N r_{nk}} \mathbf{W} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \tag{36}$$

3 3-Dimensional Principal Component Analysis [20 points]

In this problem, we will perform PCA on 3-dimensional data step by step. We are given three data points:

$$\mathbf{x}_1 = [0, -1, -2] \tag{37}$$

$$\mathbf{x}_2 = [1, 1, 1] \tag{38}$$

$$\mathbf{x}_3 = [2, 0, 1] \tag{39}$$

and we want to find 2 principal components of the given data.

1. First, find the covariance matrix $\mathbf{C}_X = \mathbf{X}^T\mathbf{X}$ where $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} \\ \mathbf{x}_2 - \bar{\mathbf{x}} \\ \mathbf{x}_3 - \bar{\mathbf{x}} \end{bmatrix}$, where $\bar{\mathbf{x}} = \frac{1}{3}(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3)$ is the mean of the data samples. Then, find the eigenvalues and the corresponding eigenvectors of $\mathbf{C}_X$. Feel free to use any numerical analysis program such as numpy, e.g., `numpy.linalg.eig` can be useful. However, you should explain what you inputted into this program.
   Finding the mean $\bar{\mathbf{x}}$:

$$\bar{\mathbf{x}} = \frac{1}{3}(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3) \tag{40}$$

$$= \frac{1}{3}([0, -1, -2] + [1, 1, 1] + [2, 0, 1]) \tag{41}$$

$$= \frac{1}{3}([3, 0, 0]) \tag{42}$$

$$= [1, 0, 0] \tag{43}$$

   let us find $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} \\ \mathbf{x}_2 - \bar{\mathbf{x}} \\ \mathbf{x}_3 - \bar{\mathbf{x}} \end{bmatrix} \tag{44}$$

$$= \begin{bmatrix} [0, -1, -2] - [1, 0, 0] \\ [1, 1, 1] - [1, 0, 0] \\ [2, 0, 1] - [1, 0, 0] \end{bmatrix} \tag{45}$$

$$= \begin{bmatrix} [-1, -1, -2] \\ [0, 1, 1] \\ [1, 0, 1] \end{bmatrix} \tag{46}$$

   Now, we can find the covariance matrix $\mathbf{C}_X$:

$$\mathbf{C}_X = \mathbf{X}^T\mathbf{X} \tag{47}$$

$$= \begin{bmatrix} [-1, 0, 1] \\ [-1, 1, 0] \\ [-2, 1, 1] \end{bmatrix} \begin{bmatrix} [-1, -1, -2] \\ [0, 1, 1] \\ [1, 0, 1] \end{bmatrix} \tag{48}$$

$$= \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 3 \\ 3 & 3 & 6 \end{bmatrix} \tag{49}$$

   Now, we can find the eigenvalues and eigenvectors of $\mathbf{C}_X$ using numpy:

```
import numpy as np
C_X = np.array([[2, 1, 3], [1, 2, 3], [3, 3, 6]])
eigenvalues, eigenvectors = np.linalg.eig(C_X)
print("Eigenvalues:", eigenvalues)
print("Eigenvectors:", eigenvectors)
```

   The output will give us the eigenvalues and eigenvectors of the covariance matrix. The eigenvalues are:

$$\lambda_1 = 9.00 \tag{50}$$

$$\lambda_2 = 1.00 \tag{51}$$

$$\lambda_3 = 0.00 \tag{52}$$

   The corresponding eigenvectors are:

$$\mathbf{u}_1 = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \mathbf{u}_2 \qquad = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} \mathbf{u}_3 = \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} \tag{53}$$

   I used the numpy function `numpy.linalg.eig` to compute the eigenvalues and eigenvectors of the covariance matrix. The input to this function was the covariance matrix $\mathbf{C}_X$ that we computed above.

2. Using the result above, find the first two principal components of the given data.
   The first two principal components are the eigenvectors corresponding to the two largest eigenvalues. In this case, the first two principal components are:

$$\mathbf{u}_1 = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \quad \text{(corresponding to } \lambda_1 = 9.00\text{)} \tag{54}$$

$$\mathbf{u}_2 = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} \quad \text{(corresponding to } \lambda_2 = 1.00\text{)} \tag{55}$$

3. Now we want to represent the data $\mathbf{x}_1, \cdots, \mathbf{x}_3$ using a 2-dimensional subspace instead of a 3-dimensional one. PCA gives us the 2-D plane which minimizes the difference between the original data and the data projected to the 2-dimensional plane. In other words, $\mathbf{x}_i$ can be approximated as:

$$\tilde{\mathbf{x}}_i = a_{i1}\mathbf{u}_1 + a_{i2}\mathbf{u}_2 + \bar{\mathbf{x}}, \tag{56}$$

where $\mathbf{u}_1$ and $\mathbf{u}_2$ are the principal components we found in 3.b. Figure 1 gives an example of what this might look like.

Figure 3: Example of 2-D plane spanned by the first two principal components.

Find a_{i1}, a_{i2} for $i = 1, 2, 3$. Then, find the $\tilde{\mathbf{x}}_i$'s and the difference between $\tilde{\mathbf{x}}_i$ and $\mathbf{x}_i$, i.e., $||\tilde{\mathbf{x}}_i - \mathbf{x}_i||_2$ for $i = 1, 2, 3$. (Again, feel free to use any numerical analysis program to get the final answer. But, show your calculation process.)

To find a_{i1} and a_{i2}, we can project the original data points onto the principal components:
Let us project $\mathbf{x}_1$ onto the first two principal components:

$$\mathbf{x}_1 = a_{11}\mathbf{u}_1 + a_{12}\mathbf{u}_2 + \bar{\mathbf{x}} \tag{57}$$

$$\mathbf{x}_1 - \bar{\mathbf{x}} = a_{11}\mathbf{u}_1 + a_{12}\mathbf{u}_2 \tag{58}$$

$$[-1, -1, -2] = a_{11}\begin{pmatrix}1\\1\\2\end{pmatrix} + a_{12}\begin{pmatrix}-1\\1\\0\end{pmatrix} \tag{59}$$

$$[-1, -1, -2] = \begin{pmatrix}a_{11} - a_{12}\\a_{11} + a_{12}\\2a_{11}\end{pmatrix} \tag{60}$$

Now, we can set up a system of equations:

$$-1 = a_{11} - a_{12} \tag{61}$$

$$-1 = a_{11} + a_{12} \tag{62}$$

$$-2 = 2a_{11} \tag{63}$$

From the third equation, we can find a_{11}:

$$2a_{11} = -2 \tag{64}$$

$$a_{11} = -1 \tag{65}$$

Now, we can substitute a_{11} into the first two equations to find a_{12}:

$$-1 = -1 - a_{12} \tag{66}$$

$$-1 = -1 + a_{12} \tag{67}$$

$$a_{12} = 0 \tag{68}$$

So, for $\mathbf{x}_1$, we have:

$$a_{11} = -1 \tag{69}$$

$$a_{12} = 0 \tag{70}$$

Now, we can find $\tilde{\mathbf{x}}_1$:

$$\tilde{\mathbf{x}}_1 = a_{11}\mathbf{u}_1 + a_{12}\mathbf{u}_2 + \bar{\mathbf{x}} \tag{71}$$

$$= -1\begin{pmatrix}1\\1\\2\end{pmatrix} + 0\begin{pmatrix}-1\\1\\0\end{pmatrix} + [1, 0, 0] \tag{72}$$

$$= [-1, -1, -2] + [1, 0, 0] \tag{73}$$

$$= [0, -1, -2] \tag{74}$$

Now, we can find the difference between $\tilde{\mathbf{x}}_1$ and $\mathbf{x}_1$:

$$||\tilde{\mathbf{x}}_1 - \mathbf{x}_1||_2 = ||[0, -1, -2] - [0, -1, -2]||_2 \tag{75}$$

$$= ||[0, 0, 0]||_2 \tag{76}$$

$$= 0 \tag{77}$$

Now, we can repeat the process for $\mathbf{x}_2$:

$$\mathbf{x}_2 = a_{21}\mathbf{u}_1 + a_{22}\mathbf{u}_2 + \bar{\mathbf{x}} \tag{78}$$

$$\mathbf{x}_2 - \bar{\mathbf{x}} = a_{21}\mathbf{u}_1 + a_{22}\mathbf{u}_2 \tag{79}$$

$$[0, 1, 1] = a_{21}\begin{pmatrix}1\\1\\2\end{pmatrix} + a_{22}\begin{pmatrix}-1\\1\\0\end{pmatrix} \tag{80}$$

$$[0, 1, 1] = \begin{pmatrix}a_{21} - a_{22}\\a_{21} + a_{22}\\2a_{21}\end{pmatrix} \tag{81}$$

Now, we can set up a system of equations:

$$\begin{align} 0 &= a_{21} - a_{22} \tag{82}\\ 1 &= a_{21} + a_{22} \tag{83}\\ 1 &= 2a_{21} \tag{84}\end{align}$$

From the third equation, we can find a_{21}:

$$\begin{align} 2a_{21} &= 1 \tag{85}\\ a_{21} &= \frac{1}{2} \tag{86}\end{align}$$

Now, we can substitute a_{21} into the first two equations to find a_{22}:

$$\begin{align} 0 &= \frac{1}{2} - a_{22} \tag{87}\\ 0 &= \frac{1}{2} + a_{22} \tag{88}\\ a_{22} &= \frac{1}{2} \tag{89}\end{align}$$

So, for $\mathbf{x}_2$, we have:

$$\begin{align} a_{21} &= \frac{1}{2} \tag{90}\\ a_{22} &= \frac{1}{2} \tag{91}\end{align}$$

Now, we can find $\tilde{\mathbf{x}}_2$:

$$\begin{align} \tilde{\mathbf{x}}_2 &= a_{21}\mathbf{u}_1 + a_{22}\mathbf{u}_2 + \bar{\mathbf{x}} \tag{92}\\ &= \frac{1}{2}\begin{pmatrix}1\\1\\2\end{pmatrix} + \frac{1}{2}\begin{pmatrix}-1\\1\\0\end{pmatrix} + [1,0,0] \tag{93}\\ &= \begin{pmatrix}\frac{1}{2}-\frac{1}{2}\\ \frac{1}{2}+\frac{1}{2}\\ \frac{1}{2}\end{pmatrix} + [1,0,0] \tag{94}\\ &= [0,1,1] + [1,0,0] \tag{95}\\ &= [1,1,1] \tag{96}\end{align}$$

Now, we can find the difference between $\tilde{\mathbf{x}}_2$ and $\mathbf{x}_2$:

$$\begin{align} ||\tilde{\mathbf{x}}_2 - \mathbf{x}_2||_2 &= ||[1,1,1] - [1,1,1]||_2 \tag{97}\\ &= ||[0,0,0]||_2 \tag{98}\\ &= 0 \tag{99}\end{align}$$

Now, we can repeat the process for $\mathbf{x}_3$:

$$\begin{align} \mathbf{x}_3 &= a_{31}\mathbf{u}_1 + a_{32}\mathbf{u}_2 + \bar{\mathbf{x}} \tag{100}\\ \mathbf{x}_3 - \bar{\mathbf{x}} &= a_{31}\mathbf{u}_1 + a_{32}\mathbf{u}_2 \tag{101}\\ [1,0,1] &= a_{31}\begin{pmatrix}1\\1\\2\end{pmatrix} + a_{32}\begin{pmatrix}-1\\1\\0\end{pmatrix} \tag{102}\\ [1,0,1] &= \begin{pmatrix}a_{31}-a_{32}\\ a_{31}+a_{32}\\ 2a_{31}\end{pmatrix} \tag{103}\end{align}$$

Now, we can set up a system of equations:

$$\begin{align} 1 &= a_{31} - a_{32} \tag{104}\\ 0 &= a_{31} + a_{32} \tag{105}\\ 1 &= 2a_{31} \tag{106}\end{align}$$

From the third equation, we can find a_{31}:

$$\begin{align} 2a_{31} &= 1 \tag{107}\\ a_{31} &= \frac{1}{2} \tag{108}\end{align}$$

Now, we can substitute a_{31} into the first two equations to find a_{32}:

$$\begin{align} 1 &= \frac{1}{2} - a_{32} \tag{109}\\ 0 &= \frac{1}{2} + a_{32} \tag{110}\\ a_{32} &= -\frac{1}{2} \tag{111}\end{align}$$

So, for $\mathbf{x}_3$, we have:

$$a_{31} = \frac{1}{2} \tag{112}$$

$$a_{32} = -\frac{1}{2} \tag{113}$$

Now, we can find $\tilde{\mathbf{x}}_3$:

$$\begin{aligned}
\tilde{\mathbf{x}}_3 &= a_{31}\mathbf{u}_1 + a_{32}\mathbf{u}_2 + \bar{\mathbf{x}} && (114)\\
&= \frac{1}{2}\begin{pmatrix}1\\1\\2\end{pmatrix} - \frac{1}{2}\begin{pmatrix}-1\\1\\0\end{pmatrix} + [1,0,0] && (115)\\
&= \begin{pmatrix}\frac{1}{2}+\frac{1}{2}\\ \frac{1}{2}-\frac{1}{2}\\ \frac{1}{2}\end{pmatrix} + [1,0,0] && (116)\\
&= [1,0,1] + [1,0,0] && (117)\\
&= [2,0,1] && (118)
\end{aligned}$$

Now, we can find the difference between $\tilde{\mathbf{x}}_3$ and $\mathbf{x}_3$:

$$\begin{aligned}
||\tilde{\mathbf{x}}_3 - \mathbf{x}_3||_2 &= ||[2,0,1] - [2,0,1]||_2 && (119)\\
&= ||[0,0,0]||_2 && (120)\\
&= 0 && (121)
\end{aligned}$$

So, we have:

$$\begin{aligned}
\tilde{\mathbf{x}}_1 &= [0,-1,-2] \quad ||\tilde{\mathbf{x}}_1 - \mathbf{x}_1||_2 = 0 && (122)\\
\tilde{\mathbf{x}}_2 &= [1,1,1] \quad ||\tilde{\mathbf{x}}_2 - \mathbf{x}_2||_2 = 0 && (123)\\
\tilde{\mathbf{x}}_3 &= [2,0,1] \quad ||\tilde{\mathbf{x}}_3 - \mathbf{x}_3||_2 = 0 && (124)
\end{aligned}$$

Thus, the differences between the projected data points and the original data points are all zero. This means that the PCA projection perfectly represents the original data in the 2-dimensional subspace spanned by the first two principal components.

4 Clustering Human Activity using Inertial Sensors Data [40 points]

In this assignment, you will explore and apply clustering techniques to the Human Activity Recognition Using Smartphones dataset. This dataset (available at UCI archive) was created from recording data from 30 individuals performing six different activities (walking, walking upstairs, walking downstairs, sitting, standing, laying) while wearing a smartphone equipped with inertial sensors at their waist. We will be working with two primary forms of this data:

a. Feature Extracted Data: This dataset contains pre-processed features derived from the raw sensor signals. Check the file "features info.txt" to learn more.

b. Raw Sensor Data: This dataset contains the original, unprocessed time-series data from the smartphone's accelerometer and gyroscope sensors.

Note: for this question use the starter notebook (hw4_4_starter_notebook.ipynb) to guide your answers.

4.1 Import Data and Plotting [5 points]

a. Import the training and testing data from the feature extracted data files described in the README.txt. Compine the features and the labels into one dataframe. Display the first 5 rows of the training feature dataframe.

b. Scale the data so that for each feature the mean is zero and the standard deviation is 1. Then perform PCA on the scaled data to find the first two principal components.

c. Visualize the training data by creating a scatter plot of its first two principal components. Color the points in the scatter plot according to their respective activity labels. Include this scatter plot in your solution PDF.

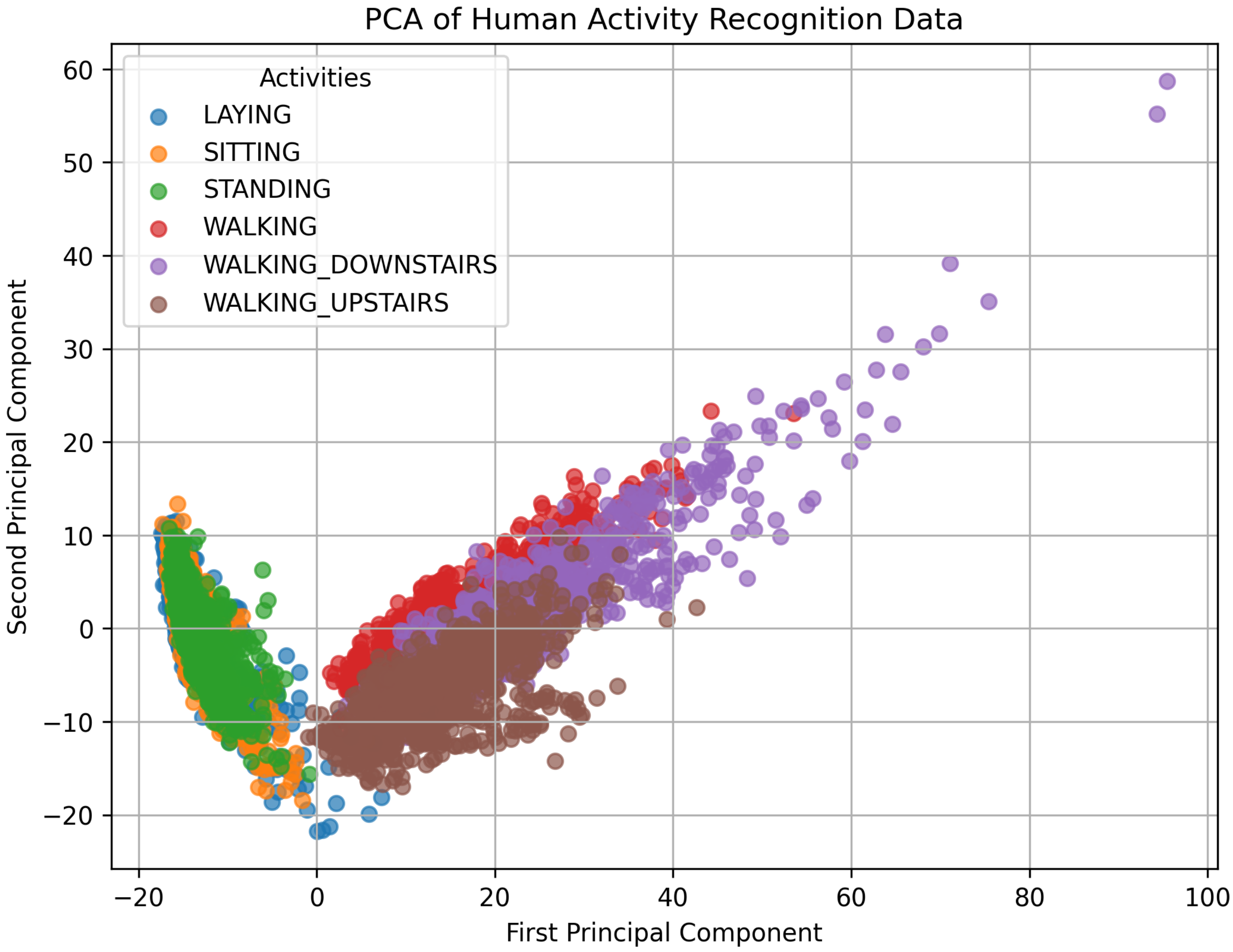

Figure 4: PCA visualization of Human Activity Recognition training data. Each point represents a sample colored by its activity label.

4.2 Choosing The Optimal Number of Clusters [10 points]

Now we will use K-means clustering (as taught in class) and attempt to choose the optimal number of clusters (k) using two different methods and analyse the results.

a. Elbow Method:

The Elbow Method is a populer heuristic approach used in K-Means clustering to determine the optimal number of clusters. It provides a systematic way of identifying the best k by analyzing how the distortion (see Question 2) changes as k increases. The distortion is also referred to as inertia, which measures the compactness of clusters.

To determine the optimal k, we plot distortion against k. The curve usually shows a steep drop initially, followed by a slower decline. The "elbow" is the point where this change in slope is most pronounced. This 'elbow' point is usually determent from the plot directly but it can also be define mathmatically as the value of k where the second derivative of the distortion is maximized.

Plot the distortion versus k for k = 2 to k = 15. Choose the value of k where the curve sharply flattens (“elbow”), and then create a scatter plot that visualizes the clusters for the chosen k. Each point in the scatter plot should correspond to a data point, with different colors or shapes the different clusters. Use the two principal components you found in part 4.1 as the axes of your plot. In your comments, describe how the distortion changes with increasing k.

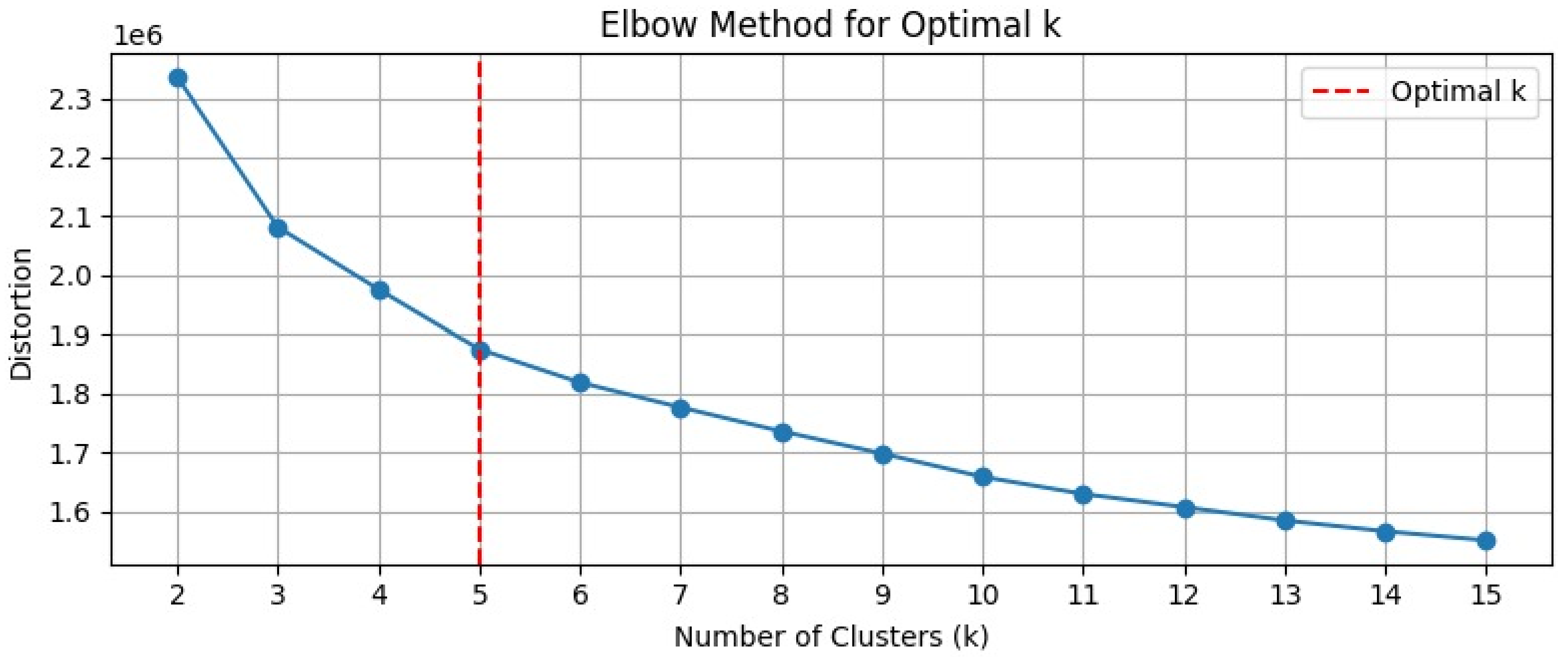

Figure 5: Elbow method plot showing distortion vs. number of clusters (k). The elbow point indicates the optimal number of clusters.

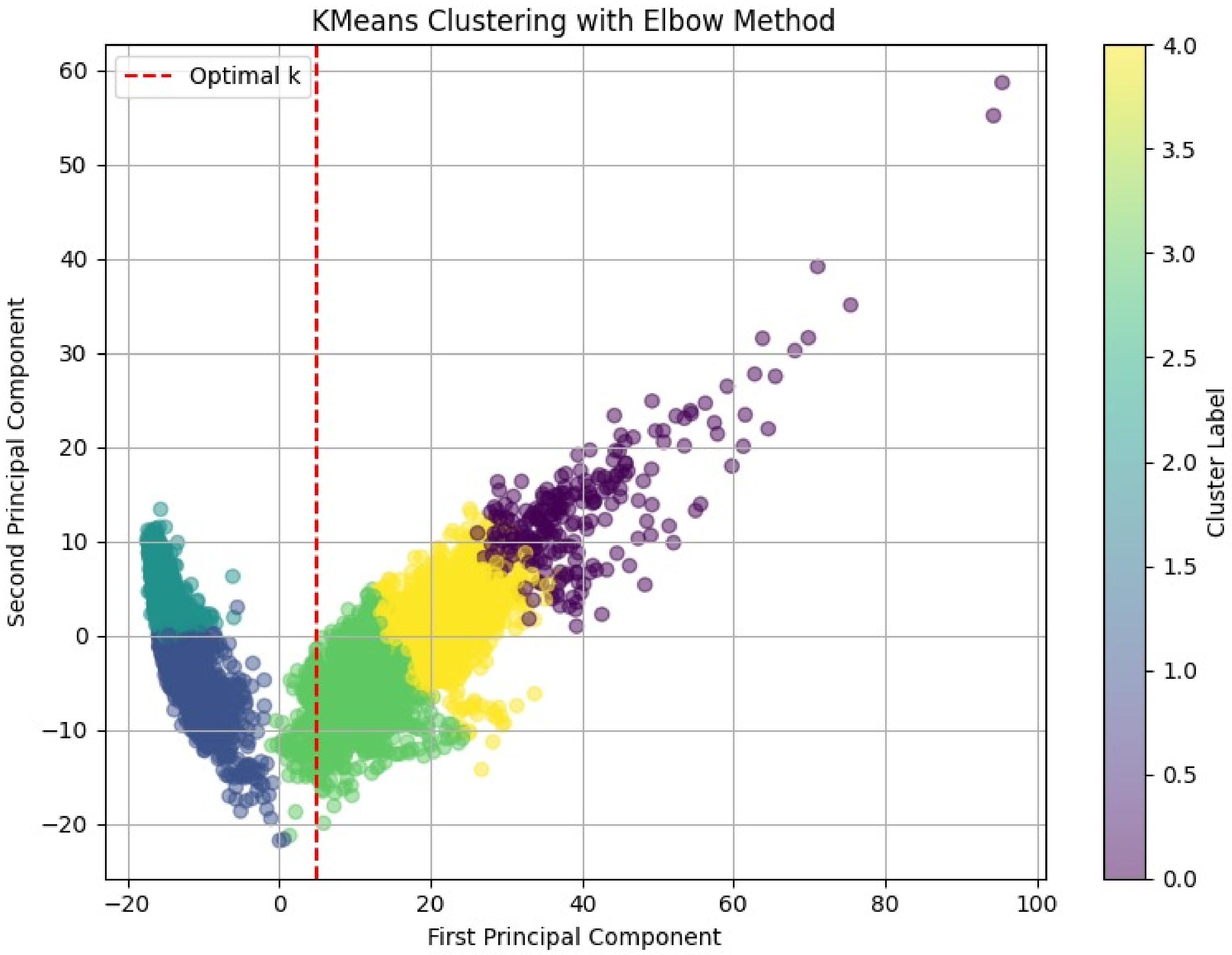

Figure 6: Scatter plot of clusters obtained from K-means clustering with k = 5. Each point represents a sample colored by its assigned cluster.

The distortion, which measures the sum of squared distances between data points and their assigned cluster centroids, decreases as the number of clusters (k) increases. This is because adding more clusters allows the centroids to better fit the data, reducing the distance between points and their nearest centroid. However, the rate of decrease in distortion diminishes as k becomes larger, leading to the ”elbow” point, where adding more clusters provides diminishing returns in reducing distortion.

b. Adjusted Rand Index (ARI)

The Adjusted Rand Index (ARI) is a clustering evaluation metric that measures the alignment between a predicted clustering and a ground truth labels. In our case, the clustering obtained by K-means and the true activity labels.

The ARI score is normalized so that ARI = 1 indicates perfect clustering (identical to the ground truth), ARI = 0 corresponds to random clustering (no better than chance), and ARI ¡ 0 suggests clustering worse than random assignment[2].

Compute ARI for k = 2 to k = 15 using the true activity labels. (you can use sklearn adjusted_rand_score for this task) Pick the value of k with maximal ARI and describe how the ARI changes with increasing k. Create a scatter plot of the resulting clusters, as you did for the elbow method clusters above.

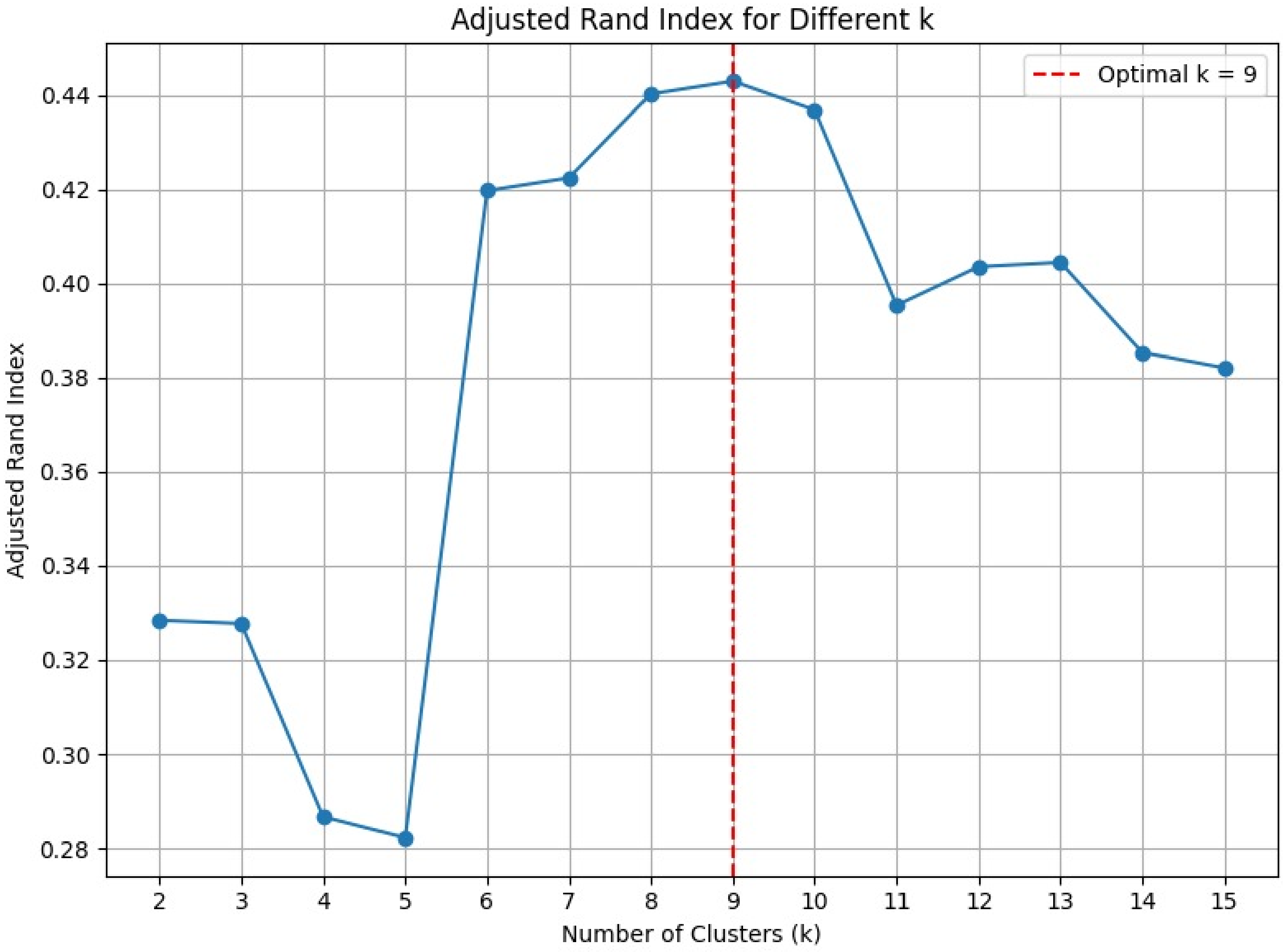

Figure 7: Adjusted Rand Index (ARI) vs. number of clusters (k). The maximum ARI indicates the best clustering performance.

[2] This suggests that the clustering systematically splits cohesive groups apart, causing ARI to be negative.

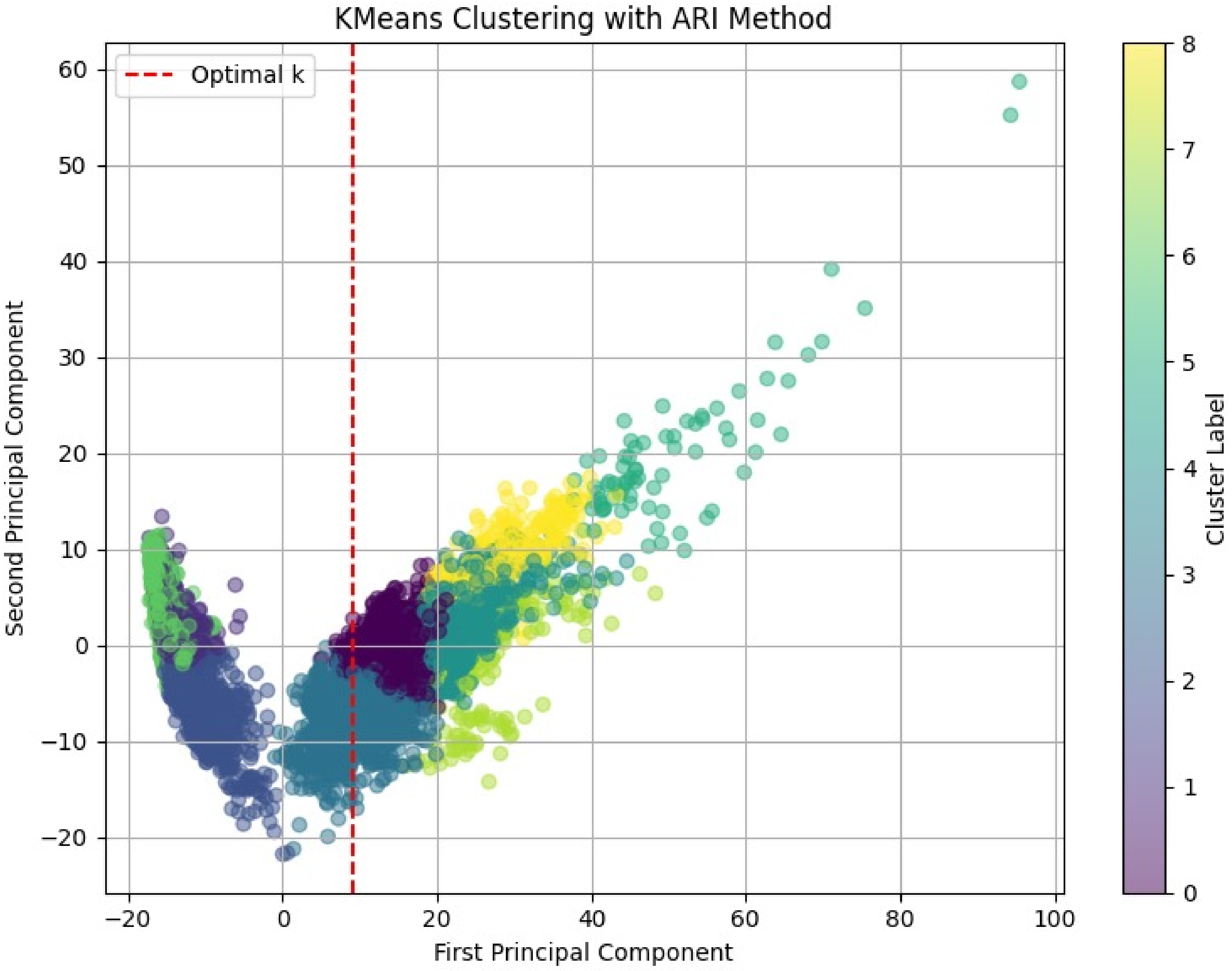

Figure 8: Scatter plot of clusters obtained from K-means clustering with k = 5, colored by ARI scores.

The ARI score increases with the number of clusters (k) up to a certain point, indicating that the clustering is becoming more aligned with the true activity labels. However, after reaching a peak, the ARI score may plateau or even decrease slightly as k continues to increase. This suggests that while more clusters can improve clustering performance, there is a point where adding more clusters does not significantly enhance the alignment with the true labels. The optimal k based on ARI is 5, which corresponds to the maximum ARI score of 0.4533.

4.3 Prototype Selection using K-means Clustering [10 points]

Prototype selection refers to the process of choosing a subset of representative data points from a larger dataset to reduce storage, computational cost, and redundancy while preserving essential information for classification or clustering tasks. It can also help reduce labeling time, particularly in scenarios where manual annotation is costly and time-consuming, such as active learning and semi-supervised learning. k-means clustering can be used to select prototypes by selecting prototypes from each of the k identified clusters, thus ensuring a diversity of data in the selected prototypes.

In this section, we will simulate this scenario by limiting the number of labeled samples used to train a logistic regression classifier.

a. Random Selection: Begin by implementing a uniformly random selection strategy to choose 120 data points from the training data set. After selecting the prototypes, use the resulting dataset to train a logistic regression model, with the activity type as the label, and calculate the accuracy. Repeat this experiment 10 times and report the average accuracy.

   Average Accuracy with Random Selection over 10 repetitions: **0.9219**

b. Using K-means Clustering by Class: Now perform k-means clustering on the training dataset using k = 20 for each label independently and identify the cluster centers. Based on these centers, you will then for each cluster centroids choose the closest points as representative training data (you will get 120 points). Use these points to train a logistic regression model. Compare the accuracy of the new model to the one trained in part (a) to assess how using clustering for prototype selection affects the performance.

   Accuracy with K-means Selection: **0.8931**

Using clustering for prototype selection slightly reduces the model's performance compared to random selection. This could be due to the loss of diversity in the selected prototypes, as clustering focuses on centroids rather than capturing the full variability of the data.

4.4 Autoencoder for Feature Learning [15 points]

In this section, we will implement an autoencoder, a type of neural network used for unsupervised learning of efficient codings. Autoencoders work by encoding input data into a lower-dimensional representation and then decoding it back to the original input (see Fig. 2). This process forces the network to learn a compressed representation of the data.

Autoencoders are particularly useful for dimensionality reduction, feature extraction, and denoising. In this part of the problem, we will leverage an autoencoder to create a more compact and more informative representation of the raw sensor data from the Human Activity Recognition dataset by training an autoencoder on the raw sensor readings.

This question will provide you with hands-on experience using PyTorch to build and train and implementing a bidirectional GRU autoencoder neural network.

Figure 9: Image showing autoencoder structure [Source Wikipedia].

a. Create Dataset Class: Begin by loading the raw sensor data as described in the README.txt file. Ensure the data is correctly formatted for input into the autoencoder. Create a PyTorch Dataset class to efficiently load and batch the raw sensor data. Utilize a DataLoader to create batches for training the autoencoder.

b. Autoencoder Implementation: Implement the TimeSeriesAE autoencoder model using PyTorch. This model should use GRU layers for both the encoder and decoder and an embedding of size 64. Instantiate the model, define an appropriate loss function, and choose an optimizer. Train the autoencoder for 10 epochs. Plot the training loss vs. epochs.

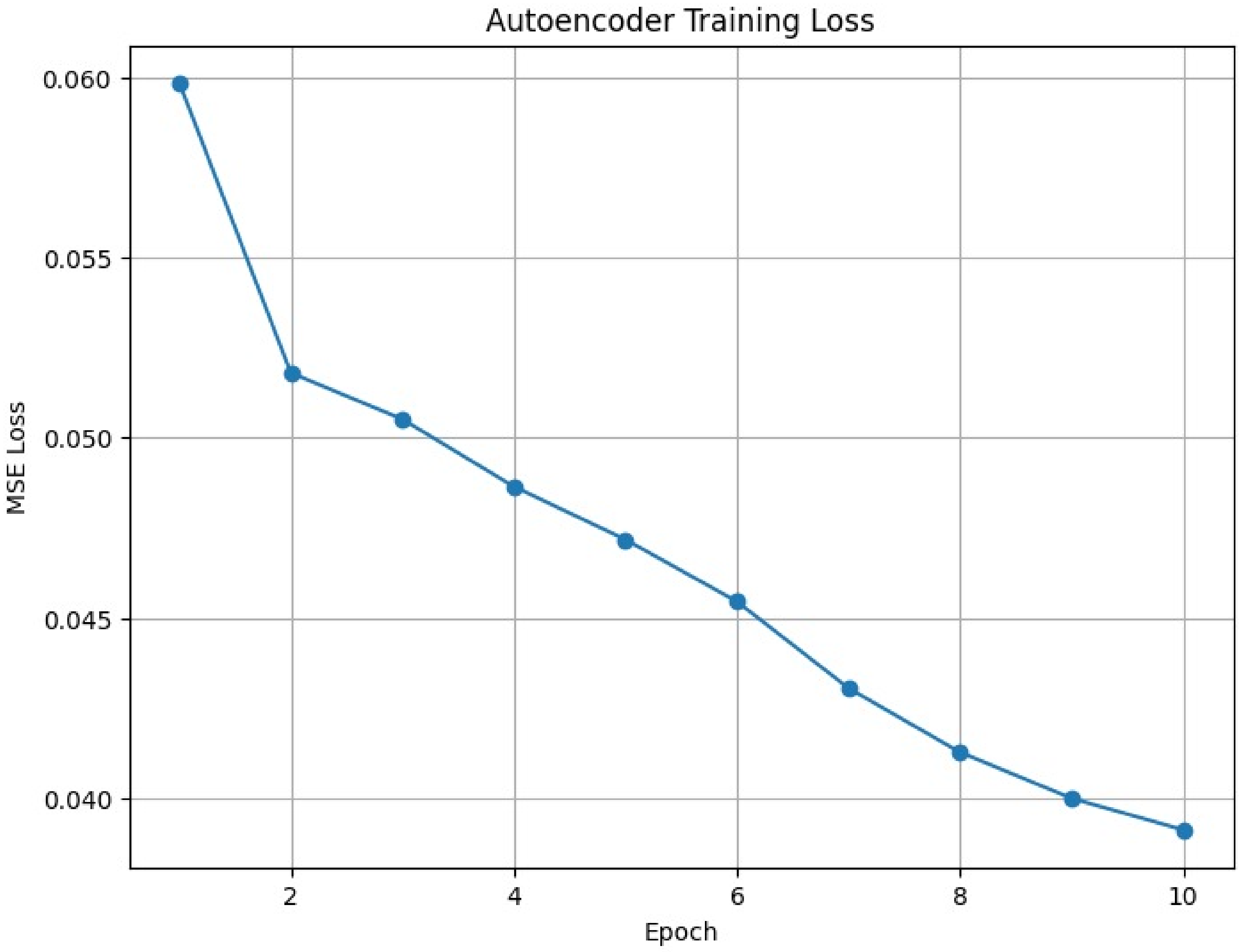

Figure 10: Training loss vs. epochs for the autoencoder model.

c. Embedding Extraction and Visualization: Extract the encoded representations (embeddings) from the trained autoencoder for the training data. To do this, pass the training data through the encoder part of the autoencoder model. Use the first two embeddings to create a scatter plot of the embeddings, colored by the activity labels. Include this plot in your solution PDF.

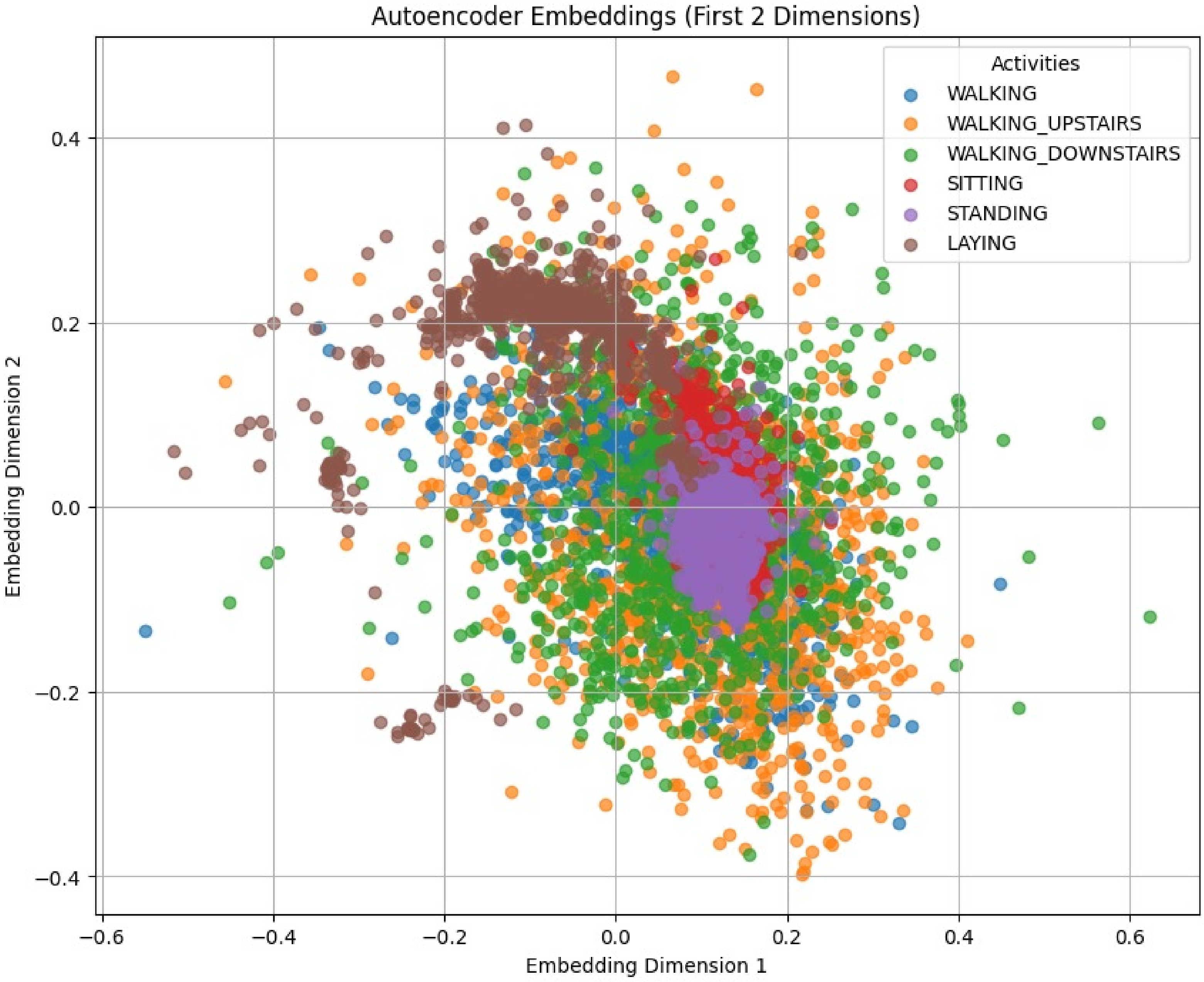

Figure 11: Scatter plot of the first two embeddings extracted from the autoencoder, colored by activity labels.

d. Clustering Evaluation and Comparison: Perform K-means clustering on the extracted 64D embeddings. Evaluate the clustering performance using the Adjusted Rand Index (ARI) by comparing the cluster labels with the true activity labels. Compare the ARI obtained by clustering the autoencoder embeddings with the clustering performance you obtained from the hand-engineered features in Part 4.2. Discuss which of the two provides the highest ARI and the potential reasons for that.

The ARI obtained by clustering the autoencoder embeddings is 0.4046, while the ARI obtained from the hand-engineered features is 0.4533. The hand-engineered features provide a slightly higher ARI compared to the autoencoder embeddings.

This difference could be attributed to the fact that the hand-engineered features are specifically designed to capture domain-specific characteristics of the data, which may align better with the ground truth labels. On the other hand, the autoencoder learns a representation in an unsupervised manner, which might not perfectly align with the clustering structure defined by the labels. However, the autoencoder embeddings still perform reasonably well, demonstrating their ability to capture meaningful patterns in the data.